Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea Magistrale in Informatica
Curriculum B - Informatica per il Management

Progettazione ed Implementazione di una Dashboard di Location Intelligence in Ambito Retail

Relatore:
Chiar.mo Prof.
Marco di Felice

Correlatore:
Federico Sitta

Presentata da:
Faienza Antonio

Sessione III
Anno Accademico 2018/2019

A mio padre ...

Introduzione

The world's most valuable resource is no longer oil, but data [1]. E' così che intitola un articolo del *The Economist*, storico quotidiano inglese, incentrato prevalentemente su temi di natura socio-economica. L'articolo oltre a toccare argomenti riguardanti privacy e regolarizzazione dei dati, asserisce come, solo nel 2017, le cinque società più importanti al mondo (Google, Amazon, Apple, Facebook e Microsoft), specializzate, ognuno a modo loro, nella gestione di grandi quantità di dati, abbiano avuto un utile nel primo trimestre di 25 miliardi di dollari.

Il valore dei dati è stato riconosciuto ormai da tutti. I dati possono essere analizzati per innumerevoli scopi, come ad esempio analisi delle performance, decision making, valutazione dei rischi.

Nelle compagnie odierne, sono una delle pietre miliari su cui è centrato il loro profitto. Che sia il focus aziendale o meno, tutte le aziende, pianificano il loro futuro, a medio o lungo termine, sulla base dei dati di cui dispongono. Tradizionalmente tutti i tipi di dati vengono acquisiti in formato grezzo per poi essere lavorati al fine di ottenere uniformità a dispetto di quella che è la fonte di approvvigionamento. In quest'ottica, il concetto di Big Data si è evoluto, ed è diventato più specifico a seconda dei campi di applicazione in cui è coinvolto: medico, economico, agricolo/alimentare, manifatturiero e così via dicendo.

La *BI*, meglio conosciuta come *Business Intelligence*, standardizza e regolamenta i processi volti alla raccolta e gestione dei Big Data in ambito aziendale. Provvede una serie di tecniche e best-practices col fine di poterne trarre vantaggio in termine economico, basandosi sul responso che sono in grado predire, analizzando prestazioni sia passate che presenti. Le tecniche di BI

sono andate sempre perfezionandosi negli anni, attraverso tool più specifici e performanti.
Una branca della BI più specifica è la *Location Intelligence*. In generale, rappresenta il processo di estrazione di informazioni significative da dati spaziali mediante l'utilizzo di tool specifici (GIS) e di darne una visione d'insieme immediata e di impatto usufruendo di mappe. L'obiettivo della tesi mira a dare una chiara esemplificazione di come questo avvenga in un contesto pratico. Il tutto è avvenuto durante un periodo di tirocinio formativo presso una società di consulenza avente sede a Casalecchio di Reno (BO).
Iconsulting è un'azienda nata nei primi anni del 2000 specializzata nella progettazione di sistemi best-in-class di Data Warehouse, Business Intelligence, Performance Management e Advanced Analytics. E' l'idea di un gruppo di ricercatori, che trae dalla ricerca stessa la propria energia. Ad oggi è partner strategico dei vendor tecnologici di riferimento come IBM, Microsoft, Microstrategy, Oracle, SAP e SAS. Ha sede a Bologna, Roma, Milano e Londra ed ha all'attivo più di 500 progetti di successo su un portfolio di oltre 150 clienti di elevato standing.
Cuore del modello di business è il laboratorio R&D a cui vengono dedicate il 25% delle attività aziendali, in collaborazione con prestigiose Business School ed Università Italiane. Gli attuali studi sono focalizzati su: Big Data, Advanced Analytics, Location Intelligence e Performance Management.
Il caso di studio preso in esame è stato incentrato attorno alla costruzione di una dashboard innovativa in ambito retail per un'azienda operante nel settore moda. Lo scopo, era quello di mostrare su mappa gli attuali punti vendita esistenti e cercare di predirne di nuovi per mezzo di un'analisi geo spaziale. Più specificatamente, sono state effettuate tre tipi di analisi. La prima è stata l'*analisi descrittiva*. In questo frangente viene offerta una panoramica riassuntiva della stato di salute dei punti vendita attuali, ponendo l'accento sui migliori negozi, i migliori Brand e il trend di vendita. E' possibile inoltre avere il dettaglio di ogni specifico punto vendita, relativo all'andamento generale, semplicemente cliccando su uno dei punti presenti sulla mappa. Parallelamente a quanto fatto per i singoli negozi, la dashboard evidenzia lo sviluppo anche a livello mondiale e regionale, mettendo in risalto i migliori paesi in termine di vendita.

La seconda parte è l'*analisi del potenziale.* Tale analisi individua la correlazione esistente tra vendite e caratteristiche dei beni venduti. Il tutto è coadiuvato da uno studio usando la Regressione Lineare. Per ogni Stato o Regione scelto è mostrato graficamente quale fattore incide maggiormente sulle vendite, così da poter appurare se le vendite sono in linea con quelle che sono le aspettative derivanti dallo studio di regressione. In caso contrario, il modello suggerisce graficamente quali possano essere i correttivi da adottare e su cui investire.
L'ultima parte, riguarda l'*analisi predittiva.*Avendo ora una panoramica della situazione degli attuali punti vendita, si pone l'accento su come identificarne di nuovi. Quest'ultima fase del progetto prende in esame le caratteristiche geospaziali di ogni punto vendita e li fonde con dati di natura statistica[1].Viene quindi elaborato un modello di Machine Learning che impara dai punti esistenti quali sono le caratteristiche geospaziali e statistiche che incidono maggiormente sul fatturato. Una volta elaborato il modello, l'*analisi della domanda* mostra su mappa se uno specifica zona è potenzialmente appetibile o meno; l'*analisi dell'offerta,*esclude le zone attualmente esistenti e che hanno costituito la base per costruire il modello predittivo.
Si dividerà l'elaborato nel seguente modo: nel primo capitolo sarà descritta la Location Intelligence e la BI in termini non specifici, risaltando le tecniche attualmente in voga sul mercato. Nel secondo e nel terzo, sarà descritto il progetto e le tecnologie adoperate in termini tecnici. Nell'ultimo, si andrà ad argomentare riguardo i vari tipi di analisi effettuate. Nell'ultima parte saranno fatte considerazioni di quello che è stato il lavoro svolto e di possibili sviluppi futuri.

[1]Essendo quest'ultimo, uno studio fatto a livello nazionale, i dati in questione provengono dal sito Istat: http://dati.istat.it/

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

I supporti alle decisioni aziendali

Le organizzazione sono sempre alla ricerca di strategie che permettano di incrementare gli utili e che consentano loro di apportare un significativo miglioramento ai processi produttivi che ne sono alla base. Per raggiungere questo scopo, quasi sempre accompagnano l'utilizzo di nuove tecnologie, al *dato*.
Quest' ultimo può essere di varia natura e di varia struttura. E' possibile infatti avere a che fare con, ad esempio, dati economici, piuttosto che con dati sanitari; come allo stesso tempo, è possibile trovare dati di tipo testo, immagine, "bit" etc. Anche i database che li contengono non sono tutti uguali: relazionali, non-relazionali, colonnari ne sono un esempio. Ciò che conta però non sono solo i dati, ma come questi vengono adoperati: possedere *Big Data* significa saperli analizzare per ottenere informazioni necessarie a prendere le migliori decisioni in contesti aziendali. Ogni dato, se non accompagnato da una fase di analisi, resta inutile. Per fare in modo che possa impattare positivamente, si devono considerare tre caratteristiche [2]:

- **volume:** ogni secondo attraversano la rete tanti dati, quanto era la totalità dei dati esistenti 20 anni fa. Dal 2012, vengono creati 2,5 *Exabyte* [1] al giorno e il numero raddoppia in circa ogni 40 mesi. Un

[1] Un Exabyte è uguale a 10^{18} Byte. Il che tradotto in GB equivale 1 miliardo di Gigabyte

caso pratico, è rappresentato da *Walmart*[2], la quale raccoglie ogni secondo circa 2,5 *Petabyte*[3];

- **velocità:** è importante almeno quanto il volume dei dati. Saper gestire grandi informazioni in poco tempo, può rappresentare un vantaggio competitivo notevole rispetto altre aziende. Da uno studio americano sui dati di vendita nel periodo natalizio, è stato possibile analizzare quante persone fossero presenti nei parcheggi di *Macy*[4] nel giorno di Natale, grazie alla velocità di localizzazione dei GPS. Durante il periodo di tirocinio, ho potuto constatare personalmente, quanto fosse importante avere un dato sempre aggiornato, sebbene il gran numero fosse un vincolo considerevole;

- **varietà:** Come già anticipato, i dati assumono "forma" diversa a seconda dei casi. Possono essere messaggi, immagini, testi scritti nei post di piattaforme social. Molte delle tecnologie che ci sembrano ormai obsolete, sono di pochi anni. I database devono saper gestire questa differenza di struttura e devono essere architetture scalabili che sappiano affrontare possibili *fault tolerance*[5].

1.1 La Business Intelligence: Cos'è e da cosa nasce

L'opportunità legata all'analisi dei dati nelle aziende, per ottenere profitto, ha generato un crescente interesse, facendo sì che potesse nascere la *Business Intelligence* [3]. Quest'ultima rappresenta una serie di *tool* e *Best-Practice* che servono da supporto alle aziende per prendere decisioni [4]. La serie di tool e *best-practices* a cui si riferisce [4] comprende [5]:

- Data mining
- Reporting
- Metriche e benchmarking delle prestazioni

[2] https://www.walmart.com/

[3] Un Petabyte equivale a 1 milione di Gigabyte.

[4] https://www.macys.com/

[5] Un sistema Fault Tolerance è un sistema che è in grado di funzionare anche in presenza di fallimenti di alcuni suoi componenti

- Analisi descrittiva
- Query
- Analisi statistica
- Visualizzazione dei dati
- Preparazione dei dati

Le aziende si servono della BI per [5]:

- Identificare le aree o i modi per aumentare i profitti
- Analizzare il comportamento dei clienti
- Confrontare i dati con la concorrenza
- Monitorare le prestazioni
- Ottimizzare le operazioni
- Prevedere il successo di nuove iniziative
- Identificare i trend del mercato
- Individuare eventuali problemi

Cosa prevede il ciclo di raccolta e produzione dei dati di output della BI? Per modellare la forza del modello di business adottato, esistono una serie di step.
Al fine di dare un esempio esplicativo, si immagini di rispondere alla domanda di un manager ai suoi dipendenti: "*Quanti sono gli ordini **per** categoria di prodotto **di** Gennaio ?*"
In questa domanda, si deve saper distinguere:

- **fatti:** rappresentano tutte le misure quantitative da mostrare nel report;
- **dimensioni:** rappresentano tutti gli attributi che identificano il contesto.

Nella precedente domanda è sia possibile estrapolare il contesto che identifica la *dimensione* (identificato dalla locuzione "*per*") sia l'istanza dalla dimensione (identificata dalla preposizione *"di"*). Il contesto e la sua istanza portano ad individuare il *fatto*, il quale risponde numericamente al quesito posto. Nella Fig. 1.1, le *dimensioni* sono "Category", "Product", "SKU";

Order Report Western Region
January 2009 (cont'd)

Category	Product	SKU	Quantity Sold	Cost	Order Dollars
Packaging	Box - Large	011-4822	700	$ 950.53	$ 1,100.00
	Box - Medium	011-4899	1,250	$ 1,001.84	$ 1,380.00
	Box - Small	011-5744	1,200	$ 1,200.72	$ 1,330.00
	Clasp Letter	011-1729	400	$ 352.82	$ 356.00
	Envelope #10	021-0011	2,000	$ 2,017.46	$ 2,080.00
	Envelope Bubble	021-0012	1,200	$ 866.51	$ 1,212.00
	All Packaging			*$ 6,389.88*	*$ 7,458.00*
Pens	Gel Pen Black	017-1999	5000	$ 116.39	$ 120.00
	Gel Pen Blue	017-2444	2990	$ 600.88	$ 624.91
	Silver Pen	017-3001	50	$ 128.46	$ 130.00
	All Pens			*$ 845.73*	*$ 874.91*
Grand Total				***$207,229.42***	***$214,896.91***

Page 10 of 10

Figura 1.1: Esempio di report che identifica fatti e dimensioni

mentre le *istanze della dimensione* sono le *tuple* appartenenti ad ognuno di esso ("Packing", "Pens", etc.). Le colonne "Quantity Sold", "Cost", "Order Dollars" denotano le *misure* o *fatti*. Il processo di raccolta e pulizia vede nel report, l'output finale. Ciò espone nient'altro, che la rappresentazione empirica di una modellizzazione conosciuta come *Star Schema.* Il suo nome è da attribuire alla sua forma tipica tentacolare che ricorda una stella. La Fig. 1.2 permette di distinguere due tipi di tabelle. Il primo tipo, chiamato *tabella delle dimensioni*[6], ne definisce il contesto, il secondo è la *Fact Table* responsabile della raccolta delle misure numeriche. Nello *Star Schema* sono presenti tutti gli attributi che saranno visualizzati nel report, tenendo conto anche del livello di granularità che è in grado di offrire. In fase di progettazione non è facile individuare univocamente i campi che determinano una

[6]La dimensione può essere riferito sia alla tabella che alla colonna. Di solito si usa per identificare dimensione degli attributi e dimensione della tabelle.

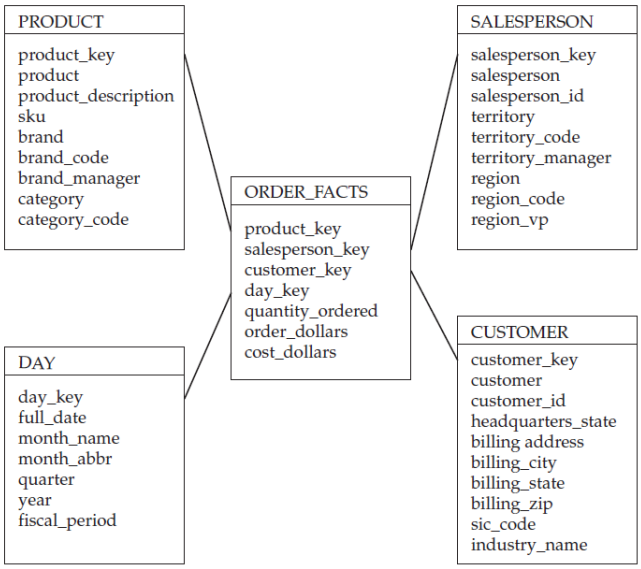


Figura 1.2: Un esempio di Start Schema per gli ordini

dimensione in quanto uno stesso campo può essere attinente a molteplici dimensioni. Generalmente si procede raggruppando gli attributi sulla base del loro *clustering* naturale, ma nulla toglie che, a seconda del contesto in cui deve essere integrato, una tabella può possedere raggruppamenti di diversa natura. Inoltre, ognuna, possiede due tipi di chiavi: la *Surrogate key*, utilizzata per tracciare i valori dal Data warehouse da cui prende il dato, e la *Natural key*, la quale determina univocamente l'ID di un'istanza della tabella.

La *Fact Table* identificabile dalla presenza delle *Surrogate key* delle varie dimensioni, deve assumere delle caratteristiche che la rendono tale e che ne determinano il comportamento. Deve perciò essere in grado di offrire vari livelli di dettaglio (*deep*); evitare confusioni (*grain*); contenere misure apparentemente non necessarie, estrapolabili dalle tabella delle dimensioni, ma necessarie per conferire ordine (*additivity*) ed evirare un numero di join eccessivi fra le varie dimensioni (*sparsity*).

Lo *Star Schema* rappresenta solo la parte finale in un processo di BI. Il centro focale è incentrato sulle modalità di raccolta e pulizia dei dati. In questo caso il protagonista è il *Data warehouse* e le varie architetture che è

possibile trovare nei contesti aziendali.

E' possibile suddividere i *Data warehouse* in tre categorie. Le prime due sono chiamate *Enterprise Data warehouse*e sono associate rispettivamente a *W. H. Inmon* e *Ralph Kimball*. Ciò che preme sottolineare è che non esiste "una migliore architettura" rispetto ad un'altra. Ognuna ha delle peculiarità che la caratterizza. Soffermandoci sull'architettura di *Inmon* chiamata

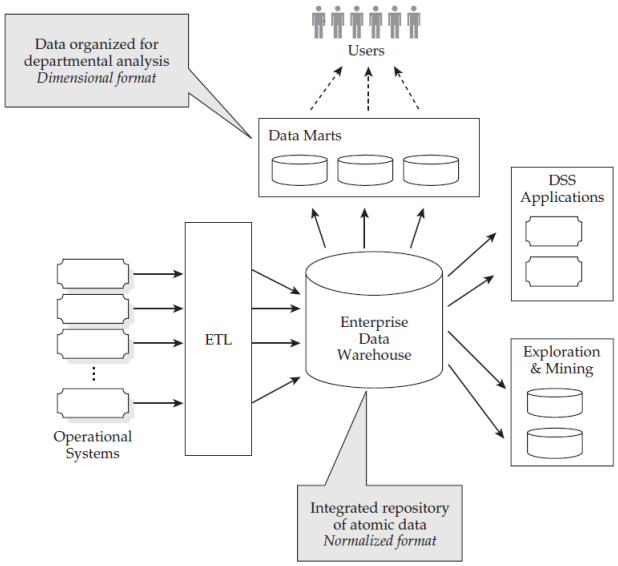


Figura 1.3: Inmon's Corporate Data warehouse Architecture

Inmon's Corporate Data warehouse, è possibile notare nella parte sinistra gli *Operational System*, i quali rappresentano la fonte di approvvigionamento dei dati: possono essere generati da dati gerarchici, dati relazionali o semplicemente da spreadsheet. Durante il processo di *ETL (extract, transform, load)* vengono acquisite le informazioni dai vari *Operational System* e aggiunti nell' *Enterprise Data warehouse* in forma atomica. Quest'ultimo infatti, rappresenta il repository centrale contenente tutti i tipi di dati con il massimo livello di dettaglio. La sua concreta rappresentazione è generalmente un database relazionale o colonnare a seconda dei casi[7]. Prima di

[7]SAP HANA è un esempio di DBMS colonnare in memory sviluppato e commercializzato dalla società SAP. La sua struttura, conferisce enorme velocità nella gestione dei dati.

essere fruibili dagli utenti, i dati vengono aggregati in *Data Mart*, secondo quello che il livello di dettaglio richiesto. Un altro tipo di architettura è conosciuta come *Kimball's Dimensional Data warehouse.*

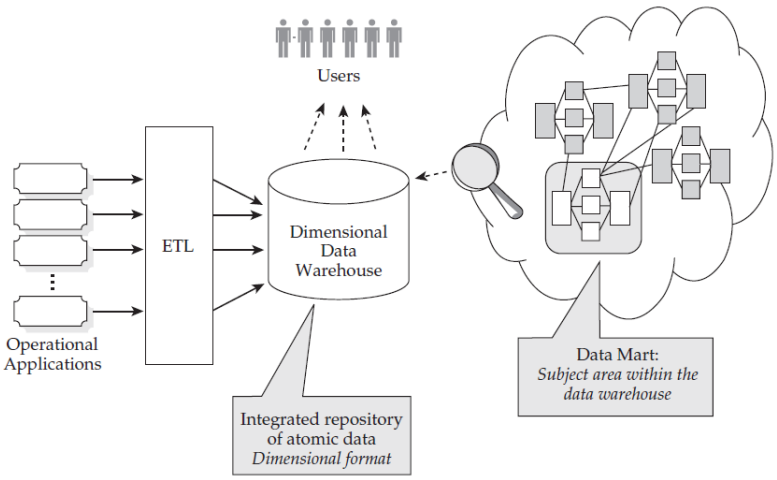


Figura 1.4: Kimball's Dimensional Data warehouse Architecture

Ugualmente a quanto visto all'architettura di *Inmon*, è possibile individuare gli *operational system* che si occupano della raccolta dei dati. Anche in questo caso, tramite un processo di *ETL* i dati vengono immagazzinati in forma atomica all'interno del *Dimensional Data warehouse.* Ciò che differenzia il *Dimensional* dall'*Enterprise Data warehouse* riguarda la possibilità di progettazione tramite *Star Schema* piuttosto che da un modello *ER* tipico nel caso precedente. Inoltre, come si può osservare dalla Fig. 1.4, è possibile accedervi in maniera diretta, senza passare tramite i Data Mart. L'ultimo tipo di architettura è denotato *Stand-Alone Data Marts.*

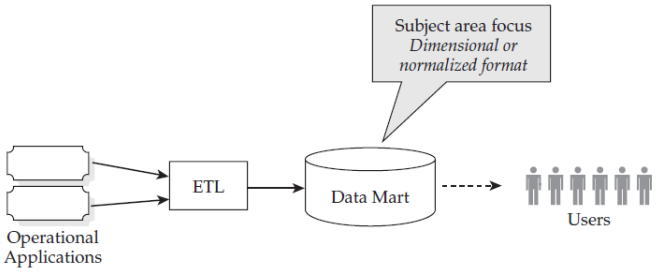


Figura 1.5: Stand-Alone Data Marts Architecture

L'architettura è *Stand-Alone* e quindi non associabile alle precedenti. Con questa espressione, si vuole intendere la sua propensione ad essere installato in vari settori aziendali. La sua utilità è volta soprattutto ad offrire in maniera rapida e con poche risorse un'architettura solida. Ha sicuramente il vantaggio di offrire elevate performance e garantire una maggiore sicurezza dovendo autorizzare l'accesso ad un insieme minore di dati. Tra i lati negativi c'è la perdita di efficienza, sia per lo scarso storage di cui dispone, sia per la possibile ridondanza del dato che lo accompagna. Si immagini un contesto aziendale che utilizzi molteplici *data mart*, uno per ogni reparto. I vari processi di ETL che leggono dagli *Operational System* e trasportano il dato possono avere settaggi diversi. Ciò comporta ridondanza come già anticipato, ma anche un diverso livello di "atomicità" , il che può portare a difficoltà di comparazione delle informazioni.

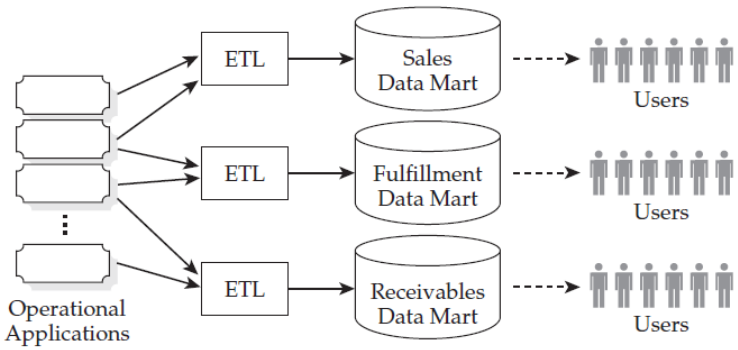


Figura 1.6: Multipli Stand-Alone data marts

1.2 Location Intelligence: storia ed evoluzione dalla BI

Questo appena visto, descrive succintamente il mondo legato alla BI. Lo scopo di questo elaborato, non è tanto dare delle chiare delucidazioni a riguardo, ma dare un overview generale della BI, per spostare il focus su un'altro aspetto ad esso correlato. Da qui in poi, ci si addentrerà infatti in una branca della BI, chiamata *Location Intelligence*, la quale partendo da quanto visto, fornisce agli utenti che ne usufruiscono informazioni aggiuntive che in un normale report non sono captabili.
Generalmente con *Location Intelligence* si intende l'insieme dei processi e delle tecniche utilizzate per ricavare informazioni significative da dati *geospaziali* ricavando modelli, tendenze, potenzialità non visibili con le normali tecnologie attualmente offerte dalla BI [6] . Può essere considerata come la sua estensione e che interagisce con i sistemi informativi attualmente in essere [7]. Ad oggi, non esiste una chiara descrizione degli elementi che la compongono.
In generale, mentre, le normali operazioni di *Business Intelligence* permettono di rispondere alle domande "*Chi*", "*Cosa*", "*Quando*", la *LI* si focalizza sul concetto di "*Dove*".

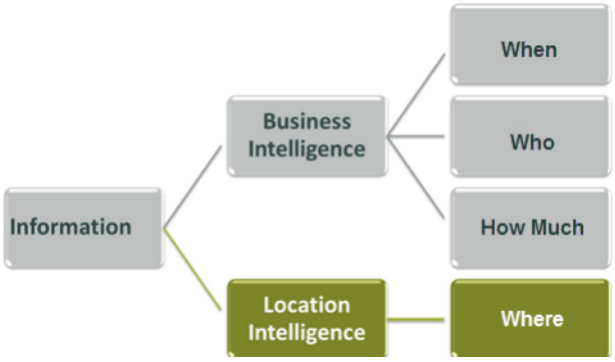


Figura 1.7: Overview Location Intelligence

Il primo caso nella storia di *Location Intelligence* è quello di John Snow. Snow è infatti considerato il pioniere di questa materia ed è attualmente ricordato soprattutto per questo. Gli viene dato merito, infatti, di aver sa-

puto individuare la ragione scatenante dell'epidemia di colera[8] [8] nel 1800 a Soho[9] quartiere di Londra. In quella circostanza, egli suppose che la propagazione della malattia, fosse da attribuire all'utilizzo dell'acqua. Durante i suoi studi, per localizzare la fonte dell'insorgenza dell'epidemia, utilizzò una piantina del quartiere di Londra come strumento per circoscrivere le zone maggiormente colpite. Questo gli permise di notare che la maggior parte dei casi erano localizzati intorno una pompa di acqua nel distretto di Soho. Bloccando l'erogazione riuscì a contrastare il propagarsi della malattia. Dopo essere stato ampiamente criticato, gli fu successivamente riconosciuto il merito di questa scoperta [9].

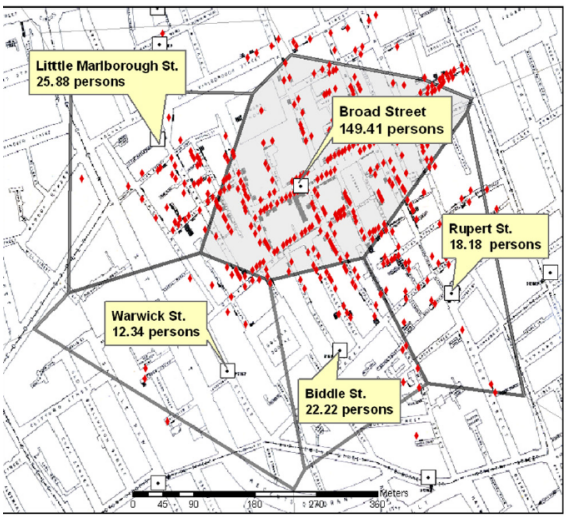


Figura 1.8: Diagramma di Voronoi per stimare il numero di morti nelle aree

Per raggiungere questo scopo, come viene mostrato nell'articolo di *Koch*e *Denike*, [9] , tassellò la superficie con un diagramma di Voronoi, stimando la concentrazione di mortalità grazie alla divisione del numero di morti per il numero di case presenti nell'area, moltiplicato per il numero di persone presenti per ogni abitazione, che a quei tempi si aggirava intorno alle 10

[8]Il colera è un'infezione causata da un batterio Vibrio Cholerae che ha come primi sintomi diarrea, vomito, crampi e acidosi. La sua trasmissione avviene per contatto orale, con alimenti contaminati. Nel 19 secolo si diffuse più volte in un'area circoscritta nei pressi del Gange.

[9]Quartiere a est di Londra

unità.
Ovviamente John Snow non sapeva di essere il precursore di un processo quale la *Location Intelligence*. Ai giorni d'oggi, la tecnologia è sempre più performante e per ottenere risultati ci si avvale di strumenti migliori. Ciò che comunque ha permesso negli ultimi anni di favorire l'incremento in termini di utilizzo della LI sono i *database spaziali*.
Non esiste una definizione univoca che lo descrive. Secondo R. H. Güting [10] , per essere considerato tale deve possedere le seguenti caratteristiche:

1. Un *database spaziale* presenta innanzitutto le caratteristiche tipiche dei Database;

2. Offre tipi di dati chiamati *spaziali* (STD) sia nel modello dei dati e sian nel linguaggio di query;

3. L'utilizzo dei dati spaziali è correlato all'utilizzo di *indici spaziali* e di funzioni/algoritmi che consentono di effettuare operazioni di carattere geometrico quali *intersezione* tra tipi, *unione* fra geometrie, etc.

1.2.1 Dati Spaziali

La precedente definizione chiarisce il concetto di *database spaziale* ma lascia ancora dubbi riguardo la definizione delle sue componenti. Cosa sono i dati *spaziali*? E cos'è un *indice spaziale*? Andando con ordine, i*tipi spaziali* sono gli elementi su cui è possibile effettuare operazioni di *selezione*, *intersezione*, *unione* su entità geometriche comunemente chiamate spaziali. Normalmente non tutti i database offrono gli stessi tipi, o le stesse operazioni di manipolazione su di essi[10]. I tipi più generici, che facile trovare fra i vari DB spaziali sono [10] [7]:

- Il **punto** raffigura la rappresentazione geometrica di un oggetto. Per il suo utilizzo è rilevante solo la posizione nello spazio di riferimento, ma è trascurabile la sua estensione. Tutto può essere modellato come punto: una casa, una macchina, una scuola, una città e così via. Può

[10]Nel prossimo capitolo sarà fatto un confronto tra le features offerte dai vari database spaziali.

essere inteso come la componente spaziale più semplice e la sua rappresentazione è connotato all'utilizzo delle coordinate (x,y) per collocarle nello spazio

- Una **linea** è l'astrazione geometrica usata per connettere punti e muoversi nello spazio. E' data da una sequenza di segmenti o da una sequenza di punti. La linea può essere retta, alternativamente può essere curva o ancora *composta*: se formata da una retta e una curva.

- Un **poligono** è l'astrazione geometrica per raffigurare una regione di spazio in cui collocare oggetti. E' definito da varie linee spezzate in cui il vertice iniziale coincide con quello finale. Per definizione, inoltre, l'area di un poligono deve essere contigua e per questo si deve poter attraversare senza uscire dai confini che lo delimitano. Sono anche da considerarsi poligoni, le regioni di spazio che hanno le caratteristiche tipiche dei poligoni, ma che al loro interno presentano dei vuoti.

- Una **collezione** presenta vari elementi geometrici. Può essere di due tipi:

  - *omogenea:* se gli elementi che compongono la collezione sono dello stesso tipo: due o più poligoni, due o più linee, duo o più punti
  - *eterogenea:* se composta da vari elementi geometrici: punti, linee o poligoni.

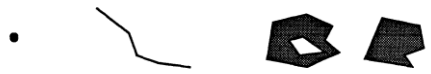


Figura 1.9: Componenti spaziali

Tutti questi sono tipi spaziali in 2D. Molti database implementano anche tipi 3D quali *superficie composta*, *solidi*, *collezione di solidi*, ma la loro trattazione è stata volutamente omessa perché esula dal contesto[11].

[11]Per maggiori informazioni, si legga (Overview of Oracle Spatial)(Güting 1994)

1.2.2 Indice spaziale

Avendo ora spiegato cosa siano i tipi spaziali, resta da spiegare la funzione dell'*indice spaziale*. Una premessa che vale la pena fare per poter consentire al lettore di capire il funzionamento dell'indice spaziale è l'*MBR*. *MBR* sta per *Minimum Bounding Rectangle*.
Come il nome suggerisce, l'MBR permette di definire il contenitore minimo che contiene la geometria a cui ci si riferisce.

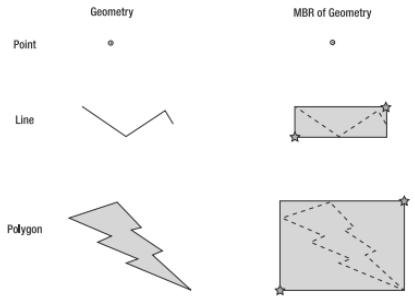


Figura 1.10: Rappresentazioni di MBR in geometrie spaziali

La comprensione degli MBR permette di capire l'utilità dell'indice spaziale. Il vantaggio che offre è incentrato sul facilitare le operazioni eseguibili tramite query riducendo il costo computazionale. Se si sta lavorando con

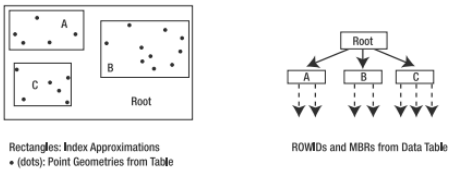


Figura 1.11: Esempio di Indice spaziale

un set di punti, come nell'esempio preso da [7], si può avere l'esigenza di effettuare operazioni su di essi, come lo può essere il calcolo della distanza massima. Il database per operare dovrebbe, in condizioni normali, calcolare punto per punto la distanza che separa l'uno dagli altri e questo comporterebbe un best effort notevole.

Tuttavia questa casistica non si palesa, grazie al MBR, il quale consente di delimitare il range spaziale del set di punti. Per cui avremo, come si può vedere dalla Fig. 1.11, tre set di punti corrispondenti a tre collezioni diverse. La radice porta il riferimento ad ognuno di essa e solo dopo aver acceduto alla singola collezione di interesse, si può puntare ad un al singolo elemento contenuto nella collezione.
Oltre alla selezione spaziale, l'indicizzazione spaziale supporta anche altre operazioni come l'unione spaziale (unione di due rettangoli), intersezione spaziale (intersezione di due linee) ecc. Concretamente, l'indice viene implementato usando delle strutture ad albero chiamate *R-Tree*[12]. Le loro elevate prestazioni soprattutto con dati reali, li ha resi adatti ai Database copiando in memoria i nodi solo quando servono ed evitando di prendere quelli superflui.

1.2.3 SRID

La parola *SRID* sta per *Spatial Reference System* e sottende la modalità in cui i dati astratti vengono espletati concretamente in un sistema di riferimento. Il sistema di riferimento in questione, banalmente, può essere accostato ai sistemi cartesiani. Le due linee rappresentate in Fig. 1.12, sono esattamente uguali, ma la loro raffigurazione nello spazio è diversa. Infatti la *A* avrà un sistema di coordinate diverso rispetto la *B*. Lo stesso concetto si pone nella proiezione della Terra, la quale possiede una forma ellissoidale, su una superficie piana in 2D. Come ciò avviene, non è univoco.

[12]Gli algoritmi di ricerca usano gli MBR per decidere se cercare o meno nel nodo figlio del nodo corrente. In questo modo la maggior parte dei nodi non viene esplorata dagli algoritmi. Per questo motivo, come per i B-tree, ciò rende gli R-tree adatti ai database, dove i nodi possono essere copiati in memoria solo quando necessario.

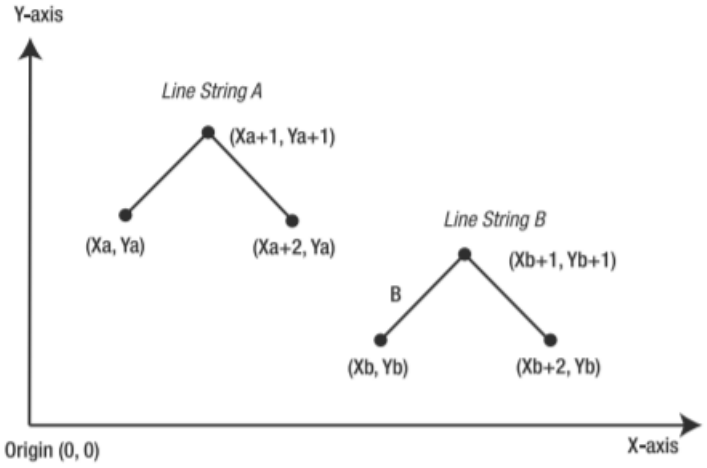


Figura 1.12: Coordinate in un sistema di riferimento

Figura 1.13: Esempio di rappresentazione su un piano 2D della Terra

La geocalizzazione di un punto, o una linea, o altro, può andare incontro a delle distorsioni rispetto la posizione reale che esso assumo nello spazio. La domanda che sorge spontanea è se è possibile rappresentare un oggetto senza distorsioni.
Esistono due approcci. Il primo ruota intorno alla possibilità di rappresentare la terra usando superfici ellissoidali e proiettarle direttamente, in toto, su un piano 2D. Il secondo, invece, converge nella proiezione della terra usando solidi tridimensionali e solo successivamente su un piano 2D.

- **Geodetic Coordinate Systems**: L'idea generale, espone come, per

individuare la posizione di un punto su una superficie terrestre sia consono proiettare tutta la superficie terrestre su un piano 2D e successivamente poter effettuare operazioni spaziali sui dati come il calcolo della distanza tra due punti. Ma ad oggi non esiste una rappresentazione conforme della superficie della terra, sia perché quest'ultima non è un'ellisse perfetto e sia perché non esiste, geometricamente parlando, una rappresentazione univoca dell'ellisse. Infatti, di norma si individua il centro della terra con coordinate *(0,0,0)*. La scelta di orientamento degli assi può differire a seconda dei casi. Il *Geodetic Coordinate Systems* non è nient altro che una convenzione, la quale prevede che l'asse z punti verso l'origine internazionale convenzionale (CIO), mentre l'asse x verso il meridiano di Greenwich. L'asse y invece andrà a completare con l'asse x la regola della mano destra[13] [11].

- **Projected Coordinate Systems**: quest' altro tipo di approccio parte dal presupposto che nell'effettuare un'operazione tra due figure geometriche, siano considerata sempre una porzione della superficie terrestre e non il tutto. In questo modo la proiezione su spazio 2D dovrebbe essere più accurata. Anche in questo contesto, il suo utilizzo non è univoco in considerazione della grandezza dei punti, la distanza, la direzione o generalmente l'utilizzo che se ne deve fare. Per tale motivo, esistono vari sistemi di proiezione come la *Marcator Projection* la quale prevede l'ausilio di un cilindro per la raffigurazione sulla terra e solo in un secondo momento la sua rappresentazione su uno spazio 2D. Avere una riproduzione della terra sotto forma di cilindro è utile se si deve effettuare operazioni spaziali nei pressi dell'equatore, perché altrimenti, per altre porzioni terrestri, eventuali operazioni risulterebbero distorte per via dell'eccessivo schiacciamento della terra sui meridiani troppo distanti dall'equatore [12] . La *Conic Projection* è un'alternativa alla *Marcator*. Come si può immaginare, in questo caso la terra viene proiettata con l'ausilio di un cono. La proiezione risulta essere meno distorta per via dell'utilizzo del centro del cono usato a mò di asse terrestre. In questo modo i meridiani e i paralleli non sono altro

[13] https://www.evl.uic.edu/ralph/508S98/coordinates.html

che un approssimazione sulla base dell'asse terrestre [13]. Infine un ultimo tipo di proiezione, è chiamata *Azimuthal Projection*. Essa, ha la caratteristica, di mantenere inalterato rispetto altre zone sia l'angolo di proiezione (azimuth) che la distanza permettendo la ricollocazione relativa di tutti i punti che compongono la terra [13] .

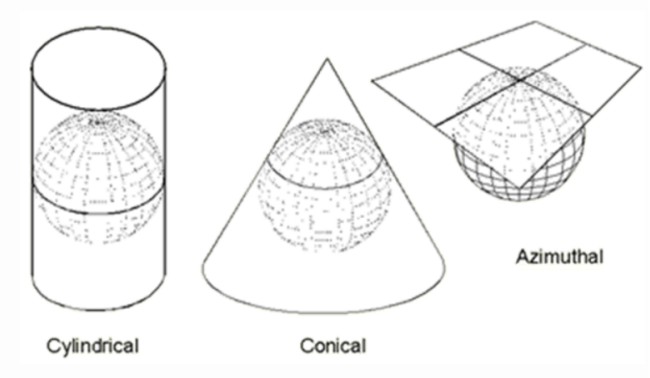


Figura 1.14: Proiezione Cilindrica, Conica e Azimutale

La Fig. 1.14 presa dal sito dell'università Humboldt State University[14] riassume i vari tipi di proiezioni sferiche utilizzando i sistemi di coordinate.
Gli ultimi due concetti che vale la pena introdurre, utili ai fini di questo lavoro di tesi, riguardano una struttura dati frequentemente usata per la manipolazioni dei dati spaziali nelle applicazioni Web: i **GeoJSON** e un formato dato geospaziale utilizzato dai software GIS: gli **ShapeFIle.**

1.2.4 GeoJSON

Fin ora si è argomentato di dati, Sistemi di proiezione e indici ma non si è ancora discusso delle modalità di rappresentazione degli oggetti spaziali sulle mappe. Non esiste una modalità univoca, ma per gli scopi del progetto si è utilizzato forse una delle più famose strutture dati che assolve a questo scopo: il formato **GeoJSON**. Il **GeoJSON** è un estensione del famoso **Json**[15]: un formato file aperto creato inizialmente per essere usato nelle

[14] http://gis.humboldt.edu/OLM/Lessons/GIS/03%20Projections/ProjectionFamilies3.html
[15] https://www.json.org/

comunicazioni client-server nelle applicazioni AJAX. E' stato sviluppato in Javascript e la sua diffusione è avvenuta sia per la facilità di utilizzo e di parsing sia per la leggibilità che lo rendono una valida alternativa ad XML. La sua struttura tipica è la seguente:

```
{
        "name": "Mario",
        "surname": "Rossi",
        "active": true,
        "favoriteNumber": 42,
        "birthday": {
                "day": 1,
                "month": 1,
                "year": 2000
        },
        "languages": [ "it", "en" ]
}
```

In questo scorcio sono presenti tutti i tipi che è possibile usare: *oggetti*, *array*, *valori*, *numeri*Interi, decimali, etc) , *whitespace*, *booleani* e json stessi. Per ognuno di esso è associabile una chiave definita "attributo" denotata da una stringa.

Il GeoJSON parte da questa struttura e la caratterizza con degli attributi specifici. Come si può apprendere dal sito[16], viene descritto come "*un formato per codificare una varietà di strutture di dati geografici.*" I dati geografici a cui fa riferimento sono: punti, linee, multi linee, poligoni e multipoligono che come visto sono i tipi di dati tipici dei database spaziali. Un esempio pratico è il seguente:

```
{
  "type": "Feature",
  "geometry": {
   "type": "Point",
   "coordinates": [125.6, 10.1]
  },
  "properties": {
   "name": "Dinagat Islands"
  }
 }
```

Da come si può vedere per ogni dato spaziale sono mappate le coordinate x,y che lo collocano nello spazio. Nell'esempio è un punto, ma vale lo stesso discorso per le linee e i poligoni o qualsiasi altra struttura supportata. Nel caso, ad esempio, fosse una linea, sarebbero contenute nella struttura tutte le coppie di punti che ne sanciscono la composizione. E' inoltre possibile

[16] https://geojson.org/

caratterizzare la struttura con altre informazioni che la identificano.

1.2.5 ShapeFile

Gli ShapeFile è un file con estensione *.hsp* contenente delle geometrie vettoriali, le quali sono usate per raffigurare i vari dati spaziali all'interno di software *GIS*[17]. Per correttezza, quando si parla di questi tipi di file si tende ad escludere i file con estensione *.shx* e quelli con estensione *.dbf*. In realtà per operare con i software GIS è fondamentale avere tutti i tipi. La ragione sta nel fatto che i *.shx* sono degli indici che hanno metadati sulle figure contenuti nei file *.shp*. Per ognuna di essa infatti, esiste l'indice di riferimento nel suddetto file. Inoltre è contenuta la sua lunghezza. Invece i file.dbf contengono informazioni aggiuntive, non strettamente correlate agli shape. E' da sottolineare che ogni file può contenere solo una geometria specifica e non geometrie eterogenee. Se uno shape contiene punti, conterrà solo punti e non anche linee o poligoni [14].

1.2.6 Casi di utilizzo

Attualmente i settori che adoperano la Location Intelligence sono:

- **Servizi Finanziari**: Gli account dei clienti sono monitorati ed utilizzati nel settore finanziario, in cui le normative volte a prevenire le frodi e il riciclaggio di denaro impongono agli enti di comprendere correttamente i collegamenti tra diversi account e identificare comportamenti potenzialmente fraudolenti. Questo può essere fatto individuando clienti e transazioni, incidenti e aree di rischio e identificando le relazioni geografiche tra i conti.

- **Settore Assicurativo**: I dati di posizione geocodificati con precisione possono fare la differenza nella valutazione di un bene in una particolare area geografica. Questa differenza può essere anche di poche centinaia di metri ma comporta enormi implicazioni in termini di

[17] Rappresenta un sistema che permette acquisizione e analisi di componenti spaziali. Opera sia con immagini vettoriali che raster. E' quindi un sistema informativo che estrae informazioni dalla rappresentazioni di dati spaziali su mappe

costi. Da un lato c'è il rischio che l'assicuratore sottoponga al cliente una polizza che ha un rischio maggiore di quello stimato; dall'altro vi è il rischio che un cliente possa pagare un prezzo eccessivo rispetto il rischio esistente con una conseguente perdita di clientela.

- **Retail**: I rivenditori che utilizzano beacon Bluetooth a bassa energia interni, RFID o Wi-Fi in combinazione con un'app basata sulla posizione corrente, possono essere in grado di analizzare il comportamento di acquisto dei clienti in negozio in una data area e, di conseguenza, ottimizzare l'inventario del negozio. Questo tipo di coinvolgimento del cliente nel punto di acquisto può aumentare gli acquisti e fidelizzare il negozio quando le offerte vengono offerte al momento giusto.
- **Customer Experience**: Le esperienze degli ospiti possono essere gestite in luoghi di grandi dimensioni, come stadi o resort, o per migliorare le esperienze negli aeroporti e nei viaggi. La ricerca di strade, l'ordinazione e persino la gestione delle code per servizi richiesti come servizi igienici o bevande possono essere migliorate con i dati e l'analisi della posizione e con i risultati prodotti nelle zone in cui si è agito.

Per avere un caso empirico del suo funzionamento, come *R. Kothuri*, *A. Godfrind* e *E. Beinat* spiegano nel loro libro [7] , si immagini di voler individuare il sito più adatto per aprire un nuovo centro commerciale. Si supponga inoltre, che la sua individuazione, presenti dei vincoli:

- deve essere locato in un'area in cui è consentita la costruzione;
- debba avere delle dimensioni adeguate;
- non sia esposto a rischi;
- debba essere situato nei pressi di strade per favorire una buona accessibilità.

Escludendo i problemi legati alla componente demografica, gli autori indicano gli step per una corretta analisi:

1. dalla mappa, vengono selezionate tutte le aree edificabili etichettabili come "aree commerciali";

2. usufruendo di una mappa fornita ipoteticamente da un agenzia immobiliare, si selezionano tutti gli edifici sufficientemente grandi in vendita;

3. partendo da una mappa dei rischi, che indica le aree di sicurezza intorno ai fiumi, si eliminano tutte quelle zone soggette ad inondazioni;

4. Sovrapponendo le varie selezioni, alla fine, si considerino solo quelle aree che sono nella vicinanza di strade.

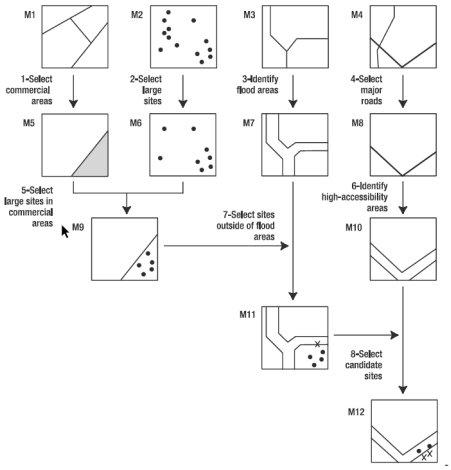


Figura 1.15: Analisi spaziale per la scelta della locazione di un nuovo punto vendita

Questo appena descritto viene riassunto dalla Fig. 1.15. Fino al passaggio 8, sono selezionate le aree che rispecchiano i vincoli che ci si era prefissati inizialmente. Nel passaggio 9 sono raffigurati i papabili nuovi centri commerciali. E' da notare che quest'ultimo è frutto della sovrapposizione tra il 5, il quale tende a scremare le aree commericale non idonee, con il 6. Si può notare come soltanto 5, dei centri commerciali di partenza, sono conformi ai vincoli. Successivamente, incrociando il passaggio 9 con il 10 e l'11 emerge che dei siti presenti in partenza, soltanto due sono conformi alle specifiche e rappresentano due candidati ideali ad ospitare un centro commerciale.
Esistono studi scientifici che sfruttano le tecnologie moderne per mettere in

piedi sistemi di *Location Intelligence* avanzati con l'ausilio di tool di appoggio come Foursquare[18] per l'inquadramento del posizionamento ottimale di un punto vendita su un'area definita [15] , oppure come mezzo sostitutivo a tecniche più dispendiose a livello economico e temporale, come lo può essere un piano urbano per la raccolta dati sulla domanda dell'utilizzo delle biciclette, *bike trip demand (BTD)* [16].
Tutte le operazioni da effettuare, atte a trovare la giusta locazione, sono il frutto di query SQL e i risultati non sono nient'altro che tuple (punti, poligoni, linee, griglie e così via) di ogni tabella adoperata per questo processo di selezione. L'analisi della posizione utilizza i dati provenienti da diverse fonti per comprendere diversi fattori geografici e spaziali e utilizza tecniche statistiche per prevedere la posizione migliore per ciascun negozio [17].
Volendo porre l'accento sugli aspetti più generali della Location Intelligence è possibile distinguere 3 *core [6]*:

- *Analisi Descrittiva*
- *Analisi del Potenziale*
- *Analisi Predittiva*

La fase di *Analisi Descrittiva* è strettamente legata al **Geocoding**[19]. Lo scopo primario è quello di associare le coordinate geografiche per mezzo di un indirizzo. Inoltre, un altro scopo a cui assolve, è legato alla correzione di eventuali errori nell'indirizzo. In questo caso si parla di *Address normalization*: correzione degli errori facendo in modo che tutti gli indirizzi siano chiari e comprensibili. La normalizzazione prevede anche l'eliminazione di eventuali duplicati che sarebbero trattate come voci separate nel database [7] .
In questo contesto, come visto nei precedenti paragrafi, ogni oggetto rappresentabile su mappa, ha almeno o più (se è una linea o poligono) coppie di coordinate (x,y) assegnate in un database Geospaziale. A questo punto, i dati assumono significato se visualizzati con una logica alla base. Selezione, unione, interazione sono solo alcune delle operazioni utilizzabili grazie alle

[18] https://it.wikipedia.org/wiki/Foursquare
[19] Detta anche Georeferenziazione

quali è possibile escludere aree a fronte di altre. Tale processo prende il nome di "*Data Refining*" . Per esempio, si immagini di voler mostrare su mappa le aziende che hanno un ROI[20] superiore ad una certa soglia, escludendone delle altre. Per raggiungere questo scopo, si può usufruire di:

- **Visualizzazione (Mapping)**: grazie alla quale si può offrire una rappresentazione colorata agli utenti in maniera non convenzionale. E' permesso quindi zoomare, selezionare ed escludere aree. Non comprende solo informazioni alfanumeriche, ma anche quelle spaziali all'interno dei report. Ovviamente, può essere accompagnata a grafici che consentono di interpretare al meglio il significato che la proiezione su mappa offre.

- **Analisi delle aree**: la quale da l'autonomia a scegliere una specifica visualizzazione piuttosto che un altra. Questa fase si cela dietro l'utilizzo di diversi layer disponibili. Normalmente in un'analisi generica di BI è possibile scegliere il livello di dettaglio delle informazioni che si desidera. Ciò è la conseguenza delle varie forme di aggregazione che è possibile dare al dato. Se un'azienda agroalimentare volesse analizzare il consumo di carboidrati in un'area geografica, potrebbe scegliere di vedere in generale per ogni area qual sia il consumo associato, ma allo stesso tempo potrebbe voler soffermarsi su uno specifico tipo di carboidrato escludendone degli altri. Analogamente nella LI è concesso avere un livello più generico di visualizzazione, come ad esempio la mappa della città, ma anche specificare solamente la rete stradale, la rete fluviale che scorre nei confini o solo il flusso di congestione del traffico e così via. Tali operazioni, sono realizzate utilizzando query soventi nei DB spaziali (*Join, Disjoin, Intersection, within*).

[20]Return of Investment: è un indice di bilancio che indica la redditività e l'efficienza economica delle aziende

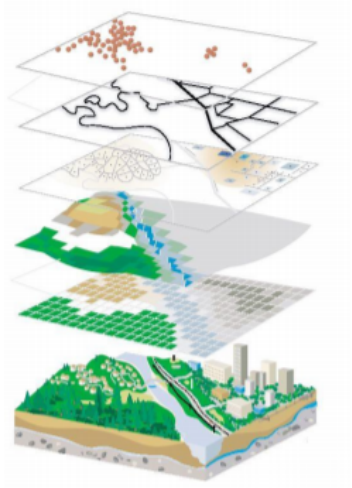


Figura 1.16: Visualizzazione dei diversi Layer

L'*Analisi del Potenziale,* può essere applicata a problemi analitici, e contemporaneamente, incorporata nei processi *Real Time* aziendali, come ad esempio nella gestione dei processi relazionali con i clienti. L'ottimizzazione può consentire ad una compagnia che sfrutta la *Location Intelligence* di individuare, sulla base dei suo profili schedulati, di ottimizzare le performance dei risultati attualmente esistenti e capire se si sta operando nel migliore dei modi o se si necessita di apportare dei correttivi. Il tutto può e deve avvenire, incrociando sia i risultati di vendita sia l'analisi del territorio in cui si sta operando. Questo aspetto può comprendere anche materie di carattere economico-strategico, come lo può essere la Basket Analysis.
L'*Analisi Predittiva* o **Forecast Analysis** rappresenta il fulcro della *LI*. Con essa è possibile riconoscere nuovi *trend*, fare previsioni su ipotetici andamenti futuri. La conoscenza e l'influenza di fattori spaziali sono un elemento fondamentale nell'applicazione dei "modelli predittivi". La considerazione di tali fattori porta ad una maggiore trasparenza (cosa succede? dove?) e dà anche la possibilità all'utente fare previsioni migliori. Ad esempio, sapendo quale sia l'andatura media degli automobilisti e sapendo quelle che sono le aree con il maggior numero di incidenti stradali, si determinano le aree più soggette ad incidenti. Si immagini che lo Stato voglia disporre delle normative che limiti la velocità. Tramite un'analisi predittiva sarà possibile stimare,

e graficamente rendere visibile su mappa come tali direttive impatteranno sul numero di incidenti. Per effettuare un'analisi predittiva di questo genere occorrono una serie di variabili, alcune delle quali esterne al dominio applicativo aziendale:

- Dati demografici come la popolazione totale per codice postale, genere e stato di occupazione;
- Fattori economici come dati di vendita e indici di prezzi;
- Altri dati come informazioni geografiche o sul terreno e posizione del concorrente.

L'accuratezza del modello predittivo può essere valutato sulla base di indicatori come ad esempio:

- Mean Absolute Percentage Error (MAPE):
- Root Mean Square Error (RMSE)
- Scarto Quadratico Medio

Grazie alla LI, i rivenditori possono rimanere all'avanguardia e ottenere il sopravvento sui concorrenti. I dati utilizzati possono includere dati interni sulle vendite, dettagli demografici e profili della popolazione. Altri indicatori macroeconomici come il potere d'acquisto della popolazione, l'indice di crescita economica e i costi connessi alla creazione di nuovi negozi, e così via, possono risultare utili per il raggiungimento dell'obiettivo [17].
Le aziende che al giorno d'oggi sono impegnate nella Location Intelligence sono in costante crescita. Sicuramente degna di menzione è **CARTO**[21]. Essa consente alle organizzazioni di ottimizzare le prestazioni operative e gli investimenti strategici. *CARTO* prevede inizialmente di caricare i propri dati (Shapefiles, GeoJSON, etc) su un database spaziale[22]. Una volta caricati, è possibile visualizzarli su mappa utilizzando il layout e le mappe che il software offre, potendo anche personalizzare o costruirne di proprie.

[21] https://carto.com/

[22] dalla documentazione, si evince come il database utilizzato sia PostGIS: https://cartodb.readthedocs.io/en/latest/install.html# postgis

Come il sito ufficiale riporta[23], tutto il software è *open-source* ed installabile gratuitamente sulle proprie macchine. La componente *Enterprise* è concentrata su un assistenza costante (anche da remoto) e sulla gestione ottimale delle prestazioni che è in grado di mettere in atto. Oltre a questo aspetto, viene tenuta in debita considerazione, la questione legato alla sicurezza, tutte tematiche centrali all'interno di un *core business* aziendale. **Geoblink**[24] è una *startup* spagnola con sede a Madrid impegnata nell'erogare servizi legati al mondo della *LI* incentrati prevalentemente sul mondo delle vendite al dettaglio, *Real-Estate*[25] and *FMCG* (*Fast Moving Consumer Goods*)[26]. Recentemente selezionata da Bloomberg come una delle 50 startup più promettenti al mondo, Geoblink è una soluzione di Location Intelligence basata su SaaS che aiuta i professionisti a prendere decisioni sulle proprie strategie di business. Combina tecniche di analisi avanzate tradizionali e non tradizionali su dati grandi e piccoli, insieme a una ricca interfaccia utente basata su mappe per visualizzare più tipi di statistiche in un modo semplice da usare e facile da capire. Grazie i dati pubblici e privati di cui dispone, analizza e incrocia informazioni con l'ausilio di tecniche di Machine Learning al fine di far ottenere il massimo dalle aziende che ne usufruiscono.

urbanData Analytics[27] è una società digitale specializzata nell'analisi dei dati, dedicata alla conoscenza in tempo reale del mercato immobiliare. In meno di cinque anni, l'*UDA* si è affermata come leader nel settore immobiliare Big Bata in Spagna ed è stata premiata a livello internazionale. Offre prodotti e servizi a società consolidate nel settore immobiliare: consulenti, sviluppatori, fondi di investimento, società, banche, investitori, portali, periti etc. Utilizza uno strumento che, segmenta e localizza il mercato raccogliendo, strutturando, analizzando informazioni da varie fonti: (privato, open data, social network, propri etc). La raccolta e l'elaborazione dei dati è

[23] https://carto.com/help/getting-started/os-vs-commercial-onprem/

[24] https://www.geoblink.com/

[25] Per Real-Estate si intende tutto il movimento legato al settore immobiliare che comprende la compravendita di beni o di terreni (JAMES CHEN 2019).

[26] I FMCG rappresentano prodotti a basso consumo che vengono venduti rapidamente e consumati in un arco temporale che va da pochi giorni ad un anno Fanno parte di questa categoria alimenti come bevande, latticini, farmaci da banco etc.

[27] http://www.urbandataanalytics.com/en/

organizzata in 160 dataset, classificati in aree: Urban People, Urban Economics, Urban Shape, Urban Move. Una volta analizzati, vengono visualizzati i dati georeferenziati attraverso mappe, grafici e infografiche, consentendo una rapida comprensione. In questo modo, sono in grado di rivelare modelli di comportamento, anomalie e variabili identificative con l'ausilio di indicatori (*urbanData Insights*) che consentono una concentrazione dinamica di un grande volume di dati in un unico valore.
Queste appena menzionate sono solo tre esempi di aziende che sfruttano la LI. Ci si sarebbe potuto soffermare su aziende come **Qlik**o **Esri**, ormai multinazionali, ma le aziende di di cui si è parlato, hanno avuto la bravura di affermarsi in poco tempo, riuscendo ad entrare a far parte delle top 100 startup del 2019 [18] perseguendo il loro ideale aziendale. Ciò che hanno in comune, al di là dell'idea su cui stanno investendo è la raccolta del "dato" e la sua divulgazione nei termini più chiari possibili. Tutto questo, è reso possibile dalle tecniche di visualizzazione offerte dalla *Spatial Analysis*. Lo stesso obiettivo, è quanto si è cercato di perseguire nell'implementazione di questo progetto, partendo inizialmente dalla raccolta del dato per poi porre l'accento e trarre vantaggio da esso.
Avendo per cui cercato di dare un'esaustiva panoramica dell'argomento generale, di qui in avanti sarà descritto il progetto nella sua interezza: dalla raccolta alla visualizzazione, passando per l'implementazione e la valutazione dei risultati delle macro aree che mettono in luce tutti gli aspetti peculiari della *Location Intelligence*: analisi descrittiva, analisi potenziale e soprattutto analisi predittiva.

Capitolo 2

Dashboard Retail: Architettura e Progettazione

Il progetto in esame che ci si appresta a descrivere rappresenta il frutto di lungo lavoro realizzato all'interno di **Iconsulting**[1]: azienda di consulenza con sede legale a Casalecchio di Reno. Specializzata in metodologie, algoritmi e tecnologie capaci di dare valore al più grande asset del mercato disponibile al giorno d'oggi: il*dato*. Nata nel 2000, dall'idea di un gruppo di ricercatori, rappresenta una Vendor indipendente con 4 sedi rispettivamente di Bologna, Milano, Roma e Londra. E' un azienda in forte espansione con una propensione alla BI affiancata da branche come il Machine Learning, BlockChain e con una forte propulsione alla Location Intelligence. Ad oggi opera nei più svariati settori che vanno dall' Automotive, al Fashion Retail, passando per Healthcare, Manufacturing e Media. Vanta una collaborazione tra le tante con *IBM*, *Microsoft*, *Oracle*, *CARTO*.

2.1 Obiettivi e Requisiti

La finalità del lavoro svolto è volta alla progettazione e successiva implementazione di un *PoC*[2] che permetta di valutare le potenzialità di una

[1] https://www.iconsulting.biz/

[2] PoC è l'acronimo di Proof of Concept. Riguarda la creazione di un progetto (a volte anche non completo) che mira a verificare il grado di fattibilità in un contesto reale. Un

Dashboard per un cliente impegnato nel settore del *Fashion Retail* con l'ausilio di dati open (numero di abitanti, popolazione maschile e femminile, reddito percepito etc), e dati *custom* del cliente, integrandoli con informazioni sul territorio. Il tutto per una duplice funzionalità: sia per analizzare e valutare l'andamento e le performance sulle base delle informazioni disponibili *(analisi descrittiva)*, cercando anche di porre in rilievo se la resa fosse in linea con le aspettative aziendali *(analisi del potenziale)*; e sia per l'individuazione di potenziali nuovi punti vendita *(analisi predittiva)*.
In una prima istanza, per far fronte alle esigenze del cliente, sono state effettuate varie sessioni di analisi con lo scopo di individuare le migliori tecnologie che meglio si adattassero alle richieste del cliente. Quest'ultimo infatti optava per una soluzione a basso costo che non implicasse l'utilizzo *tool*[3] con licenza a pagamento.
Per questo motivo, nella prima fase di analisi, l'attenzione si è soffermata sull'individuazione di quale Database, in particolare Database Spaziale, potesse essere utile alla causa. Lo stesso, è stato fatto per trovare la "migliore" tecnologia che fosse in linea con i vincoli prestabiliti.
Esistono varie tipologie di database spaziali che il mercato offre. Di seguito, un confronto fra le principali soluzioni che offre il mercato, focalizzando l'attenzione sulle caratteristiche spaziali, geometriche e sull'eventuale integrazione con gli Shapefile.

sinonimo spesso usato è "prototipo"

[3]Con tool in questa circostanza sono intesi sia database ma anche programmi o librerie di vario genere

	Oracle DB Spatial	MongoDB	RethinkDB	Geo Couch
Open Source		X	X	X
BI	X	X	X	X
Point	X	X	X	X
Line	X	X	X	X
Multilinee	X	X	X	X
Poligoni	X	X	X	X
Collezioni	X	X		X
MBR	X			
SRID	X	(WGS84)	(WGS84) (GCS)	(WGS84)
Intersect	sdo_intersection	$geoIntersects	get_intersecting, get_intersecting	$geoIntersects
Union	sdo_union			
Difference	sdo_difference		polygon_sub	
XOR	sdo_xor			
Centroid	sdo_centroid	$center		$center
Distance	sdo_within_distance	$geoWithin,$box	distance	$geoWithin,$box
Distance from Object	sdo_nn	$near, $nearSphere, $maxDistance, $minDistance	get_nearest	$near, $nearSphere $maxDistance, $minDistance
relation	inside, contains, coveredby, on_cover, touch, overlap, equal, anyinteract join, disjoin		include	
TURF		X	X	X
Shapefile	X	X		

Tabella 2.1: Benchmark tra Database Spaziali (Parte 1)

	H2Gis	Redis	Neo4j	PostGIS
Open Source	X	X	X	X
BI	X	X	X	X
Point	X		X	X
Line	X		X	X
Multilinee	X		X	X
Poligoni	X		X	X
Collezioni	X			X
MBR	X			
SRID	ALL			ALL
Intersect	ST_Intersection	sinter	spatial.intersects	ST_Intersect
Union	ST_Union		unionAll	ST_Union
Difference	ST_Difference, ST_SymDifference			ST_Difference, ST_SymDiffference
XOR				
Centroid			toCentroid	ST_Centroid
Distance	ST_Dwithin, ST_MaxDistance	geodist	spatial.withinDistance	ST_Distance
Distance from Object	ST_ClosestCoordinate, ST_ClosestPoint, ST_FurthestCoordinate		spatial.closest	ST_Dwithin
relation	ST_Contains, ST_Covers, ST_Crosses, ST_Disjoint, ST_Equals, ST_Intersects, ST_Overlaps, ST_Touches, ST_Within	geoadd, geodist, geohash, geopos, georadius, georadiusbymember	Contain, Cover, Covered, Cross, Disjoint, Intersect, Intersect, Overlap, Touch, Within, Within Distance,	ST_Within, ST_Containt, ST_Disjon, ST_Covers, ST_Equals, ST_Touches
TURF	X			
Shapefile	X		X	X

Tabella 2.2: Benchmark tra Database Spaziali (Parte 2)

Le precedenti due tabelle raccolgono ed esplicitano le caratteristiche salienti dei database spaziali. Anche se è stato premesso che la richiesta del committente fosse rivolta alla scelta di una tecnologia *open*, è stato inclusa in questa cernita anche *Oracle DB Spatial* perché dalla letteratura, essendo considerato uno dei più completi, può essere usato come metro di paragone con gli altri.
Vediamo ora in cosa differiscono i vari database:

- **Oracle DB Spatial:**[4] è parte del DBMS Oracle con licenza separata. Viene usato per gestire dati geografici e di localizzazione in manie-

[4]https://www.oracle.com/database/

ra nativa all'interno del database Oracle, potenzialmente supportando un'ampia gamma di applicazioni come i Sistemi Informativi Geografici (GIS). Come già accennato è il più completo tra i database spaziali. Supporta tutte le *features* spaziali tipici come, Linea, Poligono, Punto oltre che tutti i formati di proiezione del dato. Presenta una vasta implementazioni delle funzioni geometriche disponibili. Inoltre permette l'interazione con ShapeFile.

- **MongoDB:**[5] è un DBMS non relazionale, orientato ai documenti. Classificato come un database di tipo NoSQL, MongoDB si allontana dalla struttura tradizionale basata su tabelle dei database relazionali in favore di documenti in stile JSON con schema dinamico. Supporta tutte le caratteristiche spaziali principali fatta eccezione per gli MBR. Tra gli SRID disponibili comprende solo i (WGS84) e pecca per il numero di geometrie offerte.

- **RethinkDB:**[6] è un database *open source* e gratuito creato dalla compagnia che porta il suo stesso nome. Il database memorizza i documenti JSON con schemi dinamici. Non permette l'integrazione degli *Shapefile* e anche in questo caso offre poche funzioni geometriche.

- **Geo Couch:** [7] è un database NoSQL che utilizza JSON per memorizzare i dati, JavaScript come linguaggio di interrogazione e che utilizza MapReduce e HTTP come API. E' l'estensione di **CouchDB** ed è stato distribuito per la prima volta nel 2005 ed è successivamente diventato un progetto Apache nel 2008. Offre soprattutto funzioni legate alla vicinanza e la distanza da essa. Anche in questo caso non è supportata l'integrazione con gli shapefile.

- **H2Gis:**[8] o più semplicemente **H2** è un gestore di basi di dati relazionale scritto in linguaggio Java con tecnologia In memory. Può essere incapsulato in applicazioni java o eseguito in modalità client-server. **G2Gis** è un estensione spaziale del database H2. Supporta tutte le

[5]https://www.mongodb.com/
[6]https://rethinkdb.com/
[7]https://gdal.org/drivers/vector/couchdb.html
[8]http://www.h2gis.org/

principali features presenti anche in Oracle con annessa integrazione agli shapefile. Può essere considerato una valida opzione.

- **Redis:**[9] è un è un archivio di strutture di dati in memoria open source (con licenza BSD), utilizzato come database, cache e broker di messaggi. Supporta strutture di dati come stringhe, hash, elenchi, set, set ordinati con query di intervallo, bitmap, hyperloglog, indici geospaziali con query di raggio e flussi. Non supporta l'integrazione con gli Shapefile.

- **Neo4j Spatial** è una libreria di **Neo4j**[10] che facilita l'abilitazione di operazioni spaziali sui dati. In particolare, permette l'aggiunta di indici spaziali ai dati già localizzati e l'esecuzione di operazioni spaziali su di essi come la ricerca di punti in aree di interesse o entro una certa distanza da un punto di interesse. Inoltre vengono fornite classi per esporre i dati a tool come **GeoTools** [11] e **GeoServer**[12].

- **PostGIS** rappresenta l'estensione spaziale per il database **PostgreSQL** distribuito con licenza GPL. Fornisce i tipi di dati specificati negli standard dell'*Open Geospatial Consortium* [13].

Avendo avuto modo di testare e prendere visione di come questi database si comportassero, si è scelto di optare per **PostGIS** per due motivazioni principali. La prima perché open, ma a dispetto degli altri, la sua diffusione è stato un fattore incentivante anche in previsione di problemi. Cosa non

[9] https://redis.io/

[10] Neo4j è un database basato sui grafi. E' open-source ed è sviluppato in Java. Può essere utilizzato sia in stand-alone sia integrabile con altre applicazioni. La sua peculiarità riguarda l'immagazzinamento dei dati in cartelle. Per maggiori informazioni si può consultare https://neo4j.com/

[11] GeoTools è una libreria scritta in Java. Permette la lettura, la manipolazioni e l'analisi dei dati. E' distribuita con licenza LGPL. Per info consultare https://geotools.org/

[12] GeoServer è un server open source per la condivisione di dati geospaziali scritto in Java. E' interoperabile e pubblica i dati da tutte le principali fonti di dati spaziali (OpenLayers, Google Maps, Bing Maps o anche Google Earth, NASA World Wind) utilizzando standard aperti. Per maggiori dettagli http://geoserver.org/

[13] L'Open Geospatial Consortium (OGC) è un consorzio che comprende più di 530 imprese, agenzie governative e organizzazioni di ricerca. Ha lo scopo di rendere liberi i dati e i servizi geospaziali.

da poco, supporta tutte le principali funzioni presenti anche in Oracle, che ne ha sancito la scelta.
Quanto fatto lato Server, è stato anche realizzato per il front-end. La selezione non è stata rivolta al tipo di linguaggio di programmazione per l'implementazione ma al tipo di mappa da utilizzare e all'eventuale integrazione con **Turf.js**[14], oltre che con altri software potenzialmente utilizzabili (**Power BI**, **Tableau**).

	Leaflet	**MapBox**	**MapNik**	**OpenLayer**
Open Source	YES	NO	YES	YES
Spatial Features				
Point	YES	YES	YES	YES
Line	YES	YES	YES	YES
Multiline	YES	YES	YES	YES
Poligon	YES	YES	YES	YES
GeoJson	YES	YES	YES	YES
TURF	YES	YES	NO	YES
Include Shapefile	YES	YES	YES	YES
Maps Features				
Heatmap	YES	YES	?	YES
Cluster	YES	YES	?	YES
Markers	YES	YES	YES	YES

Tabella 2.3: Benchmark tra Mappe

Come è possibile osservare dalla tabella, tutte sembrano offrire le medesime soluzioni, ma la scelta si è incentrata tra **Leaflet.js** e **OpenLayers**, principalmente perché **MapBox** (fornitore di mappe online americano), offre una soluzione parzialmente open-source[15], mentre MapNik (toolkit di mapping basato su utilizzo Desktop e Server) pecca principalmente, ad og-

[14] libreria per l'analisi geospaziale su client

[15] Esiste sia una versione open, sia una versione Enterprise. Tuttavia quella open è limitata al numero di utilizzi offerti. Infatti dopo un certo numero utilizzi, diventa a pagamento.

gi, per la mancata integrazione con **Turf.js**, oltre al fatto che sembra più inerente all'utilizzo desktop.
Dovendo scegliere tra le due soluzioni rimanenti (entrambe sono libreria Javascript che permettono l'integrazione in pagine Web) , si è scelto **Leaflet** perché più semplice e potenzialmente integrabile in altri contesti aziendali.
In sintesi, le prime fasi di analisi, hanno indirizzato le scelte progettuali su **PostGIS** come database spaziale, mentre **Leaflet.js** è stato decretato il tool scelto per interagire con le mappe. L'ultimo *open point* ancora da scegliere è il tool/framework con cui implementare il progetto.
Le scelte in questo caso ruotavano su tre possibili opzioni:

- **Power BI**[16] come il sito cita, "*è un servizio di analisi business che offre informazioni dettagliate per favorire decisioni rapide e informate*" . E' una soluzione Enterprise concentrata prevalentemente per l'esplorazione visiva dei dati ma che supporta anche l'utilizzo di mappe.

- **Tableau**[17] è un software, anch'esso Enterprise, con una vocazione prettamente sulla BI. La sua peculiarità è la facilità di utilizzo e di interazioni con varie fonti dati, oltre che la presenza di mappe con un design elegante[18].

- **Soluzione Custom** consente di avere libera scelta sul linguaggio da utilizzare e consente l'utilizzo di svariati plugin. Tre le soluzioni disponibili attualmente maggiormente in voga: *Angular 7*, *Vue.js* o *React*. Ovviamente nulla vietava di poter scrivere una soluzione basata su HTML e Javascript, ma la scelta di uno di questi tre framework facilita l'integrazione con tutti i componenti aggiuntivi necessari e in ottica futura, consente un facile *deployment*[19]del software.

Seguendo quello che è stato il diktat adoperato nella scelta delle componenti precedenti, anche qui, si è deciso di optare per una soluzione Custom[20].

[16]https://powerbi.microsoft.com/it-it/
[17]https://www.tableau.com/it-it
[18]Le mappe di Tableau sono realizzate con MapBox
[19]In Informatica si intende la consegna e rilascio al cliente con annessa installazione del software
[20]Nulla vieta che negli sviluppi futuri il software potesse essere spostato su altre soluzioni come Power BI e Tableau per l'appunto.

Come si vedrà nel prossimo capitolo, alla fine la realizzazione *front-end* è stata implementata con Angular 7 in quanto già ampiamente usato in altri progetti aziendali.

2.2 Client side: Use Case & User Experience

Dal momento in cui, sono stati definiti tutti i requisiti che la Dashboard deve possedere, ci si può soffermare sulle scelte progettuali, lato *User Experience* e sulla definizione dell'architettura necessaria allo sviluppo del software.
Ovviamente essendo una piattaforma Custom, integrabile o nei sistemi di BO aziendali o alternativamente, sul Web deve essere fruibile dall'utente e deve rispettare i canoni base della user experience facendo in modo che possa offrire un'esperienza di navigazione soddisfacente. Per tale motivo sono stati realizzati dei mockup, che tentino di dare una visione globale di quel che sarà poi il risultato finale. Il software di cui ci si è servito per tale scopo è Balsamiq[21].

2.2.1 Features Descrittive

Lo schema logico che si è seguito, ha permesso di tracciare la suddivisione del lavoro in 3 macro aree principali; le stesse aree elencate e spiegate a livello generico nel primo capitolo. Esse riguardano *l'analisi descrittiva*, *analisi del potenziale* e *analisi predittiva.*
L'*analisi descrittiva*, avrà a livello di layout[22] una mappa che mostrerà tutte i punti vendita presenti. Lo scopo, è quello di mostrare l'andamento generale delle *door*[23] e il tutto sarà connotato dall'utilizzo di grafici come i grafici a barre per aspetti specifici, un grafico a torta che mostra le quantità di

[21] Balsamiq è un tool specializzato nella realizzazione di Mockup. E' disponibile nella versione WIndows e Mac e sebbene sia a pagamente, da la possibilità agli utenti di creare dei progetti Free su Cloud, disponibili per 30 giorni. https://balsamiq.com/

[22] E' importante ricordare che la realizzazione di Mockup è una fase transitoria e non definitiva. Il lavoro ultimato può differire rispetto quanto emerso in questa analisi.

[23] Con "doors" si intendono tutti i negozi fisici attualmente esistenti. L'attribuzione di questa associazione è frutto della scelta di politiche interne.

vendita e una trend line che sancisce l'andamento temporale. Cliccando su una specifico punto mostrato sulla mappa, saranno poste in dettaglio le informazioni di quel singolo punto, mantenendo sempre lo stesso contesto di informazione.

Figura 2.1: Esempio schermata Analisi Descrittiva

Nel dettaglio, l'*analisi descrittiva* sarà suddivisa in varie tab per soddisfare i seguenti *use case*:

- **Doors**: geocalizza su mappa tutte le *door* presenti sul territorio mondiale e da un excursus generale sulla situazione globale e singola di ogni punto vendita. In particolare viene posta l'attenzione sull'individuazione di quelli che sono i principali clienti (**Customers**), le migliori "**Door**" , il migliori prodotti elencati per codice (**SKU**), e **Brands**" . Inoltre mostra l'andamento globale dei 5 migliori punti vendita e qual è il loro venduto a livello quantitativo.

- **Sell out**: come il precedente punto, analizza la situazione delle *doors*, ma questa volta non a livello quantitativo, ma sulla base delle vendite al netto. In questa sezione inoltre è possibile vedere quante sono le vendite rispettivamente per uomo, donna, bambino e unisex, visualizzando contemporaneamente i migliori 5 prodotti per volume di vendita;

- **Country e Regions**[24]: in queste tab, l'attenzione si concentra sul rendimento globale dei rispettivi paesi che ospitano le varie *doors* . Soprattutto in questo contesto, assume una notevole rilevanza i grafici relativi ai trend. Anche la seguente area contiene grafici inerenti lo stato dei Customers, dell SKU e del Brand;

2.2.2 Features Potenziali

L'*analisi del potenziale* sarà approssimativamente simile, con la presenza della mappa centrale, ma conterrà una o più tabelle a seconda dell'area selezionata. Nell'immagine mostrata da mockup, comparirà anche un riquadro contenente il paese scelto. Anche in questo caso, andando a dettagliare i casi d'uso, presenterà:

- **Potential Analysis**: questa prima tab, mostra nazione per nazione, l'andamento delle vendite tenendo in considerazione 4 macro categorie: *genere*, *brand*, *colore* e *forma* del prodotto, mostrando se le vendite sono in linea con le aspettative;

- **Potential Analysis Italy**: analogamente a quanto fatto a livello globale, in questo frangente si valuetà l'andamento a livello nazionale. In aggiunta a quanto visto, sono state anche prese in esame alcuni informazioni scaricate dal sito *Istat* , come si vedrà meglio più avanti, che nello studio effettuato hanno inciso sull'apporto dato alle vendite.

[24]Essendo un PoC, essa comprende solo alcune regioni dei principali paesi europei: Austria, Belgio, Francia, Germania, Gran Bretagna, Irlanda, Italia, Portogallo, Russia, Spagna e Svizzera. Per ognuno dei paese elencati, saranno mostrate solo le regioni che ospitano i punti vendita.

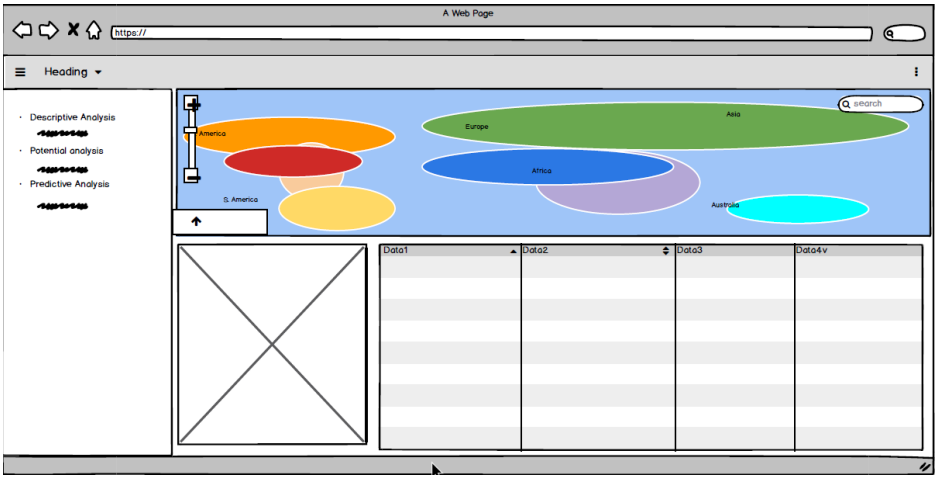


Figura 2.2: Esempio schermata Analisi Potenziale

2.2.3 Features Predittive

L'ultima macro area invece riguarda l'*analisi predittiva*. In questo frangente è la mappa e il suo contenuto ad avere un ruolo di primo piano che si distacca da tutti gli elementi di contorno. Qui saranno mostrate le potenziali nuove aree o regioni in cui collocare le door. Nel concreto, si prefiggerà di individuare, basandosi sulle caratteristiche delle varie *doors*, in combinazione con i *POI*[25] e dei dati Istat, potenziali nuove aree dove è consigliabile posizionare o non posizionare un nuovo punto vendita. Ciò si è evinto incrociando la *domanda* con *l'offerta*. Nello specifico, si vuole intendere che il lavoro finale di predizione di possibili nuove *doors* tende a scandagliare aree potenzialmente produttive dal punto di vista economico che ad oggi non sono ancora coperte fisicamente dalla presenza di un punto vendita, ma escludendo quelle zone che già ne presentano uno o più di uno, mediante uno slider con raggio definito dall'utente.

[25] Point of Interest.

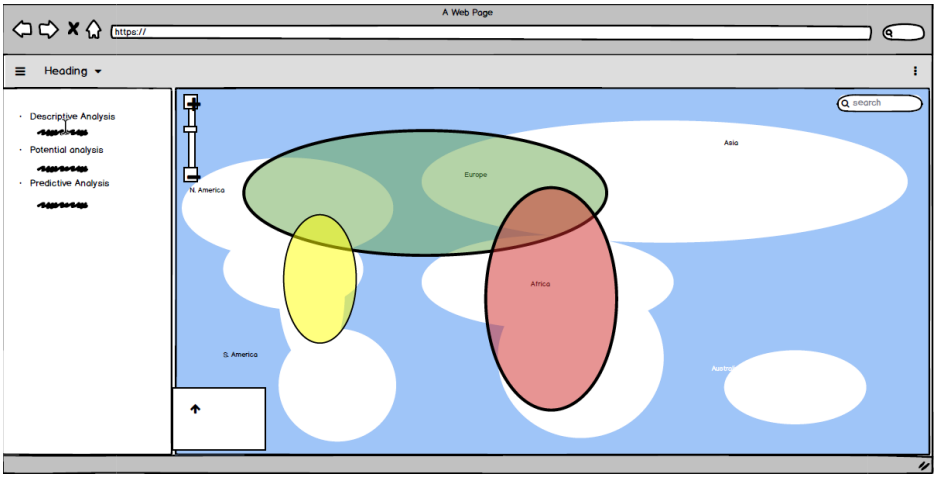


Figura 2.3: Esempio schermata Analisi Predittiva

Quanto mostrato è una succinta semplificazione dell'aspetto che la dashboard assumerà. La tabella seguente riassume la differenza fra le varie aree.

Analisi Descrittiva	Doors
	Sell out
	Country
	Regions
Analisi Potenziale	Potential Analysis
	Potential Analysis Italy
Analisi Predittiva	Predictive Analysis Italy

Tabella 2.4: Stack logico del progetto

2.3 Server side: Database Spaziale

Fino a questo momento, sono stati tracciati i vincoli generali; definito il rendering e spiegati i casi d'uso di ogni singola macro area. Per poter passare a vedere l'aspetto implementativo, occorre prima approfondire un altro aspetto legato alla gestione del dato spaziale.
Nei precedenti paragrafi di questo capitolo, sono stati messi a confronto i

principali Database spaziali e dopo una fase di approfondimento, si è optato per l'utilizzo di **PostGIS**.
La prima operazione effettuata è stata la sua installazione e successiva configurazione[26][27], per poi avviare il processo di popolamento.
I dati di partenza su cui si è lavorato, mappano i singoli punti vendita collocati nel mondo che sono stati estratti dal *DWH* aziendale. Il processo di estrazione, ha portato all'ottenimento di un file Excel. Tra le caratteristiche salienti che esso presenta, si ha:

- **ID**: rappresenta l'indicizzazione del dato nel *DWH* aziendale;
- **Door Code**: tecnicamente sta a rappresentare il codice della *door*;
- **Sales Org. - Distr. Chan.**: sta ad indicare il canale di distribuzione e l'organizzazione di vendita. In particolare si riferisce sia alla filiale nel mondo e sia alla categoria a cui è stato venduto il prodotto: se per esempio verso il mercato asiatico, per il mercato europeo etc.;
- **Customer Door Code:** indica il codice univoco dei vari clienti a cui è stato venduto il prodotto che a loro volta distribuiranno nelle doors;
- **Customer Warehouse Code:** indica il codice di riferimento del cliente nel DWH;
- **Door Name**: indica il nome del singolo punto vendita;
- **Address:** l'indirizzo del punto vendita;
- **City:** la città in cui è situato il punto vendita;
- **Postal Code:** il codice postale della città;
- **Country:** lo Stato in cui è collocato;
- **Region:** l'area geografica a cui fa capo;

[26] sono stati definiti *username*, *password*, *porta*, e *nome del database*
[27] per maggiori informazioni, si rimanda il lettore a prendere visione dell'*Appendice A*

2.3.1 Preparazione dei dati

Salta subito all'occhio la mancanza delle coordinate geografiche, fondamentali per poter effettuare un inserimento all'interno del DB. Per questa ragione, è stato necessario geolocalizzare ogni singolo punto vendita e mapparlo. Il seguente script Python assolve a questo scopo:

```
filename = "DWH_doors"
convertXlsxToCsv("./doc/" + filename +".xlsx", "./doc/csv/" + filename +".csv")
geocoding("./doc/csv/" + filename +".csv")
deleteColumnFromCSV("./doc/csv/" + filename +"_RESULTS.csv",'TempCoordinate')
convertCsvToXlsx("./doc/csv/" + filename + "_RESULTS.csv","./doc/csv/" + filename + "_RESULTS.xlsx" )
```

Dal blocco di codice, si vede come inizialmente il file viene convertito nel formato CSV con Python per una migliore gestibilità per mezzo di librerie come *Pandas*[28]; poi calcolate la coordinate, pulito il file risultante e infine riconvertito in un file con estensione Excel.
Tralasciando la conversione dei file nei vari formati, concentriamo la nostra attenzione sulla funzione che geocalizza le varie doors.

```
import xlrd

# GEOCODING IMPORTS
...

# PANDAS
from pandas.io.excel import ExcelWriter
import pandas as pd
...

def geocoding(path):

    input_csv = pd.read_csv(path)

    # You can specify the type of provider to perform geocoding
    geolocator = ArcGIS() #Nominatim(user_agent="GEOLOCALIZZAZIONE")
    geocode = RateLimiter(geolocator.geocode, min_delay_seconds=1)

    # Creation Data Frame
    df = pd.DataFrame(input_csv)

    # Take the address, or Door name if missing
    new_clean_dataframe = []
    for i in df.index :
        if pd.isnull(df['Address'][i]) :
            new_clean_dataframe.append( df['Door Name'][i])
        else :
            new_clean_dataframe.append( df['Address'][i])

    # Conversion to Data Frame
    df['TempCoordinate'] = new_clean_dataframe
```

[28] Libreria Open Source per l'analisi dei dati. Verrà usata più avanti per la fase di predizione delle nuove doors

```
32
33      df['Coordinate'] = df['TempCoordinate'].progress_apply(geocode)
34      df['Latitude'] = df['Coordinate'].progress_apply(lambda location: (location.latitude) if
35                          location else None)
36      df['Longitude'] = df['Coordinate'].progress_apply(lambda location: (location.longitude) if
37                           location else None)
38
39
40      # Save
41      input_csv.to_csv(path[:-4]+"_RESULTS.csv",index=False,encoding='utf-8')
```

La funzione, inizialmente legge il file che è stato convertito in csv e lo assegna ad un dataframe[29]. Subito dopo, instanzia le variabili necessarie per la geolocalizzazione[30]. In questo caso, la quasi totalità del lavoro, viene svolta dalla libreria *geopy*[31]. Vengono ciclate tutte le righe ed assegnate ad un array, verificando che ognuna di essa contenga l'indirizzo (su cui è effettuare la geolocalizzazione) o in alternativa il nome del punto vendita. Dopodichè, è lanciata la funzione di geolocalizzazione *geocode*[32].
Alla fine di questo processo di mappatura, si creerà un nuovo file.csv da pulire per evitare l'inserimento di colonne superflue e infine convertito nuovamente in un Excel.
Il file risultante sarà identico a quello originale, ma arricchito di altre due nuove colonne: la colonna della *latitudine* (coordinata x) e la colonna della *longitudine* (coordinata y). Bisogna specificare che questa convenzione adottata non è sempre riconosciuta. Sono molti i software che invertono la latitudine con la longitudine. Un esempio è **Turf.js** come si vedrà più avanti.

2.3.2 Popolamento del Database

In questo momento è stato messo il primo mattoncino necessario a creare l'architettura lato Server. Si posseggono tutte le componenti necessarie per

[29] L'utilizzo del Dataframe è reso possibile dall'installazione della libreria Pandas

[30] Nell'assegnamento del Geocode, è stato calcolato un tempo di latenza che evita i recuperi delle coordinate più lente e quindi errori nello script.

[31] Libreria Python che consente di "individuare le coordinate di indirizzi, città, paesi e punti di riferimento in tutto il mondo utilizzando geo coders di terze parti e altre fonti di dati" https://geopy.readthedocs.io/en/stable/

[32] La funzione progress_apply è superflua. Serve per monitorare la situazione con l'ausilio di una progress bar da Shell.

poter inserire le righe nel Database. Quest'ultima operazione può avvenire in due modi:

- attraverso uno script Python;
- attraverso le opzioni di PostGIS[33]

Qui di seguito è mostrato come creare una tabella e successivamente come inserire le righe che la compongono via script. Perché non usare la funzionalità offerta da PostGIS? il motivo di questa scelta è dovuto al fatto che si sta popolando il DB con dati spaziali. Certo PostGIS supporta ovviamente questa *features* di inserimento ma avendo parlato nel Capitolo 1 del *dato* e dell'*indice spaziale* permette di capire come esso tratti i dati spaziali in previsioni di query dello stesso tipo.

```
1  def createTableDoors(self,table,schema = 'public'):
2          try:
3              schema = schema
4              tableName = table
5              self.cursor.execute("SELECT to_regclass('{}.{}');".format(schema, tableName))
6              fetch = self.cursor.fetchall()
7              tableList = [fetch[i][0] for i in range(len(fetch))]
8              if table not in tableList:
9                      self.cursor.execute("CREATE TABLE {} (ID SERIAL PRIMARY KEY,"
10                                                 "Door_Name VARCHAR, "
11                                                 "Address VARCHAR, "
12                                                 "City VARCHAR,"
13                                                 "Country VARCHAR,"
14                                                                 ...
15                                                                 ...
16                                                 "Coordinate VARCHAR,"
17                                                 "Latitude FLOAT,"
18                                                 "Longitude FLOAT, "
19                                                 "Geom GEOGRAPHY(POINT));"
20                                         .format(tableName))
21                     index_name = tableName+"_index"
22                     self.cursor.execute("CREATE INDEX {} ON {} USING GIST(Geom)"
23                     .format(index_name,tableName))
24                     print("Create Spatial Index")
25
26             else: print("Table already exist")
27         except psycopg2.DatabaseError as e:
28             logger.warning("error creating table: %s", e)
```

La funzione `createTableDoors` è parte di una classe *Database*[34]. In quanto tale ha con se 3 parametri. Il primo è l'istanza della classe grazie al quale è possibile accedere agli attributi e i metodi presenti all'interno. Tale funzione ne è un esempio. Il secondo parametro invece definisce il nome

[33] Si guardi l'appendice A.3.3

[34] Si veda l'appendice A sezione A.3.2

della tabella, mentre il terzo lo *Schema*[35] che di default è *Public*.
Nella riga 5 viene eseguita una query che si accerta che la tabella che si sta creando non esiste[36]. Nel caso in cui la tabella non esista, ne viene creata una nuova. E' da notare come le colonne, sono le stesse del file ottenuto dopo la geocalizzazione, denotate da un *nome* e da un *data type*[37], ma comprende anche una nuova colonna di tipo *Geometry*. Qui si sta indicando al Database, che la sua tabella conterrà dei dati spaziali. Nelle ultime righe, come da prassi in questi casi, è creato l'indice spaziale per permettere di effettuare operazioni.
Ora che è stata creata la tabella, si possono inserire le righe:

```
1  def addDoor(self,path, table, schema = 'public'):
2
3          df = None
4          # Check if file Exist and assign to a Dataframe
5          try:
6              schema = schema
7              tableName = table
8              self.cursor.execute("SELECT to_regclass('{}.{}');".format(schema, tableName))
9              ...
10             else:
11                 length = df.shape[0]
12                 nullValue = None
13                 for index, row in tqdm(df.iterrows(), total=length):
14                     ID = row['ID']
15                                 ...
16                     Door_Name = nullValue if pd.isnull(row['Door Name']) else row['Door Name']
17                     Latitude = nullValue if pd.isnull(row['Latitude']) else row['Latitude']
18                     Longitude = nullValue if pd.isnull(row['Longitude']) else row['Longitude']
19                     point = nullValue if (Latitude is None or Longitude is None) else 'POINT(%s %s)'
20                             % (Longitude, Latitude) # LON/LAT:
                                   https://postgis.net/2013/08/18/tip_lon_lat/
21
22
23                     query = sql.SQL("INSERT INTO {} VALUES
24                     (%s,..,%s,"
25                      "ST_PointFromText(%s,4326));").format(sql.Identifier(tableName))
26                     self.cursor.execute(query, [ID, ... , Door_Name, Address, City, Postal_Code,
                               Country, Region,
27                                                 Coordinate, Latitude, Longitude, point])
28                     self.conn.commit()
29
30         except psycopg2.DatabaseError as e:
31             logger.warning("error creating table: %s", e)
```

[35]Gli schemi assolvono ad un duplice scopo. Da una parte consentono di organizzare il Database in varie sezioni logiche e organizzative. Dall'altro consente un accesso simultaneo da parte di più utenti. (Postgres Tutorial 2019)

[36]La sintassi utilizzata da Python permette di settare i parametri cercati solo alla fine della query mantenendo il codice pulito

[37]Qui di seguito è possibile trovare i datatype di Postgres

La funzione ha dei parametri come la precedente. L'unica differenza da questo punto di vista è il *path* relativo al file Excel che conterrà i punti vendita. Nella parte centrale è verificata l'esistenza del Database e in caso affermativo, vengono ciclate le righe del file passato. In questo frangente sono gestiti anche i valori nulli da inserire in caso di campo mancante per evitare errori da parte dello script. Per ogni riga viene effettuata una query SQL. E' qui che il database è popolato. In particolar modo per ognuna di esse è creato un punto che fa riferimento alla colonna *geom* precedentemente descritta a cui sono passate due coordinate rispettivamente la *latitudine* e la *longitudine*. Come già preannunciato è possibile vedere che PostGIS prevede l'aggiunta prima della *longitudine* e poi della *latitudine* e non come è conforme fare nella pratica comune.
Per i successivi inserimenti, si è scelto di usare la tecnica descritta nell'Appendice A sezione 3.6. In particolare modo per i dati di vendita estratti dal *DWH* aziendale così come fatto per le *doors*.

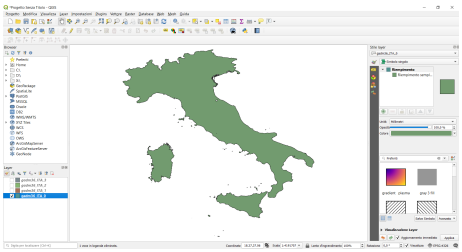
2.3.3 Creazione degli Shape File

La parte finale di questo capitolo sarà focalizzata sull inserimento di tabelle contenenti *Shapefile* utilizzando il software free **QGIS**. Sono stati già introdotti i software GIS all'interno del paragrafo riguardante la descrizione degli Shapefile nel primo capitolo. Cercando in rete la parola *"GIS Software"* compariranno innumerevoli risultati relativi al numero elevato di software disponibili sul mercato.
QGIS è uno dei più famosi GIS, multipiattaforma, disponibili in varie lingue e Open, rappresenta una delle migliori alternative disponibili sul mercato. L'utilizzo di questo strumento in questo progetto è stato volto alla creazione delle seguenti tabelle:

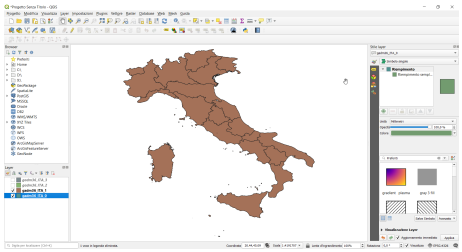
- **shape_world:** tabella che ha al suo intenrno tutti gli stati del mondo;
- **shape_regions:** tabella avente tutte le regioni dei principali stati europei: Austria, Belgio, Francia, Germania, Gran Bretagna, Irlanda, Italia, Portogallo, Russia, Spagna e Svizzera;
- **shape_italian_regions:** tabella con tutte le 20 regioni italiane;

- **shape_town:** tabella con tutti i comuni italiani;

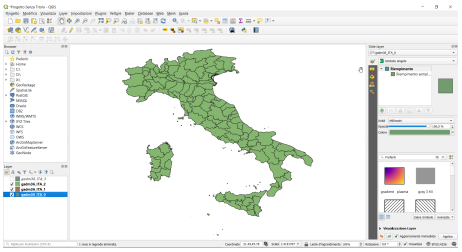
La creazione di queste tabelle è vincolato all'utilizzo ai fini progettuali. Solo l'ultima è ad uso esclusivo dell'analisi predittiva mentre le altre sono usate per le altre due fasi di analisi.
I file utilizzati ed importati in QGIS sono stati scaricati dal sito GADM[38] nella sezione Download. Esso presenta nel suo Set di dati lo Shapefile che racchiude i confini degli stati mondiali in toto, ma allo stesso tempo anche quello dei singoli.
Aprendo QGIS basterà trasportare il file con estensione .shp per poterlo visualizzare. I file tra di loro, sono sovrapponibili e consentono di aggiungere o togliere i layer di interesse.



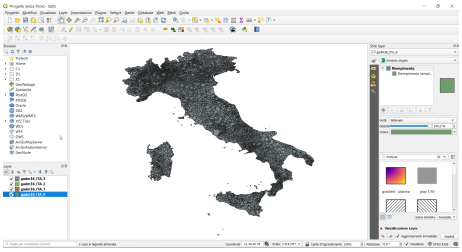
(a) ShapeFile Stato



(b) ShapeFile regioni



(c) ShapeFile provincie



(d) ShapeFile comuni

Figura 2.4: Esempi di ShapeFile in QGIS

I file in questione sono pronti per essere caricati. Prima però bisogna connettersi al database:

- Nella parte sinistra di QGIS, in prossimità a PostGIS, selezionare con il tasto destro '*Nuova Connessione*';

[38] https://gadm.org

- Immettere un nome e le credenziali definite al momento della creazione del DB;
- Nella 'Barra dei Menu', selezionare la voce 'DB Manager';
- Selezionare '*Import Layer* accertandosi di aver selezionato il DB corretto;
- Dopo aver individuato il file nell'area di lavoro, conferire il nome alla tabella. Lo schema di default è quello *public*;
- settare le proprietà facendo ben attenzione a checkare sia l'indice spaziale e sia la creazione della colonna *geom* di tipo Geometry;
- Importare la tabella

A questo punto, se tutto è andato per il verso giusto, la nuova tabella dovrebbe essere visibile nello schema del db.
Quanto descritto ultima la fase di "*preparazione dei dati*" . Adesso si dispone di tutti gli elementi necessari per poter creare il progetto e conferirgli una struttura.

Capitolo 3

Implementazione

In questo capitolo, sarà descritta l'implementazione del progetto. Sarà dato spazio per cui, al codice necessario per strutturare la Dashboard finale, seguendo la struttura dettata dal *mockup* visto nel capitolo precedente. Tale descrizione non riguarderà solo la parte *front end*, ma anche quella *backend*,cercando di dare una visione esaustiva delle chiamate effettuate e delle query adoperate per presentare i dati su mappa.

3.1 Tecnologie Usate

Le tecnologie adoperate sono state scelte al fine di cercare di ottenere un'esperienza utente e di fruizione che fossero soddisfacenti e in linea con i vincoli definiti nei capitoli precedenti. Le principali esse sono:

- **Angular 7** (o più genericamente **Angular**) è una framework e allo stesso tempo una piattaforma per scrivere applicazioni *web* e *mobile*. I linguaggi che utilizza sono HTML e Typescript. Quest'ultimo è lo stesso con cui è stato implementato. E' la naturale Evoluzione di *AngularJS*[1]. L'evoluzione del linguaggio, ha portato a mantenere i costrutti di base ma cambiando le modalità di implementazione (come si vedrà più avanti), portandole ad essere più semplici. Ogni app Angular ha un modulo *root*, chiamato convenzionalmente *AppModule*, che

[1]Data la vasta diffusione, è ancora scaricabile ed utilizzabile in alternativa ad Angular 7

insieme al file *index.html* fornisce il boot per l'avvio dell'applicazione. Un'app in genere contiene molti moduli funzionali

  - **Angular CLI** è un interfaccia a riga di comanda ideata per installare e generare le componenti necessarie: *componenti, moduli,*ambienti di test etc, oltre che per lanciare il progetto ad ogni avvio. E' installabile da *npm.*
  - **Angular Material**: è un'implementazione che segue le linee di progettazione definite in *Google Material Design*[2]. Fornisce un set di componenti di interfaccia integrabili nei progetti *Angular.* La sua installazione è legata all'utilizzo di *Angular CLI.*
  - **ngx-charts**[3] framework pensato per Angular dalla versione 2 in avanti per e la creazione di grafici interattivi. Sfrutta la libreria D3[4]

- **Node.js**[5]: piattaforma Open Source per l'implementazione di operazioni server-side in Javascript.

- **npm**: gestore di pacchetti utilizzato di default da *Node.js.* usato per condividere pacchetti tra le organizzazioni e per gestire lo sviluppo privato. Tra i vari pacchetti utilizzati, i più importanti, in virtù dell'importanza che hanno assunto ai fini del progetto, sono:

  - **Concurrently**: per lanciare multipli comandi in modo *concorrente.* Ai fini progettuali, il suo utilizzo è connotato al riavvio del *client* e del *server* ad ogni modifica rilevata.
  - **Express.js**[6]: è un framework scritto in Node.js che consente la creazione di un server veloce e flessibile.
  - **Nodemon**[7]: è un utility ideata per monitorare ogni cambiamento nel codice lato server e automaticamente riavviarlo ad ogni cambiamento. E' usato in combinazione con *concurrently.*

[2]https://material.io/archive/guidelines/#
[3]https://swimlane.github.io/ngx-charts/# /ngx-charts/bar-vertical
[4]Libreria Javascript per manipolare i documenti partendo dai dati. https://d3js.org/
[5]https://nodejs.org/en/
[6]https://www.npmjs.com/package/express
[7]https://nodemon.io/

  - **node-postgres (pg)**[8] è un client PostGres interamente scritto in Javascript. Grazie ad esso, è possibile eseguire query sul Database.

- **R**[9]: è un linguaggio e un ambiente (con R Studio) per l'elaborazione statistica. Offre un'ampia varietà di tecniche statistiche (modellistica lineare e non lineare, test statistici classici, analisi di serie temporali, classificazione, raggruppamento, etc.) e tecniche grafiche ed è altamente estensibile.

- **Leaflet.js** [10]: è la libreria Javascript Leader per mappe interattive e ottimizzate sui dispositivi mobile. La sua facilità di utilizzo, insieme alla larga adozione da parte degli sviluppatori ha fatto sì che nel tempo, fosse integrata con svariati plugin divenuti man mano ufficiali ed elencati sul sito ufficiale. Quelli usati sono:

  - *leaflet-boundary-canvas*: serve per la creazione di mappe statiche a mò di immagine;
  - *leaflet-easybutton*: permette l'inserimento di bottoni
  - *leaflet-search*: permette di cercare i punti o qualsiasi tipo di dato spaziale partendo da un valore di un attributo prestabilito.
  - *leaflet.heat*: dà la possibilità di creare mappe di colore, meglio conosciute come *heatmap*.
  - *leaflet.markercluster:*consente di clusterizzare i vari punti su mappa per fornire un rendering e uno scroll dinamico.
  - *leaflet.slider: slider* dinamico integrabile direttamente su mappa.

- **Turf.js**[11]:libreria per l'analisi geospaziale su client. Fornisce molte features e dati spaziali e il loro utilizzo è estremamente facile e veloce.

[8] https://www.npmjs.com/package/pg
[9] https://www.r-project.org/about.html
[10] https://leafletjs.com/
[11] https://turfjs.org/

3.2 Struttura del Progetto

Le tecnologie che sono state elencate, sono da considerarsi requisiti essenziali. Il loro utilizzo, ha apportato un notevole vantaggio nella stesura del codice.
Il passo iniziale è stato l'installazione di *Node.js*, più comunemente chiamato *Node*, senza del quale tutte le operazioni successive, sarebbero vane. Per installarlo, bisogna andare sul sito ufficiale, nella sezione Download[12] e scaricare la versione relativa all'architettura delle propria macchina. La sua installazione, porterà in aggiunta con sé, anche il packet manager `npm` grazie al quale sarà possibile accedere a tutte le librerie necessarie. Una volta installato, si può passare all'installazione di *Angular CLI*:

```
1   > npm install -g @angular/cli
```

L'ultimo comando racchiude la sintassi[13] tipica di `npm`. Con la dicitura `install` si sta chiedendo al packet manager di installare in una cartella denominata `node_modules` la libreria `angular/cli`. Il comando `-g` stabilisce che la sua installazione debba essere effettuata a livello globale nel sistema e non solo nella cartella corrente in modo tale da poter essere lanciato in qualsiasi parte del File System. Ora il sistema possiede i comandi di Angular CLI. Per cui:

```
1   > ng new location_intelligence
2   > cd location_intelligence
3   > ng serve
```

Lanciando i comandi appena visti, viene creato un nuovo progetto a cui è stato dato il nome "*location_intelligence*". In particolar modo con il primo comando, è creato il progetto in sé. In questa circostanza, verrà chiesto all'utente di settare le preferenze relative all'impostazione dei file di *routing* e al formato usato per i documenti di stile. La scelta si è soffermata su *Sass*[14] per avere in un secondo momento la possibilità di definire dei

[12] https://nodejs.org/en/download/

[13] Per maggiori informazioni, come i comandi per disinstallare un pacchetto, si può far riferimento alla documentazione ufficiale: https://docs.npmjs.com/

[14] Estensione di CSS. Permette la dichiarazione di variabili e di organizzare lo stile in più file diversi.

propri temi. Dopo avervi acceduto con il comando `cd`, lo si può lanciare digitando `ng serve`. Aprendo il browser, basterà aprire una tab all'indirizzo `http://localhost:4200` per vedere la nuova app in esecuzione.
Il contenuto della cartella *location_intelligence* rappresenta una struttura vergine comprendente solo delle parti necessarie al suo funzionamento.

```
location_intelligence :.
|   angular.json
|   package.json
|    ...
|
+---e2e
|      ...
|
+---node_modules
|       ...
|
+---src
    |  index.html
    |   ...
    +---app
    |       app-routing.module.ts
    |       app.component.html
    |       app.component.sass
    |       app.component.spec.ts
    |       app.component.ts
    |       app.module.ts
    |
    +---assets
    +---environments
```

Tutti i progetti creati con Angular CLI condividono la stessa struttura di configurazione [19] [20] [21] :

- **angular.json**: È un file che racchiude tutte le configurazioni di base per il progetto come la collocazione dei file di stile, la struttura di compilazione, la gestione dei test: TSLint, Karma e Protractor;

- **package.json**: Json che da informazioni sul servizio di Host usato[15], il nome, la versione e gli autori. Soprattutto contiene tutte le librerie adoperate nel progetto. Aprendo il file si può notare che ci sono due tipi di librerie: quelle riguardanti esclusivamente il progetto e quelle che riguardano lo sviluppo. La loro collocazione nel file dipende dalla sintassi adoperata durante l'installazione[16]. Infine, ha al suo in-

[15] In questo caso è GitHub https://github.com/

[16] Due principali tipi di installazioni:

npm install –save nome_pacchetto # installazione per uso all'interno del progetto

npm install –save-dev nome pacchetto # installazione per uso di sviluppo.

terno i comandi necessari per lanciare il progetto in varie modalità: produzione, sviluppo, debug, test etc.;

- altri file riguardanti la configurazione di Typescript o relativi ai test;
- **node_moduels**: come già introdotto è una folder rappresentante la destinazione di tutti i pacchetti scaricati;
- **e2e** sta per *end-to-end* ed è la directory all'interno del quale scrivere i test finali. Nello specifico i test in questione sono automatizzati e cercano di simulare il comportamento degli utenti: accesso al sito, compilazione di form e disconnessioni. Non è presente all'interno di *src* perché è un'app separata con delle proprie dipendenze.
- **src**: è il fulcro del progetto. Ha al suo interno il codice dell'implementazione ed è diviso in varie sottocartelle.
  - **app:** la struttura tipica comprende:
    - un file *.html* per la definizione del template e i metatag che orientano il comportamento dell'applicazione;
    - un file *.sass* dove è definito il layout;
    - un file *.module* che è il contenitore di tutte le componenti specifiche adoperate;
    - un file *.component* dove viene implementata la logica dell'intera applicazione (in Typescript) o del singolo componente se l'app comprende molteplici componenti[17];
    - un file di *routing*, eventualmente compreso nel modulo, dove si definiscono le policy per la navigazione. Se il modulo è unico allora ne basterà uno. Se i moduli che compongono l'app sono molteplici, allora per ognuno bisognerà definire il file di routing.
  - **index.html**: è il file che viene richiamato in ogni progetto Angular. E' qui che avviene il boot delle varie sezioni che costituiscono il progetto presenti nella cartella *app*: moduli, componenti servizi etc.

[17] Come si approfondirà più avanti

  - **asset**: tutto ciò che è contenuto in questa spazio è considerato un asset del progetto: loghi, immagini, eventuali librerie personali. In parole povere, è il luogo in cui si possono archiviare risorse statiche.
  - **environment**: contiene i file (generalmente sono solo 2, ma è possibile averne anche più) dove si possono definire gli ambienti in cui si sta operando. Tra le tante opzioni, comprendono anche gli indirizzi server a cui puntare per i vari ambienti.

Quanto descritto è come si presenta un progetto Angular. Come è abbastanza intuibile, la sua versione standard è strutturata in modo che tutta l'applicazione sia scritta in un solo modulo e da una o più componenti. In questo specifico caso, dovendo rispettare il layout definito nei mockup, un solo modulo, non sarebbe stato sufficiente. Oltre questo, la struttura standard è poco scalabile e la crescita delle dimensioni del progetto comporta scompensi sia organizzativi che di prestazioni.
Per questa serie di ragioni, è stata modificata la struttura secondo le *best practice* usate in questi contesti [22] [23] [24].
La parte interessante del cambiamento, riguarda il contenuto della cartella `/src`. Essa è stata riorganizzata secondo le seguenti tre macro componenti:

- **Core Module**: Rappresenta un modulo al cui interno sono contenute e implementate le singole istanze, o più come è comunemente chiamate *singleton*. Un esempio calzante può riguardare la definizione delle chiamate HTTP, oppure i dati relativi all'impostazione delle applicazioni;
- **Feature Module**: Si riferiscono ai vari moduli di cui è composta l'applicazione è che presentano la struttura tipica della directory "/app" definita poc'anzi
- **Shared Module**: Tutta la logica e le features[18] che vengono condivise tra le varie parti del progetto devono essere racchiuse qui. Questi componenti non importano e non iniettano servizi dai loro core o altre

[18]Da questo momento, per features si intenderà le tab della dashboard. Per questo durante il resto della trattazione i due termini saranno usati scambievolemente.

funzionalità nei loro costruttori ma sono considerati dei meri contenitori. Ricevono dati come parametri dai componenti *features* che li contengono e non hanno dipendenze tra di loro e/o verso altre parti del progetto. Ad esempio, i grafici o le mappe possono essere un esempio chiarificatore;

Una volta chiarificata la struttura, vediamo come si è arrivato ad ottenere quanto descritto. Per creare il *core:*

```
1  > ng generate module core
```

Al suo interno sono presenti vari servizi, definiti con:

```
1  > ng generate service 'nome_servizio'
```

Essi sono:

- *HTTP*: per effettuare le chiamate al server;
- *logging*: per definire uno stack della navigazione;
- *settings*: potenzialmente per contenere qualsiasi dato del progetto come la lingua o le impostazioni dell'utente, ma in realtà è stato creato per chiamare dinamicamente i vari temi contenuti nella cartella `/theme`. Essi sfruttano RxJS[19] per gestire gli stati dell'applicazione;

In secondo luogo, si è passati alla definizione delle *features*. Esse sono directory che si riferiscono alle tab presentate nella tabella 2.4. Per cui si avranno le seguenti:

- Doors
- Sell out
- Country
- Regions
- Potential Analysis

[19]NGRX è un gruppo di librerie "ispirate" dal modello Redux che a sua volta è "ispirato" dal modello Flux (Santiago García da Rosa 2018).

- Potential Analysis Italy
- Predictive Analysis Italy

Ognuna di esse comprenderà i file presi in esame prima. Tali tab saranno visualizzate in una Sidebar. La sua implementazione è definita nella cartella che prende lo stesso nome. Lo stesso discorso vale per la *navbar*. Nella cartella `/shared` invece ci saranno, rispettando la logica esplicitata nella definizione, i grafici, le tabelle, i filtri e le mappe. Tutte componenti statiche, che implementano la logica di funzionamento, ma che si limitano a visualizzare quanto gli viene passato e a restituire eventualmente il risultato alle features da cui vengono chiamate. La struttura finale si presenterà nel seguente modo:

```
location_intelligence :.
|   .env
|   angular.json
|   package-lock.json
|
+---e2e
|   +--- ...
+---node_modules
+---src
    +---app
    |   |   app-routing.module.ts
    |   |   app.module.ts
    |   +---app
    |   |   |   app.component.html
    |   |   |   app.component.scss
    |   |   |   app.component.scss-theme.scss
    |   |   |   app.component.spec.ts
    |   |   |   app.component.ts
    |   |   +---components
    |   |       +---navbar
    |   |       +---sidebar
    |   +---core
    |   |   |   core.module.ts
    |   |   |   core.state.ts
    |   |   +--- http-request
    |   |   +--- ...
    |   +---features
    |   |   |   features.component.scss-theme.scss
    |   |   +---country
    |   |   +---doors
    |   |   +---potential-analysis
    |   |   +---potential-analysis-italy
    |   |   +---predictive-analysis
    |   |   +---regions
    |   |   +---sellout
    |   +---shared
    |       +---modules
    |           |   shared.component.scss-theme.scss
    |           +---charts
    |           |   |   charts.component.scss
    |           +---filter
    |           +---map
```

```
|           |   +---map-country
|           |   +---map-door
|           |   +---map-potential
|           |   +---map-predictive
|           |   +---map-region
|           |   +---map-sellout
|           |   +---static-map
|           +---mat-card-image
|           +---mat-card-information
|           +---mat-card-potential
|           +---table
+---assets
+---environments
+---styles
+---theme
...
```

Prendendo visione della struttura finale, non si può non spendere due parole per la gerarchia che compone la creazione delle mappe. Essendo mappe diverse fra di loro a seconda delle funzioni, si è optato per la creazione di una mappa generica che integrasse caratteristiche generiche quali: gli zoom, i filtri, la ricerca etc. Poi per ogni tipo di mappa, secondo la collocazione all'interno della features di di riferimento, ne sono state create di nuove, le quali ereditano dalla principale, secondo i principi dell Object Oriented, e l'arricchiscono con delle features specifiche.
L'ultima precisazione riguarda l'aspetto legato al layout. Ogni modulo definito nel progetto che espone una parte grafica, richiama al suo interno, la componente Angular Material che lo riguarda.
La parte server, contenuta nella stessa cartella di quella client, ha al suo interno i file riguardanti la gestione del database descritti nel Capitolo 2; gli algoritmi di Machine Learning testati, ma soprattutto il codice che definisce la logica del server. Il file che assume maggiore importanza è `app.js` e `/bin/www`. Quest'ultimo richiama il primo per avviare il server sulla porta specificata. Nella cartella `/routes` sono gestite le chiamate dal client. Qui oltre a ricevere la richiesta, vengono scaricati i dati e rimandati al client. Generalmente con dati non sono solo da intendere quelli contenuti nel Database, ma anche i risultati del modello predittivo. Esso è situato in `script_py`. Il modello a cui fa riferimento è un file su cui è stato fatto training tra i vari modelli presenti in `/ML`

```
location_intelligence:.
|
+---server
|   |   app.js
```

```
|   +---bin
|   |       www
|   +---management_db_postgis
|   +---ML
|   +---postgres-postGIS
|   +---routes
|   +---script_py
|       |   predict_doors.py
+---+---saved_model
```

Quanto visto finora è la struttura del progetto creata lato client-server. Per motivi espositivi, le due strutture sono state divise, ma come si può notare dalla `root`, fanno riferimento alla stessa directory. Le librerie e tutte le dipendenze dell'uno e dell'altro sono contenute nello stesso file: `angular.json`. Il motivo di questa scelta progettuale è data dalla volontà di non volere separare due strutture che comunicano in maniera sincrona. Lanciando il comando:

```
npm run dev
```

Il progetto, avvia simultaneamente[20] sia il client che il server grazie al pacchetto `npm`, visto prima, `concurrently`.

```
"scripts": {
   "ng": "ng",
   "dev": "concurrently -c \"yellow.bold,green.bold\" -n \"SERVER,BUILD\" \"nodemon
        ./server/bin/www\" \"ng serve \"",
   "build": "concurrently -c \"yellow.bold,green.bold\" -n \"SERVER,BUILD\" \"node
        ./server/bin/www\" \"ng build --watch\"",
    ...
 },
...
```

Ciò che ci si chiede è come avvenga la comunicazione nel concreto. La seguente immagine prova a chiarire questo aspetto.

[20] Per l'implementazione, viene lanciato sempre il progetto in modalità sviluppo, ma per la fase di deployment invece, si userà il comando build

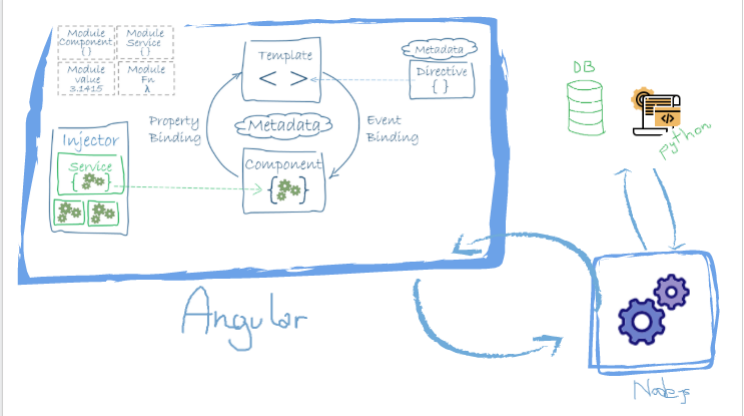


Figura 3.1: Architettura del progetto client-server

Angular, che nell'immagine rappresenta la parte client, è costituito da diverse componenti. La parte centrale a cui viene dato spazio riguarda la comunicazione tra la `Component` e il `Template`. Queste ultime definiscono insieme un `Angular View`. In particolare, nella `Template` sono definiti i metadati che lo identificano come tale e nella `Component` invece, ci sono i riferimenti ai puntatori che consentono lo scambio di dati e di informazioni. Nel `Template` è possibile eseguire loop, e dichiarare variabili sulla base del contenuto della `Component`. Ancora, tornando a parlare della `Component`, sono richiamate tutte le `Service` che occorrono a quella specifica classe secondo il paradigma del `Dependency Injection` [25]. Tale paradigma, è una delle peculiarità di Angular e consiste nell'*iniettare* solo ciò che è strettamente dipendente in una specifica classe. Con questo si innesca un meccanismo di dipendenza tra le varie parti in gioco. Se vengono rispettate i canoni e le Best Practice cha Angular suggerisce, ci si troverà ad avere una struttura modulare e ben organizzata che non sia bloccante, ma che sia definita in modo che ogni parte assolva alla propria funzione correttamente.
Nel caso del progetto, oggetto di discussione, un esempio, come si vedrà sarà rappresentato dai servizi Http.
Ed è proprio in questo specifico frangente che vengono definiti i metodi che effettuano le chiamate al server. Quest'ultimo, le riceve con l'ausilio di funzioni `GET` o `POST`, effettua le query al Database o allo script Python e restituisce alla componente Angular il risultato che carica al suo interno il

Servizio HTTP preposto alla ricezione di quella specifica chiamata. Successivamente, quanto ottenuto verrà passato al template di riferimento, il quale lo mostrerà o per mezzo di tabelle o, come la maggior parte dei casi, tramite mappe.
Con questo si conclude la sezione relativa alla struttura del report. Adesso, si darà spazio all'aspetto implementativo.

3.3 Angular

Nell'ultima parte del precedente paragrafo, si è argomentato circa la struttura di comunicazione tra *client* e *server* spiegando succintamente come essa avviene. Si è visto inoltre, quale sia l'organizzazione di Angular nel complesso senza entrare troppo nei dettagli. L'obiettivo di questo paragrafo e del successivo è mostrare dettagliatamente come sono state implementate le singole parti che costituiscono il client e in quello successivo la parti che compongono il server.

3.3.1 Index.html

All'avvio del progetto, Angular richiama il file `Index.html`. Questo può essere visto come l'unico file convenzionale alla tipica programmazione da Web ed è il primo file che viene eseguito quando la pagina viene caricata.

```
1  !doctype html>
2 <html lang="en">
3 <head>
4   ...
5 </head>
6 <body>
7   <app-root>
8     <!-- loading layout replaced by app after startupp -->
9     <div class="app-loading">
10      ...
11    </app-root>
12 </body>
13 </html>
```

Sono presenti tutte le parti che costituiscono un file html standard[21], ma ciò che lo caratterizza è la presenza della chiamata all'`AppModule` che

[21] E presente anche il Layout di uno Spinner in fase di caricamento

racchiude il vero `boot` dell'applicazione. Questo è il compito principale del file `Index.html`.

3.3.2 AppModule

L'`AppModule` può essere considerato, per ovvie ragioni, il modulo più importante dell'intero progetto ed è situato, come ogni tipico progetto Angular nella cartella `/app`. Consta di due file principi: `app.module` e `app-routing.module`. Mentre una piccola applicazione potrebbe avere un solo Module, la maggior parte delle app ha molti più moduli di funzionalità [26] come in questo caso.

```
1  import { NgModule } from '@angular/core';
2 import { BrowserModule } from '@angular/platform-browser';
3 import { HttpClientModule } from '@angular/common/http';
4 import { AppComponent } from './app/app.component';
5 import { AppRoutingModule } from './app-routing.module';
6 ...
7 @NgModule({
8   declarations: [
9     AppComponent,
10    NavbarComponent,
11    SidebarComponent
12  ],
13  imports: [
14    BrowserModule,
15    AppRoutingModule,
16    HttpClientModule,
17     ...
18  ],
19  bootstrap: [AppComponent],
20  providers: []
21 })
22 export class AppModule { }
```

Essendo Angular 7 implementato in Typescript, segue la sua sintassi. Nella parte iniziale del file, sono importate tutte le dipendenze. Quelle essenziali sono le seguenti:

- **NgModule**: Decorator[22] che contrassegna una classe come *modulo* e fornisce i metadati di configurazione racchiusi nel JSON `@NgModule`;

[22] I Decorator rappresentano un pattern di progettazione utilizzato in vari linguaggi di programmazione come Javascript o Python. La sua peculiarità riguarda l'aggiunta di un singolo oggetto staticamente o dinamicamente senza che si influisca con il suo comportamento su altri oggetti. Tutto ciò per aderire al concetto di Single Responsibility, il quale prevede che ogni componente debba assolvere ad un unica funzione per la quale è stata creata(Bojan Gvozderac 2017).

- **BrowserModule**: Modulo che permette di esportare l'app sul Browser;
- **HttpClientModule**: Modulo che predispone l'app alle chiamate Http;

Gli altri due import permettono di aggiungere rispettivamente il `Component` inerente il Modulo e il file di configurazione del `Routing`.
I metadati di configurazione a cui si è accennato prima, permettono di settare tutte le componenti.

- **declarations:** racchiude tutte le direttive, e componenti che appartengono a seguente modulo e che sono dichiarate per essere usate ai fini implementativi della logica che espone;
- **imports**: contiene tutte le classi che sono necessarie per conferire un modello di comportamento al modulo. Ad esempio, senza l'`imports` de modulo Http non è possibile comunicare con il server;
- **exports**: permette di esportare le funzionalità che questo modulo implementa e definisce all'esterno, in modo che gli altri moduli possano usufruirne secondo il modello della *Dependency Injection*,visto poc'anzi.
- **bootstrap**:ha al suo interno la vista principale del modulo. Dice al motore Angular quale component avviare. In questo caso,è l'AppComponent;
- **providers**: grazie a questo meta dato, si definisce un modulo come un Service e permette di esportarlo al di fuori, così come viene fatto con il metadato *exports* per le componenti.

In generale si può asserire come gli import, rendono disponibile al modulo il loro utilizzo, mentre i metadati, definiscono la funzione da usare.
Tra gli import presenti nello script, quello di routing, è molto importante[23].
La sua funzione è di switchare tra le varie tab dell'applicazione:

```
1 import { NgModule } from '@angular/core';
2 import { Routes, RouterModule } from '@angular/router';
```

[23] può anche essere incluso nello stesso file

```
const routes: Routes = [
  {
    path: '',
    children:
      [
        {
          path: '',
          redirectTo: '/doors',
          pathMatch: 'full'
        },
        {
          path: 'doors',
          loadChildren: './features/doors/doors.module#DoorsModule'
        },
        {
          path: 'sellout',
          loadChildren: './features/sellout/sellout.module#SelloutModule'
        },
        {
          path: 'country',
          loadChildren: './features/country/country.module#CountryModule'
        },
        ...
      ]
  }
];

@NgModule({
  imports: [RouterModule.forRoot(routes)], // important: this is forChild and not forRoot
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Come si può vedere, esso sembra un modulo a se stante. Lo si può supporre dall'import `NgModule` e dai relativi metadati. Tuttavia, è diverso. Ciò che lo contraddistingue, è la presenza del `routing` con il quale si da la possibilità all'utente di cambiare la view con un click. Per ognuna esiste un percorso. Ad esempio, se si vuole vedere la tab delle Country, si dovrà digitare l'indirizzo `http://localhost:4200/country` o tramite url o tramite la corrispondente tab nella Sidebar, la quale rimanderà allo stesso url. Di default, se nessun percorso è assegnato, viene chiamato il Modulo relativo alle Doors.

```
    {
        path: '',
        redirectTo: '/doors',
        pathMatch: 'full'
    },
```

Il routing opera sulla `Sidebar`. La sua collocazione è posta ad un livello gerarchico superiore rispetto gli altri Moduli del progetto, quindi nell'`AppModule`. I metadati `@NgModule` inizializzano il router e lo metto-

no in ascolto. L'import configura `route` in un solo passaggio chiamando `RouterModule.forRoot(routes)`, mentre l'export lo rende accessibile attraverso l'app, nel caso specifico dal file `/app/app/app.component.html`. Esso, è come segue:

```
<!-- Define NavBar -->
 <app-navbar></app-navbar>

 <!-- Define sidebar -->
 <app-sidebar></app-sidebar>

 <div class="theme-wrapper {{theme$ | async}}">
     <div class="main-container">
         <router-outlet></router-outlet>
     </div>
 </div>
```

Al suo interno, è settato il punto di accesso tra le varie componenti dell' applicazione del `router`. Il `main-container` ha il compito di ospitare il rendering delle diverse tab selezionate attraverso la *Sidebar*. La dicitura `theme-wrapper {{theme$ | async }}` asserisce come tutto il container deve rispettare i canoni di layout definiti da file css.
Il file che definisce la logica del template html è un file con estensione ts: `app.component.ts`

```
import { ChangeDetectorRef, Component, OnInit, OnDestroy } from '@angular/core';
...

@Component({
    selector: 'app-root',
    templateUrl: './app.component.html',
    styleUrls: ['./app.component.scss']
})
export class AppComponent implements OnInit, OnDestroy {
    // variable
    private _mobileQueryListener: () => void;
    mobileQuery: MediaQueryList;
    theme$: Observable<string>;


    constructor(changeDetectorRef: ChangeDetectorRef, media: MediaMatcher, private store: Store<State>) {
        this.mobileQuery = media.matchMedia('(max-width: 600px)');
        this._mobileQueryListener = () => changeDetectorRef.detectChanges();
        this.mobileQuery.addListener(this._mobileQueryListener);
    }

    ngOnInit() {
      ...
    }

    ngOnDestroy(): void {
      ...
    }
}
```

Questo file rappresenta il vero fulcro della logica di ogni modulo. Viene decretato come Component per via dell'import dal `@angular/core`[24] del Decorator "*Component*" . Con esso, si definisce

- un selettore identificativo dell'url di riferimento, `app-root`;
- il template di riferimento con `@angular/templateUrl`, in questo caso rappresentato dal file `app.component.html`;
- il file di stile che sancisce il layout: `styleUrls`, il quale fa riferimento a `app.component.scss`.

Nel costruttore sono importati i servizi necessari alla root del progetto. Avendo come obiettivo quello di definire il comportamento a livello *mobile*, sono importati i `ChangeDetectorRef` e `MediaMatcher`.
I metodi `ngOnInit()` e `ngOnDestroy()` fanno parte del "*Ciclo di Vita*" della Component e sono metodi che mettono a disposizione degli eventi ad ogni singolo passaggio dalla sua creazione alla sua distruzione. i più importanti sono:

- **ngOnChanges()**: chiamato prima di `ngOnInit()`, ogni volta che cambia una o più proprietà di input associate ai dati;
- **ngOnInit()**: chiamato una volta, dopo il primo `ngOnChanges()`. Inizializza tutte le variabili che vengono istanziate per la prima volta. Eventuali rilevazioni di cambiamenti, non saranno captati da tale metodo.
- **ngOnDestroy()**: Quando il cicli di vita della component arriva a compimento, tale metodo conterrà le chiamate necessarie per la pulizia di tutte le direttive o qualunque variabile impiegata nella logica implementata.

Eventuali altri metodi riguardanti il ciclo di vita, sono consultabili sulla guida ufficiale [26].

[24] rappresenta uno dei package principali di Angular. Importa tutte le funzionalità, i servizi di basso livello e le utility necessarie ad Angular.

3.3.3 Navbar & Sidebar

Nel file html dell'`AppModule` si fa riferimento a due componenti: la Navbar e la Sidebar. Si ricorda che il tag che le identifica, come può esserlo `<app-navbar></app-navbar>` per la navbar, corrisponde al selettore delle singole componenenti. Queste ultime due non presentano un file *module* perchè incluse nell' `AppModule`.
Il `[routerLink]="['/doors']"` lega la l'hyperlink della tab con il path visto nel file di routing.

```
<div class="theme-wrapper {{theme$ | async}}">
    <div id="sidebar">
        <mat-nav-list (mouseleave)="hoverElemTop=-56">
            <a (mouseenter)="hoverItem($event)" mat-list-item [routerLinkActive]="'active'"
                [routerLink]="['/doors']">
                <mat-icon svgIcon="store"></mat-icon>
                Doors
            </a>
            ...
            ...
            </a>
            <a (mouseenter)="hoverItem($event)" mat-list-item [routerLinkActive]="'active'"
                [routerLink]="['/predictive_analysis_italy']">
                <mat-icon svgIcon="predictive_analysis"></mat-icon>
                Predictive analysis Italy
            </a>
          </mat-nav-list>
        ...
    </div>
</div>
```

In questo caso il file Typescript assume poca rilevanza in termini di logica che importa. Il layout, è impostato con il file avente estensione `scss`. La posizione della Sidebar è sempre fissa a dimensioni definite.

```
@import '../../../../ styles/variables';


#sidebar {
    width: $sidebar-base-width;
    position: fixed;
    top: 56px;
    bottom: 0;
    box-shadow: 3px 0 6px rgba(0, 0, 0, 0.24);
    z-index: 8000;
    ...
}

...
...

@media screen and (max-width: 992px) {
    #sidebar {
        left: -$sidebar-base-width;
    }
```

```
}
```

Nella Media query viene settata la visualizzazione in relazione alla dimensione della finestra.
Per quanto riguarda la Navbar invece, contiene un logo a sinistra e un bottone a destra per la scelta dei temi.

```
<button type="button" mat-icon-button class="visible-md" (click)="toggleSidebar()">
    <mat-icon aria-label="Side nav toggle icon">menu</mat-icon>
</button>
...
...
<button mat-menu-item *ngFor="let t of themes" [value]="t.value" (click)="onThemeSelect(t)">
    {{ t.label }}
</button>
```

Come si può notare la presenza di due funzioni, una nel toggle di navigazione e l'altra per i temi. Entrambe sono situate nel file Typescript. E' per tale motivo che si sostiene che è nel file Typescript la logica descrittiva.

```
theme$: Observable<string>;

  // define theme
  themes = [
    {value: 'default-theme', label: 'blue'},
    { value: 'light-theme', label: 'light' },
    { value: 'nature-theme', label: 'nature' },
    { value: 'black-theme', label: 'dark' }
  ];

  constructor(public router: Router, private store: Store<State>) {
    this.router.events.subscribe(val => {
      if (val instanceof NavigationEnd && window.innerWidth <= 992 && this.isToggled()) {
        this.toggleSidebar();
      }
    });
  }
...
toggleSidebar() {
    const dom: any = document.querySelector('body');
    dom.classList.toggle(this.pushRightClass);
  }
```

In questo contesto, il toggle di navigazione è sempre presente nel file html, ma è nel file Typescript, in combinazione con il file scss, che si decide se mostrarlo o meno a seconda della grandezza della finestra. In caso di visualizzazione da Mobile, la Sidebar scompare e il Toggle appare.

```
...
.visible-md {
    display: none;
}
.visible-sm {
```

```
6       display: none;
7   }
8   ...
9   @media screen and (max-width: 768px) {
10      ...
11
12      .nav-brand {
13          width: 100%;
14      }
15  }
16  @media screen and (max-width: 768px) {
17      .hidden-sm {
18          display: none;
19      }
```

Ora la Navbar e la Sidebar, sono implementate. L'attenzione ora si sposterà dapprima sulle componenti situate nella cartella `/shared` , successivamente sul container che ospiterà la view e che a seconda della tab scelta, cambierà il contenuto.

3.3.4 Shared Modules

Nella cartella `/shared` sono contenuti i moduli condivisi fra le varie tab. Come il nome fa intuire, sono associabili a dei tasselli orchestrati tra le varie tab secondo gli scopi per i quali sono stati ideati. L'idea con la quale sono state programmati, è quella di ottenere delle componenti il più possibile riutilizzabili, evitando di crearne "ad hoc" per ogni diversa esigenza. Il motivi alla base di questa scelta sono principalmente due: il primo è legato alla possibilità di ampliare il progetto, ponendo l'attenzione su nuovi dati; il secondo invece riguarda la maggiore manutenibilità del codice. Infatti il programmatore, potendo contare sulla sua modularità, è in grado di individuare e/o correggere eventuali bug, poter modificarlo e ampliarlo, senza gravare sulle performance della dashboard, in maniera trasparente. Dal canto suo, Angular, incentiva il programmatore alla suddivisione modulare proprio perché essa rappresenta una delle peculiarità più importanti che lo hanno reso popolare.
In questa sezione, si dettaglierà i vari moduli, secondo il loro grado di importanza.

3.3.4.1 Leaflet

Leaflet.js è la libreria Javascript leader di mappe interattive e ottimizzate sui dispositivi mobile. Offre elevate performance su tutti i tipi di piattaforme, sfruttando HTML5 e CSS. Ha a disposizione una vasta gamma di plugin facili da usare e ben documentati. Ai fini progettuali sono stati definiti varie tipologie di mappe:

- mappa delle door, sellout, country e region;
- mappa del potenziale;
- mappa predittiva;
- mappa statica

Ad ognuna di essa, è associato un elemento `div` con medesimo ID condiviso tra tutte, che fungerà da contenitore. La sua collocazione è da attribuirsi nel punto in cui si desidera visualizzare la mappa. L'insieme di tutte le mappe, fanno parte di un unico Angular Module. L'approccio utilizzato per la sua creazione è il seguente:

```
1 # Generate Module without Routing
2 ng generate module --routing=false shared/modules/map
3
4 # Generate components: scss, html, Typescript files
5 ng generate component shared/modules/map
```

Con la prima riga viene creato il modulo come quanto fatto nell' `AppModule`. Il comando successivo, crea le componenti: i file scss, il file html, i file Typescript. L'ordine di chiamata è importante. Se fossero invertite, il motore di Angular sarebbe andato ad importare i file generati nell'`app.module.ts`. Invece, creando prima l'apposito modulo, Angular capisce che deve far riferimento ad esso quando prova ad importare le componenti. Nulla impedisce lo spostamento manuale, cancellandole da un file ed aggiungendole in un altro, ma questa operazione è *error prone*.
In questo file, il file di routing è omesso perchè stiamo parlando di componenti statiche orchestrate ad un livello di gerarchia superiore.
Le mappe implementare in questo progetto sfruttano l'ereditarietà, tipica dei linguaggi di programmazione *Object Oriented*. Typescript appartiene

alla cerchia di questi linguaggi di programmazione avendo in dote la possibilità di creazione ed utilizzo delle classi.

```
1  ...
2  import { MapDoorComponent } from './map-door/map-door.component';
3  import { UtilitiesService } from '../../../ core/ utilities / utilities .service';
4  import { MapSelloutComponent } from './map-sellout/map-sellout.component';
5  import { MapCountryComponent } from './map-country/map-country.component';
6  import { MapRegionComponent } from './map-region/map-region.component';
7  import { MapPotentialComponent } from './map-potential/map-potential.component';
8  import { DynamicPanelService } from 'src/app/core/utilities/dynamic-panel.service';
9  import { StaticMapComponent } from './static-map/static-map.component';
10 import { MapPredictiveComponent } from './map-predictive/map-predictive.component';
11
12
13 @NgModule({
14   declarations: [MapDoorComponent, MapSelloutComponent, MapCountryComponent,
         MapRegionComponent, MapPotentialComponent, StaticMapComponent, MapPredictiveComponent],
15   imports: [
16     MatGridListModule,
17     MatCardModule
18   ],
19   providers: [... , DynamicPanelService],
20   exports: [ MapDoorComponent, MapSelloutComponent, MapCountryComponent, MapRegionComponent,
         MapPotentialComponent, StaticMapComponent, MapPredictiveComponent]
21 })
22 export class MapModule { ... }
```

Il modulo racchiude la totalità delle mappe e funge da snodo per la comunicazione con le parti che le richiamano. Tralasciando gli *import* relativi al settaggio del modulo, vengono importate le singole componenti delle mappe. Nell'array `declarations` è presente tutto ciò che verrà utilizzato nel modulo ai fini implementativi. Nell'array degli *imports*, aggiunti i moduli riguardanti il *Material Design*. Nel `providers`, aggiunti i servizi utili e successivamente, nell'`exports` reso tutto disponibile all'esterno.
Avendo sfruttato l'ereditarietà, basterà richiamare dal codice di una specifica tab `map.module.ts` per avere accesso a tutte le restanti mappa, specificando semplicemnete il selettore di interesse.
E' nel file `map.component.ts` che viene creata la mappa utilizzando i cicli di vita menzionati poc'anzi e riciclata nei figli che la ereditano.

```
1 // Import Leaflet and plugins
2 import 'leaflet';
3 import 'leaflet.markercluster'
4 import 'leaflet-search'
5 import 'leaflet-zoombox'
6 import 'src/assets/ leaflet-slider-master/dist/leaflet-slider.js';
7 import 'leaflet.heat';
8 import 'leaflet-easybutton';
9 declare let L;
```

```
10
11 @Component({
12   template: ''
13 })
14
15 export abstract class MapComponent implements OnInit {
16   protected dynamic_panel: DynamicPanelService;
17   // Input data. They derive from father component.
18   @Input() data: any;
19
20   // Output data. They are used to comunicate with the sibling and father
21   @Output() open_popup = new EventEmitter<Object>();
22   @Output() close_popup = new EventEmitter<Object>();
23   @Output() toPredict = new EventEmitter<Object>();
24
25   // declaration variables
26   private map;
27   private latitude = 44.5075;
28   private longitude = 11.3514;
29   // default zoom
30   protected zoom = 4;
31   ...
32
33   constructor() { ... }
34
35   /**
36    * Define the creation of map
37    * @param latitude.
38    * @param longitude
39    * @param zoom
40    * @param layer
41    */
42   ngOnInit() {
43
44    var mapboxlight = ... ;
45    var openstreetmaps = L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.pdf', {
46        attribution: '(c) <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a>
              contributors'
47     });
48    var mapboxdark = ... ;
49     // Create the new map
50     this.map = L.map('map', {
51       center: [this.latitude, this.longitude],
52       zoom: this.zoom,
53       layers: [mapboxlight, openstreetmaps, mapboxdark]
54     });
55     var control = L.control.zoomBox(options);
56     this.map.addControl(control);
57
58   }
59  ...
60 }
```

Salta subito all'occhio l'elevato numero di plugin che viene adoperato. Uno per la definizione dei cluster, un altro per definire gli slider, un altro per i bottoni; un altro ancora per per le heatmap e così via. Guardando il codice, è facile accorgersi di due caratteristiche che differenziano questa component da quella dell'`AppModule`.

La prima è la mancanza di un template html che sancisce la posizione della

mappa e la seconda è il settaggio della classe come *abstract*. Questo fa presupporre come non tutti i metodi siano definiti e che ereditando la classe, si dovrà andar a implementare quelli di cui si possiede solo la firma[25]. Nella dichiarazione delle variabili, sono presenti quattro tipi diversi. I due più comuni sono `private` e `protected` con significato rispettivamente di utilizzo esclusivo nella classe di appartenenza, in questo caso *MapComponent* o di utilizzo in tutto il package: la directory `/shared`. Gli altri due tipi sono tipici di Angular. Le variabili contrassegnate come `@input()` sono `decorator` che identificano un input proveniente da componenti esterne. Sarà premura di tale componente esterno passare attraverso dei metadati i dati di interesse. In questo caso è utilizzato per trasmettere le informazioni estrapolate da query e visualizzabili su mappe. Analogamente l'`@output()` assolve alla funzione opposta. Trasmetterà alla componente sorgente il risultato. Questo meccanismo tipico di Angular è sovente per la comunicazioni tra componenti anche se non è l'unico modo, o per lo meno non è il più pratico, come si vedrà più avanti.

Nell'*init* della classe è inizializzata la mappa e sono settate i layers da mostrare. La mappa ha delle caratteristiche proprie date dalle variabili private *latitudine, longitudine*, e lo *zoom*. Nella parte finale, sono aggiunti i plugin utilizzati a livello generale e che tutte le mappe dovranno possedere[26].

Sorvolando sui `gettere` `setter`, per estrarre e settare informazioni[27] vediamo quali sono i metodi astratti:

```
1  /**
2    * Define a Color of Shape
3    */
4   abstract getColor(d: any): String;
5
6   /**
7    * This abstract Function is used to define a specific behavior
8    * of child component.
9    * Increase a behavior of component started from openPopup
```

[25] In informatica, nella programmazione orientata agli oggetti, la firma di un metodo è costituita da un insieme di informazioni che identificano univocamente il metodo stesso fra quelli della sua classe di appartenenza. Tali informazioni includono generalmente il nome, il numero e il tipo dei suoi parametri. https://it.wikipedia.org/wiki/Firma_(programmazione)

[26] come ad esempio lo zoombox che consente di zoomare selezionando aree di spazio con il mouse.

[27] Anche questi metodi sono tipici della programmazione ad oggetti.

```
10    */
11   abstract behaviorMapChild(feature: Object, layer: Object): void;
12
13   /**
14    * Create specific layer
15    */
16   abstract setMapChild(): void;
17
18   /**
19    * Set a abstract method for filter
20    */
21   abstract setFilter(geojson: Object): void;
```

- **getColor()** assolve alla definizione dei colori per le regioni di spazio delle specifiche mappe e dei cluster;
- **behaviorMapChild()** prende in input un layer e le proprietà di un json per impostare regole di comportamento inerenti le info mostrare da *popup*[28];
- **setMapChild()**configura il tipo di mappa con le caratteristiche che deve assumere;
- **setFilter()**: mostra su una mappa figlia i dati filtrati;

Dei restanti metodi, quelli più importanti sono:

```
1  ...
2  ...
3  areaGeoJSONLayerCountry(geojson) {
4
5      // Define a style based on the sellout
6      var customeStyle = (feature) => {
7        return {
8          fillColor: '#034e7b',
9          weight: 2,
10         opacity: 1,
11         color: 'white',
12         dashArray: '3',
13         fillOpacity: 0.7
14       };
15     }
16     var geoJsonLayer = this.getL().geoJSON(geojson,
17       {
18         style: customeStyle,
19         onEachFeature: this.definePopup
20       });
21
22     return geoJsonLayer;
```

[28]In informatica, i pop-up sono degli elementi dell'interfaccia grafica, quali finestre o riquadri, che compaiono automaticamente durante l'uso di un'applicazione ed in determinate situazioni, per attirare l'attenzione dell'utente. https://it.wikipedia.org/wiki/Pop-up

```
23
24  }
25
26
27  /**
28   * Create a GeoJSON layer to bind to the map
29   * @param geojson geoJSON param
30   */
31  markersGeoJSONLayer(geojson) {
32
33
34    // Set Default Style Icon
35    var iconDefault = this.defaultIcon();
36
37    // GeoJSON Result
38    var geoJsonLayer = L.geoJSON(geojson,
39      {
40        onEachFeature: this.definePopup,
41        pointToLayer: function (feature, latlng) {
42          return L.marker(latlng, { icon: iconDefault })
43        }
44      });
45
46    return geoJsonLayer
47  }
```

Il primo sulla base di un layout, dato dalla variabile `customeStyle` riempie le aree di spazio definite da un Geojson come input e il secondo analogamente, sulla stessa scia assolve alla medesima funzione ma si sofferma sulla definizione dei cluster.
Altri metodi presenti nella classe sono `utilities`, simili a quanto visto o metodi che caratterizzano il layout per tipi di markers specifici, come quelli dell'analisi predittiva: musei, aeroporti, ristoranti, B&B.
Le singole mappe, individuabili dal `nome-mappa.component.ts`, e contenute in directory che portano lo stesso nome, hanno tutte la stessa struttura.

```
1  import { Component, OnInit, ViewEncapsulation, Input } from '@angular/core';
2  import { MapComponent } from '../map.component';
3  import { Panel } from 'src/app/core/utilities/dynamic-panel.service';
4
5  @Component({
6    selector: 'app-map-potential',
7    template: '<div id="map"></div>',
8    styleUrls: ['../map.component.scss'],
9    encapsulation: ViewEncapsulation.None
10 })
11 export class MapPotentialComponent extends MapComponent implements OnInit {
12
13   private layerGroup;
14   private prevColor: string = '';
15   @Input() zoom: any;
16
17   constructor() {
18     super();
```

```
19  }
20
21  ngOnInit() {
22    super.ngOnInit();
23    this.setMapChild();
24  }
25
26  setMapChild(){
27    this.getMap().setZoom(this.zoom);
28
29    this.layerGroup = this.getL().layerGroup().addTo(this.getMap());
30    // Define a new GeoJson layer from country Sell out
31    this.shapeBoundaries = this.areaGeoJSONLayer(this.data);
32    // Add to the map
33    this.layerGroup.addLayer(this.shapeBoundaries);
34    // add Search Control
35    this.searchControl(this.shapeBoundaries, 'name', this.getMap());
36    // define a legend for the country
37    this.mapLegend();
38    // define a div info for the map
39    this.mapInfo();
40  }
```

A scopo esplicativo, è stata presa in considerazione la mappa che dovrà visualizzare il potenziale delle aree. Come ogni Component importa i requisiti essenziali che la certificano come tale. Tra le direttive, il `template`[29] definisce il `div` con ID uguale a `map` per l'utilizzo della mappa, mentre il selettore, setta il `tag` richiamabile tra le varie parti dell'applicazione.
La classe `MapPotentialComponent` eredita da `MapComponent` ed estende i metodi del ciclo di vita necessari ai suoi scopi. Nel costruttore è richiamato costrutto `super()` il quale da accessibilità a tutti i metodi ereditati dalla classe padre. Non appena è creata la classe, il compilatore di Angular ridarà un errore. Questo perché esige che nella classe appena creata siano implementati i metodi astratti. Nell' `init` viene esplicitamente richiamata l'`init` del padre per creare la mappa e aggiungere i plugin di base come visto prima. La seconda chiamata invece, chiama il primo metodo astratto `setMapChild()`. Nel metodo, sono definite le impostazioni che caratterizzano la mappa:

- E' settato lo zoom con un valore pari al valore presente nella classe padre;

[29] Non è templateUrl perché non rimanda a nessun file html, ma lo incorpora nella direttiva stessa. Questa tecnica si usa quando il contenuto del file html è esiguo come in questo caso.

- Viene creato un LayerGroup[30] e assegnato alla mappa ereditata.
- Viene ritornato un GeoJson dalla chiamata ad `areaGeoJSONLayer` implementato in `MapComponent` ed aggiunto ad un layer della mappa. Ora grazie a questo metodo, tutte le regioni di spazio, individuate da GeoJson passate come input, sono colorate sulla base del sell out, insito nelle properties del geoJson, e del metodo `getColor()`. Quest'ultimo è richiamato dal metodo padre, ma punta alla sua implementazione nella classe corrente. Ciò è una delle proprietà dell'Object Oriented;
- Aggiunto il plugin della ricerca che si baserà su uno degli attributi estrapolati dalla precedente chiamata;
- essendo una mappa che mostrerà delle regioni di spazio, verrà settata l'area delle info e della legenda della mappa;

Degli altri metodi astratti, `behaviorMapChild` si premura di definire un comportamento specifico nel momento in cui viene cliccato uno specifico paese. Tale comportamento si rifà alla visualizzazione dei dettagli dello stato del paese:

```
behaviorMapChild(feature, layer) {
    // bind an event when i click on a specific markers and NOT for all
    layer.on({
      click: this.colorShape
    })
  }

  /**
   * Color the shape of a specif country
   */
  colorShape = (e) => {
    var layer = e.target;

    // if this layer already exist clear and return to
    // the previus style else add to the layer group
    if(layer.options.fillColor == 'red'){

      layer.setStyle({
        fillColor: this.prevColor,
        weight: 2,
        opacity: 1,
        color: 'white',
        dashArray: '3',
        fillOpacity: 0.7
      })
      this.dynamic_panel.decreasePanel(layer);
```

[30] Il LayerGroup è un raggruppamento di livelli usato per una corretta manipolazione e gestione.

```
27      }else {
28        this.prevColor = layer.options.fillColor;
29        layer.setStyle({
30          fillColor: 'red',
31          weight: 2,
32          opacity: 1,
33          color: 'white',
34          dashArray: '3',
35          fillOpacity: 0.7
36        });
37
38
39        this.dynamic_panel.increasePanel(layer);
40
41      }
42  }
```

Quanto segue sopra, verifica che il country non sia mai stato selezionato. In caso negativo, lo evidenzia di rosso e richiama uno dei metodi del servizio `DynamicPanelService` importato nel modulo[31]. Al contrario, invece verrà colorato del suo colore originale[32]. L'ultimo dei metodi astratti da implementare è il `setFilter`. Esso pulisce tutti i layer appartenenti al Group inizializzato prima e lo ripopola con l'unico layer scelto passato come parametro.

Le altre mappe sono implementate usando tutte la stessa logica. Certo, ognuna avrà delle caratteristiche che le differenzia dalle precedenti, ma il criterio implementativo adottato sarà pressoché lo stesso.

Un discorso a parte lo merita `MapPredictiveComponent`. La mappa risultante da questa classe, permette la gestione dei punti predittivi. La *tab* che la richiama, contiene soltanto essa. Non esiste nessun altro elemento a supporto.

```
1  import * as turf from '@turf/turf'
2  ...
3
4  @Component({
5    selector: 'app-map-predictive',
6    ...
7  })
8  export class MapPredictiveComponent extends MapComponent implements OnInit {
9
10   ...
11
12   // Zoom
13   @Input() zoom: any;
14
```

[31] In questo caso, il layer, è aggiunto ad un array il quale verrà ciclato nella features che sfrutta la mappa.

[32] verrà eliminato dall'array da ciclare.

```
15  // points
16  @Input() istat_information: any;
17  @Input() current_doors: any;
18  @Input() restaurant: any;
19  @Input() museum: any;
20  @Input() aeroporto: any;
21  @Input() bb: any;
22  @Input() bus: any;
23
24   ...
25
26  constructor(private spinner: NgxSpinnerService) {
27    super();
28  }
29
30  ngOnInit() {
31    super.ngOnInit();
32    this.setMapChild();
33    this.mapLegendPrediction();
34  }
35
36  setMapChild() {
37
38    // offer  analysis
39    this.currentDoorsGroup = this.getL().layerGroup().addTo(this.getMap());
40    // demand analysis
41    this.museumsGroup = this.getL().layerGroup().addTo(this.getMap());
42     ...
43     ...
44
45
46    // SET THE RESULTS GROUPS
47    this.notRecommendedGroup = this.getL().layerGroup().addTo(this.getMap());
48    this.moderatlyRecommendedGroup = this.getL().layerGroup().addTo(this.getMap());
49    this.recommendedGroup = this.getL().layerGroup().addTo(this.getMap());
50    this.highlyRecommendedGroup = this.getL().layerGroup().addTo(this.getMap());
51    this.veryMuchReccomendedGroup = this.getL().layerGroup().addTo(this.getMap());
52
53
54
55    // Define a new Layer from GeoJSON result
56    this.museumGeoJson = this.museumGeoJSONLayer(this.museum);
57     ...
58     ...
59
60    // Create a Clusters from geoJSON Layer
61    this.markersMuseum = this.clustersMarkers(this.museumGeoJson);
62     ...
63     ...
64
65    // Add to map the Cluster Group
66    this.museumsGroup.addLayer(this.markersMuseum);
67     ...
68    // TURF
69    this.museumCollection = turf.featureCollection(this.museum.features);
70
71    var centroid = this.getCentroid(this.istat_information);
72    centroid.forEach((c) => {
73      var res = this.definePOI(c);
74      // Display the points on the map
75      var points = this.getL().geoJSON(res);
76      this.pointsToPredictGroup.addLayer(points);
77      this.features.push(res);
78    });
79
```

```
80      this.sliderDistance();
81
82      // convert Features into FeatureCollection to define a complete GeoJson
83      var featuresCollection = turf.featureCollection(this.features);
84
85      this.getL().easyButton('<img src="assets/img/predict_icon.svg" style="width:34px">', (btn, map) => {
86        // Start the loading spinner
87        this.showSpinner();
88        var toPredict = JSON.stringify(featuresCollection);
89        // Send the value to the server to start the prediction
90        this.predictValue(toPredict);
91      }).addTo(this.getMap());
92
93      // heatmap
94      this.getL().easyButton('<img src="assets/img/icons8-heat-map.svg" style="width:34px">', (btn, map)
            => {
95
96        ...
97      }).addTo(this.getMap());
98
99    }
```

Così come è stato fatto per `MapPotentialComponent` e per le altre mappe, `MapPredictiveComponent` eredita dalla classe padre e richiama i metodi che le sono necessari nell'`init`.

Le prime righe di codice di `setMapChild` si premurano di definire tutti i `LayerGroup` necessari al prosieguo. Fanno parte di questi gruppi i POI per l'analisi predittiva, ma anche i punti da predire e i risultati di tali predizioni: `notRecommendedGroup`, `moderatlyRecommendedGroup`, `recommendedGroup`, `highlyRecommendedGroup`, `veryMuchReccomendedGroup`.

Sono così convertiti in GeoJson, come fatto in `MapPotentialComponent` ed assegnati a variabili con lo stesso nome ma con suffisso diverso (esempio: `museumGeoJson`). Affinchè ora possano essere mostrati sulle mappe, viene passato il risultato a dei clusters methods. Poi ognuno viene aggiunto al corrispettivo gruppo, il quale è già aggiunto alla mappa, ma essendo stato fino ad ora vuoto non ha mostrato nulla.

Inoltre qui viene inizializzato una delle variabili più importanti di tutta la classe: i POI (Point of Interest). Essi racchiudono una serie di Punti, che come si vedrà nel prossimo capitolo, sono stati considerati di interesse al fine di decretare un punto come appetibile o meno. Partono dall'individuazione dei centroidi. Questi ultimi, nella libreria Turf, si rifanno al centro della città. La loro individuazione è di facile riuscita. Basterà passare il GeoJson di cui si sta cercando il centroide alla funzione preposta. In questo caso è quello istat. Ognuno di essi viene ciclato e richiamato il metodo `definePOI()`.

Esso effettua delle operazioni spaziali che mirano all' individuazione delle seguenti caratterisctiche per ogni centroidi, quindi per ogni città:

```
definePOI(targetPoint: turf.helpers.Feature<Point>) {

   var pol = [];
   ...
     if (value.properties.city == targetPoint.properties.city) {
       ...
       ...
       var nearestAeroporto = turf.nearestPoint(targetPoint,
             <turf.helpers.FeatureCollection<Point>>this.aereoCollection);

       var nearestBb = turf.nearestPoint(targetPoint,
             <turf.helpers.FeatureCollection<Point>>this.bandbCollection);
       var nearestMuseum = turf.nearestPoint(targetPoint,
             <turf.helpers.FeatureCollection<Point>>this.museumCollection);
       var nearestRestaurant = turf.nearestPoint(targetPoint,
             <turf.helpers.FeatureCollection<Point>>this.restaurantCollection);
       var nearestBus = turf.nearestPoint(targetPoint,
             <turf.helpers.FeatureCollection<Point>>this.busCollection);

       targetPoint.properties["nome_aeroporto"] = nearestAeroporto.properties.name;
       targetPoint.properties["dist_aeroporto"] = nearestAeroporto.properties.distanceToPoint;
       ...
       ...
     }

   });

   return targetPoint;
 }
```

Il metodo cerca tutti i punti più vicini al centroide passato come parametro, rispetto:

- Aereoporti;
- B&B;
- Musei;
- Ristoranti;
- Fermata Bus

I risultati, sono salvati e aggiunti ad un array con annesso nome di riferimento.
La chiamata `var featuresCollection = turf.featureCollection(this.features);` mette insieme tutte le informazioni raccolte e le raggruppa sotto forma di un GeoJson di tipo `FeaturesCollection`.
A questo punto il tutto è pronto è può essere passato al server per effettuare

la predizione.

Nella parte finale di `setMapChild()`, sono aggiunti i plugin dello slider per determinare l'analisi dell'offerta e dei bottoni. Uno è istanziato per richiamare il server all'avvio della predizione mentre l'altro per la creazione della heatmap.

Tutti i punti di interesse (POI) sono mostrati sulla mappa e il tutto è inizializzato. E' possibile ottenere la predizione del modello semplicemente pigiando il bottone apposito e aspettando l'esito della chiamata al server.

Nel metodo `setFilter()`, metodo ereditato dalla classe padre, vengono gestiti i risultati del modello predittivo.

```
1  setFilter(geoJsonPredicted) {
2      var parsedGeoJson = JSON.parse(geoJsonPredicted);
3
4      // Stop the Spinner
5      this.hideSpinner();
6
7      // Clear the actually point showed into map for prediction
8      this.pointsToPredictGroup.clearLayers();
9
10     // cleat hte buffer
11     this.bufferGroup.clearLayers();
12
13     // clear the group fow avoid the overlapping
14     this.heatmapGroup.clearLayers();
15
16     // clear the heatmap value
17     this.heatmap = [];
18
19     // clear all previusly layers
20     this.notRecommendedGroup.clearLayers();
21     this.moderatlyRecommendedGroup.clearLayers();
22     this.recommendedGroup.clearLayers();
23     this.highlyRecommendedGroup.clearLayers();
24     this.veryMuchReccomendedGroup.clearLayers();
25
26
27     var notRecommended = {
28       radius: 8,
29        fillColor: "#e34a33",
30       color: "black",
31       weight: 1,
32       opacity: 1,
33        fillOpacity: 0.8
34     }
35
36     var moderatlyRecommended = { ... }
37
38     var recommended = { ... }
39
40     var highlyRecommended = { ... }
41
42     var veryMuchReccomended = { ... }
43
44     var L = this.getL();
45     var notRecommendedGeoJson = L.geoJSON(parsedGeoJson,
```

```
46      {
47        onEachFeature: this.definePopup,
48        pointToLayer: (features, latlng) => {
49
50          var prediction = features.properties.prediction;
51          var lat = features.geometry.coordinates[1];
52          var lng = features.geometry.coordinates[0];
53          if (prediction == 0) {                                    // NOT RECOMENDED
54            return L.circleMarker(latlng, notRecommended);
55          }
56        }
57      });
58    var moderatlyRecommendedGeoJson = { ... }
59    var recommendedGeoJson = { ... }
60    var highlyRecommendedGeoJson = { ... }
61    var veryMuchReccomendedGeoJson = { ... }
62
63    this.notRecommendedGroup.addLayer(notRecommendedGeoJson);
64    this.moderatlyRecommendedGroup.addLayer(moderatlyRecommendedGeoJson);
65    this.recommendedGroup.addLayer(recommendedGeoJson);
66    this.highlyRecommendedGroup.addLayer(highlyRecommendedGeoJson);
67    this.veryMuchReccomendedGroup.addLayer(veryMuchReccomendedGeoJson);
68
69    this.createBaseLayerWithResult();
70
71  }
```

Il metodo, a cui ritornerà l'esito dal server, inizialmente parserà il risultato. Dopodichè pulirà i vari layers associati ai diversi gruppi derivanti da predizioni precedenti.
Subito dopo, verranno definite e settate i colori dei diversi markers a seconda dell'esito che potrebbero avere. Per ogni risultato parsato, si leggerà la proprietà `features.properties.prediction` a cui sarà assegnato un valore compreso tra 0 e 4. 0 nel caso in cui non sia un punto raccomandato, 4 se sia un punto altamente raccomandato. A seconda, del valore del punto, è richiamata la variabile che ne sancisce il colore.
E' da notare che per quei punti considerati "raccomandati" , oltre all' assegnazione del colore, è settato un filtro.

```
1  var veryMuchReccomendedGeoJson = L.geoJSON(parsedGeoJson,
2      {
3        ...
4        ...
5        },
6        filter : (features, properties) => {
7          var lat = features.geometry.coordinates[1];
8          var lng = features.geometry.coordinates[0];
9          var ptTemp = turf.point([lng, lat])
10         var c = this.bboxPolygons.find(elem => {
11           return turf.booleanPointInPolygon(ptTemp, elem);
12         })
13         return c == undefined ? true : false
14       }
15     });
```

La funzionalità del filtro è quella di mostrare, sulla base del valore booleano, il punto se e solo se è incluso nel range definito dallo slider. La funzione `sliderDistance`,è inizializzata nell' `setMapChild()` come visto prima.

```
/**
* Create a slider for deleting a point from RecommendedGrop that is present
* in a specific area
*/
sliderDistance() {
var buffer;
var bboxBuffer;
var bufferGeoJson;
var sldDistance = this.getL().control.slider((value) => {
  this.bufferGroup.clearLayers();
  this.bboxPolygons = [];
  this.currentDoorsCollection.features.forEach((d) => {
    // Create a new Buffer based on value of slider
    buffer = turf.buffer(d, value, { units: 'kilometers' });  meters kilometers

    // Convert into GeoJson and add to the Group Buffer
    bufferGeoJson = this.getL().geoJson(buffer)
    this.bufferGroup.addLayer(bufferGeoJson);
    bboxBuffer = turf.bbox(buffer);
    this.bboxPolygons.push(turf.bboxPolygon(bboxBuffer));
  });
}, {
  max: 101,
  min: 1,i
  value: 1,
  step: 10,
  size: '250px',
  orientation: 'horizontal',
  position: 'topleft',//'bottomleft',
  id: 'sliderDistance'
});
sldDistance.addTo(this.getMap());
}
```

La funzione dello slider, in generale, è quella di non mostrare i punti che non appartengono all'area di interesse delimitata dal raggio dello slider. Per ottenere questo risultato, nel metodo `control.slider` che è sempre attivo, viene creato un *buffer* in prossimità delle door attualmente esistenti mostrate sulla mappa. Tale buffer aumenta o diminuisce a seconda dell'interesse dell'utente. Il suo valore viene mostrato costantemente su mappa e poi aggiunto ad un array, lo stesso array adoperato per il filtraggio dei punti "raccomandati" .
In questo modo è implementata l'analisi dell'offerta. Io utilizzatore della dashboard non voglio vedere i punti potenzialmente raccomandati che sono già compresi nelle aree dei punti vendita attualmente esistenti.

Con questo si chiude la sezione riguardante l'implementazione delle mappe e da adesso in poi si darà spazio ad altre componenti ritenute rilevanti ai fini progettuali.

3.3.4.2 NgxChart & Angular Material

Nelle varie tab, eccezion fatta per "*Predictive Analysis Italy*" , le mappe sono accompagnate da diversi grafici o tabelle per dare maggiore granularità alle informazioni esposte dalle mappe. Ad esempio, nella tab delle "Doors" i grafici all'inizio mostrano lo stato di salute generale dell'attività economica delle Doors. Cliccando su una specifica, il focus dell'analisi si sposta esclusivamente su di essa, mantenendo lo stesso contesto di informazione.
I grafici e i contenitori che li contengono, sono creati con la combinazione di due tecnologie diverse che poggiano ambedue su Angular. Esse sono rispettivamente `NgxChart` e `Angular Material`. Sempre nella cartella `/shared` è possibile trovare i moduli che contengono le loro implementazioni. Essendo due tipologie di plugin che lavoro insieme, sono importate entrambe negli stessi moduli. Prendiamo ad esempio, a scopo esplicativo il contenuto della cartella `shared/modules/charts/horizontalBarChart`. E' ovvio che quanto applicato e descritto vale anche per le altre tipologie di grafici come "`line-chart`, `pie-chart` , `pie-chart-advanced` .
Il file `horizontalBarChart.module.ts` si presenta:

```
1 import { NgModule } from '@angular/core';
2 import { CommonModule } from '@angular/common';
3 import { HorizontalBarChartComponent} from './horizontalBarChart.component'
4 @NgModule({
5   declarations: [HorizontalBarChartComponent],
6   imports: [ CommonModule ]
7 })
8 export class HorizontalBarChartModule { }
```

Da qui si è partito arricchendo il modulo con l'aggiunta delle componenti `NgxChartsModule`, e quelle necessarie di Angular Material. Infine, il modulo è stato esportato al fine di essere richiamabile dalle altre parti dell'applicazione.

```
1   ...
2 import { NgxChartsModule } from '@swimlane/ngx-charts';
3 import { MatGridListModule, MatIconModule, MatCardModule} from '@angular/material';
4
5 @NgModule({
```

```
  declarations: [HorizontalBarChartComponent],
  imports: [
    CommonModule,
    NgxChartsModule,
    MatGridListModule,
    MatCardModule,
    MatIconModule
  ],
  exports: [HorizontalBarChartComponent]
})
export class HorizontalBarChartModule { }
```

E' ovvio che tutte le componenti di cui si necessita possono essere aggiunte man mano che la loro utilità si palesa. Angular Material ad esempio, consta di molte parti e sarebbe superflue aggiungerle interamente all'intero progetto. NgxCharts invece non è scomponibile, ma aggiungendolo, si acquisiscono tutti grafici disponibili.

```
<mat-card class="card">
  <mat-card-header>
    <div mat-card-avatar>
        <mat-icon>{{icon}}</mat-icon>
    </div>
    <mat-card-title>{{title}}</mat-card-title>
    <mat-card-subtitle>{{subtitle}}</mat-card-subtitle>
  </mat-card-header>
   <mat-card-content>
    <div id="chart-parent-simple">
        <ngx-charts-bar-horizontal
        class="chart-container"
        [scheme]="colorScheme"
        [results]="chartValue"
        [gradient]="gradient"
        [xAxis]="showXAxis"
        [yAxis]="showYAxis"
        [xAxisLabel]="xAxisLabel"
        [yAxisLabel]="yAxisLabel"
        [legendPosition]="legendPosition"
        (select)="onSelect($event)">
    </ngx-charts-bar-horizontal>
    </div>
  </mat-card-content>
</mat-card>
```

I tag esterni definiscono la Card di Angular Material e sarà l'involucro esterno che conterrà il grafico. I valori, contrassegnati da una duplice parentesi graffa saranno gli input gestiti nel file component. La sintassi del grafico è abbastanza elementare. Cambiando il tipo di tag cambierà il tipo di grafico. Gli attributi e i valori servono a personalizzare il suo layout finale. Il file component si presenta come segue:

```
...
```

```
@Component({
  selector: 'app-horizontalBarChart',
  ...
})

export class HorizontalBarChartComponent implements OnInit, OnChanges {

  @Input() data: any;
  @Input() icon: string;
  @Input() title: string;
  @Input() subtitle: string;

  reload_riginal_data: any[] = [];
  chartValue: any[] = [];
  tempChartValue: any[] = [];
  // Width - height
  view: any[] = [];

  ...
  constructor() { }

  ngOnInit() {

    this.data.forEach((item, i, data) => {
      this.chartValue.push({ name: (item.name), value: (item.value) })
    });
    this.reload_riginal_data = this.chartValue;
    Object.assign(this, this.chartValue);
  }

  ngOnChanges() { }

  /**
   * This function define the data chart data to update. It his called
   * by father for each component
   * @param data {Object} the data to update
   */
  updateDataChart(data){
    // clear array
    this.tempChartValue.length = 0;

    // Fille the temp array
    data.forEach((item, i, data) => {
      this.tempChartValue.push({ name: (item.name), value: (item.value) })
    });


    //  spread operator TypeScript. Remember that all value will be read from @chartValue
    this.chartValue = [... this.tempChartValue]
    Object.assign(this, this.chartValue);
  }


  /**
   * Reload original data when a particular event is trigger.
   * It has called by doors or any parent component
   */
  reloadOriginalData(){
    this.chartValue = [... this.reload_riginal_data]
    Object.assign(this, this.chartValue);
  }
}
```

Le variabili dichiarate sono abbastanza esplicative. All'inizio, nella init ven-

gono ciclati i file di input e assegnati al grafico con la chiamata `Object.assign(this, this.chartValue);`. E' ovvio che per ogni grafico, il metodo di assegnamento valori-grafico è differente e questo risulta essere l'unica variante che distingue le tipologie di grafici.
Attraverso un array, vengono salvati i valori originali. Il motivo è legato al comportamento che la dashboard assumerà quando si selezionerà qualcosa di specifico sulla mappa. I grafici, vengono ricaricati ed aggiornati per mezzo di `updateDataChart(data)`. Ma una volta effettuata la deselezione, è necessario mostrare i dati originali e per questo, caricati i valori di default attraverso il metodo `reloadOriginalData()`.

3.3.4.3 Service Http

I servizi Http non sono situati nella cartella `/shared` . La loro collocazione è da attribuirsi nella cartella `/core`. Tuttavia, il loro grande utilizzo all'interno del progetto, in condivisione fra le varie parti, ha fatto sì che fosse collocato in questa trattazione nella corrente sezione. Anche per i servizi collocati nella cartella `/core`, c'e un solo modulo a cui si fa riferimento. Bypassando volutamente la parte relativa al modulo, si descrive direttamente il file `httpServerRequest.service.ts`. L'estensione del file, fa subito intendere che non si tratta di un file di tipo Component. I Service in Angular hanno una funzione diversa. Accedono ai dati e si premurano di salvarli o eliminarli. Per sottolineare ciò per cui è stata creata, prende il nome `HttpServerRequest`.

```
1 -- Create Service
2 ng generate service  src/app/core/http-request
```

Le operazioni principali che offre, sono semplici richieste GET, o POST al server:

```
1  import { Injectable } from '@angular/core';
2  import { Observable, of } from 'rxjs';
3  import { catchError, map, tap } from 'rxjs/operators';
4  import { HttpClient, HttpHeaders } from '@angular/common/http';
5
6  import { environment } from 'src/environments/environment'
7
8  @Injectable({
9    providedIn: 'root'
10 })
11
```

```
12 /**
13  * Define the API to communicate with Server
14  */
15 export class HttpServerRequest {
16   private _endpoint = environment._URL;
17   constructor(private http: HttpClient) { }
18
19 ...
20 ...
21
22 getDoors(): Observable<any> {
23     return this.http
24       .get(this._endpoint+'/api/doors')
25       .pipe(
26         tap(_ => console.log('fetched doors')),
27         catchError(this.handleError('getDoors')),
28         map(this.extractData)
29       );
30   }
31
32 getDoor(id): Observable<any> {
33     return this.http
34     .get(this._endpoint + '/api/getDoor?kering_door_code=' + id)
35     .pipe(
36       tap(_ => console.log('fetched get Door')),
37       catchError(this.handleError('sellOut')),
38       map(this.extractData)
39     );
40   }
41
42
43 // Send the Geojson to the server to obtains the prediction
44   putToServer(doors): Observable<any> {
45     return this.http
46     // .get(this._endpoint + '/predictive/putPoints?geojson=' + doors)
47     .post(this._endpoint + '/predictive/putPoints', doors)
48     .pipe(
49       tap(_ => console.log('*** fetched put Doors to the server ***')),
50       catchError(this.handleError('sellOut')) ,
51       map(this.extractData)
52     );
53   }
54  ...
55  ...
56 }
```

All'inizio viene importato il decorator `@Injectable`. Esso permette di contrassegnare una classe come disponibile per essere fornita e iniettata alle componenti che lo richiedono. Il suo range di azione è su tutto il progetto, come definito dall'assegnazione `providedIn: 'root'`. Nel costruttore, viene importato HttpClient. Essa offre delle API HTTP client che si basano sull'interfaccia `XMLHttpRequest` esposte dai browser[33]. Il metodo di richiesta dei dati presenta pressoché sempre la stessa struttura. Per operare lavora in concomitanza con `Observable`. In particolare, grazie a questo

[33]In aggiunta offrono funzionalità di testabilità, intercettazione di rich

componente, viene offerta la gestione delle chiamate asincrone. All'interno, il metodo `pipe()` si occupa di combinare più funzioni, in un unica funzione, eseguendo quelle di cui è composta in modo sequenziale. Nei metodi implementati sono le seguenti:

- **tap()**[34] Intercetta tutte le emissioni provenienti dal server restituendo il risultato della funzione, che in questo caso è un messaggio di log; a patto che non si verifichino errori;
- **catchError()**: gestisce gli eventuali errori in maniera "innocua" ossia facendo in modo che l'applicazione continui a funzionare [27];
- **map()**[35] passa ciascun valore di origine attraverso una funzione di trasformazione per ottenere i valori di output corrispondenti. I valori ritornati, saranno mappati e organizzati;

Adesso, si possiedono tutti gli elementi di cui le nostre features necessitano[36]. Si hanno le mappe, per la visualizzazione dei punti vendita o delle informazioni di sell out; si hanno i grafici per verificare il rendimento che essi possiedono; si hanno i metodi per estrarre le informazioni dalla sorgente dati. Il passo seguente, è quello di organizzare e mettere insieme i tasselli costruiti finora, per arrivare ad ottenere un risultato conforme alle aspettative sia a livello di info, sia a livello grafico.

3.3.5 Main Container

Il `MainContainer` è il contenitore che orchestra e mette insieme le componenti viste fino a questo momento. E' realizzato mediante un `div` con delle dimensioni definite, ed occupa la porzione di spazio centrale lasciata libera dalla Navbar e dalla Sidebar.
Il contenuto che ospita non è mai fisso, ma cambia a seconda della tab scelta. Le tab a cui si fa riferimento, si riferiscono alle stesse definite, nella Tab 2.4[37].

[34] https://rxjs-dev.firebaseapp.com/api/operators/tap
[35] https://rxjs-dev.firebaseapp.com/api/operators/map
[36] Sono stati trattati gli elementi essenziali.
[37] Sono situate nella directory "src/app/features" secondo l'organizzazione del progetto vista precedentemente.

Al fine di evitare di essere ridondanti, si raggrupperà le features per macro-categorie e si procederà alla descrizione degli elementi essenziali e univoci che le caratterizzano.
La fase di analisi descrittiva, comprende le seguenti tab: *Doors, Sell out, Country, Region.*

```
# Generate Module and Routing
ng generate module --routing src/app/features/nome_tab

ng generate component src/app/features/nome_tab
```

Inizialmente, la creazione dello scheletro di ogni features è scollegata al resto dell'applicazione. Per ovviare a questo entrano in gioco i due file di routing. Il primo si rifà a `app-routing.module.ts` descritto nella sezione AppModule. Il secondo, da quello generato da shell.
A scopo esemplificativo, prendiamo la prima tab, quella delle `Doors`. I due file menzionati, mostrano rispettivamente quanto segue:

```
const routes: Routes = [
  {
    path: '',
    children:
      [
        ...
        {
          path: 'doors',
          loadChildren: './features/doors/doors.module#DoorsModule'
        },
        ...
        ...
      ]
  }
];
@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
...
```

Nel primo pezzo di codice, viene caricato il modulo relativo al componente di interesse. Il percorso è lo stesso defintio al momento della creazione. La sintassi specifica sia il file che si cerca e sia il nome della classe preceduta dal simbolo di cancelletto. Il caricamento è inserito in un array, il cui attributo è "children" . Cosi facendo, si specifica che non deve cambiare tutta la view della pagina ma solo ciò che è all'interno di essa. Angular capisce quale contenuto cambiare, dalla posizione del tag interno al `div` del

`main-container`.

```
<div class="main-container">
    <router-outlet></router-outlet>
</div>
```

Dal suo punto di vista, ogni file di routing corrispettivo ad una specifica tab, avrà una struttura come questa:

```
...
...

const routes: Routes = [
  {
      path: '',
      component: DoorsComponent
  }
];

@NgModule({
  imports: [RouterModule.forChild(routes)],
  exports: [RouterModule]
})
export class DoorsRoutingModule { }
```

Qui la differenza la fa sia la variabile `routes` che carica la specifica Component e non il Module, ma soprattutto come essa viene importata. In questo caso, è `forChilds()` anzichè `forRoot()`. I due metodi definiscono il contesto di applicazione: il primo ci sta dicendo che deve essere usato quel percorso e quella Component esclusivamente nel modulo corrente; mentre il secondo, deve essere applicato a livello globale [28].
`Doors.module.ts`, è la stessa tipologia di file ormai vista più volte in questo elaborato:

```
import { FlexLayoutModule } from '@angular/flex-layout';
@NgModule({
  declarations: [DoorsComponent],
  imports: [
    ...
    FlexLayoutModule.withConfig({addFlexToParent: false})
  ]
})
export class DoorsModule { }
```

Il *FlexLayout* è una tipologia di organizzazione di contenuti che si contrappone a quella che è più solita usare: Bootstrap. In particolare, *FlexLayout* è la libreria Angular che implementa la proprietà FLEX di CSS. Gli elementi flex possono essere usati per la distribuzione dello spazio in maniera "elasti-

ca" e proporzionale, cercando di occupare gli spazi liberi e allo stesso tempo di evitare la sovrapposizione o le eccedenze tra gli elementi protagonisti[38]. In questo progetto è stata usata settando la proprietà `addFlexToParent`,che di norma sancisce lo stile di direzione tenendo conto della posizione dei componenti genitori, a falsa [29]. Se fosse stata settata a vera, lo spazio sarebbe stato organizzato tenendo conto della direzione della Sidebar e Navbar, occupando tutto il contenuto nella parte alta della finestra.
Nel file html sono raggruppati gli elementi usati per improntare un significato alle tab.

```
<div class="map-container" *ngIf="doors; else loading">
      <div class="mb-20" fxLayout="row" fxLayout.lt-md="column" fxLayout.lt-lg="column" fxFlex
          fxLayoutGap="20px">
        <div fxFlex>
          <mat-expansion-panel class="filter">
              <mat-expansion-panel-header>
                  <mat-panel-title>
                      <mat-icon aria-label="icon">filter_list</mat-icon>
                  </mat-panel-title>
              </mat-expansion-panel-header>
                  <div fxLayout='row'>
                      <div fxFlex>
                          <app-filter [data]="doors" [ filterField ]="'name'"
                                      [placeholderTitle]="'Doors'"
                                      [titleDropDown]="'Name'"
                                      (selectedValue)="filterOnMap($event)">
                          </app-filter>
                      </div>
                                  ... other filters ...
                      </div>
                      <div fxLayout='row'>
                          ... other rows ...
                      </div>
          </mat-expansion-panel>
          </div>
      </div>

      <!-- Map -->
      <div fxLayout='row' class="map-container">
          <div fxFlex>

              <app-map-door [data]="doors" (open_popup)="loadData($event)"
                  (close_popup)="reloadOriginalData()"></app-map-door>
          </div>
      </div>

      <!-- Bar Chart -->
      <div class="mb-20" fxLayout="row" fxLayout.lt-md="column" fxLayout.lt-lg="column" fxFlex
          fxLayoutGap="20px">
          <div fxFlex>
              <div *ngIf="topCustomers">
                  <app-horizontalBarChart [icon]="'bar_chart'" [title]="'Customers'"
                      [subtitle]="'Show the top 5 customers'" [data]="topCustomers">
```

[38]https://developer.mozilla.org/it/docs/Web/CSS/flex

```
                                </app-horizontalBarChart>
                            </div>
                    </div>
                    <!-- Top Doors -->
                    <!-- Top SKU -->
                    <!-- Top Brands -->
            </div>

            <!-- ... other charts  ... -->
```

Tutti gli elementi sono posti sfruttando gli attributi del FlexLayout come ad esempio:

- **fxLayout="row"**: dispone gli elementi interni a livello di righa;
- **fxLayout.lt-md="column"** e **fxLayout.lt-lg="column"**: dispone gli elementi interni a livello di colonna. Usato per il layout da mobile;
- **fxFlex fxLayoutGap="20px"**: definisce la larghezza del padding dai bordi esterni.

Dovendo maneggiare un gran quantitativo di dati, è sovente avere uno scostamento temporale dal momento in cui l'utente carica la pagina a quanto i dati siano disponibili. `*ngIf="doors; else loading"` gestisce questa attesa, mostrando i dati al momento del caricamento, altrimenti uno spinner di attesa. Gli altri elementi della pagina sono organizzati allo stesso modo. L'utilizzo del costrutto Angular `ngif` si combina alla ricezione dei dati da parte del file `Doors.component.ts`.

```
...
// Charts
...
// Map
import { MapDoorComponent } from '../../shared/modules/map/map-door/map-door.component'


@Component({
  selector: 'app-doors',
  templateUrl: './doors.component.html',
  styleUrls: ['./doors.component.scss']
})



export class DoorsComponent implements OnInit {

  // Comunicate with charts
  @ViewChildren(HorizontalBarChartComponent) horizontalBarCharComponents:
        QueryList<HorizontalBarChartComponent>;
  ...
  // Filter value on Map
  @ViewChildren(MapDoorComponent) mapComponent: QueryList<MapDoorComponent>;
```

```
23
24   // Variable Type
25   doors: any;
26
27   // charts
28   topCustomers: any;
29    ...
30   constructor(private serverRequest:HttpServerRequest) { }
31
32   ngOnInit() {
33     // get the result from db
34     this.serverRequest.getDoors().subscribe(res => this.doors = res);
35     this.serverRequest.getTopCustomers().subscribe(res => this.topCustomers = res);
36      ...
37   }
38
39   /**
40    * Pass the value to filter on the map
41    */
42   filterOnMap(event: Event){
43     this.mapComponent.forEach(component => {
44       component.setFilter(event);
45     })
46   }
47
48   loadData(event: Event){
49     // It need to explicitly tell TypeScript the type of the HTMLElement which is your target.
50     // The way to do it is using a generic type to cast it to a proper type:
51     var door = (<any>event.target).feature.properties;
52
53     // Update Bar Chart Component
54     this.horizontalBarCharComponents.forEach(component => {
55         if(component.title == "Customers"){
56          this.serverRequest.getCostumer(door.customer_code).subscribe(res => {
57              component.updateDataChart(res);
58          });
59         }
60         if(component.title == "Doors"){
61           ...
62         }
63          ...
64   }
65
66   /**
67    * Clear the chart and update the value
68    */
69   reloadOriginalData(){
70     // clear bar chart
71     this.horizontalBarCharComponents.forEach(component => {
72       component.reloadOriginalData();
73     });
74
75      ...
76   }
77
78 }
```

L'utilità di questa classe è duplice:

1. effettua le richieste alla sorgente dati. La loro ricezione è gestita asincronicamente grazie *Observable* visto nella sezione "Service Http" [30]

.

2. Orchestra la comunicazione tra gli elementi che compongono la tab.

Andando con ordine, all'inizio vengono importante le Component necessarie. E' da sottolineare che questi imports sono possibili solamente nel caso in cui il modulo importa a sua volta i moduli delle corrispondenti Component. In caso contrario, Angular darà errore. Dichiarata la classe e i metodi del ciclo di vita utilizzati, nel costruttore è richiamato il Service `HttpServerRequest`. Esso, dà accesso ai metodi delle `GET` e delle `POST`. Nella `init` infatti, avviene la chiamata. Se il servizio, non implementasse Observables la chiamata porterebbe all'errore. Infatti, solo nel momento in cui i dati richiesti sono disponibili, avviene l'assegnamento con la variabile dichiarata. La direttiva `ngif` nell' `html` gestisce proprio questa casistica. Mostra gli elementi che sta cercando di caricare solo quando disponibili. Nella parte successiva del codice, i restanti metodi organizzano la comunicazione tra le Component figlie. Normalmente esse non comunicherebbero tra di loro. Angular per fare in modo che la comunicazione possa avvenire mette a disposizione vari costrutti. Sul primo si è già discusso. Esso prevede l'utilizzo di due Decorator, `@Input()` e `@Output()`. Sono utilizzati contemporaneamente. Il primo si premura di inoltrare i dati alle component figlie. Queste ultime, devono devono elaborare una risposta e rimandarla al padre. Proviamo a schematizzare l'interazione tra due componenti [21] :

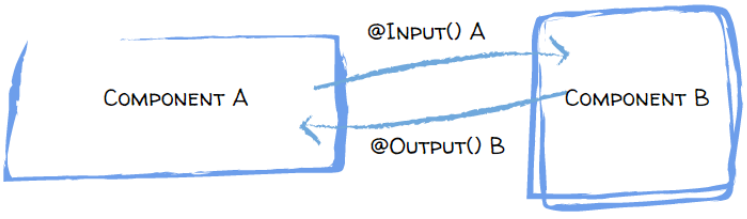


Figura 3.2: Interazione tra due Component utilizzando i Decorator

Se la comunicazione avvenisse solo tra due Component il risultato sarebbe quanto descritto dalla Fig. 3.2.

```
// ComponentA.ts
this.serverRequest.getData().subscribe(res => this.request = res);

// ComponentA.html
<app-component [data]="request" (responseValue)="showResult($event)"> </app-component>

// ComponentB.ts
@Input() data: any;
@Output() responseValue= new EventEmitter();
... do something with data ...
this.responseValue.emit(res);

// ComponentA.ts
showResult(event: Event){
    // ... Show Results ...
  }
```

Il pezzo di codice, rappresenta una semplificazione della comunicazione tra due Component. E' stato semplificato per dare una chiara visione d'insieme a come possa avvenire la comunicazione [31] [32] , ma quanto scritto lo si può trovare anche nella descrizione delle Doors in molteplici occasioni: nella definizione del filtro, nella comunicazione tra la Component padre e la component figlia mappa etc.
Ora però complichiamo il modello della Fig. 3.2 supponendo che ci siano più component che debbano comunicare tra loro a fronte di un input dato dallo stesso padre. Lo schema è:

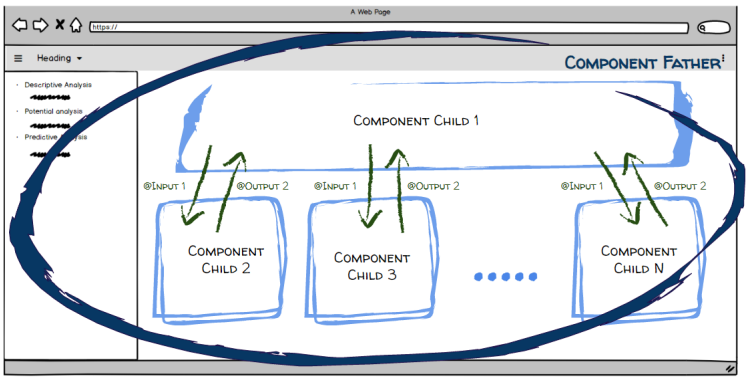


Figura 3.3: Interazione tra multipli Component utilizzando i Decorator

E' evidente che la coordinazione tra le varie Component risulta più ostica

da maneggiare. L'approccio precedente dovrebbe prevedere da parte del padre (coordinatore) l'invio e la ricezione di molte variabili a seconda del numero di componenti figlie:

```
// ComponentFather.ts
this.serverRequest.getData().subscribe(res => this.input1 = res);

// ComponentFather.html
<app-component-child1 [data]="input1" (responseValue)="showResult($event)">
      </app-component-child1>

// ComponentChild1.ts
@Input() data: any;
@Output() output2= new EventEmitter();
... do something with data ...
this.output2.emit(res);


// ComponentFather.ts
showResult(event: Event){
    this.serverRequest.getData().subscribe(res => this.input2 = res);
}

// ComponentFather.html
<app-component-child2 [data]="input2" (responseValue)="showResult($event)">
      </app-component-child2>

// ComponentChild2.ts
@Input() data: any;
@Output() output3= new EventEmitter();
... do something with data ...
this.output3.emit(res);

...   ComponentChild3 ...
...   ComponentChildN ...
```

Per ovviare a questa situazione, Angular utilizza `QueryList` e `ViewChildren` nella Component padre, in questo caso la Door. Ogni qual volta un elemento figlio è aggiunto o rimosso la `QueryList` aggiunge, toglie o aggiorna i valori contenuti in un Array che certifica la presenza dell'elemento nella View. In questo modo,si può avere un maggiore controllo sulle component figlie da parte del padre il quale detiene tutto il flusso di esecuzione. Questo cosa comporta? Torniamo al precedente file `Doors.component.ts`. Qui vengono dichiarate le componenti che la features ospita. Per sincronizzare l'invio e la ricezione dei dati, si prosegue come visto:

- nell'init vengono acquisiti i dati dal server;
- vengono passati tramite metadati alle Component figlie nel file html usando il Decorator `@Input`.

- i dati vengono elaborati e rispediti al padre con il Decorator `@Output`.
- una volta che il controllo passa di nuovo nella mani del padre, per evitare che sia ripetuto il flusso di esecuzione per ognuno dei figli, nel metodo di ricezione, richiama le funzioni di interesse.

Vediamo un flusso pratico di comunicazione:

```
// doors.component.ts
@ViewChildren(HorizontalBarChartComponent) horizontalBarCharComponents:
      QueryList<HorizontalBarChartComponent>;
  ...
@ViewChildren(MapDoorComponent) mapComponent: QueryList<MapDoorComponent>;

<!-- Doors.component.html -->
<app-map-door [data]="doors" (open_popup)="loadData($event)"
                                (close_popup)="reloadOriginalData()"></app-map-door>

// Map component
@Output() open_popup = new EventEmitter<Object>();
@Output() close_popup = new EventEmitter<Object>();

// Function that call the output
openAndEmit = (e) => {
    this.open_popup.emit(e);
 }

// doors.component.ts
loadData(event: Event){
    // It need to explicitly tell TypeScript the type of the HTMLElement which is your target.
    // The way to do it is using a generic type to cast it to a proper type:
    var door = (<any>event.target).feature.properties;

    // Update Bar Chart Component
    this.horizontalBarCharComponents.forEach(component => {
        if(component.title == "Customers"){
         this.serverRequest.getCostumer(door.customer_code).subscribe(res => {
             component.updateDataChart(res);
         });
        }
        if(component.title == "Doors"){
          ...
        }
        ...
  }

  /**
   * Clear the chart and update the value
   */
  reloadOriginalData(){
    // clear bar chart
    this.horizontalBarCharComponents.forEach(component => {
      component.reloadOriginalData();
    });
```

Nella funzione di gestione della risposta, sono chiamati, come detto, i metodi di interesse. Come si può notare ci possono essere più risposte per lo stesso input. Ognuna sarà accompagnata da una funzione specifica. Il

valore che deve essere trasmesso, viene passato come parametro in questo modo la comunicazione è come se avvenisse solo fra due Component e le restanti ne beneficiano incorporando il risultato della comunicazione.
Un metodo alternativo, prevede l'utilizzo dei Service. In questo progetto è usato nelle tab del Potenziale. Nella sezione Leaflet del capitolo corrente si è mostrato il metodo `getColor()` dove si è asserito per l'occasione che venisse prima colorato lo shape e successivamente chiamato un metodo del Service `DynamicPanelService` importandolo nel modulo. Lo scopo della specifica funzione usata è quella di aggiungere il layer ad un array. Tale layer sarà adoperato della features `"Potential Analysis"` e `"Potential Analiysis Italy"`.

```
/**
 * Color the shape of a specif country
 */
colorShape = (e) => {
    var layer = e.target;
    ...
    ...
    this.dynamic_panel.increasePanel(layer);
    }
}
```

Il Service in questione è implementato come segue:

```
import { Injectable } from '@angular/core';


export class Panel {
  tableData: any;
  mapGeoJSON: any;
}

@Injectable({
  providedIn: 'root'
})
export class DynamicPanelService {

  private nPanels: any = [];

  constructor() { }


  increasePanel(panel: Panel) {
    this.nPanels.push(panel);
  }

  decreasePanel(panel: Panel) {
    const index = this.nPanels.indexOf(panel, 0);
    if (index > -1) {
      this.nPanels.splice(index, 1);
    }
  }
  getIndex(panel){
```

```
30      return this.nPanels.indexOf(panel, 0);
31    }
32    getPanels(){
33      return this.nPanels;
34    }
35
36     ...
37
38  }
```

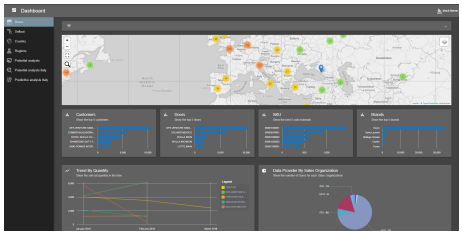
Essendo un servizio, assume gli stessi requisiti assunti dal servizio `HttpServerRequest`. Esso contiene una serie di features che manipolano un array di pannelli (anch'essi importati con Angular Material). Le features che le utilizzano, aggiungono nel costruttore il servizio:

```
1  export class PotentialAnalysisComponent implements OnInit {
2  constructor (...,
3               public dynamicPanel: DynamicPanelService) { }
4  ...
5  }
```

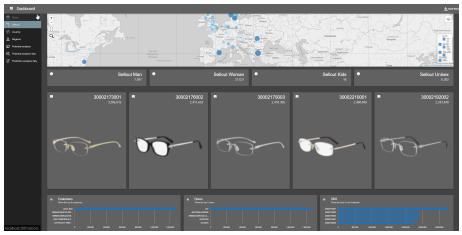
e in un secondo tempo lo adoperano per mostrare i dettagli relativi al paese scelto da mappa:

```
1  <div *ngFor="let panel of dynamicPanel.getPanels() ; let i = index" [attr.data-index]="i">
2      <div class="mb-20" fxLayout="row" fxLayout.lt-md="column" fxLayout.lt-lg="column" fxFlex
           fxLayoutGap="20px">
3        <div fxFlex>
4          <app-mat-card-potential [type]="'world'" [icon]="'table_chart'"
               [title]="panel.feature.properties.name"
5            [subtitle]="''" [data]="panel.feature.properties" [geoJSON]="panel" [index]='i'>
6          </app-mat-card-potential>
7        </div>
8      </div>
9    </div>
10 </div>
```

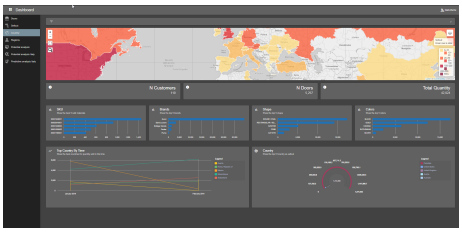
L' `ngFor` cicla i pannelli in ordine di aggiunta e li rimuove a seconda della scelta dell'utente.
Con quanto descritto sono stati coperte tutte le macro aree inerenti l'implementazione del codice. Il risultato finale è mostrato di seguito:



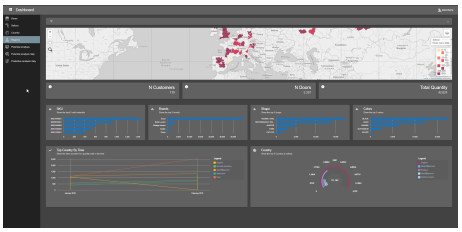
(a) Schermata Doors



(b) Schermata Sell Out



(c) Schermata Country



(d) Schermata Regions

Figura 3.4: Schermate Analisi Descrittiva

Adesso, si sposterà l'attenzione sulle scelte che hanno accompagnato l'implementazione del Server.

3.4 Express.js

In questa sessione saranno analizzate le implementazioni salienti e tutte le scelte che hanno accompagnato la stesura del progetto lato server-side.
`Express.js` è un framework leggero, scritto in `Node.js` che consente di scrivere applicazioni Web server side in Javascript.
Quello che preme sottolineare è la differenza con `Node.js`. La differenza riguarda il livello di astrazione: Node è una piattaforma per scrivere applicazioni server-side event-driven[39], applicazioni input/output usando Javascript. Express, invece, è un framework basato su Node per scrivere Web-application che usa principi e approcci basato su di esso. Questo differenza implica che tutto quello che può essere realizzato in Express può essere rea-

[39]Event-driven sta per "programmazione ad eventi" e rappresenta un paradigma di programmazione. Con questa espressione si intende che il flusso del programma è determinato da eventi esterni. https://it.wikipedia.org/wiki/Programmazione_a_eventi

lizzato in Node, ma significherebbe voler rinunciare a una serie di comode features già definite, oltre che voler riscrivere un qualcosa che già esiste. La definizione data da Express fa intuire che può essere usato interamente per la definizione anche della parte client side. Si possono usare vari template di *view engine* che renderizzano il contenuto del server in una pagina html dinamica. Esistono esempi in rete che spiegano come avvenga [33] [40].
Nel progetto corrente, è stato adoperato per fungere da semplice server tralasciando la parte legata al rendering.

3.4.1 Server Setting

Tutti i file riguardanti la parte server side, sono situati nella folder `/server` . `/App.js` e `/www/bin` sono i file responsabili del suo avvio. Essi si differenziano per i ruoli che svolgono. Partendo a ritroso, il file `/www` richiama il primo, inizializza il server e si mette in ascolto sulla porta principale.

```
1  /**
2   * Module dependencies.
3   */
4
5  var app = require('../app');
6  var debug = require('debug')('prova:server');
7  var http = require('http');
8
9  /**
10  * Get port from environment and store in Express.
11  */
12
13 var port = normalizePort(process.env.PORT || '3000');
14 app.set('port', port);
15
16 /**
17  * Create HTTP server.
18  */
19
20 var server = http.createServer(app);
21 ...
22 ...
```

`/App.js`, invece si premura di configurare tutti i requisiti di cui express necessita:

- importa i package scaricati da `npm` per la corretta configurazione;

[40] A Questo proposito, esistono varie alternative. Una delle quali è rappresentata da Meteor.js. Il modello di architettura adottato è diverso (Guillermo Rauch November 4, 2014).

- setta il server come di tipo Express;
- importa i file di routing che dovranno comunicare con il client gestendo la dimensioni dei corpi delle richieste;
- gestisce gli accessi da localhost;

Tutto questo si traduce nel seguente file:

```
1  var express = require('express');
2  var dotenv = require('dotenv');
3  ...
4
5  var app = express();
6
7  // Rounting of different type of analysis
8  var doors = require('./routes/doors');
9  ...
10 var predictive = require('./routes/predictive');
11
12 ...
13 ...
14
15
16 app.use(express.json({limit: '50mb'}));
17 app.use(express.urlencoded({limit: '50mb'}));
18
19 // support parsing of application/json type post data
20 app.use(bodyParser.json());
21 //support parsing of application/x-www-form-urlencoded post data
22 app.use(bodyParser.urlencoded({ extended: true }));
23 app.use(bodyParser.urlencoded( { limit: 52428800 } )); //50mb
24 ...
25
26 // Managment routing of Query
27 app.use('/', doors);
28 ...
29 ...
30 app.use('/', predictive);
31
32
33 // catch 404 and forward to error handler
34 app.use(function(req, res, next) {
35   next(createError(404));
36 });
37
38 ...
39 ...
40
41 module.exports = app;
```

Una nota che è importante specificare riguarda l'utilizzo del pacchetto `/dotenv`. Questo package consente di definire un file esterno in modalità nascosta che contiene credenziali di qualsivoglia natura. Nel progetto, esso

presenta le credenziali per il Database PostGIS oltre che la porta su cui il server deve mettersi in ascolto.

```
# Define Server Port
PORT=3000

# Database
PG_USER= ...
PG_PASSWORD= ...
PG_HOST= ...
PG_DATABASE= ...

# Client
CLIENT= ...
```

Se un utente decidesse di cambiare le porte di accesso, affinché il server le identifichi come nuove, si necessita dell'aggiornamento del file.
E' evidente come i file mostrati siano esigui. Questo si spiega perché è nei file di routing che viene effettuato il lavoro vero e proprio.

3.4.2 Routing & Views

Nella precedente sezione si è asserito che il file `App.js` contenesse anche i percorsi di routing. Per chiarezza è necessario specificare che contiene le variabili che dichiarano i file di routing. Ad esempio:

```
var doors = require('./routes/doors');
...
app.use('/', doors);
```

Osservando, dal path della variabile, si fa riferimento ad un file `doors.js`
In generale la cartella routes ha al suo interno i file che gestiscono le operazioni di routing. Continuando con l'esempio, `doors.js`, così come tutti gli altri file presenti nella cartella, implementa tutte le `GET` e le `POST` che riguardano gli utenti.
Andando con ordine, soffermiamoci sulle `GET`. A scopo esplicativo consideriamo l'operazione di estrapolazione dei dati dal Database PostGIS mediante una chiamata, che nello specifico ha percorso `api/doors.js` .

```
/* GET the map page */
router.get('/api/doors', function (req, res) {

  // Setup our Postgres Client
  var client = new Client(conString);

  // connect to the client
  client.connect();
```

```
9
10  // Run our Query
11  var query = client.query(new Query(show_doors));
12  query.on("row", function (row, result) {
13    result.addRow(row);
14  });
15
16  // Pass the result to the map page
17  query.on("end", function (result) {
18
19    // Save the JSON as variable data
20    var data = result.rows[0].row_to_json
21    res.json(data);
22    client.end();
23  });
24 });
```

Questa operazione è molto semplice. Per tutte le operazioni GET esiste la medesima gestione. Richiamando dalla barra degli indirizzi, il seguente URL: `http://localhost:3001/api/doors`, viene:

- inizializzato il client;
- effettuata la connessione al Database;
- effettuata la query;
- restituiti i risultati al client, in questo caso il browser, sotto forma di JSON.

La query, è una variabili di tipo Stringa che utilizza la sintassi tipica di PostGIS:

```
1  var show_doors = "SELECT row_to_json(fc)" +
2   "FROM ( " +
3   "SELECT 'FeatureCollection' As type, array_to_json(array_agg(f)) As features " +
4   "FROM ( " +
5   "SELECT 'Feature' As type, ST_AsGeoJSON(lg.geom)::json As geometry," +
6   "(select row_to_json(_) from (select lg.id," +
7   "lg.kering_door_code," +
8    ...
9    ...
10   ...
11  "lg.coordinate," +
12  "lg.latitude," +
13  "lg.longitude) as _) as properties " +
14  "FROM doors As lg " + //LIMIT 100
15  ") As f" +
16  ") As fc";
```

Questa semplice query, è la responsabile della visualizzazione di tutte le door su mappa nella tab "Doors" . Estrapola i dati in formato GeoJSON. Si

faccia ben attenzione al fatto che la query ricalca la struttura tipica del GeoJSON vista nel primo capitolo. La dicitura `ST_AsGeoJSON(lg.geom)::json As geometry` setta la geometria del tipo spaziale. Il resto della query invece assegna le proprietà del GeoJSON risultante.
Per quanto riguarda la connessione al Database, ogni file di routing, contiene nell'intestazione le seguenti righe di codice:

```
var pg_conn = require('./pg_connection');

// Connection from database
const { Client, Query } = pg_conn.connection_obj;
const conString = pg_conn.connection_info;
```

L'import di `/pg_connection` espone agli altri file sia l'URL di connessione al Db e sia l'oggetto Postgres.

```
var dotenv = require('dotenv');
var express = require('express');
// var router = express.Router();
var pg = require('pg');
// pg.defaults.poolSize = 20;
/* PostgreSQL and PostGIS module and connection setup */
const { Client, Query } = pg;


const resultDotenv = dotenv.config()
if (resultDotenv.error) {
  throw resultDotenv.error;
}

// DEBUG
// console.log(resultDotenv.parsed)

////////////// SETUP CONNECTION //////////////

// sandbox username
var username = process.env.PG_USER;

// read only privileges on our table
var password = process.env.PG_PASSWORD;

// host
var host = process.env.PG_HOST;

// database name
var database = process.env.PG_DATABASE;

// Your Database Connection
var conString = "postgres://" + username + ":" + password + "@" + host + "/" + database;



module.exports = {
    connection_info: conString,
    connection_obj: pg
};
```

Esso si rifà al file *.env* descritto pocanzi e viene importato in ognuno dei file di routing che effettuano query al Database Postgres. Tra questi è compreso `predictive.js`, il quale assume le caratteristiche analoghe degli altri file. L'unica variante è rappresentata dal metodo `POST`. A differenza delle altre `GET` soventi in questi tipi di file, questo fa da tramite tra il client e lo script Python che si occupa delle predizioni.

```
const spawn = require("child_process").spawn;
...
...
router.post('/predictive/putPoints', function (req, res) {

    var result = JSON.stringify(req.body).replace(/'/g, " ");
    const pythonProcess = spawn('python',[path.resolve(__dirname, '../script_py/predict_doors.py')]);

    pythonProcess.stdin.write(result);
    pythonProcess.stdin.end();


    pythonProcess.stdout.on('data', (data) => {
        if (!res.headersSent) res.json(data.toString(), 0, 500);
    });

    pythonProcess.stderr.on('data', (data) => {
        console.error(data.toString());
    });

    pythonProcess.on('exit', (code) => {
        console.log(`Child exited with code ${code}`);
    });
```

Node.js è una delle tecnologie di sviluppo web più adottate ma manca di supporto alle librerie di machine learning, deep learning e intelligenza artificiale. `child_process`[41] cerca di sopperire a questa mancanza non direttamente ma facendo da intermediario tra linguaggi diversi tra cui Python. Si è così in grado di implementare qualsivoglia algoritmo di Machine Learning o Deep Learning e poi richiamarlo da Node stesso. Il codice da utilizzare è molto semplice. Dopo aver importato `pythonProcess` nel metodo che lo ospita, si indica il percorso dello script Python[42]. Una volta istanziato il processo, vengono passati i parametri dati da un Json proveniente dal Client. Il Json contiene le informazioni utili per la predizione e rappresenta la totalità dei punti di interesse con annesse le distanze dai POI e integrato con dei dati Istat sulla popolazione. Una volta che il file Python li ha elaborati, li rimanda al `child_process`, il quale, nella funzione preposta, li

[41] https://nodejs.org/api/child_process.html# child_process_child_process

[42] Il contenuto dello script sarà descritto minuziosamente nel prossimo capitolo.

rinvia al client tramite JSON. Il file d'origine, si discosterà da quello finale per un attributo. Tale attributo ha valore `prediction`. Alla fine una volta che tutto è stato inviato viene chiusa la connessione.
Con quanto detto, si chiude questo capitolo in cui si è argomentato riguardo l'implementazione del codice. Nel prossimo capitolo, si darà spazio alle analisi effettuate sui dati e ci si soffermerà sui risultati prodotti.

Capitolo 4

Validazione dei risultati

A questo punto della trattazione, la dashboard è ultimata. Adesso, si posseggono tutti i moduli e le componenti per mostrare i dati. Il database è impostato e le query funzionanti. A livello implementativo il progetto è completo e i dati sono visualizzabili sia sulle mappe che sui grafici, il tutto con un layout in linea con le richieste del cliente.
Tutto in questo progetto ha ruotato intorno ai dati, ma finora non si è dato il giusto spazio al "significato" che assumono. Quest'ultimo capitolo, per certi aspetti è il più importante per il lavoro di tesi svolto. I protagonisti saranno i due tipi di analisi *l'analisi del potenziale* e *l'analisi predittiva*. Il primo, ha come obiettivo l'individuazione del rapporto esistente tra il fatturato, con le caratteristiche dei prodotti venduti più una serie di dati demografici, cercando di intravedere se esista una crescita che sia correlata e proporzionale ad esso; il secondo invece punta alla scoperta di nuovi punti da posizionare sul territorio italiano partendo da una valutazione di quelli esistenti, secondo un modello predittivo realizzato con tecniche di Machine Learning.

4.1 Analisi del potenziale

L'aumento del fatturato è l'obiettivo perseguito da tutte le imprese, sulla cui base viene data valenza agli investimenti passati e futuri. Tuttavia, è spesso difficoltoso individuare quali sono le componenti che incidono sulla

crescita o sulla decrescita del sistema economico aziendale.
Un motivo che spiega le difficoltà in tal senso è l'individuazione dei fattori che tra i tanti, hanno maggiore impatto sulle performance. Quando un'azienda dispone di molte informazioni riguardanti la vendita di un prodotto, si cela l'importanza che possono assumere sul totale. E' per questo che è nata la BI, come si è discusso nel primo capitolo, proprio per dare un senso al valore dei dati. Se in aggiunta a queste informazioni, si considerano anche quelle "esterne" , ossia quei dati che non sono strettamente correlati alle vendite tutto diventa più arduo.
Il tentativo di questo capitolo è cercare di quantificare e stabilire l'apporto che ogni fattore arreca, o per lo meno cercare quelli che per tale scopo, sono da considerarsi rilevanti a fronte di altri. Questa certificazione, passa per una serie di fasi. La prima è la raccolta dei dati di interesse. La seconda invece è la loro analisi in funzione di una variabile di riferimento, in questo caso rappresentata, ovviamente, dal dato di vendita e l'ultimo è la visualizzazione degli output secondo dei criteri. Ciò che ci si aspetta da questa analisi, è il saper giustificare le performance (di una data nazione o regione) rispetto i ricavi ad essi correlati. Ancor più importante, sarebbe cercare di carpire se le performance, seppur alte siano al massimo del potenziale. In parole più spicciole, questa situazione potrebbe presentarsi se il fatturato in un dato periodo storico o in una specifica area in esame, presenti dei valori di vendita (ad esempio il numero di prodotti da uomo), al di sotto delle aspettative nonostante il fatturato elevato e le componenti demografiche congeniali.

4.1.1 Raccolta dei Dati

Il passo iniziale per l'avvio del lavoro di analisi del potenziale, è, come il titolo del paragrafo esplicita, la raccolta dei dati. I dati presi in esame e oggetto di studio sono stati di due "tipologie" diverse. I primi sono strettamente legati ai ricavi, e per convenzione chiameremo interni; i secondi, invece, sono estranei alle dinamiche di vendita, ma sono per lo più riguardanti il territorio, e per questo, li chiameremo "esterni".
I dati "interni" considerati, non sono altro che le colonne del Database della tabella di sell out. Comprendono:

- Numero vendite per uomo, donna, bambino;
- Numero di vendite per ogni marchio;
- Numero di vendite per i diversi colori;
- Numero di vendite per tipologia di forma del prodotto;

I dati esterni, presi da organi statistici sono:

- Stranieri residenti;
- Popolazione Residente al primo gennaio;
- Redditi IRPEF su base comunale;
- Numero di Convivenze

La gestione delle due tipologie di dati, è diversa. Infatti per i primi, già disponibili, è bastato estrarli dal Database[1].

city	material_gender_name	material_brand_name	material_front_main_color_name	material_shape_name
BARI	Woman	Bottega Veneta	GREY	ROUND / OVAL
ACIREALE	Woman	Gucci	GOLD	AVIATOR
BOLOGNA	Woman	Cartier	BLACK	ROUND / OVAL
BOLOGNA	Man	Bottega Veneta	HAVANA	ROUND / OVAL
...	...	...	...	...

Tabella 4.1: Esempio di Sell out estratti dal DB per analisi del Potenziale

Partendo dalle informazioni poste in colonna, la seguente query, manipola i dati conferendogli la struttura desiderata.

```
1  select lg.city,
2  -- Material Gender Name
3  SUM(CASE WHEN so.material_gender_name = 'Man' THEN so.sell_out_quantity ELSE 0 END) AS 'N
       man',
4  SUM(CASE WHEN so.material_gender_name = 'Woman' THEN so.sell_out_quantity ELSE 0 END) AS "N
       woman",
5  SUM(CASE WHEN so.material_gender_name = 'Kid' THEN so.sell_out_quantity ELSE 0 END) AS 'N kid',
6  SUM(CASE WHEN so.material_gender_name = 'Unisex' THEN so.sell_out_quantity ELSE 0 END) AS 'N
       Unisex',
7  -- Material Brand Name
8  SUM(CASE WHEN so.material_brand_name = 'Balenciaga' THEN so.sell_out_quantity ELSE 0 END) AS '
       N Balenciaga',
9  ...
10 ...
11 -- Material Color Main
```

[1] In questi esempi, si fa riferimento a città italiane, ma è ovvio che il tutto si applica anche a città straniere.

```
12 SUM(CASE WHEN so.material_front_main_color_name = 'BLACK' THEN so.sell_out_quantity ELSE 0
       END) AS 'N BLACK',
13 ...
14 -- Shape
15 SUM(CASE WHEN so.material_shape_name = 'RECTANGULAR / SQUARE' THEN so.sell_out_quantity
       ELSE 0 END) AS 'N RECTANGULAR / SQUARE',
16 ...
17 -- Value
18 SUM(so.sell_out_quantity) as quantity,
19 SUM(so.estimated_sell_out_value_based_on_srp_euro5) as net_sell_out_value_euro_act,
20 (SUM(so.estimated_sell_out_value_based_on_srp_euro5)/SUM(so.sell_out_quantity )) as sellout_update
21 FROM doors As lg INNER JOIN sellout as so ON lg.kering_door_code = so.Internal_Door_Code
22 WHERE so.operation_type_name = 'Sell Out' and lg.country = 'Italy (IT)' and lg.city != 'null'  GROUP
       BY lg.city
```

Con quanto scritto, si è estratto il numero di vendite per tipologia di genere, materiale, brand, colore e forma. Adesso, i dati di partenza, visti in precedenza, hanno la seguente forma trasposta:

city	N man	...	N Azzedine Alaïa	...	N BLACK	...	N RECTANGULAR / SQUARE	...
Acireale	3	9	0	0	11	0	3	10
Alessandria	13	29	0	0	22	1	20	17
Bari	20	66	0	4	35	4	57	22
Bologna	732	1053	0	0	1190	73	938	791
...	20	26	0	0	25	5	23	22

Tabella 4.2: Estrazione dei dati in forma trasposta

Per l'analisi del potenziale globale, la raccolta dei dati è terminata. Invece per quella italiana, sono stati incrementati con altrettanti dati ottenuti da organi statistici. Ad ognuna delle città italiane individuate dalla query, è stata aggiunto il valore statistico di riferimento. E' ovvio che entrambe i tipi di analisi, sono correlati a valori di sell out.

city	quantity	net_sell_out_value_euro_act	sellout
Acireale	15	4985	55887.33.00
Alessandria	49	13170	129519.42.00
Bari	108	35510	133044.03.00
Bologna	2129	541105	26792.17.00
...	55	11589	118391.49.00

Tabella 4.3: Valori Sell out

Alla fine entrambe le tipologie, sono state aggregate tra loro e poste all'interno di un file Excel. Da qui in poi si passerà alla fase di analisi.

4.1.2 Analisi

L'analisi dei dati è stata effettuata usando come linguaggio `R` e come IDE di supporto `R Studio`. `R` è un linguaggio per l'elaborazione statistica. E' disponibile per varie piattaforme Unix, GNU/Linux, macOS, Microsoft Windows ed è sotto licenza GNU GPL, quindi lo si annovera tra i software *open source*. Offre un'ampia varietà di tecniche statistiche ed è integrabile anche con sistemi e Database GIS, come ad esempio GRASS GIS. Per questa trattazione è stata usata una tecnica di *regressione lineare*.
La *regressione lineare* è una tecnica adoperata per prevedere il valore di una variabile Y sulla base dei valori assunti da una o più input X. L'obiettivo di fondo è stabilire in che grado le variabili di input X influenzino il valore della variabile Y [34] .
Un esempio esplicativo, può essere quello che prevede l'utilizzo di una sola variabile semplice X e una variabile dipendente Y. Rispettivamente esse possono essere la velocità e la distanza. Com'è la variazione dell'una al variare dell'altra? Il rapporto sarà sicuramente lineare e decrescente, ossia all'aumento della velocità diminuirà la distanza. Questo implica correlazione tra le due. Ovviamente l'esempio è molto intuitivo, ma nella maggior parte dei casi, come il nostro, l'influenza assunta da una variabile (o carattere) sulle altre non è sempre intuibile ad occhio, per questo occorre un'analisi più approfondita.

$$Y = a + bX + \epsilon \tag{4.1}$$

La (1) esprime la formula matematica della regressione:

- Y è la variabile dipendente;
- X la variabile indipendente;
- a rappresenta l'*intercetta*[2] con l'asse Y;
- b indica la pendenza della retta;
- ϵ è il termine di errore, la parte di Y che il modello di regressione non è in grado di spiegare.

[2] è il valore espresso tra la retta e l'asse di riferimento

Graficamente si presenta:

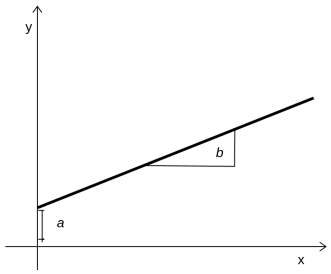


Figura 4.1: Reppresentazione della regressione lineare

Le variabili a e b dette anche stime dei coefficienti di regressione sono calcolate automaticamente da R sulla base dei valori assunti da X e Y[3]. Graficamente, la retta stima l'andamento dei punti.

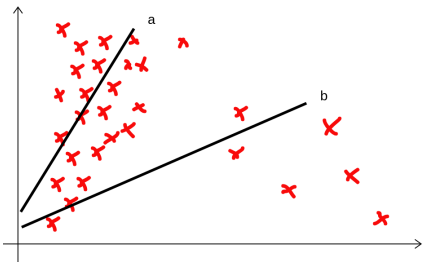


Figura 4.2: Esempio grafico di regressione lineare

Più la retta sintetizza al meglio quanto detto, decretato dalla presenza dei punti lungo la sua direzione, e più si può asserire che il modello è statisticamente significativo. Nell'immagine vi è differenza fra la retta a e b. La prima, esprime una statistica migliore di quanto faccia la seconda. Per avere informazioni più dettagliate sulla regressione lineare si rimanda il lettore a letture più approfondite [35] . La descrizione della logica della regressione lineare, considera solo due variabili. Più precisamente, considera solo una variabile indipendente X. Normalmente, negli studi che la adoperano, le variabili in gioco, sono più d'una, così come nel nostro caso. E' logico quindi

[3] il "Metodo dei minimi quadrati" è un metodo di solito adoperato per calcolare le stime

pensare che una rappresentazione solo grafica, non chiarisca il modello che esprime. Generalmente, quando si considerano un gran numero di caratteri, R valuta delle metriche sulla base delle osservazioni passate come input:

- **t-value**: esprime un valore il quale più è alto e più il coefficiente calcolato non è diverso da zero per puro caso;
- **p-value**: è un valore che esprime la probabilità di ottenere un *t-value* uguale o "più estremo" di quello osservato supponendo l'esistenza dell'ipotesi nulla. Il valore di riferimento è 0.05. Per cui per valori uguali o inferiore alla soglia, la variabile è da considerarsi influente; al contrario per valori più alti della soglia, no, perché l'esistenza dell'ipotesi nulla è falsa e quindi il valore non è attendibile;
- **R-squared**: esplicita in che misura la variazione di una variabile spiega la variazione di una seconda. Per cui generalizzando il tutto, l $\mathbf{R^2}$ esprime di quanto i risultati ottenuti siano variabili. Più è tendente ad uno e più è indice di buon modello;
- **Standard Error**: indica se le variabili indipendenti X adottate costituiscono un buon modello.

Con queste basi è possibile passare all'implementazione in R. I due tipi di analisi, sia quella globale che quella italiana, sono pressoché simili ma variano per alcuni aspetti. Partendo dalla prima, all'inizio dello script vengono importate le librerie e caricato il file Excel che ospita le variabili raccolte in un Dataframe:

```
1 library(readxl)
2 library(xlsx)
3 library(MASS)
4 library(corrplot)
5
6 analysis_sell_db_world <- as.data.frame(read_excel("MYDATA.csv", sheet = "db_world", range =
      "A1:AY131"))
7
8 # Clear Data from Character column and exclude a specific column
9 analysis_sell_db_world <- analysis_sell_db_world[sapply(analysis_sell_db_world, is.numeric)]
```

Come si può notare il dataframe in oggetto viene pulito dalle colonne che non contengono dati numerici. Successivamente, viene applicata la formula `lm`. Essa, è usata per creare il modello. Prende in input due parametri:

1. Il primo definisce la formula per la regressione. Si specifica quale colonna dovrà essere la variabile Y e dopo la tilde specificate le colonne che fungeranno da variabili X. In questo caso, le vogliamo tutte, per cui si utilizza il punto per segnalare ciò all'interprete di R.

2. definisce il dataset di riferimento.

```
fullModelBDWorld <- lm(`sellout`~ .,data=analysis_sell_db_world)
print(fullModelBDWorld)

Call:
lm(formula = sellout ~ ., data = analysis_sell_db_world)
Coefficients:
                  (Intercept)              `N man`            `N woman`             `N kid`       ...
                    2.588e+02            2.005e-02            1.977e-02            5.303e-01       ...
```

La `print` ci da una serie di informazioni. Prendendo in considerazione il grafico, ci indica il valore dell'intercetta e per ognuna delle variabili, definisce i coefficienti delle singole rette. Per cui potenzialmente per N variabili, avremo N rette.

```
 summary(fullModelBDWorld)

Call:
lm(formula = sellout ~ ., data = analysis_sell_db_world)

Residuals:
    Min      1Q  Median      3Q     Max
-92.065  -8.493   0.000  12.639  59.890

Coefficients: (5 not defined because of singularities)
                                  Estimate Std. Error t value Pr(>|t|)
(Intercept)                      2.588e-02  3.718e+00  69.616  < 2e-16 ***
`N man`                          2.005e-02  9.620e+01  -2.084 0.040177 *
`N woman`                        1.977e-02  9.561e+01  -2.068 0.041710 *
`N kid`                          5.303e-01  1.015e+03   5.223 1.24e-06 ***
`N Unisex`                       1.917e-02  9.690e+01  -1.979 0.051106 .
`N Alexander McQueen`            2.287e-05  4.828e+01  -0.474 0.636975
`N Azzedine Alaia`               4.123e-02  1.627e+02  -2.535 0.013081 *
`N Puma`                         1.205e-01  2.293e+02  -5.255 1.08e-06 ***
`N Stella McCartney`             2.813e-03  1.124e+03  -2.504 0.014189 *
`N BROWN`                        9.342e-02  2.389e+02   3.911 0.000184 ***
`N BURGUNDY`                     5.494e-03  2.722e+01   0.202 0.840551
`N CRYSTAL`                      7.689e-04  1.482e+03   5.188 1.43e-06 ***
`N FUCHSIA`                      2.785e-01  2.526e+01  -1.103 0.273202
`N ORANGE`                       1.307e-01  2.470e+03  -5.291 9.33e-07 ***
`N PINK`                         3.059e-01  5.532e+01  -5.530 3.46e-07 ***
`N YELLOW`                       4.673e-02  1.705e+01  -2.740 0.007476 **
net_sell_out_value_euro_act 4.907e-02 1.315e-02 3.731 0.000343 ***
 ...
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 30.2 on 85 degrees of freedom
Multiple R-squared: 0.7518,     Adjusted R-squared: 0.6233
F-statistic:  5.85 on 44 and 85 DF, p-value: 1.913e-12
```

Abbiamo così ottenuto il nostro modello. Passiamo a fare un paio di considerazioni. La chiamata *summary* da una panoramica del risultato. A livello globale, il modello è positivo,per via del valore espresso da R^2 che è uguale a 0,75. Come già indicato in precedenza più alto è il valore, più alta è l'attendibilità. Ovviamente non sono mostrate tutte le variabili in gioco, ma sono mostrate quelle che apportano valore al sell out. Lo si capisce dagli asterischi posti alla destra che rappresentano il grado di influenza di ognuna. Il modo in cui le variabili incidono sulla Y è dato dal segno del valore della colonna *Estimate*. Per esempio, consideriamo il precettore "*N kid*" che corrisponde alla vendita di prodotti per bambini. Si può notare come sia altamente correlata alla dipendente. Il suo valore incide positivamente sul fatturato. Si può per cui constatare che per ogni unità da bambino venduta, ci sia un incremento del fatturato pari a 0.503. Analogamente, per ogni unità da donna vendute, ci sia un aumento del 0.0197.
E' possibile verificare la validità del modello ? Esistono vari criteri. Le più semplici sono l'AIC e il BIC.

```
1 AIC(fullModelBDWorld) # AIC => 591.75
2 BIC(fullModelBDWorld) # BIC => 424.892
```

L'*AIC* (criterio di informazione dell'Akaike) e il *BIC* (criterio di informazione bayesiana) sono misure della bontà di adattamento di un modello statistico stimato e possono anche essere usati per selezionare il modello. Entrambi i criteri dipendono dal valore massimizzato della funzione di probabilità per il modello stimato. Il primo considera il numero di parametri, il secondo invece considera la dimensione del campione. Il modello è considerato valido perché i valori risultanti dal calcolo, sono bassi e di conseguenza tutte le variabili si prestano bene a questo studio.
Riguardo il secondo tipo di analisi, quella italiana, l'approccio adottato è pressocchè simile, sebbene ci siano aspetti divergenti. Tale divergenza è frutto del risultato nullo ottenuto con l'ausilio di tutte le variabili come operato per quella globale. Il motivo,deriva dal numero di osservazioni esiguo rispetto ai precettori in gioco. Per questa ragione, si è dovuto individuare quali variabili prendere in esame. Prendiamo in considerazione la *matrice di correlazione*.

```
# load package ...
# stepwise regression  library
library(caret)

analysis_sell_db <- as.data.frame(read_excel("MYDATA.csv", sheet = "db_italy", range = "A1:BJ43"))

# Clear Data from Charachter column and exclude a specific column
analysis_sell_db <- analysis_sell_db[sapply(analysis_sell_db, is.numeric)]

# We calculate the correlation for the Total Model
correlation_pearson <- round(cor(analysis_sell_db, use="pairwise.complete.obs", method="pearson"),2)
```

Dopo aver caricato i dati e puliti omettendo le variabili non numeriche, è stata calcolata la *matrice*[4]. Quest'ultima, indica la relazione esistente tra le variabili disposte in forma matriciale. Tale "relazione" rappresenta l'andamento dell'una in funzione dell'altra e viene quantificata con un valore compreso tra -1 (perfetta correlazione negativa, le variabili si muovono in maniera opposta) e 1 (perfetta correlazione positiva,le variabili seguono lo stesso andamento). In caso di mancata relazione, il valore sarà uguale a 0.

	N man	**N woman**	**N kid**	**N Alexander McQueen**	**N McQ**	**sellout**
N man	1.00	0.99	-0.08	-0.02	0.66	0.10
N woman	0.99	1.00	-0.08	-0.01	0.02	0.88
N kid	-0.08	-0.08	1.00	0.33	0.44	0.63
N Alexander McQueen	-0.02	-0.01	0.33	1.00	0.66	0.28
N McQ	0.66	0.02	0.44	0.66	1.00	0.35
sellout	0.10	0.88	0.63	0.28	0.35	1.00

Tabella 4.4: Matrice di correlazione

l valori disposti sulla matrice sono simmetrici perché il valore assunto dalla variabile X_{ij} è uguale al valore assunto dalla stessa nella posizione X_{ji}. Per lo stesso motivo, sulla diagonale ci sono tutti 1. Per ovvie ragioni, la matrice di correlazione mostrata è solo uno scorcio di quella originale. A questo punto, è possibile vedere, come la variable "sell out" su cui è basata la statistica, sia correlata con la variabile "N woman" e "N kid" ma non con la le altre.

```
Coefficients: (1 not defined because of singularities)
                        Estimate Std. Error t value Pr(>|t|)
`N man`                 7.933608   4.075708  -1.947 0.065772 .
`N woman`              10.033887   2.609432  -3.845 0.001010 **
`N kid`                27.215638  10.582042  -2.572 0.018199 *
`N Alexander McQueen`  48.242856  26.313837  -1.833 0.081676 .
`N McQ`                21.458311  10.745675   1.997 0.059624 .
```

[4]https://en.wikipedia.org/wiki/Correlation_and_dependence#Correlation_matrices

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Apparentemente si potrebbe pensare ad eliminarle dall'insieme dei precettori utilizzati in quanto non utili, ma questo tipo di approccio è sbagliato. Dalla regressione fatta su questo sottoinsieme di variabili il risultato cambia. Togliendo infatti le variabili "N man" , "N Alexander McQueen" , "N McQ" , ne consegue:

```
Coefficients :
                          Estimate Std. Error t value Pr(>|t|)
(Intercept)              59.861989 118.761030   0.504   0.61923
`N woman`                 3.032433   3.322420  -0.913   0.37129
`N kid`                  24.868593  11.800997  -2.107   0.04672 *
```

In generale le variabili tendono ad influenzarsi a vicenda. Ciò comporta che una variabile apparentemente "innocua" non incida direttamente sulla variabile Y ma influenzi l'andamento delle altre. Questo è possibile vederlo anche dalla matrice stessa focalizzando l'attenzione sulla relazione esistente tra due variabili X.
Come fare una scelta ponderata in questi casi ? Un aiuto ci viene incontro dalla *stepwise regression*. Essa esprime un metodo statistico pensato per individuare un sottoinsieme di attributi che meglio si adattano allo variabile dipendente da analizzare [36] [37] . Esistono anche altre tecniche per ottenere il medesimo risultato tra le tante: Adj R-Squared[5], Akaike information criterion[6],PCR[7].
In genere in R è sovente usare questo tipi di tecnica e per questa ragione è stata adoperata anche in questa casistica. Esistono 3 tipi di stepwise:

- **Forward selection:** parte da un insieme vuoto e iterativamente aggiunge il carattere con il grado di influenza elevato. Si stoppa nel momento in cui non ci sono più variabili influenti;

- **Backward selection:** al contrario del Forward, parte dall'insieme pieno e elimina iterativamente le variabili che meno contribuiscono alla spiegazione della variabile Y;

[5] https://en.wikipedia.org/wiki/Coefficient_of_determination# Adjusted_R2
[6] https://en.wikipedia.org/wiki/Akaike_information_criterion
[7] https://en.wikipedia.org/wiki/Principal_component_regression

- **Stepwise selection:** è un mix tra il Forward e lo Stepwise. Parte dall'insieme vuoto dei caratteri e man mano lo aggiunge con lo stesso criterio adottato dalla Forward. Per ogni nuovo carattere aggiunto, rimuove eventuali variabili che non forniscono più un miglioramento nell'adattamento del modello.

Queste tre tipi di tecniche non possono essere adottati in maniera indiscriminata. La prima e l'ultima tecnica si addicono a situazioni in cui il numero di variabili è maggiore delle osservazioni, la seconda invece si addice per trovare il miglior set quando ci sono un numero maggiore di osservazioni rispetto le variabili.

```
# Set seed for reproducibility
set.seed(1000)
# Set up repeated k-fold cross-validation
train.control <- trainControl(method = "repeatedcv", number = 20)
# Train the model
step.model <- train(`sellout` ~., data = analysis_sell_db,
                    method = "leapSeq", #leapForward
                    tuneGrid = data.frame(nvmax = 30:35),
                    trControl = train.control)
```

All'inizio dello script viene settato il *seed* come è tipico fare in questi casi per riprodurre il set di dati random. Successivamente viene settato il *trainControl*[8]. L'obiettivo di questa funzione è quella successiva è di allenare il modello per scoprire le variabili potenzialmente utili. In particolare, nella prima è definito la suddivisione del dataset in train e test set e settato il numero di incroci da effettuare ad ogni iterazione per il cross-validation[9]; nella seconda, indicata la variabili Y; preso come parametro il dataframe; stabilita la tipologia di stepwise. In questa occasione è usato il "*leapSeq*" che corrisponde alla terza opzione, Stepwise selection. Le altre disponibili sono: "*leapForward*" e "*leapBackward*" che indicano rispettivamente *Forward* e *Backward* selection. Il parametro *nmax* specifica il numero di modelli massimi ottenibili che esprimono al meglio la selezione.
Per visualizzare il risultato ottenuto, eseguire il seguente comando:

[8] Per avere maggiori informazioni sui metodi di R basta digitare dalla console ?nomeMetodo i.e ?trainControl.

[9] è una delle varie tecniche di convalida del modello simile per valutare come i risultati di un'analisi statistica si generalizzeranno a un set di dati indipendente. https://en.wikipedia.org/wiki/Cross-validation_(statistics)

```
step.model$results

  nvmax     RMSE Rsquared      MAE RMSESD RsquaredSD MAESD
1     30 295.7201 0.9497052 243.3112 267.8590 0.2235086 216.8095
2     31 269.4639 0.9516680 228.4196 255.5376 0.2137591 210.9130
3     32 396.5677 0.9591939 324.6625 553.6945 0.1822778 425.0775
4     33 353.8408 0.9726065 291.3332 565.6378 0.1224425 439.6518
5     34 286.7121 0.9637961 236.8060 481.1102 0.1586500 375.0169
6     35 286.7121 0.9637961 236.8060 481.1102 0.1586500 375.0169
```

L'output, mostra una serie di metriche al fine di poterle comparare. Ognuno si rifà al modello di pertinenza tenendo in considerazione le variabili adottate. Le metriche per la valutazione sono:

- **nvmax**: indica il numero di variabili adoperate per elargire quel modello. Ad esempio, nvmax = 30, specifica il miglior modello a 30 variabili
- **RMSE** e **MAE** sono due metriche usate per prevedere l'errore previsto da ciascun modello. Più i valori corrispondenti ai corrispettivi *nvmax* sono bassi e più il modello è migliore.
- **R-squared** è lo stesso indicatore visto in precedenza. In questo caso, la variabilità dei risultati si contrappone a quella dei valori previsti. Ancora un volta, maggiore è il valore, migliore è il modello.

Per capire quale set di variabili sia meglio usare:

```
step.model$bestTune

# nvmax
# 2     31
```

Il comando estrae la migliore selezione dal set ottenuto in maniera automatica. Per conoscere a quali variabili fa riferimento, si utilizza il comando:

```
coef(step.model$finalModel, 31)

 (Intercept)              `N Alexander McQueen` `Popolazione al 31 dicembre - Totale`    ...        `Numero di Famiglie`

-1.201608e+03              -1.290590e+01          1.456630e-02                -3.425283e-02
```

Ponendole nella funzione `lm()` e lanciando il comando, si ottiene:

```
fullModelBD <- lm(`sellout`~
                    `N Alexander McQueen` +
                    `N Azzedine Alaia +
```

```
                    `N Christopher Kane`+
                    `N McQ`+
                    `N BLACK` +
                    `N BLUE` +
                              ...

                  ,data=analysis_sell_db)
summary(fullModelBD)

Residuals:
    Min      1Q  Median      3Q     Max
−51.450 −5.987 −0.313   7.333  37.210

Coefficients :
                                          Estimate Std. Error t value Pr(>|t|)
(Intercept)                               1.106e+03 4.136e+02 −2.675  0.03179 *
`N Alexander McQueen`                     1.312e−02 5.976e+01  2.195  0.06424 .
`N Christopher Kane`                      1.267e−02 5.319e+01 −2.382  0.04873 *
`N BLACK`                                 8.076e−01 2.173e+01 −3.716  0.00749 **
`N BLUE`                                  1.002e−02 2.749e+01 −3.644  0.00825 **
`N BROWN`                                 7.524e−01 8.405e+01 −0.895  0.40039
`N BURGUNDY`                              7.052e−01 2.028e+01 −3.478  0.01030 *
`N CRYSTAL`                               5.960e−02 1.845e+02 −3.230  0.01445 *
`N FUCHSIA`                               2.484e−02 9.511e+01  2.612  0.03481 *
`N GOLD`                                  9.932e−01 2.735e+01 −3.631  0.00838 **
`N GREEN`                                 1.019e−02 2.473e+01 −4.120  0.00446 **
`N GREY`                                  7.417e−01 2.754e+01 −2.693  0.03096 *
`N HAVANA`                                1.076e−02 2.433e+01 −4.423  0.00307 **
`N LIGHT−BLUE`                            1.502e−02 5.162e+01 −2.910  0.02264 *
`N MULTICOLOR`                            1.935e−02 7.246e+01 −2.670  0.03200 *
`N NUDE`                                  2.759e−02 7.428e+01 −3.715  0.00751 **
`N PINK`                                  6.797e−01 3.203e+01 −2.122  0.07151 .
`N RED`                                   1.021e−02 2.704e+01 −3.776  0.00693 **
`N SILVER`                                7.275e−01 3.095e+01 −2.350  0.05106 .
`N WHITE`                                 2.079e−02 7.143e+01 −2.911  0.02264 *
`Popolazione al 31 dicembre − Totale`     1.377e−02 5.760e−03  2.391  0.04812 *
`Numero di Famiglie`                     −4.781e−02 1.595e−02 −2.998  0.02001 *
`Numero di Convivenze`                   −8.602e−01 3.998e−01 −2.152  0.06845 .
`Numero medio di componenti per famiglia` 2.657e+01 7.096e+01 3.744  0.00723 **
`eta media`                               1.743e+01 8.376e+00  2.081  0.07598 .
`reddito totale`                          6.543e−07 2.083e−07  3.141  0.01636 *
−−−
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 33.46 on 7 degrees of freedom
Multiple R−squared: 0.9713,     Adjusted R−squared: 0.8322
F−statistic: 6.979 on 34 and 7 DF, p−value: 0.006189


AIC(fullModelBD) # AIC => 410.8074
BIC(fullModelBD) # BIC => 473.3635
```

Il risultato è ancor più attendibili di quanto ottenuto dal modello globale. Infatti R^2 è uguale a 0.97 e i valori dell AIC e BIC sono più bassi dei precedenti. Ciò implica un miglior adattamento delle variabili all'analisi.

4.1.3 Visualizzazione

Una volta individuati i precettori, e decretato come influenzano le vendite, si passa alla fase di visualizzazione. Affinchè questo avvenga, in Angular è stata creata una classe ad hoc che gestisce il grado di "importanza" assunto dal precettore. Si è deciso di suddividere i possibili output risultanti dalla regressione in 5 varianti.

```
export class ColorCorrelation {
    hightPositiveCorrelation() {
        return '#0dff00';
    }

    mediumPositiveCorrelation() {
        return '#00fff2';
    }

    lowPositiveCorrelation() {
        return '#00b7ff';
    }

    noCorrelation() {
        return null;
    }

    lowNegativeCorrelation() {
        return '#fbff00';
    }

    mediumNegativeCorrelation() {
        return '#ff9d00';
    }

    hightNegativeCorrelation() {
        return '#ff0000';
    }
}
```

La classe `ColorCorrelation` suddivide le tipologie di casistiche in base al numero di asterischi nel risultato della regressione lineare considerando inoltre il segno del coefficienti dato dalla colonna `Estimate`. L'ordine dei metodi va dai massimi (`hightPositiveCorrelation}()` e `hightNegativeCorrelation()`) contrassegnato da tre asterischi, passando per i minimi (`noCorrelation()`). La classe `DataProperties` invece, si occupa di gestire il valore numerico del singolo paese o regione, la label che indica la variabile specifica e il colore.

```
export class DataProperties {
    value;
    label;
    color;

```

```
6       constructor(value, label, color) {
7           this.value = value;
8           this.label = label;
9           this.color = color;
10      }
11  ...
12  }
```

Nella tab della Dashboard, cliccando su uno specifico paese o regione, verrà effettuata la seguente chiamata:

```
1  dataP = new DataProperties(this.data.nkid, "N Kids", this.cCorrelation.hightPositiveCorrelation());
2  this.gender.addObject(dataP);
```

Ogni oggetto della classe, verrà aggiunto ad un array per essere agganciato alla tabella di riferimento. Il risultato lato front-end è il seguente:

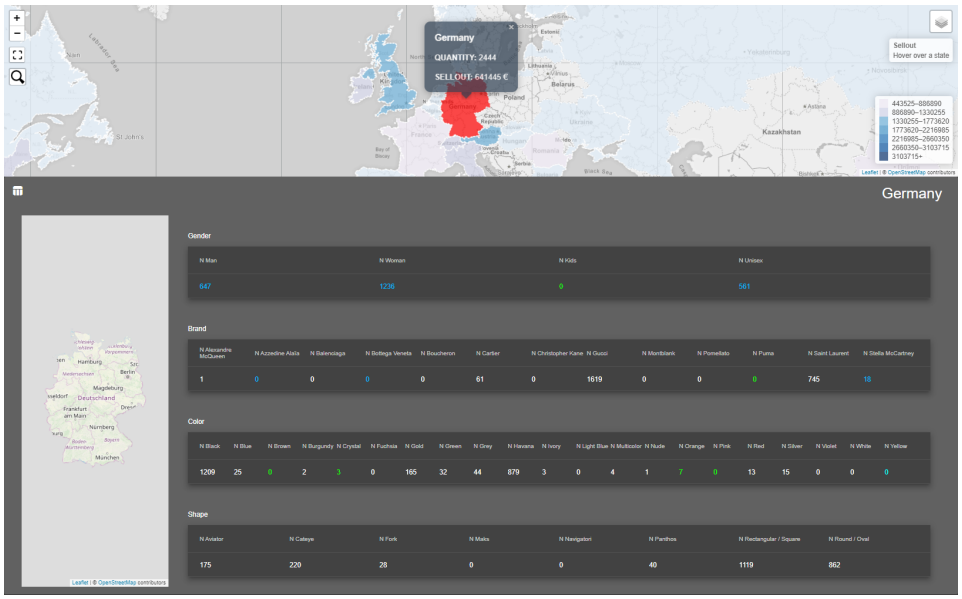


Figura 4.3: Analisi del potenziale globale

Generalmente, i due tipi di analisi sono diversi. Il primo presenta meno variabili influenti ma che sono altamente correlate. Il secondo invece è più equiparato, e la correlazione è più uniforme. Il motivo di questa differenza dipende dall'aggiunta delle variabili demografiche, che come detto, incidono sull'analisi totale. Negli esempi mostrati, si può vedere come la Germania, avente delle vendite comprese tra 400 e 800 mila euro, presenti dei margini di miglioramento. Infatti, le vendite del marchio Puma o di colori quali rosa

e arancione sono a zero, sebbene esse assumono una gran rilevanza ai fini del risultato economico.
Lo stesso discorso, è possibile appurarlo per l'analisi a livello italiano, usando Padova come esempio.

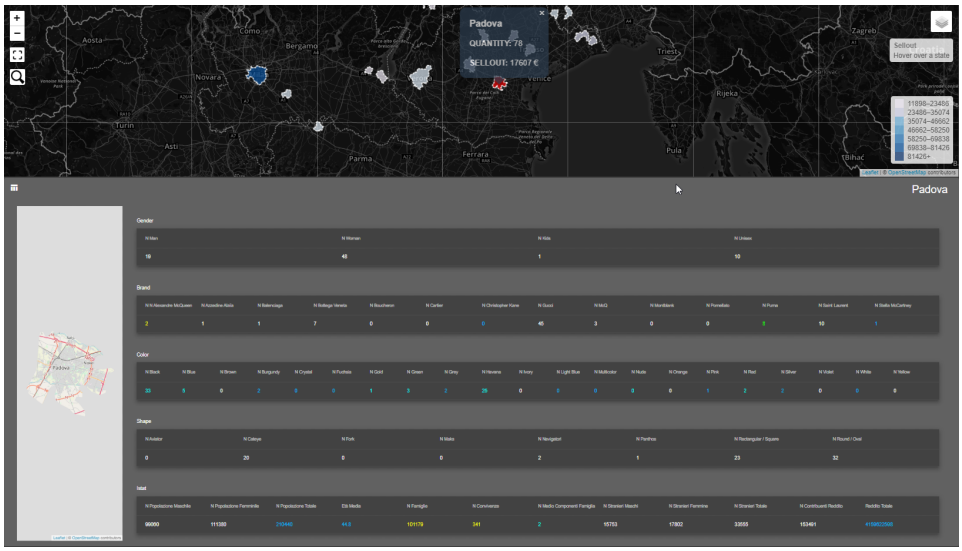


Figura 4.4: Analisi del potenziale regionale

In questo frangente, le variabili che impattano maggiormente sulla vendita sono legate al Brand e al colore. Non presentano significatività nella valutazione, quelle relative al genere e alla forma. Tuttavia in quest'ottica, entrano in gioco anche quelle demografiche. Tra esse alcune non apportano valore alla rendita come il "numero di stranieri" , ma altre quali come il "numero di famiglie" e "numero di convivenze" hanno un'influenza negativa. Questo dato poteva essere captato anche dal coefficiente visto nella regressione lineare. Si traduce quindi che all'aumentare delle suddette variabili, diminuisce il fatturato in proporzione al valore posto nella colonna Estimate del modello. Conclusa la fase di analisi potenziale, si può passare a quella predittiva. dove saranno prese in considerazione altri tipi di variabili e si seguirà un approccio diverso da quanto visto fin'ora.

4.2 Analisi predittiva

La selezione del sito, meglio conosciuta come *Site Selection* descrive la necessità di trovare una collocazione ottimale di un luogo di interesse. La scelta del sito più idoneo, può comportare in molti casi, il successo o il fallimento di un azienda [38]. L'attuazione di questi criteri di ricerca è diversificato e le aziende mettono in piedi i processi più congeniali al loro core business. Uno dei più famosi in tal senso è il *Multicriteria Decision Analysis*. Quest'ultima rappresenta una metodologia ben strutturata che considera fattori ben identificati. Dopo la loro raccolta, essi vengono classificati sulla base di un peso dato, mediante formule matematiche. Il risultato finale è scelto sulla base dei luoghi che sono in linea con pesi dei diversi fattori [39]. Un altro approccio alternativo prevede la costruzione di aree di interesse con l'ausilio di statistiche e algoritmi di percorrenze medie [40]. Con l'avvento della digitalizzazione però, le tecniche sono aumentate notevolmente. Anche in quest'ottica la raccolta dei Big Data insieme alle moderne tecniche di Machine Learning hanno portato un notevole incremento sulle performance dei risultati e una riduzione dei tempi di produzione dell'output.
In quest'ultima parte del capitolo e del lavoro di tesi verrà dato spazio proprio a questo aspetto. Qui la dashboard vista finora verrà ampliata di una nuova features che le conferisce maggior completezza. Dopo essere passati per la descrizione dei valori delle vendite ed aver applicato una regressione lineare per decretare l'impatto assunto dalle variabili, ora ci si concentrerà sulla scoperta di nuove *doors* sfruttando la combinazione di dati a tecniche di Machine Learning. Per il raggiungimento dell'obiettivo prefissato, si è per prima passato attraverso la scelta, la raccolta e la pulizia dei *POI* (Point of Interest). In seguito presi in considerazione degli algoritmi di ML. E' stato per cui effettuato un confronto sulla base delle performance offerte e valutati con metriche specifiche [41]. Al miglior algoritmo trovato sono stati dati in pasto i valori raccolti dalla dashboard[10] e riproiettati con il valore di output specifico che ne decretava la corrispettiva colorazione. In aggiunta si è ritenuto opportuno dare la possibilità di scegliere le aree di interesse a fronte di altre che magari erano già coperti da una domanda soddisfacente.

[10] Grazie all'aiuto di Turf.js

4.2.1 Point of Interest (POI) e manipolazione dei dati

Come primo passo effettuato vi è stata la raccolta dei dati. Uno studio [42] articola riguardo le relazioni spaziali esistenti tra diverse categorie di luoghi e quantifica la loro interconnessione, anche detta *frequenza di attrazione*. Con "frequenza di attrazione" si intende l'interconnessione di passaggio tra due luoghi diversi. Volendo essere più espliciti, spiega se c'è una relazione al passaggio di una persona in due posti diversi. Se il valore tra due luoghi è più alto del valore medio, allora sono da considerarsi attendibili. Al contrario, i due luoghi sono più propensi a respingersi. L'articolo [15], partendo da questo aspetto, classifica i luoghi di maggior interesse sulla base dei dati raccolti dalla piattaforma Foursquare[11]. Ne consegue che luoghi di interesse come aeroporti, musei, hotels, ristoranti e fermate dei bus sono tutti luoghi con un coefficiente più alto rispetto la media.
A livello italiano, la piattaforma a cui si è fatto riferimento per il reperimento dei dati è *DatiOpen.it*[12]. Da qui sono state raccolte le seguenti informazioni a livello italiano:

- musei
- ristoranti
- aeroporti
- B&B
- fermate autobus

In aggiunta sono stati considerati i dati demografici come fatto per l'analisi del potenziale. I dati così ottenuti sono stati poi caricati su tabelle singole poste all'interno del database[13]. Il tutto per poter eseguire query spaziali che potessero sancire la distanza minima esistente tra le singole doors dai rispettivi POI.

[11] https://en.wikipedia.org/wiki/Foursquare_(company)
[12] http://www.datiopen.it/
[13] Per il caricamento dei dati, si faccia riferimento all'appendice A sezione 3.6

```
SELECT
lg.door_name,
lg.city,
SUM(so.sell_out_quantity) as quantity,
SUM(so.estimated_sell_out_value_based_on_srp_euro5) as net_sell_out_value_euro_act,
(SUM(so.estimated_sell_out_value_based_on_srp_euro5)/SUM(so.sell_out_quantity )) as sellout,
ist.popolazione_maschile,
...
ist.reddito_totale,
closest_aero.nome_aeroporto,
closest_aero.dist_aeroporto,
...
...
center_distance.dist_center
FROM
doors As lg LEFT JOIN sellout as so ON lg.kering_door_code = so.Internal_Door_Code
                        LEFT JOIN istat as ist ON lower(lg.city) = lower(ist.comune)
                    -- aeroporti
                        CROSS JOIN LATERAL
                          (SELECT
                                  aero.nome as nome_aeroporto,
                                  ST_Distance(aero.geom, lg.geom) as dist_aeroporto
                                  FROM aeroporti_it as aero
                                  ORDER BY lg.geom <-> aero.geom
                                 LIMIT 1
                           ) AS closest_aero
                        -- Strutture ricettive: Agriturismi, B&B
                ...
                        -- Musei
                ...
                        -- ristoranti
                ...
            -- Fermate Autobus
                           ...
                        -- distanza dal centro
                           CROSS JOIN LATERAL
                          (SELECT
                                  ST_Distance(lg.geom, ST_Centroid(town.geom)) as dist_center
                                  FROM shape_town as town
                                  WHERE ST_CONTAINS(town.geom, lg.geom::geometry)
                                                                    ) AS center_distance
WHERE so.operation_type_name = 'Sell Out' and lg.country = 'Italy (IT)' and lg.city != 'null'
GROUP BY
lg.door_name,
lg.city,
ist.popolazione_maschile,
...
ist.reddito_totale,
closest_aero.nome_aeroporto,
closest_aero.dist_aeroporto,
...
...
center_distance.dist_center
```

La query vista, semplificata per essere più chiara, assolve proprio a questa funzione. Come si può notare, per ogni POI è stato effettuato un *CROSS JOIN LATERAL*. Il *CROSS JOIN* come la documentazione ufficiale espone[14], non presenta condizioni di corrispondenza come per il `LEFT` perché il

[14] http://www.postgresqltutorial.com/postgresql-cross-join/

suo scopo è di effettuare un prodotto cartesiano tra le righe della tabella. La parola `LATERAL` si rifà alla possibilità di poter fare riferimento a colonne fornite da elementi `FROM` precedenti. Senza `LATERAL`, ogni sottoquery viene valutata in modo indipendente e quindi non può fare riferimenti incrociati a nessun altro elemento `FROM`[15].
Fatta questa premessa, si può semplificare che ogni POI è trattato quasi indipendentemente dagli altri e alla fine, i risultati aggregati in un unico output. All'interno di ogni subquery è calcolata la distanza e ordinata[16] in modo crescente. Analogamente a quanto fatto per i POI, è stata calcolata la distanza dal centro:

nome door	città	...	label sellout	sellout (€/pcs)	popolazione maschile	...	nome aeroporto	dist aeroporto (m)	dist center
Door 1	MAZARA DEL VALLO	...	NON CONSIGLIATO	235.39	25470	...	Aeroporto di Trapani Livio Bassi	14874.64	7.69
Door 2	MILANO	...	MOLTO CONSIGLIATO	960.83	654973	...	Aeroporto di Crotone Sant'Anna Pitagora	43727.76	4.36
Door 3	BARI	...	MODERATAMENTE CONSIGLIATO	257.73	155452	...	Aeroporto di Bari Karol Wojtyla	8473.18	2.59
Door 4	SABAUDIA	...	NON CONSIGLIATO	204.14	10713	...	Aeroporto di Treviso Sant'Angelo Antonio Canova	49413.67	0.23
...	TORRE DEL GRECO	...	CONSIGLIATO	530	41218	...	Aeroporto Internazionale Giovan Battista Pastine	8647.46	1.97

Tabella 4.5: Dati vendite, spaziali e demografici per l'analisi predittiva

Come si può vedere i dati sotto forma tabellare sono ottenuti dall'intersezione tra i dati di vendita sia con quelli demografici e sia con quelli spaziali. Spicca la colonna "label sellout" che mostra i dati di vendita del singolo prodotto suddivisi in sezioni. Da 0 a 250 (€/pcs) il valore sarà "NON CONSIGLIATO"; da 250 (€/pcs) a 500 (€/pcs), la label mostrerà "MODERATAMENTE CONSIGLIATO"; da 500 (€/pcs) a 750 (€/pcs), "CONSIGLIATO"; da 750 (€/pcs) a 10000 (€/pcs), "MOLTO CONSIGLIATO" e per valori superiori, "ALTAMENTE CONSIGLIATO" La ragione per cui è stato fatto quanto descritto dipende dall'incertezza del tipo di algoritmo

[15] https://www.postgresql.org/docs/9.4/queries-table-expressions.html
[16] https://www.postgresql.org/docs/current/pgtrgm.html# PGTRGM-OP-TABLE

di Machine Learning che sarà usato. E' necessario predisporre il dataset con l'aggiunta di label meglio note come "categorie" perché la presenza o l'assenza, decreta l'utilizzo di una tipologia di algoritmo a fronte di un altra. A livello generate esistono due tipologie di algoritmi: *supervisionato* o *non supervisionato [43]* . Il primo, meglio noto come *apprendimento supervisionato*, parte dal presupposto che il set di dati, serva per predire valori non conosciuti. In pratica l'algoritmo impara dal modello. Ogni dataset è composto da valori numerici e in aggiunta ha un valore categoriale che ne identifica il responso associato a ciascuna osservazione, come in questo caso. Per cui il nostro set di dati, presenta sia valori numerici relativi alla distanza e altre informazioni demografiche significative e sia un responso che sancisce l'output derivato da quelle informazioni. Volendo essere più chiari, lo si può leggere nel seguente modo: essendo la *Door1* distante 14 km dall'aeroporto più vicino e presente in una città con una popolazione maschile pari a 25000 persone, più un'altra serie di informazioni, ne consegue che un nuovo posizionamento in quell'area non è consigliato. Lo stesso ragionamento, vale anche per le altre door dove l'output è diverso o per altre situazioni, dove i set di dati può riguardare ad esempio informazioni sulle cellule e il responso può asserire se si tratta di cellule tumorali o meno. Gli algoritmi maggiormente utilizzati in questi tipo di contesto sono quelli di classificazione e di regressione. Il primo è il processo di predizione di valori discreti o categorie, il secondo invece, punta a predire valori continui, come lo possono essere l'emissione di Co2 partendo da informazioni quali consumo di carburante, cilindrata etc; oppure può servire per predire valori di vendita partendo da informazioni come numero di vendite da uomo, donna, bambino etc[17]. L'*apprendimento non supervisionato*, è l'altra macro tipologia di algoritmi di Machine Learning. Essa è chiamata in questo modo, perché a differenza della precedente, non parte da informazioni conosciute a priori per cercare di trovarne di nuove, ma gli algoritmi, tentano di trovare informazioni e pattern tra i dati non visibili ad occhio. La peculiarità di questi algoritmi è che i dataset di cui necessitano, debbano contenere solo numeri e non label. Gli algoritmi in questione, sono più complessi perché non sono

[17] Come fatto nell'analisi del potenziale

supportati da nessun tipo di output associato ai dati. Il *clustering* è uno degli algoritmi maggiormente consociuti. Raggruppa il set di dati partendo da affinità strutturali, o da anomalie comune fra i dati e definisce dei gruppi che presentano le medesime caratteristiche [44] . Tornando al nostro dataset, così come fatto per lo studio precedente il set di dati a disposizione non è sufficientemente grande per essere dato in pasto ad un algoritmo di ML. Non è inusuale trovarsi in una situazione di questo genere. Infatti sono molti gli articoli scientifici che argomentano casistiche di questo tipo [45] [46] . Molte volte, soprattutto con studi sperimentati è sovente trovarsi a che fare con dataset detti "sbilanciati" . Il modo, come si vedrà tra poco, per ottenere un insieme soddisfacente è quello di utilizzare dei dati detti "sintetici" . Con dati sintetici si intende dati creati ad hoc che seguono l'andamento di quelli reali. Sono creati nel momento in cui non si riesce ad ottenerli in maniera diretta. Il modo più semplice per generate è crearne random, ma in realtà in questo lavoro, saranno creati appositamente con librerie ad hoc. Nel nostro caso, partendo dal dataset originale, vi è stato un incremento quantitativo di valori in maniera random facendo sì che fossero compresi tra la media più e meno la varianza[18]. In questo modo, si è ottenuto un dataset che comprendesse molte più osservazioni da quelle di partenza per un totale di 500 righe. Ora possediamo il dataset che sarà usato per creare il modello predittivo. Proviamo a fare due considerazioni su quanto ottenuto. Ci avvaliamo di Python, linguaggio di programmazione Server Side, che fa della facilità di utilizzo e vastità di campi di applicazione uno dei suoi punti di forza. Per cui, per prima cosa, carichiamo il nostro set di dati in formato csv[19].

```
1 import matplotlib.pyplot as plt
2 import pandas as pd
3 import pylab as pl
4 import numpy as np
5 import scipy.optimize as opt
6 import seaborn as sns
7
8 df = pd.read_csv("../data/sellout.csv", delimiter=";", thousands=',')
9 print(df.head())
```

[18] La varianza calcola quanto si discostano i valori dalla media. https://en.wikipedia.org/wiki/Variance

[19] L'esportazione in csv è stata fatta da Excel.

La prima operazione da effettuare è il caricamento delle librerie necessarie. Tra queste in particolar modo, sarà usata *Pandas*, utilizzata per la manipolazione e l'analisi dei dati. Grazie ad essa, viene letto il file csv, facendo bene attenzione a specificare il separatore dei valori[20] e il separatore delle migliaia. Subito dopo, proviamo a leggere con la funzione `head()` le prime `n` righe (il valore di default è 5) per verificare rapidamente se l'oggetto contiene il giusto tipo di dati.

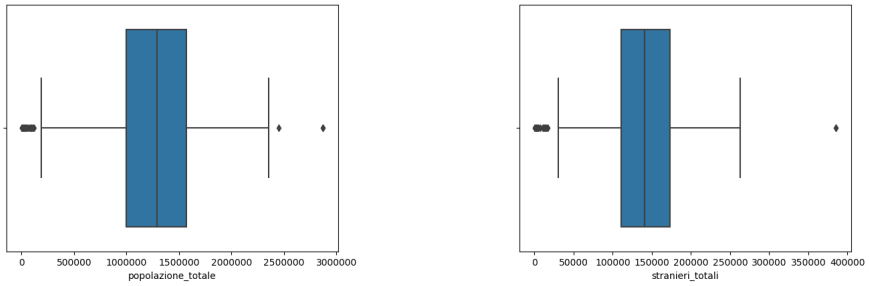
```
1      door_name    city  quantity     sell_out  ...  dist_bus  ...  dist_center_m  ...
2  0       Door1   CAPRI         9         3200  ...     98.25  0.10        8354.08  8.35
3  1       Door2    ROMA        32         8870  ...     28.81  0.03        4874.85  4.87
4  2       Door3 FIRENZE        29         8165  ...   6506.23  6.51        1286.34  1.29
5  3       Door4  MILANO        52        16470  ...    145.27  0.15        2542.17  2.54
6  4       Door5  MILANO        79        23135  ...    180.33  0.18        2013.51  2.01
7
8  [5 rows x 34 columns]
```

Avendo appurato che il nostro dataset comprenda tutte i dati necessari, proviamo ad estrarre info avvalendoci di uno strumento grafico come il *boxplot*. Esso è una rappresentazione grafica che ci consente di verificare la distribuzione di un set campione. Per raggiungere questo intendo, vengono sfruttati i *quartili*[21]. Il primo (Q1), è la media tra il valore più piccolo e la mediana; il secondo (Q2) è dato dalla mediana mentre il terzo (Q3) è la media tra la mediana e il valore massimo. Proviamo a visualizzare la distribuzione delle nostre variabili sfruttando la libreria python `seaborn`, prendendo a mò di esempio due tra le variabili X.

```
1 sns.boxplot(x=df['popolazione_totale'])
2 plt.show()
3 sns.boxplot(x=df['stranieri_totali'])
4 plt.show()
```

Le variabili prese in esame, presentano entrambe degli *outlier:* punti che differiscono notevolmente dalla distribuzione raccolta. Nelle due immagini (Fig. 4.5), è evidente che per entrambe le variabili esistono più punti non compresi tra il Q1 e Q3. Per avere un dataset pulito che non influisca negativamente sul nostro modello predittivo è necessario rimuove i valori anomali. Esistono vari modi per raggiungere tale obiettivo [47] . Quello adoperato per questo lavoro di tesi è stato il *IQR score [48]* sia per la facilità

[20] Excel usa esportare il file con separatore ";" ma è facile trovare in questi casi la virgola.
[21] https://en.wikipedia.org/wiki/Quartile



(a) Boxplot popolazione totale (b) Boxplot numero di stranieri

Figura 4.5: Rappresentazione di Outlier su Attributi

di utilizzo e sia per la larga adozione.

Per prima cosa acquisiamo informazioni sul nostro dataset con la chiamata `df.info()`.

```
print(df.info())

#Output
Data columns (total 34 columns):
door_name                          44 non-null object
city                               44 non-null object
quantity                           519 non-null int64
net_sell_out_value_euro_act        519 non-null int64
label_sellout                      519 non-null object
sellout                            519 non-null float64
...
...
dist_center                        519 non-null float64
dtypes: float64(14), int64(12), object(8)
memory usage: 121.7+ KB
```

Questa funzione ci fornisce una serie disparata di informazione dall'uso della memoria ai valori non nulli. Ciò che preme sottolineare è che il nostro set di dati presenta variabili di tipo *object* ossia valori non numerici. Di questi, solo l'attributo "label_sellout" è stato mantenuto perché rappresenta la variabile Y su cui è costruito il modello. Eventuali valori nulli avrebbero inficiato sul risultato della pulizia e sarebbero stati da eliminare. Per questa ragione sovrascriviamo il dataframe con le variabili di interesse.

```
df = df[['quantity','sellout',label_sellout,'popolazione_totale',...,'dist_center']]
```

Ora possiamo concentrarci sul *IQR*. L'*IQR* anche detto *scarto interquartile* è la differenza fra il terzo e il primo quartile. E' simile alla deviazione

standard o la varianza, ma è maggiormente robusta contro gli *Outliers*. Dal suo utilizzo si cerca di ottenere la dispersione di una variabile statistica intorno a un indicatore medio di posizione.

```
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1
print(IQR)

#Output
quantity                                5.900000e+01
net_sell_out_value_euro_act             1.204450e+04
sellout                                 1.393100e+02
popolazione_maschile                    5.016745e+05
popolazione_femminile                   4.074255e+05
popolazione_totale                      5.748845e+05
numero_di_famiglie                      4.526935e+05
numero_di_convivenze                    3.905000e+02
numero_medio_componenti_famiglia 5.000000e-01
stranieri_maschi                        4.483100e+04
straniere_femmine                       6.107950e+04
stranieri_totali                        6.244100e+04
reddito_numero_contribuenti             6.282715e+05
reddito_totale                          1.237792e+10
dist_aeroporto                          2.498000e+01
dist_bb                                 1.251500e+01
dist_musei                              1.196166e+31
dist_ristorante                         1.485000e+00
dist_fermata_bus                        2.390000e+00
dist_center                             4.225000e+00
dtype: float64
```

L'output risultante rappresenta l'IQR per tutte le colonne che compongono dataframe. Per sapere quali colonne contengono gli outliers,

```
print( (df < (Q1 - 1.5 * IQR)) | (df > (Q3 + 1.5 * IQR)) )
```

L'output che ne deriverà sarà una matrice composta da `True` o `False`. In caso di valori False significa che i valori sono tutti compresi tra Q1 e Q3. In caso di valori True allora la colonna conterrà outliers. Si può notare come all'interno della condizione booleana ci sia un valore uguale a 1.5. Tale valore, rappresenta una regola d'uso. Rifacendoci ad esempio all Fig. 4.5b infatti, non è chiaro quanti siano il numero di outliers presenti. Per convenzione, in questi casi, si usa moltiplicare il IQR ± 1.5 perché i valori anomali bassi sono inferiori a

$$\text{Q1} - 1.5 \cdot \text{IQR}$$

e quelli alti superiori a

$$\text{Q1} + 1.5 \cdot \text{IQR}$$

.

```
     dist_aeroporto  dist_bb  ...  stranieri_maschi  stranieri_totali
0             False    False  ...             False             True
1             False    False  ...             False             True
2             False    False  ...              True             True
3             False    False  ...             False            False
4             False    False  ...             False            False
..              ...      ...  ...               ...              ...
514           False    False  ...             False            False
515           False    False  ...             False            False
516           False    False  ...             False            False
517           False    False  ...             False            False
518           False    False  ...             False            False

[519 rows x 21 columns]
```

Adesso passiamo all'eliminazione vera e propria dei valori ritenuti superflui.

```
print(df.shape)  # => (519, 21)
df_out = df[~((df < (Q1 - 1.5 * IQR)) |(df > (Q3 + 1.5 * IQR))).any(axis=1)]
print(df_out.shape) # => (327, 21)
```

Possiamo vedere che da un dataframe composto da 519 righe iniziali per 21 colonne, si è passati ad un altro con lo stesso numero di colonne ma con un numero di righe inferiori, ripulite dai vari outliers.

4.2.2 Comparazione di vari algoritmi di Machine Learning

Ora siamo in grado di poter confrontare i vari algoritmi tra loro. Generalmente in questi tipi di lavori si procede testandoli ponendo su una base di confronto tramite metriche predefinite [49]. Da questo confronto sarà decretato quello che meglio si adatta ai dati passati come parametro e quindi usato nel progetto oggetto di studio. Per questo progetto di tesi, sono stati utilizzati i seguenti algoritmi[22]: Support Vector Machine (SVM), Logistic Regression, K-nearest neighbors e Decision Tree.

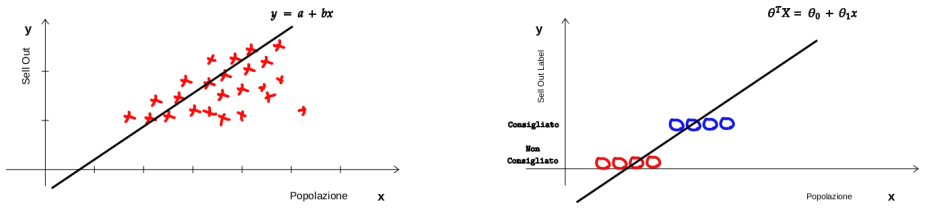
4.2.2.1 Logistic Regression

La Logistic Regression[23], è una tecnica statistica e di Machine Learning che serve per classificare dei valori partendo da un set di dati in input.

[22] La loro spiegazione si è basata sui corsi Python realizzati da IBM sulla piattaforma Coursera https://www.coursera.org/

[23] https://en.wikipedia.org/wiki/Logistic_regression

Così come visto con la regressione lineare, prevede l'utilizzo di variabili indipendenti e una variabile dipendente. E' un algoritmo di apprendimento supervisionato e il suo output cerca di definire l'appartenenza o meno ad una data categoria. Si presta generalmente a predizione di valori binari come 0/1, Si/No, True/False, etc. In tal senso, alcuni esempi che meglio fanno comprendere i campi di utilizzo riguardano ad esempio la probabilità di essere colpiti da attacchi cardiaci sulla base di pressione sanguigna, età, livello di attività fisica, etc; oppure la propensione degli individui a comprare specifici prodotti a dispetto di altri e così via dicendo.
Quanto detto, sottoscrive la prima grande differenza rispetto la semplice regressione lineare. Infatti la regressione semplice, sebbene utilizzata precedentemente per verificare l'influenza tra variabili, se applicata in questi contesti, predice nuovi valori attenendosi alla continuità dei valori della variabile Y. Per intenderci, alla crescita di un precettore X, come può esserlo la variabile "popolazione totale" , crescerà o decrescerà linearmente la variabile Y "sell out" .
Se però provassimo a considerare la variabile "sellout_label" che mostra le categorie di sell out, la quale per semplicità sarà solo "Consigliato" e "Non consigliato", la modalità di rappresentazione dei punti sul piano di assi cartesiano cambierà.



(a) Plot Linear Regression (b) Plot Logistic Regression

Figura 4.6: Differenza fra Linear Regression e Logistic Regression

Confrontando le due immagini, è palese la differenza che assume quella di destra (Logistic Regression) da quella di sinistra (Linear Regression). Le due equazioni che descrivono la retta sono equivalenti. Assumendo di voler provare a predire dove si cataloga una nuova door risolvendo l'equazione di riferimento, il valore risultante si troverà o al di sopra o al di sotto della

linea che divide le due label.

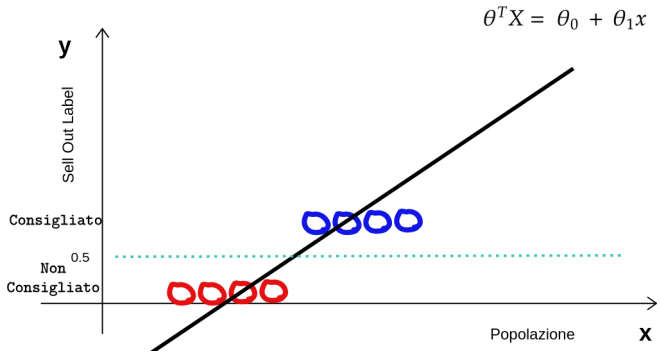


Figura 4.7: Soglia di appartenenza ad una categoria

Per cui si può sintetizzare dicendo

$$\hat{y} = \begin{cases} 1 \text{se} \theta^T X < 0.5 \\ 0 \text{se} \theta^T X \geq 0.5 \end{cases} \tag{4.2}$$

L'equazione spiega come per valori maggiori di 0.5, allora apparterrà alla categoria "Consigliato" , mentre per valori inferiori alla soglia, sarà "Non Consigliato" . Ciononostante non è spiegato quale sia la probabilità che una door appartenga ad una delle categorie. Per questa ragione, viene usata la funzione σ (Sigma). Tale funzione spiega la probabilità di appartenere ad una classe invece che cercare di predire il valore in maniera diretta.

$$\sigma(\theta^T X) = \frac{1}{1 + e^{-\theta^T X}} \tag{4.3}$$

L'Output della funzione sarà un numero uguale compreso tra 0 e 1 in dipendenza dal valore che assume la variabile lineare θ $^{\mathrm{T}}$X. Il grafico spiega la funzione, perché al crescere di θ $^{\mathrm{T}}$X al denominatore, si otterrà un valore più vicino ad 1, al contrario 0.

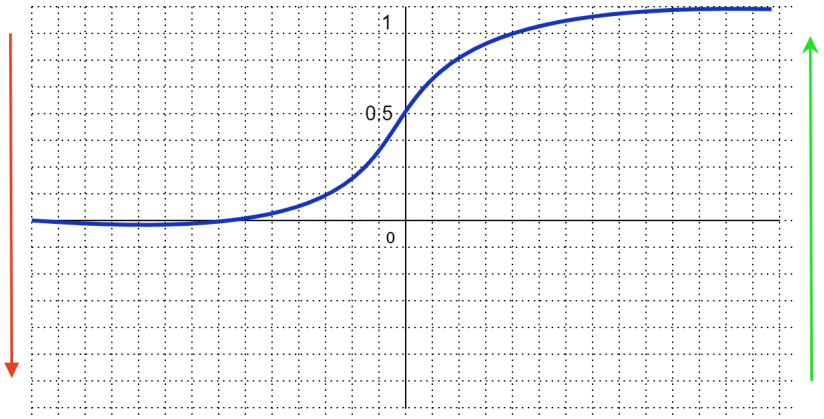


Figura 4.8: Rappresentazione grafica funzione sigmoidea

A livello pratico, affinché la probabilità abbia senso vi è la necessità di individuare un valore di θ che meglio riesca a predire la variabile Y ed è qui che entra in gioco l'algoritmo di Machine Learning. Infatti grazie all'algoritmo, è definito un θ random e poi iterativamente viene calcolata la funzione costo, la quale più assume un valore minore e più riesce a predire l'output del precettore Y.
In Python, trasformiamo il set di dati etichettando le nostre variabili di output con valori numerici.

```
1  y = df['label_sellout'].values
2
3  from sklearn import preprocessing
4  le_sellout = preprocessing.LabelEncoder()
5  le_sellout.fit(['NON CONSIGLIATO','MODERATAMENTE CONSIGLIATO','CONSIGLIATO','MOLTO
       CONSIGLIATO', 'ALTAMENTE CONSIGLIATO']) print(le_sellout)
6  y = le_sellout.transform(y)
```

Nella fase successiva, carichiamo le nostre variabili X e le standardizziamo. Da qui in poi ha inizio la fase di training del modello.

```
1  X = df[['popolazione_maschile'...'dist_center']] .values
2
3  # Transform Value Between 0 and 1
4  X = preprocessing.StandardScaler().fit(X).transform(X.astype(float))
5
6  from sklearn.model_selection import train_test_split
7  X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.3, random_state=4) # 70% training
       and 30% test
8  print ('Train set:', X_train.shape, y_train.shape) #Train set: (247, 17) (247,)
9  print ('Test set:', X_test.shape, y_test.shape) # Test set: (106, 17) (106,)
```

Una volta ottenuti i due set di dati, vengono passati i risultati alla funzione Logistic Regression, la quale nella riga successiva prova a predire da quanto appreso nella fase di training, i risultati delle variabili X di test e a calcolarne la probabilità.

```
from sklearn.linear_model import LogisticRegression
LR = LogisticRegression(C=0.01, solver='liblinear').fit(X_train,y_train)

yhat = LR.predict(X_test)
# print(yhat)

# predict_proba returns estimates for all classes, ordered by the label of classes.
# So, the first column is the probability of class 1, P(Y=1|X), and second column is probability of # class
      0, P(Y=0|X):
# yhat_prob = LR.predict_proba(X_test)
```

E' chiaro che allo stato attuale non è interessante sapere il valore predetto sul test set quanto su quello passato come input dal server della dashboard[24].

4.2.2.2 KNN

Il *K-Nearest Neighbors*[25] è un algoritmo di classificazione supervisionato, basato sull'idea di categorizzare i valori sconosciuti in un set discreto di classi. L'algoritmo è abbastanza intuitivo. Consideriamo la variabile dipendente Y divisa in 3 categorie "Non consigliato" , "Consigliato" , "Molto Consigliato" . Il set di dati disposto su un piano di assi cartesiano.

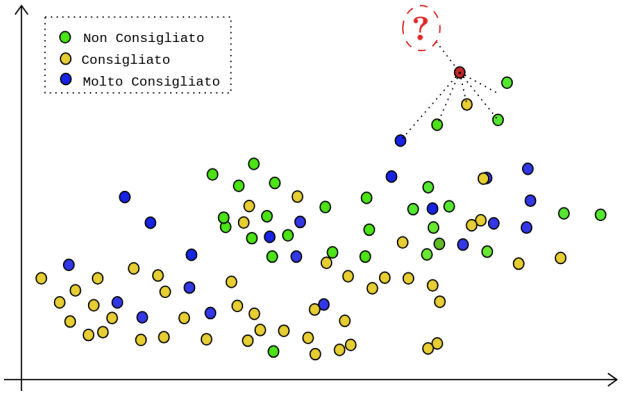


Figura 4.9: Esempio funzionamento KNN

[24] Questo si vedrà in una sezione successiva.
[25] https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

L'algoritmo prova a catalogare le osservazioni sconosciute (il pallino rosso) sulla base della vicinanza con gli altri casi. Se il numero dei *neighbors* presi in considerazioni fosse 1 allora la classe di appartenenza del punto sarebbe quella "Consigliato" . Se diversamente il numero fosse 5, allora sarebbe "Non Consigliato" . I passi necessari per l'implementazione sono:

- scelta del k;
- calcolare la distanza tra i casi sconosciuti da tutti gli altri;
- calcolare la distanza minima dei punti sconosciuti con quelli conosciuti;
- individuare la classe di appartenenza del punto a distanza minima.

La distanza è decretata sulla base della formula Euclidea:

$$Dis(x_1, x_2) = \sqrt{\sum_{i=0}^{n}(x_{1i} - x_{2i})} \tag{4.4}$$

Resta da capire quale sia il miglior `k` su cui basare l'algoritmo. Non esiste una soluzione univoca a questo. Di norma, si tendono ad escludere valori estremi come $k = 1$ o $k = 20$. Entrambi i casi presupporrebbero una cattiva predizione. Il primo perché il punto risultante potrebbe essere un caso di overfitting, il secondo sarebbe uno troppo generico. Quale `k` allora scegliere? Fortunatamente la libreria scikit-learn seleziona il punto migliore per noi, sulla base di una metrica di comparazione che sarà descritta più avanti.

```
...
...

from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics

Ks = 10
mean_acc = np.zeros((Ks-1))
std_acc = np.zeros((Ks-1))
ConfustionMx = [];
for n in range(1,Ks):

    #Train Model and Predict
    neigh = KNeighborsClassifier(n_neighbors = n).fit(X_train,y_train)
    yhat=neigh.predict(X_test)
    # print("VEDIAMO I VALORI CHE HA PREDETTO")
    # print(yhat)
    mean_acc[n-1] = metrics.accuracy_score(y_test, yhat)


    std_acc[n-1]=np.std(yhat==y_test)/np.sqrt(yhat.shape[0])

```

```
23
24
25 plt.plot(range(1,Ks),mean_acc,'g')
26 plt.fill_between(range(1,Ks),mean_acc - 1 * std_acc,mean_acc + 1 * std_acc, alpha=0.10)
27 plt.legend(('Accuracy ', '+/- 3xstd'))
28 plt.ylabel('Accuracy ')
29 plt.xlabel('Number of Nabors (K)')
30 plt.tight_layout()
31 plt.show()
```

Il codice, dopo aver pulito e splittato il dataset come effettuato per tutti gli algoritmi, inizializza la variabile preposta all'algoritmo. Essa viene eseguita in un loop. Ad ogni iterazione c'è una fase di training e di test con un diverso *k*. Tutti le iterazioni vengono valutate con la funzione `accuracy_score` e il risultato aggiunto in un array. Il valore massimo sancirà sia qual è l'accuratezza più alta e sia il `k` relativo. Nel caso specifico:

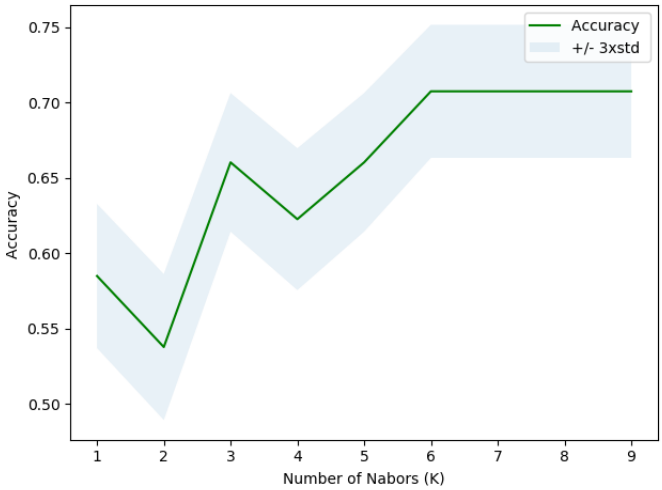


Figura 4.10: Output accuratezza miglior KNN

```
1 print( "The best accuracy was with", mean_acc.max(), "with k=", mean_acc.argmax()+1)
2 #The best accuracy was with 0.7075471698113207 with k= 6
```

4.2.2.3 Decision Tree

Il Decision Tree[26] è un algoritmo di classificazione che prevede la costruzione di un albero topdown per poter mappare un percorso di attributi

[26] https://en.wikipedia.org/wiki/Decision_tree

ottimale per future predizioni. L'albero è costituito dai nodi (attributi) che vengono splittati in branch (rappresentano l'associazione dell'output a quell attributo), e da nodi foglia (quando la scelta di attributi si è esaurito e viene individuata la predizione associata ad uno specifico percorso). In questo paragrafo, sarà spiegato come avviene la costruzione sia a livello teorico che a livello pratico.

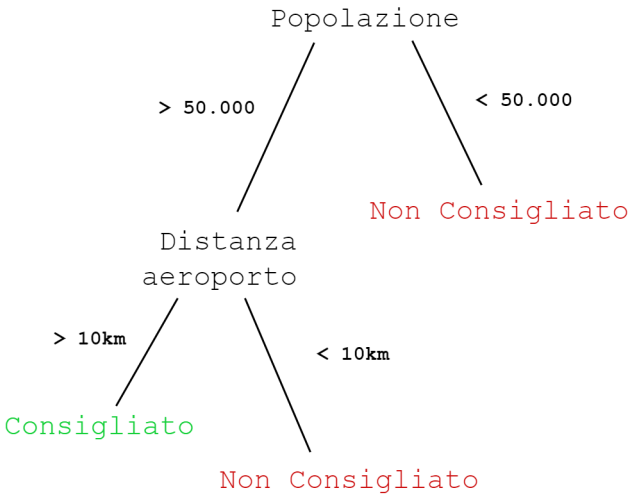


Figura 4.11: Esempio Decision Tree

La costruzione di un Decision Tree avviene grazie alla partizione ricorsiva[27]. Il processo è definito ricorsivo perché ogni sottopopolazione può a sua volta essere suddivisa per un numero indefinito di volte fino a quando il processo di divisione termina al raggiungimento di un criterio di arresto. Gli step che si utilizzano per costruire un decision tree sono:

1. scelta dell'attributo del dataset;
2. calcolo dell'adattamento dell'attributo scelto;
3. calcolo dell'adattamento dell'attributo scelto dopo aver effettuato lo split dei dati;
4. tornare al passo 1;

[27] https://en.wikipedia.org/wiki/Recursive_partitioning

I passi 2 e 3 sono fondamentali per la corretta prosecuzione dell'algoritmo perché puntano a verificare se i valori ottenuti dallo split siano "puri" o "impuri" ossia se è possibile l'associazione univoca di un risultato ad un attributo, o se alternativamente l'attributo presenti più output permettendo la suddivisione del dataset in altre sottoparti. A livello matematico corrisponde al calcolo dell'*entropia* dei nodi.

$$Entropia = p(A)\log(p(A)) - p(B)\log(p(B)) - \cdots - p(n)\log(p(n)) \quad (4.5)$$

$$= -\sum p(X)\log(p(X)) \quad (4.6)$$

La formula, per ogni nodo, misura gli output accomunati ad esso. Ad esempio immaginando che il dataset sia composto da 6 osservazioni di cui 3 avranno come output "Non Consigliato" , due, "Consigliato" e una "Molto Consigliato" , la formula calcolerà:

$$Entropia = \frac{3}{6}\log(\frac{3}{6}) - \frac{2}{6}\log(\frac{2}{6}) - \frac{1}{6}\log(\frac{1}{6}) \quad (4.7)$$

$$= \frac{1}{2}\log(\frac{1}{2}) - \frac{1}{3}\log(\frac{1}{3}) - \frac{1}{6}\log(\frac{1}{6}) \quad (4.8)$$

$$= 0.31 \quad (4.9)$$

Iterativamente, se ne sceglierà un altro. Quando l'output risulterà essere omogeneo, e quindi appartenente solo ad uno specifico precettore X, allora l'entropia sarà 0. Al contrario, se l'output sarà equamente distribuito allora l'entropia arriverà ad assumere un valore massimo di 1. Per classificare la bontà di un nodo, l'entropia sarà calcolata prima dello split (passaggio 2) e sia dopo (passaggio 3). L*'Information Gain* permette di valutare la bontà di un nodo:

$$InformationGain = (Entropiaprimadellosplit) - (pesoentropiadopolosplit) \quad (4.10)$$

Tornando all'esempio, delle 6 osservazioni precedenti, una colonna stima la percentuale di donne rispetto agli uomini, dove per semplicità, se maggiore

nel primo caso segnerà F, se minore M. Supponendo che 3 osservazioni siano F e 3 M, la formula diventa:

$$InformationGain = 0.31 - ((\frac{1}{2}(p(F)) + \frac{1}{2}(p(M))) \tag{4.11}$$

Essendo un processo ricorsivo, è ovvio terminerà solo dopo la la valutazione di tutti i nodi che decreterà la struttura dell'albero risultante.

```
...
...

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=3)

DecTree = DecisionTreeClassifier(criterion="entropy", max_depth = 4)
DecTree.fit(X_train,y_train)

# predTree = DecTree.predict(X_test)
```

Abbiamo visto come a livello di codice, Python tramite la liberia *scikit learn* assumano l'onere della costruzione dell'albero in maniera trasparente. Come da prassi, il dataset è diviso in training e test set. La funzione `DecTree` prende in input i valori del training e crea l'albero che sancirà la costruzione del modello predittivo.

4.2.2.4 SVM

Il *Support Vector Machine*[28] è un algoritmo supervisionato che classifica gli output individuando dei separatori. L'algoritmo si fonda su due concetti principali:

- Effettuare una mappatura dei punti da un piano 2D ad un piano multidimensionale;
- Trovare l'iperpiano che meglio separi le categorie di punti.

La necessità che si cela dietro al dover trasformare un piano da bidimensionale a multidimensionale, è spiegata dalla seguente immagine:

[28] https://en.wikipedia.org/wiki/Support-vector_machine

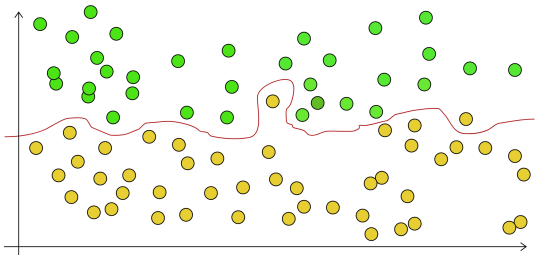


Figura 4.12: Separazione attributi su piano 2D

Immaginando che i punti nel piano siano due delle variabili indipendenti X, trovare una linea che separi distintamente le due features risulta essere alquanto difficile. Tuttavia immaginando il piano di assi visto, su uno spazio 3D, risulta più facile immaginare come si comporta:

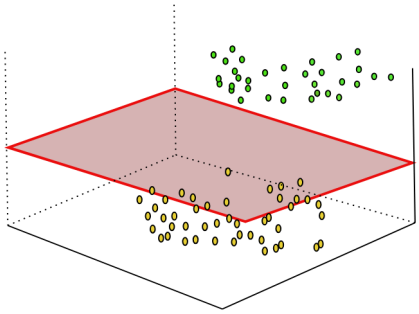


Figura 4.13: Separazione attributi su piano 3D

Quindi, come è possibile trovare il piano, o la retta se lo si immagina su un piano 2D, che separi le due variabili indistintamente? Per assolvere a questo scopo, si utilizza il processo denominato *kerneling*:

$$\phi(X) = [x, x^2] \tag{4.12}$$

La funzione mostrata rappresenta una generalizzazione del concetto di Kernel, che suppone la considerazione di una sola variabile x. Infatti ne esistono di diversi tipi: lineare, polinomiale, sigmoidea etc. Avendo ora definito la mappatura in uno spazio multidimensionale, l'obiettivo del SVM è quello di trovare il miglior piano che riesca a separare al meglio le due classi di

variabili. Il piano trovato è tanto migliore quanto è maggiore la distanza tra le due classi.

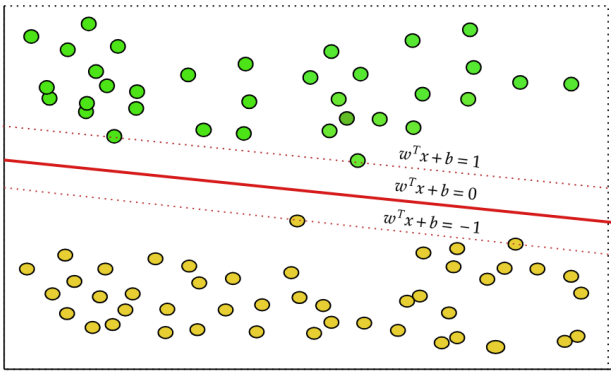


Figura 4.14: Individuazione del piano

L'algoritmo è chiamato Support Vector perché utilizza i vettori di supporto rappresentati dalle linee tratteggiate. L'obiettivo è massimizzare la distanza che intercorre tra la retta e i vettori. Per valori sconosciuti, risolvendo l'equazione, nonché obiettivo dell'algoritmo che trova i valori di w e b, è possibile capire se un valore appartiene ad una classe o quella sottostante.

```
...
...
from sklearn import svm
clf  = svm.SVC(kernel='rbf')
clf . fit (X_train, y_train)

# yhat = clf.predict(X_test)
```

L'algoritmo prende in input il tipo di kernel che si preferisce usare. In questo caso è usato quello di default: *radial basis function* basato sulla distanza da punti fissi e prova a costruire il modello sul training set, come per tutti gli altri algoritmi.

4.2.2.5 Valutazione dei risultati

Ora che gli algoritmi sono stati creati e sviluppati, bisogna verificare quale tra questi menzionati sia più consono al nostro dataset, in quanto non tutti sono applicabili unicamente. Per avere dei risultati che siano il più possibili in linea con i valori reali, la scelta deve basarsi unicamente

sull'output delle metriche. A riguardo, ne esistono molteplici, ma quelle maggiormente utilizzate sono:

- Mean Absolute Error
- Jaccard similarity score
- F-1 Score

La *Mean Absolute Error* (MAE) è sicuramente una delle più semplici:

$$MAE = \frac{\sum_{i=1}^{n} |y_i - \hat{y_i}|}{n} \tag{4.13}$$

Già come la formula sancisce, rappresenta la media in valore assoluto del discostamento tra i valori predetti e i valori reali. L'errore può anche essere espresso in forma percentuale, nella sua variante denominata MAPE. Un'altra misurazione per testare l'efficienza dell'algoritmo è il coefficiente di similarità di Jaccard meglio conosciuto come Jaccard Index. Questa misurazione lavora sui due insiemi derivanti dai valori predetti e dai valori reali:

$$J(y, \hat{y}) = \frac{|y \cap \hat{y}|}{|y \cup \hat{y}|} = \frac{|y \cap \hat{y}|}{|y| + |\hat{y}| - |y \cap \hat{y}|} \tag{4.14}$$

Immaginiamo di avere a disposizione due set di valori di 10 elementi rispettivamente. Il primo contiene i valori reali (y), il secondo quelli predetti (ŷ). Di questi, 8 sono uguali a quelli reali, per cui la formula diventa:

$$J(y, \hat{y}) = \frac{8}{10 + 10 - 8} = 0.66 \tag{4.15}$$

E' chiaro il valore può oscillare da un minimo di 0 (nessun valore è stato predetto) a un valore uguale a 1 (i valori dei due insiemi sono esattamente gli stessi). L'ultima misurazione, che ci accingiamo a descrivere, è l'*F1-score*. Questa metrica di valutazione sfrutta la *matrice di confusione*.

y \ ŷ	Consigliato	Non Consigliato
Consigliato	**6** *TP*	**9** *FN*
Non Consigliato	**1** *FP*	**24** *TN*

Tabella 4.6: Matrice di confusione

La matrice quantifica l'affidabilità, la precisione dei valori predetti e sancisce la corretta separazione delle classi. Le righe corrispondono ai valori reali mentre le colonne ai valori predetti. Supponiamo che il nostro dataset sia composto di 40 osservazioni, tangibile dalla somma dei numeri. Concentriamoci sulla prima riga. Dei 15 valori reali che etichettano il nostro output come "Consigliato" , 6 sono stati interpretati correttamente dall'algoritmo (*True Positive*) mentre 9 no (*False Negative*). Analogamente, dei 25 output "Non Consigliati" ,1 soltanto è interpretato in maniera errata (*False Positive*) mentre i restanti 24 sono stati catalogati correttamente (*True Negative*). La matrice si serve di due misurazioni matematiche:

$$Precision = \frac{TP}{TP + FP} \tag{4.16}$$

$$Recall = \frac{TP}{TP + FN} \tag{4.17}$$

La *Precision* misura l'accuratezza della corretta predizione mentre la *Recall* misura il tasso con cui il modello prevede correttamente i valori realmente positivi[29]. Da queste due misurazioni, viene calcolato l'*F1-score*: la media armonica[30] delle due misurazioni:

$$MediaArmonica = \frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}} \tag{4.18}$$

[29] https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall

[30] Generalmente usata per le situazioni in cui si desidera la media dei tassi.

$$F_1 - score = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{4.19}$$

Dal valore derivante dell *F1-score* riusciamo a classificare l'attendibilità del nostro modello. Quest'ultimo può oscillare da un valore minimo di 0, che corrisponde ad una pessima predizione ad un valore massimo di 1, massima accuratezza dei valori predetti. E' chiaro che quanto visto è solo una semplificazione della matrice reale ottenuta che considera un numero maggiore di variabili. Per ognuno degli algoritmi visti in precedenza sono state effettuate le medesime misure servendoci, come sempre, della libreria scikit learn:

```
def evaluationAlghotihm(y_test,yhat, encodeLabel = False):

    if(encodeLabel==True):
        le_sellout = preprocessing.LabelEncoder()
        le_sellout.fit(['NON CONSIGLIATO',... 'ALTAMENTE CONSIGLIATO'])
        yhat = le_sellout.transform(yhat)
        y_test = le_sellout.transform(y_test)

    print("Classification Report: ")
    confusionMatrix(y_test, yhat)

    # Calculate the absolute errors
    errors = mean_absolute_error(y_test, yhat)
    # Print out the mean absolute error (mae)
    print('Mean Absolute Error: %.4f' % round(np.mean(errors), 2), 'degrees.')
    print("Accuracy: %.4f" % accuracy_score(y_test, yhat))

    # Lets try jaccard index for accuracy evaluation. we can define jaccard as the size of the
      intersection divided by the size of the union of two label sets.
    # If the entire set of predicted labels for a sample strictly match with the true set of labels,
    then the subset accuracy is 1.0; otherwise it is 0.0.
    print("Jaccard similarity score: %.4f" % jaccard_similarity_score(y_test, yhat))

    print("Precision Score: %.4f" % precision_score(y_test, yhat, average='weighted'))
    print("Recall Score: %.4f" % recall_score(y_test, yhat, average='weighted'))
    print("F-1 Score: %.4f" % f1_score(y_test, yhat, average='weighted', labels=np.unique(yhat)) )



def confusionMatrix(y_test, yhat):
    cm = confusion_matrix(y_test, yhat)
    ax= plt.subplot()
    cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
    sns.heatmap(cm, annot=True, ax = ax); #annot=True to annotate cells

    # labels, title and ticks
    ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
    ax.set_title('Confusion Matrix');
    ax.xaxis.set_ticklabels(['NON CONSIGLIATO',... 'ALTAMENTE CONSIGLIATO']);
    ax.yaxis.set_ticklabels(['NON CONSIGLIATO',...,'ALTAMENTE CONSIGLIATO'])
    plt.show()

    print(classification_report(y_test, yhat))
```

Il codice, definisce la matrice di confusione settando le label di output, per poi calcolare le metriche. La valutazione è applicata alle *y_test*, i valori reali e alle *y_hat,* i valori predetti. Lanciando gli algoritmi visti prima[31], i risultati sono simili a quanto segue:

```
1
2                precision    recall  f1-score
3             0       0.00      0.00      0.00
4             1       0.00      0.00      0.00
5             2       0.00      0.00      0.00
6             3       0.00      0.00      0.00
7             4       0.71      1.00      0.83
8
9      accuracy                           0.71
10    macro avg       0.14      0.20      0.17
11 weighted avg       0.51      0.71      0.59
12
13 Mean Absolute Error: 0.6400 degrees.
14 Accuracy: 0.7075
15 Jaccard similarity score: 0.7075
16 Precision Score: 0.5054
17 Recall Score: 0.7075
18 F-1 Score: 0.6378
```

Saltano subito all'occhio due incongruenze. La prima è la presenza di molti zeri per tutti gli output (etichettate con interi) fatta eccezione per l'ultima, la quale si riferisce alla predizione "Altamente Consigliato" . La seconda è che sebbene la presenza degli zeri, l'accuratezza sia in generale elevata. Quanto ottenuto è spiegabile dallo sbilanciamento del dataset. Infatti tutti i valori sono concentrati in prossimità dell'ultima label. Situazioni di questo tipo, si hanno quando su 10 valori, 8 hanno come output lo stesso risultato. Per ovviare, si è utilizzato una libreria che gestisce queste casistiche: *SMOTE.*

```
1  from imblearn.over_sampling import SMOTE
2
3  print ('Train set:', X_train.shape, y_train.shape)
4  print ('Test set:', X_test.shape, y_test.shape)
5  print('Count Category ', np.bincount(y_train))
6  # Output
7  # Train set: (247, 17) (247,)
8  # Test set: (106, 17) (106,)
9  # Count Category [ 5   5  49   5 183]
10
11 smote = SMOTE(kind='regular',k_neighbors=6)
12
13 X_res, y_res = smote.fit_sample(X, y)
14 X_train, X_test, y_train, y_test = train_test_split( X_res, y_res, test_size=0.3, random_state=4) # 70%
       training and 30% test
15
16 print ('Train set:', X_train.shape, y_train.shape)
17 print ('Test set:', X_test.shape, y_test.shape)
18 print('Count Category ', np.bincount(y_train))
```

[31] Per la spiegazione, si è preso in esame la Logistic Regression

```
# Output
# Train set: (903, 17) (903,)
# Test set: (387, 17) (387,)
# Count Category [185 172 182 188 176]
```

La libreria, non fa altro che amplificare il dataset bilanciando gli output risultanti prendendo come parametri il tipo di amplificazione e il numero dei vicini da considerare. Ciò si palesa sia vedendo lo shape del dataset prima e dopo, ma soprattutto vendendo il conteggio delle varie categorie, significativamente migliorato. Rilanciando di nuovo il codice precedente che tiene conto del nuovo dataset, il risultato è quanto segue.

```
                precision    recall  f1-score
           0         0.61      0.90      0.73
           1         0.70      0.74      0.72
           2         0.49      0.39      0.44
           3         0.55      0.94      0.69
           4         0.86      0.07      0.13

    accuracy                           0.60
   macro avg       0.64      0.61      0.54
weighted avg       0.65      0.60      0.54

Mean Absolute Error: 0.7800 degrees.
Accuracy: 0.5995
Jaccard similarity score: 0.5995
Precision Score: 0.6492
Recall Score: 0.5995
F-1 Score: 0.5385
```

Adesso i valori sono notevolmente diversi. La matrice di confusione che ne deriva è:

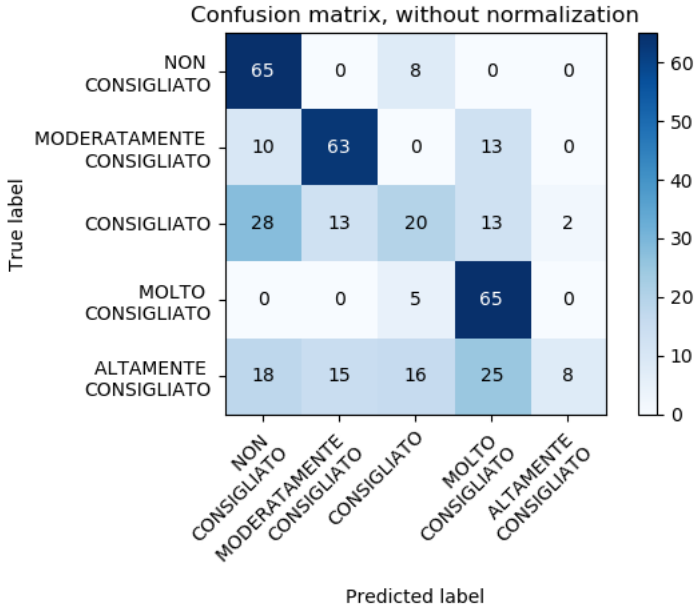


Figura 4.15: Matrice di confusione risultante dal dataset Bilanciato

Mettiamo a confronto i diversi algoritmi.

	Decision Tree	KNN	Logistic Regression	SVM
Mean Absolute Error	0.67	0.640	0.640	0.690
Jaccard similarity score	0.651	0.708	0.708	0.690
Precision Score	0.522	0.501	0.505	0.689
Recall Score	0.651	0.708	0.708	0.690
F-1 Score	0.596	0.829	0.638	0.640

Tabella 4.7: Confronto tra gli algoritmi con dataset sbilanciato

	Decision Tree	KNN	Logistic Regression	SVM
Mean Absolute Error	0.61	0.40	0.78	0.89
Jaccard similarity score	0.71	0.83	0.60	0.20
Precision Score	0.74	0.83	0.65	0.04
Recall Score	0.74	0.82	0.60	0.20
F-1 Score	0.68	0.78	0.54	0.34

Tabella 4.8: Confronto tra gli algoritmi con dataset bilanciato (SMOTE)

Da un rapido confronto emergono alcuni aspetti interessanti. L'utilizzo di un dataset bilanciato porta ad una diminuzione dei valori dell' *F1-score*. L'errore assoluto nel primo caso si attesta oltre il 50% mentre nel secondo varia tra il 40% e il 60% . Lo Jaccard Index diminuisce dal 68% di media al 58% . In generale mentre la prima tabella mostra un andamento omogeneo e quasi indistinto tra gli algoritmi, la seconda li valuta meglio palesandone una maggiore espressività determinata dalla varie metriche non omogenee. Risulta ovvio che quello che si presta meglio al dataset sia il KNN con numero di neighbors considerati pari a 6. Ora che è stato sancito l'algoritmo di riferimento, bisogna integrarlo nella dashboard. La prima operazione effettuata a tal riguardo è il salvataggio dei valori predetti da usare. Una volta scelto il modello di riferimento, è sovente utilizzare librerie come *Pickle* o *Joblib* per serializzare i risultati e renderli disponibili esternamente. Il motivo è duplice. Innanzitutto la creazione di un modello può essere dispendiosa sia in termini di tempo che in termini computazionali e potrebbe portare ad attese prolungate. Il secondo, invece, riguarda la comodità di condivisione del modello con fonti esterne per poter essere comparato o come in questo caso, per poter essere applicato a nuovi contesti. La libreria usata, *Pickle* è estremamente facile e intuitiva.

```
1 import pickle
2 filename = 'saved_model/KNN.sav'
3 # We save the file HERE
4 pickle.dump(neigh, open(filename, 'wb'))
```

Per ogni algoritmo, è integrata nello script aggiungendo sia il nome, sia l'algoritmo da predire. E' chiaro che non occorrono i valori predetti sul test set ma soltanto il modello prodotto. Al momento di utilizzo, la libreria con un comando, importa il file con estensione *.sav*[32] ed è pronta ad effettuare nuove predizioni su dati nuovi.

```
1 knn_file_location = script_location / 'saved_model/KNearestNeighboars.sav'
2 load_knn_model = pickle.load(open(knn_file_location, 'rb'))
3 ...
4 res = load_knn_model.predict(dfColumnSelected)
```

[32] Tipica nell'utilizzo dei videogiochi.

4.2.3 Integrazione nella Dashboard

Facciamo un passo indietro, e torniamo alla dashboard nella tab "*Predictive Analysis Italy*" . All'apertura della pagina, sono mostrati su mappa i POI (musei, ristoranti, aeroporti, B& B, fermate autobus) più i punti vendita correnti. Le informazioni sono estrapolate con chiamate GET in GeoJson dal Database, esattamente come visto nella sezione "Routing & Views" .

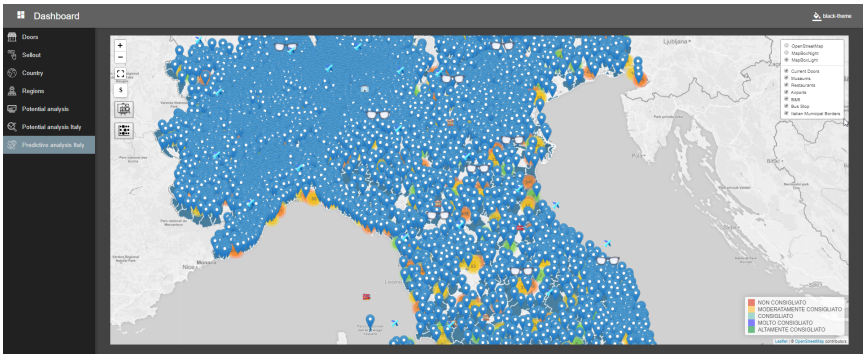


Figura 4.16: Predictive Analysis Italy: punti da predire

Nella mappa, ogni comune è identificato da un centroide di riferimento. Lanciando l'algoritmo con il tasto apposito situato nella parte sinistra della mappa, è possibile iniziare ad effettuare la predizione. Qui riprendiamo la trattazione già affrontata nella sezione *server-side*. In quel frangente, si è descritto il funzionamento di `child_precess`, responsabile della mediazione tra il server Node.js e lo script Python adibito a trovare nuovi punti vendita.

```
router.post('/predictive/putPoints', function (req, res) {
...
const pythonProcess = spawn('python',[path.resolve(__dirname, '../script_py/predict_doors.py')]);
pythonProcess.stdin.write(req.body);
...
});
```

Come si può osservare, viene lanciato uno script chiamato `predict_doors`. Il suo contenuto è il seguente:

```
import sys
# Import pickle to load my model
import pickle

...
...
def main():
```

```
8
9       # Read file
10      script_location = Path(__file__).absolute().parent
11
12      knn_file_location = script_location / 'saved_model/KNearestNeighboars.sav'
13      load_knn_model = pickle.load(open(knn_file_location, 'rb'))
14
15      data = read_in(sys.stdin)
16
17      # Return a Dataframe to manipulate by ML algorithm
18      df = json_normalize(data['features'])
19
20      # Take only the column in which we are interest
21      dfColumnSelected = df[['properties.popolazione_maschile',..., 'properties.dist_center']]
22
23       ...
24       ...
25
26      res = load_knn_model.predict(dfColumnSelected)
27
28
29      df['properties.prediction'] = res
30
31
32      for index, row in df.iterrows():
33          df.loc[index, "geometry.coordinates.long"] = row['geometry.coordinates'][0]
34          df.loc[index, "geometry.coordinates.lat"] = row['geometry.coordinates'][1]
35
36      del df['geometry.coordinates']
37
38      finalResult = convert2geojson(df)
39      readJson(finalResult)
40
41
42  #start process
43  if __name__ == '__main__':
44      main()
```

Dopo aver aggiunto il modello selezionato, sono stati caricati i dati provenienti da server. Essi sono in formato GeoJson. Di quello che arriva allo script, solo una parte è di interesse, in particolar modo le properties che contengono le informazioni su cui è stato costruito il modello. Puliti i dati, è possibile passarli al modello caricato `KNN`, il quale provvederà ad effettuare le predizioni. Una volta ottenuto il risultato, verrà aggiunta una nuova colonna con la predizione, ricreato il GeoJson originale, e inviati i risultati al client con `readJson(finalResult)`.

```
1  ...
2  pythonProcess.stdout.on('data', (data) => {
3          if (!res.headersSent) res.json(data.toString(), 0, 500);
4      });
5  ...
```

Il Server, provvederà ad inviare i risultati al client ottenendo la mappa con i punti dove è consigliato o non consigliato aprire un nuovo negozio.

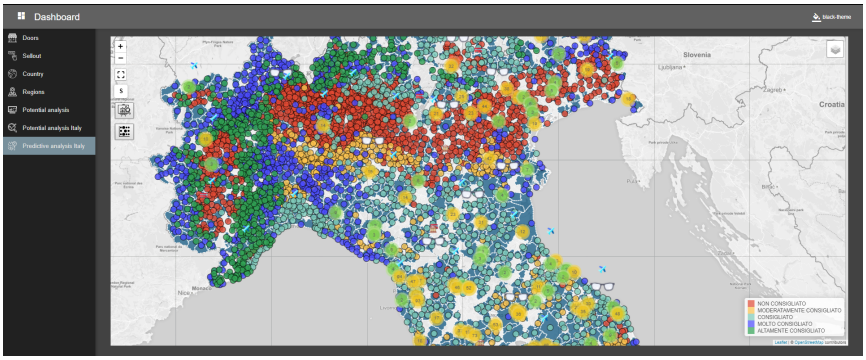


Figura 4.17: Punti predetti dall'algoritmo di ML KNN

L'output mostra su mappa per tutti i comuni italiani la predittività sulla base delle informazioni estrapolate dai POI. Nella Fig. 4.17 lo slider è stato impostato a 1 km per cui si chiede alla dashboard di non escludere nessuna zona, ma di evidenziare tutte le aree a dispetto degli attuali punti vendita. Grazie al menù in alto a destra, è possibile fare pulizia di ciò che non è ritenuto rilevante, come i POI, le Door e i punti non consigliati. Con l'ausilio

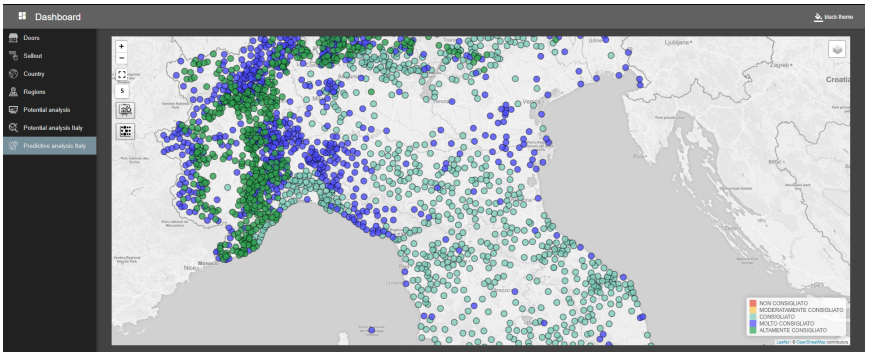
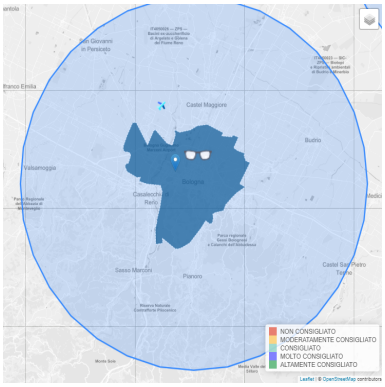


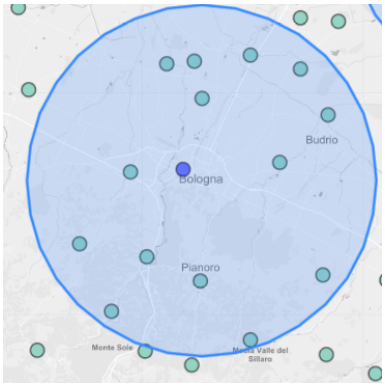
Figura 4.18: Visualizzazione dei punti di interesse

dello slider (Fig. 4.19a) è possibile delimitare il range entro cui effettuare la ricerca di ipotetici nuovi punti vendita. L'aggiunta di questa features è spiegata dalla volontà di voler escludere le zone che sono già coperte da door esistenti (analisi dell'offerta) ed effettuare la ricerca solo per quelle che si trovano ad una certa distanza da esse, definita dall'utente. In questo modo si provano a scovare le aree potenzialmente appetibili.

E' chiaro che sulla base della predizione ottenuta, si può scegliere se aumentare o diminuire il range. Ad esempio, intorno l'area di Bologna, settando il raggio a 61 km, il modello esclude i punti interni (Fig. 4.19b, Fig.4.19c).



(a) Scelta del Range

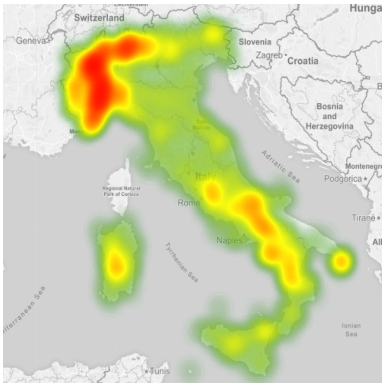


(b) Esclusione dei Punti dal range

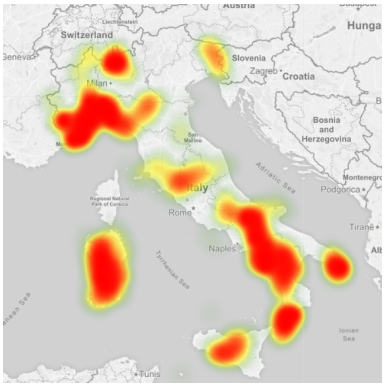
(c) Output esclusione

Figura 4.19: Analisi dell'offerta

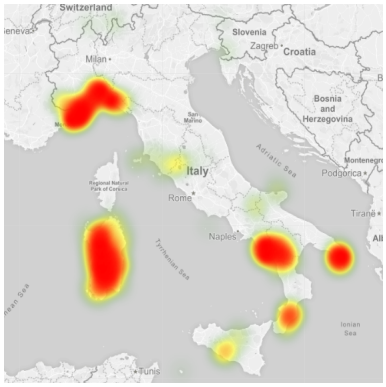
Infine con l'ausilio di una Heatmap, si può sommariamente vedere le zone italiane più predisposte all'apertura di nuove door.



(a) Nuove aree (Complessive)



(b) Nuove aree (slider a 60km)



(c) Visualizzazione nuove aree (slider a 90km)

Figura 4.20: Visualizzazione aree di nuovi punti vendita con Heatmap

Si può desumere, quindi, che a livello italiano, le zone dove è maggiormente consigliato aprire un nuovo punto vendita sono situata intorno all'area nord-est del paese. In aggiunta a queste, zone centro meridionali, presentano margini di ampliamento dell'offerta se comprati alla situazione attuale. Infatti, soprattutto nell'ultima zona, non sono molte le door presenti.
Aumentando il range dello slider a 60 e 90 km si evidenzia come questa scelta si assottiglia. La liguria, la Sardegna e parti della Calabria e Puglia sono zone a largo potenziale di investimento.

Conclusioni

La Business Intelligence racchiude i processi che cercano di scandagliare informazioni, in contesti aziendali, per individuare relazioni nascoste tra i dati al fine di accrescere il rendimento economico aziendale,portando così all'identificazione di nuove aree, analizzare il comportamento dei clienti e della concorrenza, oltre che monitorare le prestazioni. Per questa serie di ragioni, il tool di cui si serve sono generalmente report, analisi statistica e query. Ciononostante la BI presenta dei limiti. Quello che analizza sono dati in forma atomica gestiti all'interno di Data Warehouse che non danno informazioni sulla posizione che assumono nello spazio se non in maniera statica. La Location Intelligence sopperisce a questo. La LI è la branca che unisce l'analisi dei dati con i loro attributi spaziali per influenzare il processo decisionale per l'incremento dell'efficienza operativa, la crescita delle entrate o avere una migliore efficacia di gestione. Mentre la BI, cerca di rispondere a domande "Chi", "Cosa", "Quando", la LI si concentra sul "Dove". Per la lavorazione sulla posizione geografica dei dati si usano i Database Spaziali che tramite query spaziali riescano ad effettuare operazioni spaziali sui dati. L'obiettivo di questo lavoro di tesi, è stato la creazione di una dashboard che mettesse insieme questi due aspetti e che mostrasse su mappa informazioni utili relativi ai dati di un cliente impegnato nel fashion retail sia per dare informazioni descrittive sugli attuali punti vendita, ma soprattutto per analizzare il loro potenziale e poter stabilire con tecniche di Machine Learning potenziali nuovi negozi sfruttando le relazioni esistenti con luoghi di interesse (POI). La Dashboard è stata costruita con Angular, framework open source per la creazione di applicazioni lato client side, e con Leaflet, libreria Javascript leader per l'interazione con le mappe. La prima parte del

lavoro, si è focalizzata sulla costruzione della dashboard a livello strutturale sfruttando i pattern architetturali soventi in questi casi. Nella seconda parte invece, ci si è soffermati su tre tipi di analisi: descrittiva, potenziale e predittiva. Il primo tipo di analisi, è un'analisi statica che mostra i punti vendita con il loro relativo fatturato. Nelle tab preposte, con l'ausilio di grafici sono evidenziati i migliori punti vendita, i migliori marchi e materiali venduti. Il secondo tipo di analisi, invece, quantifica l'apporto di ogni caratteristica dei prodotti venduti sul fatturato. L'obiettivo è capire se le performance, seppur alte, siano al massimo del potenziale. In tal senso, sono state fatte due tipi di analisi. Il primo a livello mondiale mentre il secondo a livello italiano. In quest'ultimo caso oltre i dati strettamente legati alle caratteristiche del prodotto, sono state considerati quelli demografici come "numero medio di componenti per famiglia", "popolazione maschile" etc.. Quanto emerge dall'analisi è che a livello globale, le vendite di prodotti per bambini, siano maggiormente influenti sul fatturato, rispetto quelli da donna e da uomo. Inoltre, l'acquisto di prodotti con colorazioni come arancio, azzurro abbiano un elevato impatto. A livello di brand il marchio Puma è quello maggiormente influente. Per quanto concerne l'analisi italiana, la situazione varia. Sia per il brand, colorazione, e genere, è abbastanza omogenea. Qui, quello che salta all'occhio è l'influenza che assumono le variabili demografiche. Ovviamente in presenza di zone con alto reddito, le vendite aumentano, ma ciò di più interessante è che la componente familiare ha un influsso negativo, ossia in zone con numero di famiglie elevate, le vendite calano. L'ultimo tipo di analisi, quella predittiva, aiuta gli utenti a trovare nuovi luoghi idonei per l'apertura di nuovi punti vendita. La metodologia utilizzata, ha previsto la raccolta di punti di interesse (POI) come musei, alberghi, ristoranti dati in pasto a vari algoritmi di Machine Learning. Sulla base di un confronto realizzato con metriche idonee alla valutazione di algoritmi, è risultato che il migliore è il K-nearest neighbors (KNN), con un valore di k pari a 6. Dal modello che n'è derivato, sono stati dati in input potenziali nuovi punti vendita che considerano sia la domanda attualmente esistente rispetto gli attuali punti vendita e sia la distanza dai POI. Ciò che ne è risultato mediante l'utilizzo di una heatmap è che le zone idealmente idonee all'apertura di un nuovo punto vendita, sono la Sardegna e la Ligu-

ria, più parti di Calabria e Puglia. Una delle maggiori difficoltà incontrate durante lo sviluppo è stato sicuramente il reperimento di dati. L'offerta di piattaforme open data risulta essere ancora molto acerba. Studi analoghi effettuati, acquisiscono i dati tramite applicazioni come Foursquare e a livello italiano non esiste un'applicazione analoga. L'utilizzo di dati real time che sfruttino il segnale del GPS può essere sicuramente uno sviluppo che in prospettiva può migliorare di molto l'analisi a livello predittivo. Inoltre, sarebbe molto interessante poter effettuare un'analisi che consideri i dati dei social network. Allo stato attuale, l'integrazione di questa tipologia di dati, non è stata possibile per due ragioni. In primo luogo perché gli utenti di piattaforme come Instagram, raramente associano l'acquisto di un prodotto alla posizione. In secondo luogo, dallo scoppio del caso Cambridge Analytica, è stata avviata una politica di profonda restrizione all'utilizzo di dati personali. Per tale ragione, l'utilizzo delle API necessarie, richiede il rilascio di numerosi permessi che avrebbe rallentato di molto il lavoro.

Appendice A

Installazione PostGIS

A.1 Windows

- Sul sito PostreSQL scaricare l'ultima versione EnterpriseDB
- Per installare PostGIS, su Osgeo selezionare la corretta versione basandosi sulla versione di Postgres scaricata

A.2 Linux

Come il sito Osgeo suggerisce,

- Verificare quale versione di ubuntu si sta usando:

```
1  sudo lsb_release -a
```

- Importare la chiave di repository:

```
1  sudo apt-get install curl ca-certificates gnupg curl
       https://www.postgresql.org/media/keys/ACCC4CF8.asc |
       sudo apt-key add -
```

- Individuare la versione corrente di Linux che si sta utilizzando

```
1  lsb_release -c
```

- Dare il comando

```
sudo sh -c 'echo "deb
    http://apt.postgresql.org/pub/repos/apt/ $(lsb_release
    -cs)-pgdg main" > /etc/apt/sources.list.d/pgdg.list'
```

- Aggiornare il repository:

```
    sudo apt-get update
```

- Installare *pgadmin*:

```
    sudo apt-get install postgresql-10 pgadmin4
```

Per maggiori informazioni si può consultare il sito

A.2.1 Accedere al DB

- Accedere al Database

```
sudo -u postgres psql
```

- Settare la password di default:

```
ALTER USER postgres PASSWORD 'newPassword';
```

A.3 Configurazione DB

A.3.1 Creare un DB da PgAdmin

- In pgAdmin `'Create a Server...'` come ad esempio: `PostgresSQL`
  - `host: localhost`
  - `Port: 5432`
  - `database: postgres`
  - `Username: postgres`
  - `Password: 'newPassword'`

- Creare un nuovo Database `Nome_Database`

A.3.2 Creare un DB da Python

```
#REF:https://www.programcreek.com/python/example/2032/psycopg2.connect
#Example 3
class Database():

    def __init__(self, name, user, passwd=None, reconnect=False):
        """
        name: the name of the database to connect to
        user: the username to use to connect
        base: a file containing the SQL implementation for Ravel's
            base
        passwd: the password to connect to the database
        reconnect: true to connect to an existing database setup,
            false
        to load a new instance of Ravel's base into the database"""
        self.name = name.lower()
        self.user = user
        self.passwd = passwd
        self.cleaned = not reconnect
        self._cursor = None
        self._conn = None

        self.create()

    def create(self):
        """If not created, create a database with the name specified
            in
           the constructor"""
        conn = None
        try:
            conn = psycopg2.connect(database="postgres",
                                    user=self.user,
                                    password=self.passwd)
            conn.set_isolation_level(ISOLEVEL)
            cursor = conn.cursor()
```

```
32          cursor.execute("SELECT datname FROM pg_database WHERE " +
33                         "datistemplate = false;")
34          fetch = cursor.fetchall()
35          dblist = [fetch[i][0] for i in range(len(fetch))]
36          if self.name not in dblist:
37              cursor.execute("CREATE DATABASE %s;" % self.name)
38              logger.debug("created databse %s", self.name)
39              # Add the extension for the created Database
40              print("CREATED DB " + self.name)
41              self.add_extension()
42          else: print("CONNECTION ESTABLISH WITH DB...")
43      except psycopg2.DatabaseError as e:
44          logger.warning("error creating database: %s", e)
45      finally:
46          conn.close()
47
48  @property
49  def conn(self):
50      "returns: a psycopg2 connection to the PostgreSQL database"
51      if not self._conn or self._conn.closed:
52          self._conn = psycopg2.connect(database=self.name,
53                                        user=self.user,
54                                        password=self.passwd)
55          self._conn.set_isolation_level(ISOLEVEL)
56      return self._conn
57
58  @property
59  def cursor(self):
60      """returns: a psycopg2 cursor from DB for the PostgreSQL
61      database"""
62      if not self._cursor or self._cursor.closed:
63          self._cursor = self.conn.cursor()
64      return self._cursor
65
66
67  def add_extension(self):
68      """If not already added, add extensions"""
69      try:
```

```
70          extension_util =
                ["postgis","postgis_topology","postgis_sfcgal",
71          "fuzzystrmatch","address_standardizer",
72          "address_standardizer_data_us",
                "postgis_tiger_geocoder","pgrouting","ogr_fdw",
73          "pointcloud", "pointcloud_postgis"]
74          self.cursor.execute("SELECT extname FROM pg_extension;")
75          fetch = self.cursor.fetchall()
76          # print(fetch)
77          for ext in extension_util:
78              if (ext in fetch)==False:
79                  self.cursor.execute("CREATE EXTENSION IF NOT
                        EXISTS {};".format(ext))
80                  print("ADD EXTENSION: "+ ext)
81                  logger.debug("created extensions", ext)
82      except psycopg2.DatabaseError as e:
83          logger.warning("error loading extensions: %s", e)
```

A.3.3 Esportare Database in Postgres usando pgAdmin

- Tasto destro sulla tabella desiderata e selezionare l'opzione `Backup`
- Settare `Filepath/Filename` e cliccare sul formato deisderato (PLAIN o TAR)
- Nella seconda tab, selezionare `Pre Data` `Data` `Post Data`
- Premere il tasto Backup

A.3.4 Importare un Database usando pgAdmin

- Creare un nuovo Database con le stesse credenziali
- Tasto destro sul database creato
- Cliccare su `Restore`
- Selezionare il percorso del Database (Per questo progetto, nell'esportarlo ho usato l'esensione .tar)

- Restore

NOTA: Anche se viene mostrato un errore di mancata compilazione, fare Refresh sullo Schema e vedere se il Database appare.

A.3.5 Esportare tabelle in Postgres usando pgAdmin

- Tasto destro sulla tabella e selezionare l'opzione `Import/Export..`
- Selezionare il Percorso dove salvare il file
- Selezionare il formato che si preferisce (è suggerito quello CSV)
- Settare se nell'Export si vogliono salvare Le intestazioni di colonna, l'ID e la regola di separazione (più eventuali altre regole presenti nella seconda TAB come l'esclusione di specifiche colonne)
- Dare `Ok`

A.3.6 Importare tabelle in Postgres usando pgAdmin

- Tasto destro su una delle tabelle e scegliere `Create`
- Creare una nuova tabella che presenti gli stessi requisiti della tabella esportata
- Dare il comando `Ok`
- Selezionare una delle tabelle e scegliere `Import/Export..`
- Selezionare il cursore su `Import`
- Selezionare il file che si è esportato o che si vuole immettere ex novo.
- Selezionare il formato con cui lo si sta importando
- Se si stanno importando ance il numero di riga e l'instestazione di colonna settare l'OID e l'Header
- Selezionare il delimitatore più eventuali altre opzioni
- Dare `Ok`

Appendice B

Lavorare con Python

B.1 Lavorare con PostGIS tramite Python

- Installare `psycopg2`

```
pip install psycopg2
```

- **NOTA:** Per verificare la versione:

```
python -c "import psycopg2;
    print(psycopg2.\_\_version\_\_)"}
```

- Alternativamente è possible lavorare con Python PostGIs come mostrato in `test_postgis.py`

```
pip install cython
pip install psycopg2
pip install postgis
```

B.2 PostGIS: Query Utili

Individuare le tabelle che non hanno indice spaziale

– Selezionare le tabelle postgis che non hanno indice spaziale:

```
SELECT g.*
FROM
  (SELECT
     n.nspname,
     c.relname,
     c.oid AS relid,
     a.attname,
     a.attnum
   FROM pg_attribute a
   INNER JOIN pg_class c ON (a.attrelid=c.oid)
   INNER JOIN pg_type t ON (a.atttypid=t.oid)
   INNER JOIN pg_namespace n ON (c.relnamespace=n.oid)
   WHERE t.typname='geometry'
   AND   c.relkind='r'
 ) g
LEFT JOIN pg_index i ON (g.relid = i.indrelid AND
    g.attnum = ANY(i.indkey))
WHERE i IS NULL;
```

B.3 Psql: comandi utili da Shell

- Visualizzare lista dei db:
  - `\list`
  - `\l`
- Connessione da command line al db:
  - `\connect nome_db`
  - `\c`
- lista tabelle all'interno di un db:
  - `\dt`

B.4 Python: Lettura degli Excel

```
pip install xlrd
pip install xlsxwriter
```

B.5 Python: Lavorare con i csv

```
pip install pandas

# per la prograss bar
pip install tqdm
```

Appendice C

QGIS

C.1 Dove Scaricare gli Shapefile

- Download World Shape File
- Download European Shape File

Per importare lo Shape file nel Database, eseguire i seguenti comandi

- Scaricare QGIS Standalone
- Aprire QGIS (Current version: 3.6)
- Dal pannello sinistro, selezionare con il tasto destro `New Connection...`
- Riempire i campi in accordo con le proprie credenziali
- Inserire un nome identificativo;
- Lasciare vuoto il campo `Service`;
- `Test Connection;`
- Premere OK. Adesso in prossimità del pannello sinistro è possibile vedere le informazioni del DB.

C.2 Come caricare Shape File

- In `Layer Panel` aggiungi il tuo Shape File;

- una volta caricato, nella Menù Bar selezionare `Database`;
- Selezionare `DB Manager`;
- Aprire una connessione PostGIS e aggiungere le proprie credenziali;
- Aprire `Public Schema` e Selezionare `Import Layer/File`:
  - **Input:** è il nome dello Shapefile
  - **Table:** è il nome della tabella. Dare un nome, se si vuole creare una nuova tabella;
  - Selezionare **Primary Key**;
  - Selezionare **Geometry Column**;
  - Selezionare **Create Spatial Index**;
  - Selezionare **Convert field names to lowercase**
  - **NOTA:** Non selezionare `SOURCE SRID`. In questo modo, QGIS caricata lo shape file con l'SRID di default.

NOTA: Tieni a mente che ogni Shapefile puà avere un diverso SRID. Per verificare se quello che si sta usando è lo stesso per ogni shape usato, dal pannello sinistro provare a sovrepporre i vari layer con i quali si sta lavorando. Se si sovrappongono usano lo stesso SRID.

C.3 Verifica sistema coordinate dal BD

Eventualmente per visualizzare lo SRID da PostGIS:

```
1   ST_SRID(GEOM_COLUMN_NAME) FROM TABLE_NAME LIMIT 1;
```

C.4 Come aggiornare gli attributi da QGIS

A volte prima di essere importato in PostGIS può essere utili rinominare i campi direttamente da QGIS

- Selezionare `Processing`

- Selezionare `Toolbox`
- Selezionare `Vector Table` > `Refactor Field`

Una volta finito, cliccare su `Esegui`

Bibliografia

[1] Leaders. *The world's most valuable resource is no longer oil, but data.* 2017. URL: https://www.economist.com/leaders/2017/05/06/the-worlds-most-valuable-resource-is-no-longer-oil-but-data.

[2] McAfee A e Brynjolfsson E. «Big data: the management revolution.» In: *Harvard business review* 90, 23074865 (10 ott. 2012), pp. 60–6, 68, 128.

[3] «Business Intelligence and Analytics: From Big Data to Big Impact». In: *MIS Quarterly* 36.4 (2012), p. 1165.

[4] Gartner. *Analytics and Business Intelligence (ABI).* 2019. URL: https://www.gartner.com/it-glossary/business-intelligence-bi/.

[5] Tableau. *Cos'è la business intelligence e perché è importante?* 2018. URL: https://www.tableau.com/it-it/learn/articles/business-intelligence.

[6] Galigeo. *The 3 cores of Location Intelligence.* 2019.

[7] «Overview of Oracle Spatial». In: *Pro Oracle Spatial for Oracle Database 11g.* Apress, 2007, pp. 19–36.

[8] Istituto Superiore Sanità. *Colera - EpiCentro - Istituto Superiore di Sanità.* 2019. URL: https://www.epicentro.iss.it/colera/.

[9] Tom Koch e Kenneth Denike. «Crediting his critics' concerns: Remaking John Snow's map of Broad Street cholera, 1854». In: *Social Science & Medicine* 69.8 (ott. 2009), pp. 1246–1251.

[10] Ralf Hartmut Güting. «An introduction to spatial database systems». In: *The VLDB Journal* 3.4 (ott. 1994), pp. 357–399.

[11] Petr Vanìcek e Robin R. Steeves. «Transformation of coordinates between two horizontal geodetic datums». In: *Journal of Geodesy* 70.11 (set. 1996), pp. 740–745.

[12] A. R. Hinks. «Maps of the World on an Oblique Mercator Projection». In: *The Geographical Journal* 95.5 (mag. 1940), p. 381.

[13] B. Jenny. «Adaptive Composite Map Projections». In: *IEEE Transactions on Visualization and Computer Graphics* 18.12 (dic. 2012), pp. 2575–2582.

[14] Scott Davis. *GIS for web developers*. 2007. Cap. 4.

[15] Dmytro Karamshuk et al. «Geo-Spotting: Mining Online Location-based Services for Optimal Retail Store Placement». In: (giu. 2013). arXiv: 1306.1704v2 [cs.SI].

[16] Longbiao Chen et al. «Bike sharing station placement leveraging heterogeneous urban open data». In: *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing - UbiComp '15*. ACM Press, 2015.

[17] Vinayak Gopinath e Ragul Radhakrishnan. *Using location analytics for optimal site selection*. 2018. URL: https://www.latentview.com/blog/using-location-analytics-for-optimal-site-selection/.

[18] Muthukumar Kumar. *2019 Top 100 Geospatial Companies and Startups List*. 2019. URL: https://geoawesomeness.com/2019-top-100-geospatial-companies-startups/.

[19] Theodor Chichirita. *Structure of an Angular 7 CLI project*. 2018. URL: https://medium.com/@theodor.chichirita/structure-of-an-angular-7-cli-project-13e2889b6ee7.

[20] Sahosoft. *Learn Angular 7 easy to learn*. 2018. URL: https://www.sahosofttutorials.com/Course/Angular7/116/.

[21] Angular. *Architecture overview*. 2019. URL: https://angular.io/guide/architecture.

[22] Tomas Trajan. *6 Best Practices E Pro Tips when using Angular CLI*. 2017. URL: https://medium.com/@tomastrajan/6-best-practices-pro-tips-for-angular-cli-better-developer-experience-7b328bc9db81.

[23] Zeljko Radic. *Best Practices for Writing Angular 6 Apps*. 2018. URL: https://blog.usejournal.com/best-practices-for-writing-angular-6-apps-e6d3c0f6c7c1.

[24] Mathis Garberg. *How to define a highly scalable folder structure for your Angular project*. 2018. URL: https://itnext.io/choosing-a-highly-scalable-folder-structure-in-angular-d987de65ec7.

[25] Tomas Trajan. *Total Guide To Angular 6+ Dependency Injection — providedIn vs providers:[ ]* 2018. URL: https://medium.com/@tomastrajan/total-guide-to-angular-6-dependency-injection-providedin-vs-providers-85b7a347b59f.

[26] Angular. *Intro to Modules*. 2019. URL: https://angular.io/guide/architecture-modules#introduction-to-modules.

[27] Angular Error Handling. *Angular Error Handling*. 2019. URL: https://angular.io/tutorial/toh-pt6#error-handling.

[28] Max Koretskyi. *Avoiding common confusions with modules in Angular*. 2017. URL: https://blog.angularindepth.com/avoiding-common-confusions-with-modules-in-angular-ada070e6891f.

[29] Caerus Karu. *Configuring Angular Layout*. 2018. URL: https://github.com/angular/flex-layout/wiki/Configuration.

[30] Luuk Gruijs. *Understanding, creating and subscribing to observables in Angular*. 2017. URL: https://medium.com/@luukgruijs/understanding-creating-and-subscribing-to-observables-in-angular-426dbf0b04a3.

[31] Manali Matkar. *Sharing Data between Angular Components*. 2019. URL: https://www.intersysconsulting.com/blog/angular-components/.

[32] Miro Koczka. *3 ways to communicate between Angular components*. 2017. URL: https://medium.com/@mirokoczka/3-ways-to-communicate-between-angular-components-a1e3f3304ecb.

[33] Antonio Faienza. *Build an open-source web platform based on OpenLayers, Express and NoSQL DB*. 2019.

[34] Selva Prabhakaran. *Linear Regression*. 2016. URL: `http://r-statistics.co/Linear-Regression.html`.

[35] SANFORD WEISBERG. *Applied Linear Regression*. A cura di Wiley-Interscience. 1980. Cap. Capitolo 2.

[36] Robert B. Bendel e A. A. Afifi. «Comparison of Stopping Rules in Forward "Stepwise" Regression». In: *Journal of the American Statistical Association* 72.357 (mar. 1977), pp. 46–53.

[37] kassambara. *Stepwise Regression Essentials in R*. 2018. URL: `http://www.sthda.com/english/articles/37-model-selection-essentials-in-r/154-stepwise-regression-essentials-in-r/`.

[38] Georgios Fotopoulos e Helen Louri. In: *Small Business Economics* 14.4 (2000), pp. 311–321.

[39] Jacek Malczewski. «GIS-based multicriteria decision analysis: a survey of the literature». In: *International Journal of Geographical Information Science* 20.7 (ago. 2006), pp. 703–726.

[40] Sebastian Baumbach et al. «Geospatial Customer, Competitor and Supplier Analysis for Site Selection of Supermarkets». In: *Proceedings of the 2019 2nd International Conference on Geoinformatics and Data Analysis - ICGDA 2019*. ACM Press, 2019.

[41] Hui-Jia Yee, Choo-Yee Ting e Chiung Ching Ho. «Optimal geospatial features for sales analytics». In: Author(s), 2018.

[42] Pablo Jensen. «Network-based predictions of retail store commercial categories and optimal locations». In: *Physical Review E* 74.3 (set. 2006).

[43] Stephen G. Odaibo. «Is 'Unsupervised Learning' a Misconceived Term?» In: (apr. 2019). arXiv: `1904.03259v1 [cs.LG]`.

[44] Hui Xiong et al. «Pattern Preserving Clustering». In: *Encyclopedia of Data Warehousing and Mining, Second Edition*. IGI Global, 2008, pp. 1505–1510.

[45] «ADASYN: Adaptive synthetic sampling approach for imbalanced learning». In: *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*. IEEE, giu. 2008.

[46] Hui Han, Wen-Yuan Wang e Bing-Huan Mao. «Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning». In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2005, pp. 878–887.

[47] Haridas N. *Outlier removal clustring*. 2016. URL: `https://haridas.in/outlier-removal-clustering.html`.

[48] Natasha Sharma. *Ways to Detect and Remove the Outliers*. 2018. URL: `https://towardsdatascience.com/ways-to-detect-and-remove-the-outliers-404d16608dba`.

[49] Mengwen Xu et al. «Demand driven store site selection via multiple spatial-temporal data». In: *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems - GIS '16*. ACM Press, 2016.

Ringraziamenti

Caro Papà,
è così difficile scrivere queste parole. Ho sempre rimandato questo momento perché non mi sono mai sentito pronto ad affrontarlo e l'ho sempre temuto, ma ahimè tu mi hai insegnato ad affrontare le paure con coraggio.
Avrei voluto che tu fossi qui con me, a sostenermi e a tranquillizzarmi come solo tu sapevi fare. Bastava una tua parola per sentirmi sicuro. Ma purtroppo il destino ha preso una piega diversa da come ce lo aspettavamo.
Da quel maledetto 9 aprile tu non ci sei più, sei volato via e con te la mia spensieratezza..
Da quel maledetto 9 aprile non c'è giorno che io non ti pensi.
Da quel maledetto 9 aprile non c'è giorno che tu non accompagni le mie giornate, i miei viaggi. Dalla mattina presto alla sera tardi, passando dalle partite dell'Inter a tutto il resto. Tu sei sempre con me.
Ho avuto la fortuna di avere al mio fianco, persone vere che mi hanno sostenuto. Sicuramente la più importante è Paola. Appena l'hai conosciuta ti sei subito accorto di quanto fosse speciale, e ogni giorno che passa lo è sempre più. A lei devo tutto. E' come se gli avessi indicato come tenermi per mano mentre soffro in silenzio.
I miei amici "bolognesi" (Anna,Fabio, Vittorio, Paola, Antonio, Fabio, Coppo, Ricky, Francesca, Giulio, Simo e tutti gli altri), che mi hanno consolato non facendomi mai sentire solo, trattandomi come sempre hanno fatto ma non facendomi mai mancare il calore di cui avevo bisogno.
I miei colleghi, Da Tommy a Tommy, passando per tutti gli altri, che mi fanno sentire a casa quando sono a lavoro. In quell'ambiente più che colleghi ho trovato amici di cui fidarmi a dispetto del ruolo, come Fede, il quale

più che un "superiore" si è dimostrato un fratello.
I miei amici di sempre Attilio, Giuseppe, Raffaele, Michele, e gli altri tutti, che ti hanno conosciuto e hanno pianto con me.
E infine mamma, Matteo e Giulia che ti pensano sempre e tutto il resto della famiglia che fanno il massimo per far sembrare normale, una situazione che normale non è, sforzandosi di far apparire piccolo qualsiasi problema.
Spero che in qualunque parte tu sia, tu mi stia vedendo e che nonostante sbagli tu mi sappia indicare la retta via da seguire.

Tuo per sempre,
Antonio