Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Informatica

Machine Learning come supporto per la valutazione dei requisiti agili

Relatore:
Chiar.mo Prof.
Paolo Ciancarini

Presentata da:
Giulio Zhou

Correlatori:
Dott. Daniel Russo.
Dott. Vincenzo Lomonaco

Sessione I
Anno Accademico 2016/2017

Sommario

L'approccio delle metodologie agili ai requisiti è meno rigoroso rispetto al processo tradizionale dell'Ingegneria dei Requisiti (RE). Tuttavia, ha la pretesa di riuscire ad adattarsi con più facilità in un ambiente in continuo mutamento. Questa capacità di adattamento è data dalla pianificazione e dall'analisi dei requisiti durante tutto il processo di sviluppo del software. Nei metodi agili come *Extreme Programming* (XP) e *Scrum*, la valutazione e stima dei requisiti viene effettuata dai programmatori a ogni iterazione durante il *Planning Game*. In questa tesi si valuteranno gli algoritmi del Machine Learning (ML) come supporto a questa fase. Gli esperimenti verranno effettuati su un dataset di requisiti Scrum per un progetto fittizio appositamente ideato e si articoleranno principalmente in due fasi: l'analisi dei dati, con algoritmi di Elaborazione del Linguaggio naturale (NLP) e di ML non supervisionati, e la stima dei requisiti, attraverso algoritmi di ML supervisionati. Si scopre quindi che il pattern preponderante delle informazioni del dataset è quello lineare. La precisione delle predizioni con un input formato da una combinazione di stime (es. predire lo sforzo a partire da stime quali le linee di codice scritte) può arrivare allo 0.9985, mentre utilizzando il modello predittivo generato a partire dai campi testuali, la precisione arriva nel migliore dei casi allo 0.3360. Questo valore può essere migliorato abbassando la complessità della stima da ottenere, infatti, valutare il livello di difficoltà del requisito permetterà una precisione anche pari a 0.68.

Indice

Elenco delle figure

Capitolo 1

Introduzione

Tra tutte le fasi del processo di sviluppo di un software, quella che concerne i requisiti è senza alcun dubbio la più critica. La causa principale del fallimento di un progetto software è, in effetti, la carenza o inaccuratezza dei requisiti[21]. Procedere alla fase d'implementazione con requisiti poco chiari o incompleti può porre il progetto a un rischio considerevole. Il costo per correggere errori nei requisiti è elevato, soprattutto se scoperti nelle fasi più avanzate del progetto. Non c'è da stupirsi se a essi è quindi dedicata un'intera branca dell'Ingegneria del Software: l'Ingegneria dei Requisiti (RE).

L'Ingegneria dei Requisiti è un processo rigoroso composto da diverse fasi ben distinte: elicitazione, analisi, documentazione, validazione e organizzazione [42]. In un processo di sviluppo tradizionale come quello *Waterfall*, le attività RE vengono svolte prima dell'inizio dell'implementazione, senza offrire la possibilità di effettuare una revisione dei requisiti nelle fasi successive. Hoffman conclude il suo studio[21] affermando che un software di successo necessita una rivisitazione costante dei requisiti in quanto essi sono mutevoli. Un approccio Waterfall è quindi semplicemente irrealistico. Requisiti ben realizzati sono frutto di una costante interazione tra il team di sviluppo e il cliente, il quale deve essere informato sui progressi del progetto, attraverso frequenti aggiornamenti e prototipi. Questo approccio è sostenuto dalle metodologie agili emerse negli ultimi due decenni, le quali, nel "Manifesto

Agile"[14], affermano di prediligere il "rispondere al cambiamento più che seguire un piano" e "la collaborazione con il cliente più che la negoziazione dei contratti".

L'analisi e la valutazione dei requisiti sono attività importanti quanto quella di elicitazione. Lo studio effettuato da Capers Jones[25] ha mostrato che i progetti che non sono riusciti a rispettare i limiti temporali o di budget o che sono stati cancellati prima del loro completamento sono accomunati da diverse problematiche tra cui una superficiale pianificazione del progetto e una scarsa stima dei costi. Nei metodi agili come *Extreme Programming* (XP) e *Scrum*, queste fasi vengono effettuate dai programmatori a ogni iterazione durante il *Planning Game* durante il quale si pianificano i requisiti da implementare e si effettuano le stime di costo, tempo e dimensione[9].

Attualmente, le stime vengono effettuate attraverso attività informali come il *Planning Poker*. Scopo di questa tesi è quello di proporre un metodo automatizzato di supporto al Planning Game, avvalendosi del *Machine Learning* (ML) per valutare l'impatto di un requisito sul software o sul processo di sviluppo, aumentando di conseguenza la produttività degli sviluppatore.

Gli algoritmi di Machine Learning si sono dimostrati di grande valore pratico in una varietà di domini. Nell'Ingegneria del Software, il ML trova un terreno fertile in quanto molte attività legate allo sviluppo e mantenimento del software possono essere formulate come problemi di apprendimento[47]. Inoltre, diversi studi sono stati effettuati in merito all'utilizzo del ML in ambito dell'Ingegneria dei Requisiti, ad esempio automatizzando il processo di prioritizzazione[36]. Tuttavia, non vi sono molti studi in merito alle applicazioni del ML nel campo dei requisiti software agili.

Gli esperimenti verranno effettuati su un dataset di requisiti Scrum per un progetto fittizio appositamente ideato. Si articoleranno principalmente in due fasi: l'analisi dei dati, con algoritmi di Elaborazione del Linguaggio naturale (NLP) e di ML non supervisionati, e la stima dei requisiti, attraverso algoritmi di ML supervisionati. Nell'elaborato si valuterà la fattibilità di questo approccio e si effettueranno considerazioni in merito ai possibili

miglioramenti rispetto a quelli proposti.

La tesi sarà quindi così composta:

- Capitolo 2: fornirà una panoramica generale dell'Ingegneria dei Requisiti, di Scrum e del Machine Learning;
- Capitolo 3: illustrerà il dominio e i componenti del dataset utilizzato come input per gli algoritmi di ML;
- Capitolo 4: analizzerà gli algoritmi utilizzati negli esperimenti e fornirà i risultati ottenuti;
- Conclusioni: riassumerà i risultati e fornirà gli spunti per poterli migliorare.

Capitolo 2

Il contesto

2.1 L'Ingegneria dei Requisiti

L'Ingegneria dei Requisiti è un processo tradizionale dell'Ingegneria del Software con lo scopo d'individuare, analizzare, documentare e validare i requisiti di un sistema da sviluppare[34], fornendo un modello del problema e di ciò di cui si ha bisogno, in modo chiaro, consistente, preciso e non ambiguo.

Si vuole quindi definire ciò che si vuol creare, prima dell'inizio dello sviluppo, in quanto più gli errori sono scoperti in modo tardivo, maggiore sarà il costo delle eventuali correzioni[4].

2.1.1 I requisiti

Nell' "*IEEE Standard Glossary of Software Engineering Terminology*"[23] un requisito è definito come:

1. una condizione o un'abilità necessaria all'utente per risolvere un problema o raggiungere un obiettivo;

2. una condizione o un'abilità che un sistema o un componente del sistema deve possedere per soddisfare un contratto, uno standard, una specifica, o altri documenti formali;

3. una rappresentazione documentata delle condizioni o abilità dei punti precedenti.

In sostanza, i requisiti sono descrizioni di ciò che il sistema dovrebbe fare, i servizi che fornisce e i vincoli sulle sue funzionalità. I requisiti riflettono i bisogni dei clienti per un sistema, il quale deve avere un qualche scopo [42].

I requisiti sono spesso raggruppati in due categorie principali:

- *Requisiti funzionali*: descrivono le funzionalità del sistema, come questo deve reagire a particolari input e come deve comportarsi in determinate situazioni. In alcuni casi, i requisiti funzionali possono anche specificare ciò che il sistema non deve fare.

- *Requisiti non-funzionali*: sono vincoli sui servizi e funzioni offerti dal sistema, ad esempio possono includere quelli temporali, di processo o altri imposti da standard. Spesso i requisiti non-funzionali si applicano all'intero sistema anziché sulle singoli componenti.

In realtà, le distinzioni tra le diverse tipologie di requisiti non sono ben delimitati. Ad esempio, un requisito utente riguardante la sicurezza come le limitazioni di accesso ai soli utenti autorizzati può apparire come un requisito non-funzionale. Tuttavia, questo requisito può generare altri requisiti chiaramente funzionali, come la necessità d'introdurre un sistema di autenticazione.

2.1.2 Processo

L'Ingegneria dei Requisiti si suddivide in cinque attività principali[42, 43].

Elicitazione

L'elicitazione dei requisiti è la fase nella quale si definisce il contesto del sistema. Si comprendono l'applicazione del dominio, i bisogni di business, le limitazioni al sistema e il problema stesso. Diverse sono le tecniche per ottenere i requisiti di un sistema:

- *Intervista*: metodo per ottenere fatti e opinioni direttamente da potenziali utenti (o altre persone d'interesse) in merito al sistema da sviluppare, in modo tale da poter identificare e chiarificare incomprensioni. Le interviste possono essere chiuse, dove le domande sono predefinite, o aperte, dove si discute apertamente su ciò che ci si aspetti dal sistema;

- *Casi d'Uso/Scenari*: i casi d'uso descrivono le interazioni tra gli utilizzatori del sistema e il sistema stesso, focalizzandosi su ciò che l'utente abbia bisogno di fare con il sistema. I casi d'uso rappresentano requisiti funzionali, mentre gli Scenari sono esempi di sessioni d'interazioni, cioè in cui si simula una possibile interazione tra l'utente e il sistema. Uno scenario include inoltre una descrizione del sistema all'inizio e al termine della simulazione;

- *Focus Groups*: piccoli gruppi di utenti provenienti da background e capacità differenti discutono sulle funzionalità di un prototipo del sistema, in modo da identificarne aspetti importanti e ciò che realmente gli utenti vogliano da esso;

- *Brinstorming*: aiuta a sviluppare soluzioni creative a specifici problemi. É costituito da due fasi: quella di generazione, dove vengono collezionate le idee, e la fase di valutazione, dove si discutono le idee trovate nella fase precedente;

- *Prototipazione*: un prototipo è una versione iniziale del sistema, disponibile nelle prime fasi dello sviluppo. I prototipi sono spesso utilizzati per elicitare e validare i requisiti e possono essere "*Throw-away*" se aiutano a capire le difficoltà dei requisiti, o "*Evolutionary*", se sono prototipi funzionanti che possono far parte del prodotto finale.

Analisi

L'analisi dei requisiti verifica la necessità, consistenza (non dovrebbero essere contraddittori), completezza (tutti i vincoli sono rispettati e non manca

nessun servizio) e fattibilità (possono essere implementati nei tempi e budget disponibili) dei requisiti.

Le principali tecniche di analisi sono:

- *Joint Application Development* (JAD): una sessione di gruppo nella quale si effettua un'analisi strutturata;
- *Prioritizzazione dei requisiti*: il cliente stabilisce l'ordine delle funzionalità da implementare tenendo in considerazione i rischi, costi e difficoltà puntualizzati dagli sviluppatori;
- *Modellizzazione*: creazione di un modello. Un modello di un sistema è un importante ponte tra l'analisi e il processo di designing in cui si cerca di effettuare una rappresentazione astratta della realtà.

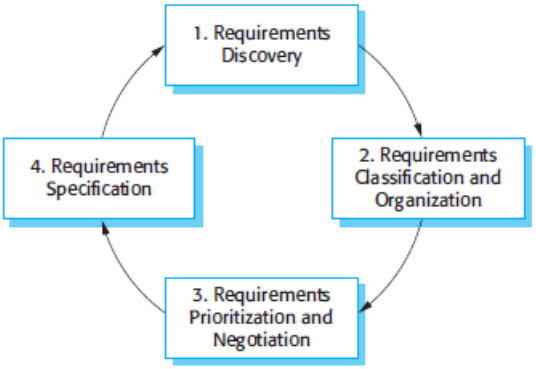


Figura 2.1: Processo di elicitazione e analisi dei requisiti[42]

Documentazione

Lo scopo della documentazione è quella di comunicare i requisiti tra gli sviluppatori e gli stakeholders. La documentazione è alla base della valutazione dei prodotti e dei processi. Una buona documentazione è non ambigua, completa, corretta, comprensibile, consistente, concisa e fattibile.

Validazione

Per validazione si intende il processo di certificazione dei requisiti, con cui si assicura che il requisito sia una descrizione accettabile del sistema da implementare.

Organizzazione

L'organizzazione dei requisiti comprende tutte le attività che concernono il controllo delle versioni e delle modifiche, il tracciamento dei requisiti e del loro status. La tracciabilità dei requisiti fornisce relazioni tra i requisiti, il design e l'implementazione di un sistema, con lo scopo di gestirne i cambiamenti.

2.2 Metodologia Agile Scrum

La pubblicazione del "Manifesto Agile"[14] poco più di un decennio fa ha portato un cambiamento senza precedenti nel campo dell'Ingegneria del Software. I metodi di sviluppo Agili possono essere visti come una reazione a quelli tradizionali. Quest'ultimi enfatizzano un approccio razionale allo sviluppo, affermando l'esistenza di soluzioni ottimi e predicibili per ogni problema. Al contrario, le metodologie agili sono indirizzati ad affrontare i problemi come impredicibili, facendo affidamento sulla creatività delle persone invece che su un processo prestabilito[11].

I processi di sviluppo di un sistema sono complessi e complicati. Per far fronte a ciò, è necessario un approccio altamente flessibile e adattivo ai problemi come *Scrum*, attualmente la metodologia agile più diffusa[13]. Scrum non è un processo standardizzato grazie al quale si può ottenere il prodotto finito rispettando tempo, budget e livello di qualità fissati. Al contrario, Scrum è un *framework* basato su un insieme di valori, principi e pratiche i quali permettono però di adattarsi alle esigenze delle singole organizzazioni, creando così metodi implementativi unici e su misura.

2.2.1 Ruoli

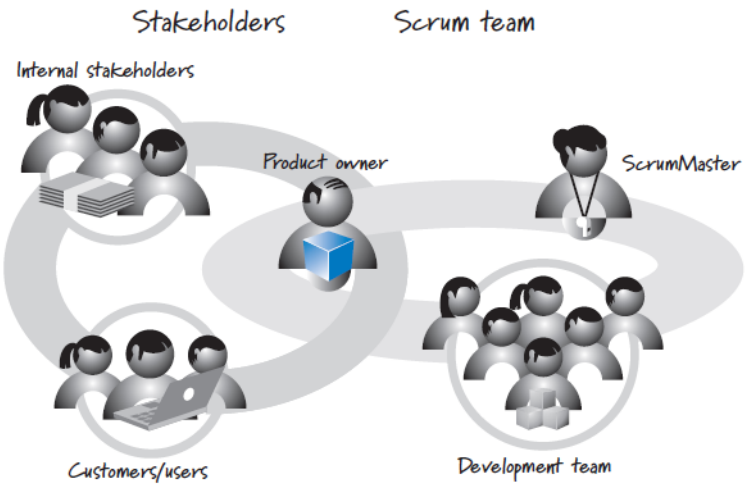


Figura 2.2: Ruoli in Scrum[39]

L'elemento fondamentale di Scrum è il *Team*, composto da un gruppo ristretto di persone (generalmente tra i tre e dieci membri). Nel Team vi sono tre ruoli:

- *Product Owner*: è l'autorità che decide quali sono le funzioni e funzionalità da realizzare, stabilendo inoltre l'ordine con la quale implementarle. Rappresenta, inoltre, gli interessi degli *Stakeholders* (coloro a cui è indirizzato il prodotto, i quali hanno determinati desideri e bisogni da soddisfare), per questa ragione è incaricato di mantenere e comunicare a tutti i partecipanti una chiara visione di ciò che il Team Scrum sta cercando di realizzare.

- *ScrumMaster*: è il responsabile del mantenimento e corretto utilizzo dei valori e principi Scrum durante il processo di sviluppo. A differenza di un *project manager*, lo ScrumMaster non ha l'autorità di esercitare un controllo sul Team, infatti, il ruolo principale dello ScrumMaster è quello di gestire le relazioni tra il Product Owner e il resto del Team, facilitando quest'ultimo attraverso la rimozione di possibili ostacoli.

- *Team di Sviluppo*: in Scrum, il Team deve essere plurifunzionale (*cross-functional*), cioè significa che, collettivamente, i membri del Team di Sviluppo hanno le competenze necessarie per svolgere tutte le attività dal designing al testing. Il Team è inoltre auto-organizzato e in continuo perfezionamento.

2.2.2 Artefatti

Il lavoro di un Team Scrum è organizzato attraverso un *Product Backlog*, composto da una serie di elementi (sezione 3.2.1) che rappresentano i desideri e bisogni degli Stakeholders. Queste richieste possono avere qualsiasi natura, tuttavia il Team di Sviluppo è tenuto a realizzare solamente quelle contenute nel Product Backlog.

Il Product Owner è quindi l'incaricato di fornire un ordine di priorità agli elementi del Product Backlog e di selezionare quelle che dovranno essere realizzate durante uno Sprint, formando quindi lo *Sprint Backlog*. Il Product Backlog non è definitivo, infatti può subire processi di raffinamento, aggiungendo elementi, togliendo quelli di disturbo ecc...

2.2.3 Rituali

In Scrum, il lavoro viene svolto in iterazioni o cicli chiamati *Sprint* i quali hanno una durata che vanno dalle due alle quattro settimane. Al termine di ogni Sprint viene rilasciato un prototipo funzionante avente del valore tangibile per il cliente o per l'utente. Generalmente gli Sprint hanno una durata fissata e uguale per tutti gli Sprint. Inoltre, durante lo svolgimento di uno Sprint non si possono effettuare delle modifiche agli obiettivi prefissati.

Sprint Planning

Durante lo *Sprint Planning*, si determina un sottoinsieme del *Product Backlog*, cioè si scelgono le attività che verranno svolte durante lo svolgimento dello Sprint. In questa fase, il Product Owner e il Team di Sviluppo si ac-

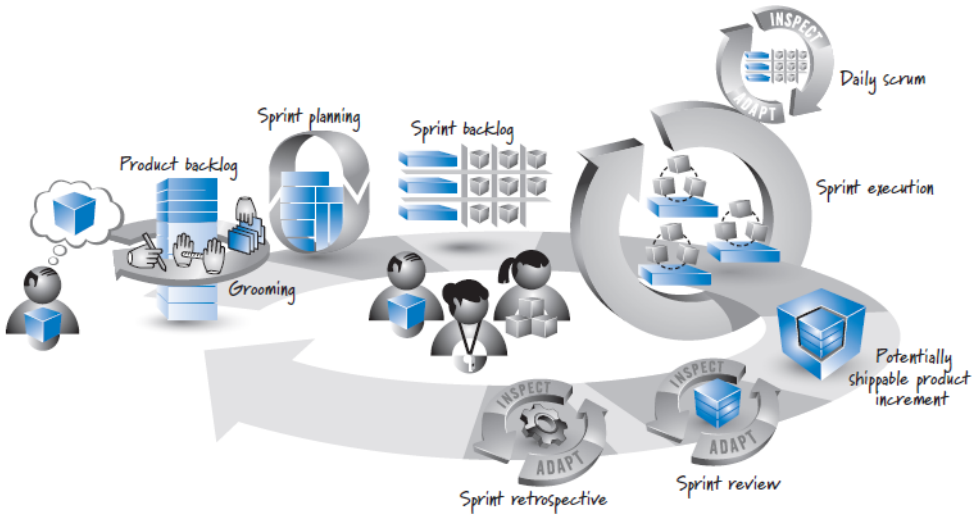


Figura 2.3: Framework Scrum[39]

cordano su quelli che sono gli obiettivi da raggiungere. Una volta stabiliti gli elementi che possono essere realisticamente realizzati, il Team effettua anche una stima dello sforzo (tipicamente in ore) necessarie al loro completamento.

Daily Scrum

Per tutta la durata dello Sprint, ogni giorno viene effettuato il *Daily Scrum*. Questi incontri non sono finalizzati alla risoluzione dei problemi riscontrati, infatti ha lo scopo di permettere a tutti i partecipanti di avere una chiara visione di ciò che sta avvenendo, dei progressi verso gli obiettivi prefissati e sulle attività che verranno svolte durante la giornata.

Scrum Review

Lo *Sprint Review* viene effettuato al termine dello Sprint. L'attività principale di questa fase è l'incontro che avviene tra tutti i partecipanti al progetto (dal Team Scrum agli Stakeholders). Questo incontro è focalizzato sulla visione delle features realizzate durante lo Sprint contestualizzate al prodotto finale. Lo Scrum Review permette una comunicazione bilaterale. Coloro che non fanno parte del team di sviluppo hanno la possibilità di essere aggiornati

sugli sviluppi del progetto. Al contempo il team viene coinvolto nella sfera del business e marketing del prodotto, ricevendo inoltre feedback frequenti.

Scrum Retrospective

L'ultima attività svolta alla conclusione dello Sprint è lo *Scrum Retrospective*. Durante questo processo, vengono discusse i problemi relative alle pratiche Scrum applicate nello Sprint appena concluso. Al termine della retrospettiva, il Team Scrum dovrebbe aver identificato un numero di possibili azioni di miglioramento da attuare durante lo Sprint successivo.

2.3 Machine Learning

Negli ultimi anni, il volume dei dati è letteralmente esploso[7], per esempio si hanno trilioni di pagine web, un'ora di video viene caricato su YouTube ogni secondo e i grandi colossi dell'e-commerce gestiscono milioni di transazioni ogni ora. Con l'avvento dell'era dei "Big Data", si è sentita la necessità di metodi automatizzati per l'analisi dei dati. Il *Machine Learning* è quindi un insieme di metodi che permettono di trovare pattern nei dati in maniera automatica, utilizzandoli per predire dati futuri oppure per effettuare varie tipologie di decisioni (come pianificare il modo di collezionare più dati)[32].

2.3.1 Tipologie di algoritmi

Apprendimento supervisionato

Lo scopo dell'apprendimento supervisionato o predittivo è quello di mappare degli input x su degli output y dato un insieme di coppie input-output $D = \{(x_i, y_i)\}_{i=1}^{N}$. D è chiamato *training set* e N è il numero degli elementi dell'insieme. Ogni elemento in x_i è un vettore di numeri D-dimensionale chiamato *feature* o attributo. Invece, gli elementi y_i sono gli output o variabili di risposta.

In base al tipo della variabile si hanno due ulteriori suddivisioni di categoria:

- *Classificazione*: l'output può assumere un numero finito di valori $y_i \in \{1, \cdots, C\}$, con C il numero di classi. Un modo per formalizzare il problema è attraverso una *funzione di approssimazione*. Si assume quindi che $y = f(x)$ per una qualche funzione f detta funzione reale. Lo scopo di un algoritmo di classificazione è quella di stimare la funzione f dato un insieme di training, in modo tale da ottenere delle predizioni $\hat{y} = \hat{f}(x)$. La difficoltà principale è quindi riuscire a effettuare predizioni su input non appartenenti al training set, visto che per predire un elemento di questo insieme basta guardare la risposta;

- *Regressione*: il funzionamento è simile alla Classificazione a eccezione del fatto che l'output può assumere valori appartenenti al continuo.

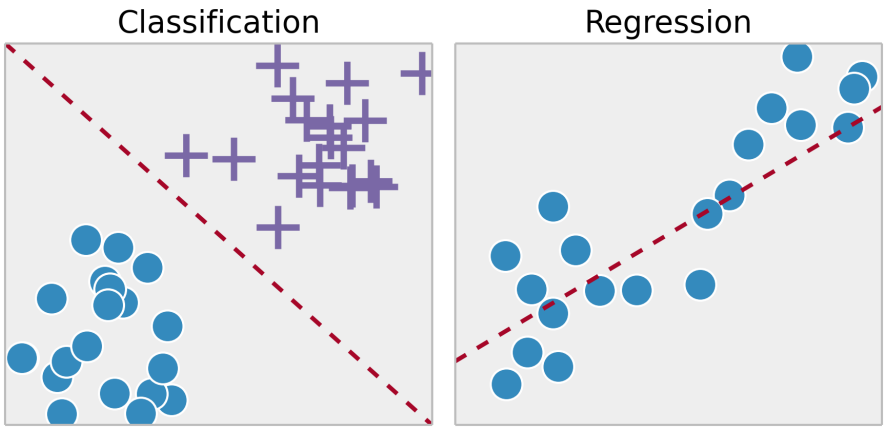


Figura 2.4: Differenza tra Classificazione e Regressione[38]

Apprendimento non supervisionato

A differenza dell'apprendimento supervisionato, in quello non supervisionato non si hanno valori di input, bensì solo quelli di output. Lo scopo è

quindi quelli di scoprire delle "strutture interessanti" nei dati (*knowledge discovery*). Nonostante non vi siano delle categorie ben specifiche, di seguito verranno descritti esempi canonici di apprendimento non supervisionato.

- *Clustering*: consiste nella suddivisione dei dati in un certo numero di gruppi. Il primo obiettivo è quello di stimare la distribuzione del numero di gruppi, $p(K|D)$, infatti, a differenza della Classificazione dove abbiamo dei raggruppamenti prestabiliti, nel Clustering si possono avere sia pochi che molti raggruppamenti. Il secondo obiettivo è quello di stimare l'appartenenza al gruppo per ogni singolo punto, quindi associare a ogni elemento i un *cluster* $z_i \in \{1, \cdots, K\}$.

- *Fattori Latenti*: quando si hanno dei dati di dimensionalità elevata, è spesso utile ridurla attraverso la proiezione dei dati su un sottospazio di dimensionalità inferiore che catturi l'"essenza" dei dati stessi, trovando quindi i fattori latenti dei dati. L'approccio più comune alla riduzione della densità è attraverso quello che viene chiamato "Analisi dei Componenti Principali" (PCA) (sezione 4.1.2).

- *Realizzazione di un grafo*: a volte dati un insieme di variabili, si ha la necessità di scoprire le correlazioni tra essi. Ciò può essere rappresentato tramite un grafo dove i nodi corrispondono alle variabili e gli archi le dipendenze tra esse.

- *Completamento di una matrice*: questi algoritmi permettono di inferire i possibili valori per gli elementi mancanti di una matrice.

Apprendimento per rinforzo

Questa terza categoria è quella meno comune, infatti spesso quando si parla di tipologia di un algoritmo di Machine Learning, ci si riferisce quasi sempre all'apprendimento supervisionato e non. L'apprendimento di rinforzo è un apprendimento che cerca di adattare il sistema alle mutazioni dell'ambiente in relazione a determinati segnali detti di "ricompensa" o di "punizione".

2.3.2 Problematiche

Bias-Variance Tradeoff

Un primo problema che si presenta negli algoritmi di apprendimento automatico, è il bilanciamento tra il *bias* e la *varianza*[16] noto come "Bias-Variance Tradeoff" o anche "Dilemma Bias-Variance".

- Il bias è l'errore generato dalle assunzioni effettuate dall'algoritmo di apprendimento. Se il livello del bias è elevato, l'algoritmo tenderà a tralasciare relazioni tra le feature e gli output le quali potrebbero essere anche rilevanti, creando così una situazione di "*underfitting*".

- La varianza è l'errore generato dalla sensibilità dell'algoritmo alle piccole fluttuazioni nei dati del training set. Una varianza elevata può generare una situazione di "*overfitting*", infatti alcuni discostamenti nel training set potrebbero essere causati da disturbi casuali e quindi non rilevanti ai fini generali.

L'errore della predizione è data quindi dalla somma del bias e della varianza dell'algoritmo di apprendimento[24], tuttavia è impossibile eliminarle contemporaneamente. Se si ha un bias basso, l'algoritmo sarà flessibile, considererà un maggior numero di situazioni facendo aumentare di conseguenza la varianza. Per questa ragione è necessario trovare un compromesso per la quale si avrà un errore finale il più basso possibile.

Complessità della funzione e dimensione del training set

In relazione alla complessità della funzione reale si ha la necessità di avere una quantità maggiore o minore di dati per il training. Se la funzione reale è semplice, l'algoritmo di apprendimento riuscirà a ottenere ottimi risultati anche utilizzando pochi dati di training con bias elevato e bassa varianza. Al contrario, se la funzione reale è complessa, l'algoritmo dovrà essere più flessibile e dovrà utilizzare una quantità elevata di dati.

Dimensionalità dello spazio dell'input

Il vettore d'input può avere una dimensionalità elevatissima. In questa situazione l'algoritmo di apprendimento riscontrerà difficoltà nell'ottenere i pattern nei dati anche nel caso in cui la funzione reale sia semplice. Per questa ragione, si dovranno selezionare manualmente o automaticamente le feature oppure effettuare una riduzione della dimensionalità.

Anomalie nei valori di input

Durante la fase di training potrebbero esserci delle anomalie nei valori di output. L'algoritmo di apprendimento dovrebbe ignorare questi valori in quanto altrimenti si creerebbe una situazione di overfitting compromettendo la precisione generale della predizione.

Altri fattori

Vi possono essere un'ulteriore serie di fattori da tenere in considerazione, ad esempio l'eterogeneità dei dati in input, la ridondanza di feature o la presenza d'interazioni non lineari.

Capitolo 3

Dataset

Come affermato precedentemente, pochi studi sono stati fatti in merito alle applicazioni del Machine Learning in ambito dell'ingegneria dei requisiti agili. Di conseguenza, per gli scopi di questa tesi è stato realizzato un dataset ex novo contenente requisiti software sotto forma di user story, a ognuna dei quali sono associati ulteriori dettagli sia dal punto di vista implementativo che progettuale.

3.1 Un progetto software virtuale

Non avendo un dataset già formato di requisiti agili e non avendo un team di sviluppo che potesse fornirci dei requisiti agili facenti riferimento a un progetto reale, si è deciso di progettare un sistema software fittizio per il quale sono stati elicitati i requisiti ed elaborati i diagrammi UML.

Il dominio di questo sistema fittizio risulta pressoché irrilevante per gli scopi di questa tesi. Al fine di agevolare la realizzazione del dataset, il sistema scelto non dovrà:

- avere dimensioni limitate. Ad esempio, sarebbe difficile ottenere un numero sufficiente di requisiti a partire da una generica applicazione per dispositivi mobili;

- richiedere conoscenze e competenze specifiche. Ad esempio, elicitare i requisiti di un sistema integrato di un satellite risulterebbe impossibile per una persona che non lavori nel campo.

Alla luce di queste considerazioni, si è deciso di utilizzare un sistema di gioco d'azzardo come base per la realizzazione del dataset. Le ragioni che hanno portato a questa inusuale scelta sono molteplici.

Per prima cosa soddisfa i requisiti sopracitati, infatti un sito di gioco d'azzardo può essere facilmente arricchito di funzionalità mantenendo comunque la coerenza del sistema stesso (es. aggiungendo nuove modalità di gioco), mentre ideare le varie componenti e le loro implementazioni risultano in gran parte operazioni fattibili da un qualunque sviluppatore.

In secondo luogo, il gioco d'azzardo è altamente regolamentato in Italia. Per poter offrire qualsiasi servizio correlato a questo campo è necessario ottenere una certificazione rilasciata dall'*Amministrazione autonoma dei Monopoli di Stato* (AAMS, organo dell'*Agenzia delle dogane e dei Monopoli*) (*art. 88 Tulps*). Per questo scopo, AAMS fornisce un esaustivo documento contenente i requisiti minimi delle piattaforme di gioco[2], il quale ha agevolato notevolmente la fase di elicitazione dei requisiti per il dataset.

Il progetto ideato si può suddividere in diverse categorie principali: i servizi relativi ai giochi, la gestione degli utenti, la gestione dei pagamenti e il servizio clienti.

3.1.1 Betting Exchange

Nonostante il sistema ideato offra diverse tipologie di gioco, il servizio principale è quello del *Betting Exchange*[6], una tipologia di scommesse su eventi sportivi o non.

Il tradizionale metodo di scommessa sportiva viene definito "scommessa a quota fissa". Con questa modalità di scommessa, l'allibratore (di seguito *Bookmaker*) stabilisce a priori le quote per gli eventi e i vari mercati. Per coprire il rischio, i Bookmaker offrono quote generalmente più basse rispetto

alla reale probabilità di vincita in modo tale da riuscire comunque ad avere un margine di guadagno nel lungo periodo. Per quanto riguarda le piazzate, la vincita potenziale viene stabilita e fissata (moltiplicando l'importo della scommessa per la quota, ad es. scommettendo a quota 2.0 un importo di 10,00 euro la vincita potenziale è di 20,00 euro). Un eventuale cambio di quota non avrà quindi alcun effetto sulle giocate già effettuate.

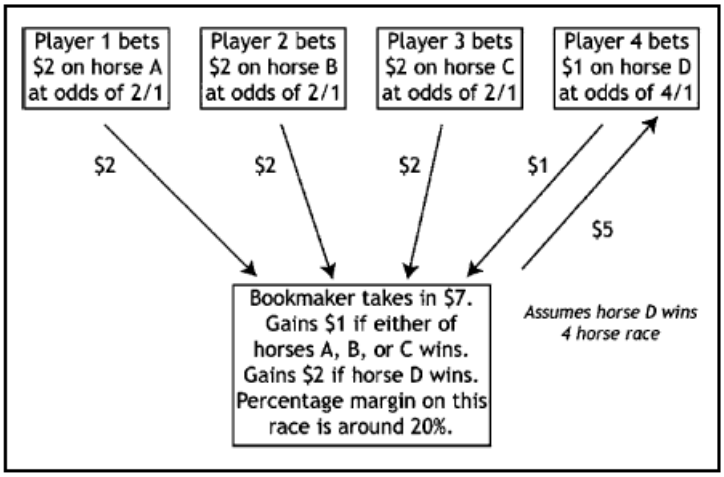


Figura 3.1: Bookmaker[10]

Analogamente le scommesse piazzate in un sistema di Betting Exchange sono anch'esse a quota fissa, nel senso che una volta stabilito la quota e l'importo, la vincita potenziale non potrà più essere modificata. La differenza profonda tra un sistema di scommesse tradizionale e un sistema di Betting Exchange risiede nel fatto che quest'ultimo è un vero e proprio mercato tra giocatori[10]. Infatti mentre nel primo caso vi è solamente un Bookmaker che decide le quote, nel secondo tutti gli utenti possono diventare Bookmaker.

A differenza di un Bookmaker, i proprietari di un sistema di Betting Exchange non hanno interesse nell'esito di un determinato evento. Essi infatti non lucrano sulle perdite dei giocatori, bensì sul numero di scommesse piazzate, trattenendo una percentuale (generalmente tra il 2 e 5 per cento) dalle vincite nette di ogni giocatore. Pertanto si limitano a offrire la piattaforma di gioco con i vari eventi, mercati e opzioni di gioco.

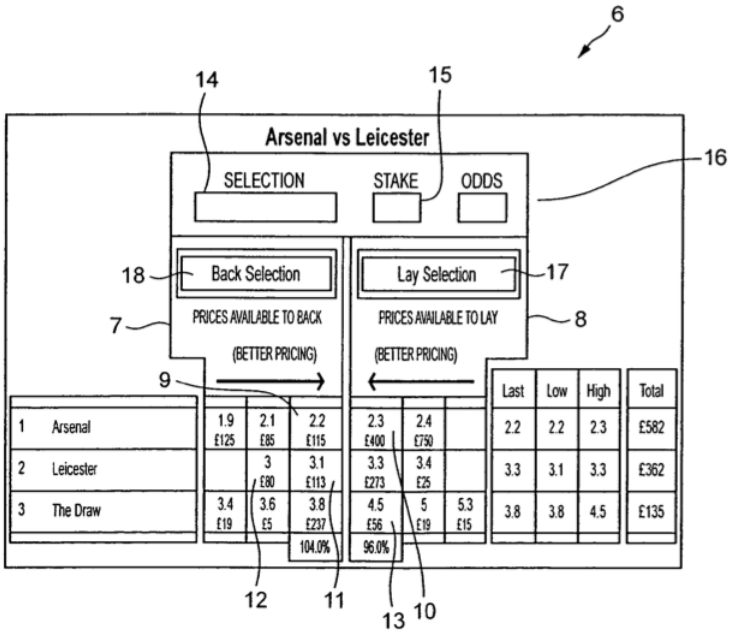


Figura 3.2: Generica schermata di gioco di un sistema di Betting Exchange[6]

Un sistema di Betting Exchange offre la possibilità di effettuare due tipologie di scommesse: la "puntata" e la "bancata". La puntata può essere considerata come la classica scommessa, un giocatore sceglie un risultato e scommette che questo risultato si verifichi. La bancata invece è esattamente il complementare della puntata, questa operazione tipica del Betting Exchange, permette all'utente di scommettere sul non verificarsi di un certo risultato. Ad esempio, su una partita di calcio si può scommettere sul risultato esatto, quindi se si vuole bancare il risultato "0 - 2" significa che si scommette sul non verificarsi dell'esito "0 - 2", e quindi il giocatore vince su un qualunque risultato diverso da "0 - 2". Inoltre - a differenza della puntata, dove il giocatore stabilisce l'importo che vuole scommettere - quando si banca, il giocatore stabilisce l'ammontare della vincita desiderata, e in base alle quote, verrà calcolata quella che si chiama "responsabilità" cioè l'ammontare che l'utente rischia di perdere.

Il punto chiave del sistema di Betting Exchange si basa sul bilanciamento

degli importi tra le puntate e le bancate, attraverso un processo di "abbinamento". Una scommessa per essere valida deve essere abbinata, ciò significa che ogni scommessa è bilanciata da una scommessa della tipologia opposta.

Ogni giocatore ha la possibilità di offrire la propria scommessa (per entrambe le tipologie) a una qualsiasi quota desiderata e a un qualunque importo. Successivamente, tre sono gli scenari possibili:

- altri giocatori hanno già effettuato la scommessa opposta alla quota desiderata e sono in attesa di essere abbinati per un importo totale (liquidità) superiore a quella della scommessa appena effettuata. La scommessa viene automaticamente abbinata (così come la controparte per lo stesso importo);

- nessun giocatore ha offerto una scommessa alla quota selezionata. La scommessa viene segnata come "non abbinata" e messa in attesa che un altro utente accetti l'offerta effettuando la scommessa opposta;

- la liquidità non copre l'importo della scommessa. La scommessa verrà parzialmente abbinata, cioè verrà abbonata per la liquidità disponibile, mentre il restante verrà lasciata in attesa di essere abbinata.

Da notare che il numero delle puntate non deve essere necessariamente uguale al numero delle bancate, infatti il bilanciamento avviene sull'importo totale scommesso e non sul numero di esse.

É quindi evidente che, mentre in un sistema di scommesse tradizionale vi è la possibilità che l'importo totale vinto dagli utenti possa essere inferiore o superiore a quello perso, in un sistema di Betting Exchange questi due importi si equivalgono. In particolar modo le vincite di un utente corrispondono alle perdite di un altro utente (o più). Ad esempio, una puntata di 10 euro a quota 1.5 può garantire una vincita di 15 euro (guadagno di 5 euro) o una perdita di 10 euro, mentre la controparte abbinata di 10 euro permette una vincita di 10 euro con una responsabilità di 5 euro. É banale osservare che per un determinato risultato solo uno tra la puntata e la bancata può vincere, di conseguenza i 5 euro vinti dall'uno sono i 5 euro persi dall'altro.

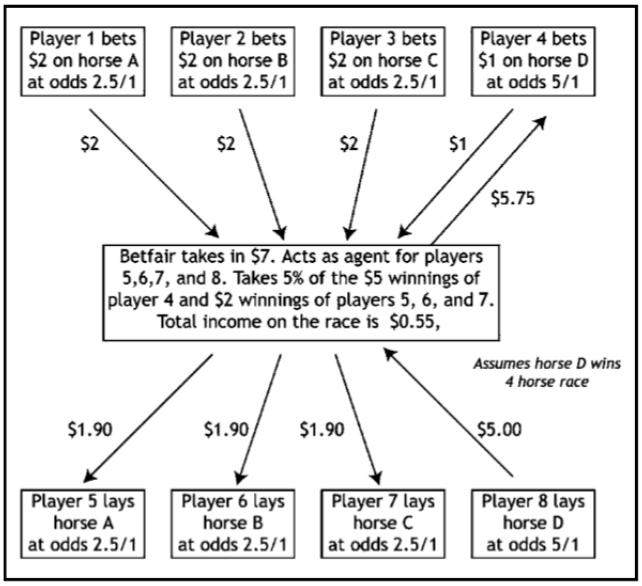


Figura 3.3: Betting Exchange[10]

3.1.2 Requisiti AAMS

Le linee guida fornite da AAMS per la certificazione della piattaforma di gioco[2] forniscono una serie di caratteristiche che il sistema deve avere in modo da tutelare gli interessi degli utenti. Per questo, gran parte dei requisiti che andranno a comporre il dataset faranno chiari riferimenti o saranno atti a soddisfare i requisiti applicativi della piattaforma di gioco indicati da AAMS.

Autenticazione e registrazione

Un utente per poter accedere alle funzionalità del sistema deve necessariamente autenticarsi. Il sistema offre un processo di registrazione del giocatore, il quale dovrà fornire i suoi dati personali (ogni persona fisica potrà creare al più un account) e accettare i termini e condizioni di gioco. Inoltre il sistema dovrà permettere l'utente di recuperare le proprie credenziali fornendo le opportune risposte di sicurezza.

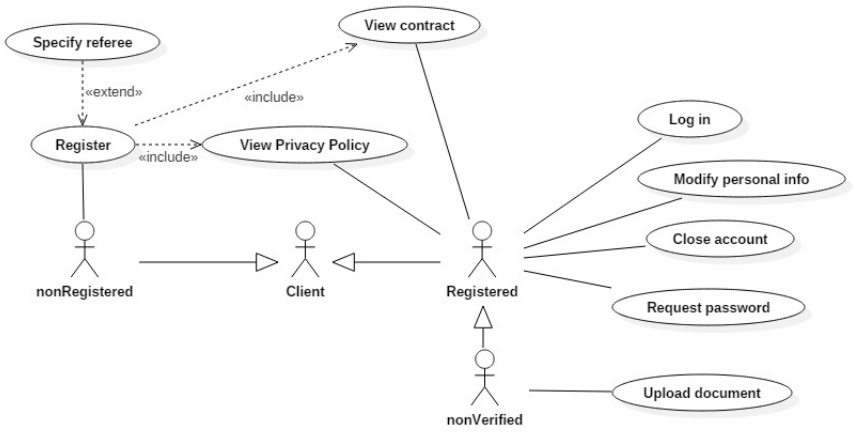


Figura 3.4: Diagramma dei casi d'uso per l'autenticazione e la verifica

Verifica dell'utente

Al fine di garantire l'autenticità degli account, è necessaria una verifica manuale dell'identità degli utenti. Il processo di verifica è necessario per le seguenti ragioni:

- impedire a soggetti minorenni di utilizzare il sistema. I più giovani sono più suscettibili allo sviluppo di patologie legate al gioco d'azzardo[44];
- prevenire collusione, riciclaggio e frode. Il gioco d'azzardo online permette facilmente ai criminali di poter convertire il denaro illecito in "soldi puliti" da poter prelevare [40];
- evitare furti d'identità.

Per forzare l'utente a verificare la propria identità inviando la documentazione necessaria (possibili documenti richiesti sono documento di identità, prova d'indirizzo di residenza e scansione della propria carta di credito), il conto di gioco viene limitato impedendo agli utenti non verificati la possibilità di prelevare. Inoltre, gli account non verificati entro un certo periodo di tempo dalla registrazione verranno sospesi.

Gioco responsabile

Con la rapida crescita del gioco d'azzardo online, anche il numero di giocatori problematici o patologici è cresciuto vertiginosamente[8, 31]. Per ridurre i rischi legati al gioco d'azzardo è necessario adottare diverse strategie per salvaguardare gli utenti:

- *Sensibilizzazione*: ogni sito che involve una qualche forma di gioco d'azzardo deve promuovere la consapevolezza sui rischi che possono occorrere utilizzando sistemi di gioco d'azzardo, fornire le indicazioni per poter riconoscere i sintomi legati alla ludopatia e pubblicizzare metodi per poterli risolvere (sia sotto forma di consigli, sia indicando possibili agenzie di supporto).

- *Autoesclusione*: è un procedimento per il quale l'attività di gioco da parte del giocatore viene bloccata per un periodo di tempo specificato dal giocatore stesso (es. 24 ore, 7 giorni, 30 giorni ecc...), il quale una volta avviato non può essere annullato fino al termine del periodo.

- *Autolimitazione*: il giocatore può stabilire dei limiti alle attività svolte sul sistema, ad esempio può stabilire un massimale di deposito per settimana, un limite di giocate possibili oppure l'importo massimo di perdite.

Conto di gioco

Il giocatore può effettuare operazioni di deposito e prelievo attraverso diverse modalità (ad es. addebito su carta di credito, pagamento tramite *PayPal* ecc...). Il giocatore deve inoltre avere pieno accesso alle informazioni riguardante i propri fondi e ai propri movimenti.

Sicurezza

I conti di gioco devono essere protetti contro una qualsiasi forma di accesso o rimozione illecito, sia da parte del personale del concessionario, sia

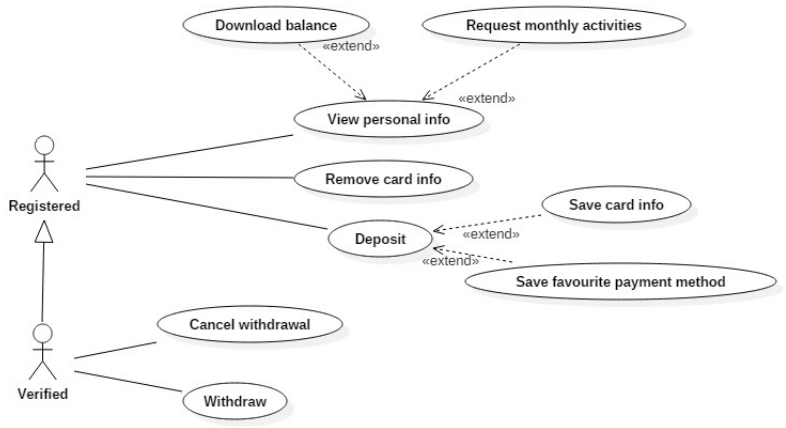


Figura 3.5: Diagramma dei casi d'uso per la gestione dei fondi

da parte di utenti malintenzionati o soggetti non autorizzati. A tal fine, il sistema deve essere in grado di eseguire procedure di blocco a fronte di eventi di malfunzionamenti e/o tentate intrusioni. Inoltre a tutela del giocatore, il sistema deve tenere traccia dell'attività e degli accessi dell'utente (orari e indirizzo IP), in modo tale che possa verificare personalmente eventuali anomalie.

Tutti i componenti critici del sistema devono prevedere misure di sicurezza atte a contrastare vulnerabilità (ad es. causate da *virus, malware, rootkit* ecc...) che possano compromettere l'integrità, la correttezza dei dati e le funzionalità del sistema stesso.

Infine, il sistema deve consentire alle apposite agenzie di vigilanza governative di poter effettuare gli appositi controlli riguardanti sia le attività dei giocatori, sia i servizi offerti dal concessionario.

3.1.3 Altre funzionalità

Oltre al sistema di Betting Exchange e le specifiche indicate da AAMS, il sistema software fittizio ideato presenta ulteriori funzionalità, sia di gioco che di supporto.

Casinò

Come accennato in precedenza, il sistema ideato è un generico sito di gioco d'azzardo. In questa categoria rientrano moltissime tipologie di gioco. Oltre alle scommesse sportive vi sono ad esempio la lotteria, il bingo, gli skill games (es. tornei di giochi con carte), il fantacalcio oppure tutti i giochi tipici dei casinò (slot machines, roulette e blackjack).

Nonostante sarebbe stato possibile estendere il progetto con tutti i sopracitati giochi, per motivi temporali si è deciso di aggiungere solamente il casinò online. Per integrare questo nuovo aspetto del gioco d'azzardo nel sistema software ideato, si è pensato di creare due sezioni separate, una per il Betting Exchange e una dedicata al casinò. Gli utenti potranno utilizzare indistintamente il proprio account con i relativi fondi nelle due sezioni.

I giochi offerti dai casinò online non sono sviluppati internamente. Questi, indipendentemente dal loro tipo, sono software prodotti da terzi sotto forma di "*flash game*" per poi essere integrati successivamente nella piattaforma di gioco. Nonostante gli amministratori di sistema non abbiano il controllo delle funzionalità dei giochi forniti dalle terze parti, essi possono comunque impostare alcuni parametri (es. l'importo della puntata minima e massima). Il gioco integrato dovrà permettere agli utenti di trasferire il denaro dal proprio fondo al fondo del gioco, e viceversa al termine della partita.

Un'importante differenza tra il casinò online e il sistema di Betting Exchange è la gestione del credito bonus. Un bonus è un importo accreditato dal sistema (a seguito del soddisfacimento di determinate condizioni o al conseguimento di promozioni offerte) utilizzabile ma non prelevabile. Al fine di convertire l'importo bonus in credito reale, l'utente dovrà giocare l'importo del bonus, tuttavia in un casinò online viene imposta un'ulteriore restrizione detta "*wagering requirement*", il quale prevede lo sblocco del credito solamente rigiocando il bonus un certo numero di volte (es. bonus di 10 euro con *wagering* = x20, l'utente potrà prelevare il bonus giocando 200 euro).

Live match

Un sistema di scommesse sportive offre molti servizi di supporto al giocatore, ad esempio, può fornire una statistica delle squadre/giocatori di un determinato evento in modo tale da guidare l'utente alla puntata più probabile.

Le scommesse sportive vengono generalmente effettuate prima della partita. Nonostante ciò, un sistema di gioco online può prevedere quello che viene definito "*live betting*" cioè scommettere mentre l'evento è in corso. A tal scopo, il sistema può fornire svariate informazioni:

- *live streaming*: l'utente può guardare in diretta l'evento sulla pagina dell'evento stesso. Il servizio è però non sempre disponibile in quanto molto spesso le reti televisive detengono i diritti delle riprese;

- *azioni importanti in live*: può essere considerata come una versione stilizzata del live streaming, dove non viene mostrato il video ma un'animazione in diretta degli avvenimenti più importanti come un'azione pericolosa o un cambio di punteggio;

- *statistiche dell'evento*: un riassunto dell'andamento dell'evento.

É da puntualizzare il fatto che le informazioni riguardanti gli eventi non vengono raccolti dalla piattaforma, bensì, si ottengono da un servizio esterno il quale è incaricato di effettuare questa operazione. La piattaforma fornisce solamente una visualizzazione delle informazioni ricevute.

Customer Service

Come ogni azienda che offre un servizio, è necessario garantire supporto ai problemi degli utenti. Oltre ai comuni metodi di supporto quali una pagina di *Frequently Asked Questions* (FAQ), un centralino telefonico o una comunicazione tramite email, è molto comune avere in un sistema di gioco d'azzardo anche una *live chat* con un operatore.

Una live chat presenta lo stesso vantaggio di una telefonata al customer service nel poter risolvere un problema immediatamente dall'operatore senza aver bisogno di lunghe attese per delle risposte. Una live chat può però essere considerato una forma di supporto più semplice ed efficiente[1], con l'aggiunta della possibilità di ottenere una copia della conversazione.

La live chat ideata per il sistema software fittizio permette quindi la diretta comunicazione tra un utente e un agente. Per velocizzare e agevolare il processo di supporto, il sistema richiederà all'utente di indicare la categoria del proprio problema e, nel caso, suggerirà all'utente sezioni della pagina di FAQ che potrebbero rispondere al suo problema. In alternativa, verrà messo in attesa di essere assegnato all'operatore di competenza. Durante l'attesa, l'utente potrà visualizzare la sua posizione in coda e la stima della durata dell'attesa. Una volta terminata la chat, l'utente potrà lasciare un feedback che servirà per migliorare la qualità del servizio.

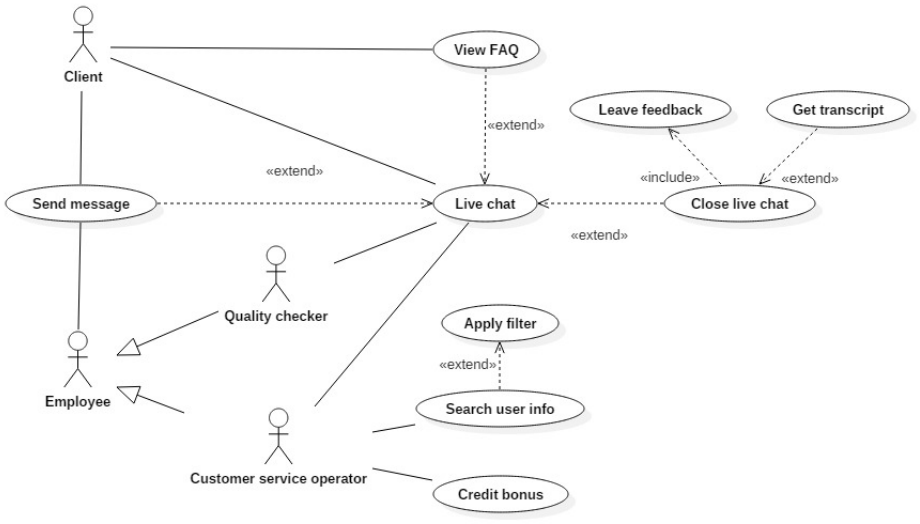


Figura 3.6: Diagramma dei casi d'uso del Customer Service

3.2 Composizione del dataset

É risaputo che il numero e la qualità dei dati forniti a un algoritmo di Machine Learning hanno un impatto notevole sui risultati ottenibili[19]. Per rispondere a questa esigenza, si è stabilito un numero di requisiti che fosse sufficiente per garantire dei risultati discreti, ma che al contempo permettesse la redazione del dataset in un tempo ragionevole. Il dataset realizzato contiene quindi 200 record ognuno dei quali composto dagli elementi di un Product Backlog Scrum e delle stime progettuali.

3.2.1 Product Backlog

Il *Product Backlog* è una lista di funzionalità, miglioramenti e bug del prodotto ordinati per priorità[39]. Il product backlog può essere quindi paragonato a un incompleto e in continuo mutamento documento di requisiti contenente le informazioni per gli sviluppatori. Le attività da svolgere nel backlog verranno smaltite a ogni iterazione spostando le attività ad alta priorità nello *sprint backlog*, il quale una volta iniziato non potrà essere modificato per tutta la durata dello sprint[34].

User Story

Una *user story* è una breve dichiarazione di intenti che descrive qualcosa che il sistema deve fare per l'utente[28]. É da puntualizzare però che le user story non sono requisiti, infatti, nonostante svolgano un ruolo simile a quello delle specifiche dei requisiti software, esse hanno delle sottili ma critiche differenze. Principalmente una user story è breve (non dettagliata) e facilmente comprensibile dagli sviluppatori, dai stakeholders e/o utenti, rappresentano quindi dei piccoli incrementi di funzionalità, sviluppabili in un breve periodo di tempo.

Una user story ha la seguente forma:

```
As a <role>, I can <activity> so that <business value>.
```

dove:

- <role> rappresenta chi effettua l'azione o chi (e talvolta cosa) riceve valore dall'attività;
- <activity> rappresenta l'azione che il sistema deve effettuare;
- <business value> rappresenta il valore ottenuto dall'attività.

ad esempio:

`As a client (<role>), I want to be notified whenever the terms and conditions of the contract change (<what the system should do>), so that I can be aware of my current duties and rights (business value I receive).`

Business Value

I requisiti contengono, in aggiunta alla descrizione della user story, una versione più dettagliata del valore che si ottiene dalla sua realizzazione.

User Story Elaboration

Contiene i dettagli implementativi della user story.

Definition of Done

La Definition of Done può essere considerata come una checklist di lavori che il team di sviluppo si aspetta vengano completati[39] prima di dichiarare la user story interamente implementata.

Expected Output

Una lista di output attesi.

3.2.2 Stime progettuali

Comparato al coding e al testing, effettuare stime può essere considerato superficiale, e per alcuni, una vera e propria perdita di tempo. Nonostante

ciò, stimare può fornire valore per diverse ragioni[28], ad esempio lo sforzo è direttamente collegato ai costi di sviluppo, senza di esso è impossibile determinare i costi, inoltre avendo una chiara visione dello sforzo necessario si può pianificare al meglio la linea di sviluppo.

Line of Codes

Stima delle Linee di Codice (LOC) modificate e aggiunte per implementare la user story.

Classes

Stima del numero di classi modificate e create per implementare la user story.

Effort

Sforzo misurato in ore/persone necessario al team per implementare la user story.

Number of Unit Tests Revisions

Il numero di revisioni alle unità di test effettuate prima del loro superamento.

Entropy

L'entropia è una metrica della complessità del codice, quantifica la complessità delle modifiche al codice[20]. In particolar modo, indica quanto sono sparse le modifiche al codice. Ad esempio, le modifiche che hanno un'elevata entropia sono difficilmente tracciabili.

L'entropia può essere quindi considerata come la distanza tra le modifiche effettuate al codice. Tuttavia, questa è una definizione piuttosto vaga. Come si può definire la distanza in un codice? Per far fronte a questa domanda si è

pensato di calcolare la distanza come il numero di linee di codice che intercorrono tra una modifica e un'altra. Sfortunatamente, anche questo metodo di misurazione risulta piuttosto vaga e astratta, in quanto le modifiche possono essere effettuati su file, pacchetti o servizi diversi rendendo impossibile un calcolo rigoroso.

Si è deciso quindi di calcolare l'entropia principalmente considerando il numero di file (o classi) modificati e creati, in quanto, più sono i file coinvolti, più sarà difficile tener traccia delle modifiche.

Services Dependencies

Il progetto fittizio alla base del dataset è stato ideato con una struttura a microservizi: piccole componenti software interconnesse tra loro, ciascuna delle quali è orientata a svolgere bene una funzionalità[33]. Questo campo del dataset è una panoramica delle interconnessioni tra i vari servizi al momento dell'implementazione della relativa user story. A causa delle modifiche apportate al codice, si possono creare nuove dipendenze tra i servizi oppure subentrarne di nuovi.

Le dipendenze tra servizi possono essere considerati come gli archi di un grafo dove i nodi sono i servizi stessi. Per rappresentare questo grafo si è deciso di utilizzare una matrice di adiacenza, dove la cella $c_{i,j} = 1$ se il servizio i dipende dal servizio j, 0 altrimenti.

Capitolo 4

Esperimenti e risultati

In questa sezione verranno descritti gli esperimenti effettuati per valutare l'utilizzo del Machine Learning in ambito di Ingegneria dei Requisiti Agili. L'idea generale è quella di ottenere il numero maggiore di informazioni dal dataset per poi provare a stimare i valori progettuali. Il dataset verrà quindi suddiviso in due parti, il *training set* e il *testing set*, i quali verranno utilizzati rispettivamente per trovare i pattern nei dati e verificare l'accuratezza del modello generato nello stimare valori date nuove user story. Gli script realizzati sono stati scritti nel linguaggio Python, utilizzando il framework *Scikit-learn*[35] per gli algoritmi propri del Machine Learning, *NLTK*[5] per la manipolazione del linguaggio naturale e *Gensim*[37] per la conversione della semantica dei testi in vettori.

4.1 Manipolazione del dataset

Processo fondamentale dell'intera sperimentazione è la determinazione e manipolazione delle feature utilizzate dagli algoritmi di Machine Learning. Nonostante il funzionamento di questi siano più o meno diversi tra loro, la scelta delle singole feature può influenzare sostanzialmente, sia positivamente che negativamente, i risultati ottenibili. Dato un *target* (valore che si vuole predire), un algoritmo di Machine Learning cercherà una correlazione tra le

feature e il target. Di conseguenza maggiore sarà la correlazione, maggiore sarà l'accuratezza della predizione. Per migliorare questo risultato, sarà necessario combinare tra loro feature correlati al target, dato che, combinando feature altamente scorrelate al target porterà a una riduzione della precisione.

4.1.1 Elaborazione del linguaggio naturale

Gli elementi del Product Backlog sono interamente testuali, questi valori non possono essere compresi direttamente da un elaboratore. Per questa ragione devono essere prima convertiti in forma vettoriale.

Lunghezza del testo

Nonostante la lunghezza di un testo non contenga molte informazioni, l'ottenimento di questa informazione è triviale, adatta per un primo approccio alle sperimentazioni.

Importanza delle parole

Per poter lavorare con campi testuali, non è possibile utilizzare ogni singola parola come valore di input in quanto testi di diversa lunghezza richiederebbero uno spazio di input diverso. Per ovviare a questo problema, uno dei metodi di *text learning* è quello di creare un *bag-of-words* contenente tutte le parole che potrebbero occorrere nel testo. A partire da esso, per ogni testo, si creerà un vettore che associa ad ogni parola un valore, ad esempio la sua frequenza nel testo.

Durante il processo di text learning vi è anche la necessità di effettuare delle modifiche ai testi:

- eliminazione delle *stopwords*: in un testo possono apparire parole che hanno la stessa probabilità di apparire in tutti i testi o che semplicemente non apportano significato al testo, perciò è preferibile rimuoverle dal testo;

- portare le parole alla loro radice: senza questa operazione, parole con lo stesso valore semantico verranno considerate separatamente (es. "*stems*", "*stemmer*", "*stemming*" si riducono tutti a "*stem*"), perdendo cosi dell'informazione.

Tra le varie metodologie esistenti, si è deciso di applicare il *Term frequency–Inverse document frequency* (Tf-Idf), la funzione di peso più utilizzata nei sistemi di *information retrieval*[3]. Per definizione, la funzione

$$tfidf(t,d) = tf(t,d) \cdot idf(t)$$

dove $tf(t,d)$ è la frequenza del termine t nel documento d, mentre $idf(t)$ è definita come

$$idf(t) = \log \frac{1+n_d}{1+df(d,t)} + 1$$

dove n_d rappresenta il numero totale di documenti e $df(d,t)$ il numero di documenti che contengono il termine t.

Il valore ottenuto cresce quindi proporzionalmente al numero delle occorrenze di una parola all'interno di un documento, il quale è tuttavia bilanciato dalla frequenza della parola tra tutti i documenti.

Somiglianza semantica

La somiglianza semantica (*semantic similarity*) è una metrica definita su un insieme di documenti o termini. La distanza tra due testi è dato da quanto essi sono simili tra loro in termini di significato o contenuto semantico.

Questa informazione può essere di particolare supporto per lo scopo di questa testi. Intuitivamente due requisiti aventi un'alta somiglianza semantica potrebbero avere anche delle stime progettuali simili tra loro.

Ovviamente, è necessario ottenere la semantica dai testi prima di tutto. Nonostante vi siano diverse metodologie per la rappresentazione della semantica[17], come per la frequenza delle parole, vi è la necessità di ottenere una rappresentazione vettoriale di dimensione fissata indipendente dalla

lunghezza del testo. A tal scopo si è utilizzato un algoritmo non supervisionato in grado di rappresentare ogni documento come vettore denso, formato in modo tale da poter predire le parole del documento stesso[27].

Il framework Gensim fornisce un'implementazione di questo algoritmo chiamato *doc2vec*. I dettagli del suo funzionamento vanno ben oltre gli scopi di questa tesi. Esso non è un solo algoritmo monolitico, bensì, può essere considerato come un albero di algoritmi. doc2ved utilizza due modelli distinti, *Cointinuous Bag of Words* (CBOW)[29] e *Skip-gram*[30], i quali utilizzano a loro volta due tipologie differenti di addestramento (con o senza esempi negativi) e altre variazioni.

Per quanto riguarda l'utilizzo pratico di questa funzione, il processo si divide in due fasi:

- creazione del modello (creazione del dizionario e training su documenti forniti);
- trasformazione del testo in formato vettoriale.

É possibile evitare la prima fase utilizzando un modello pre-addestrato. Alcuni di essi sono addestrati su una grande quantità di documenti e sono in grado di inferire il vettore semantico di un testo in maniera più accurata[26].

4.1.2 Riduzione della dimensionalità

Una volta ottenute le informazioni dal dataset, è generalmente utile effettuare un'ulteriore raffinazione, ottenendo feature latenti, selezionando le feature più rilevati oppure semplicemente riducendo la dimensionalità dei dati in input.

Estrazione delle feature

L'estrazione delle feature (*feature extraction*) è il processo di trasformazioni dei dati da uno spazio a elevate dimensione a uno di dimensionalità inferiori. Un metodo per effettuare l'estrazione delle feature è l*'analisi delle*

componenti principali (PCA), una procedura statistica che usa trasformazioni ortogonali per convertire un insieme di valori possibilmente correlati in un altro insieme di valori linearmente non correlati detti componenti principali (*Principal Components*)[22, 45].

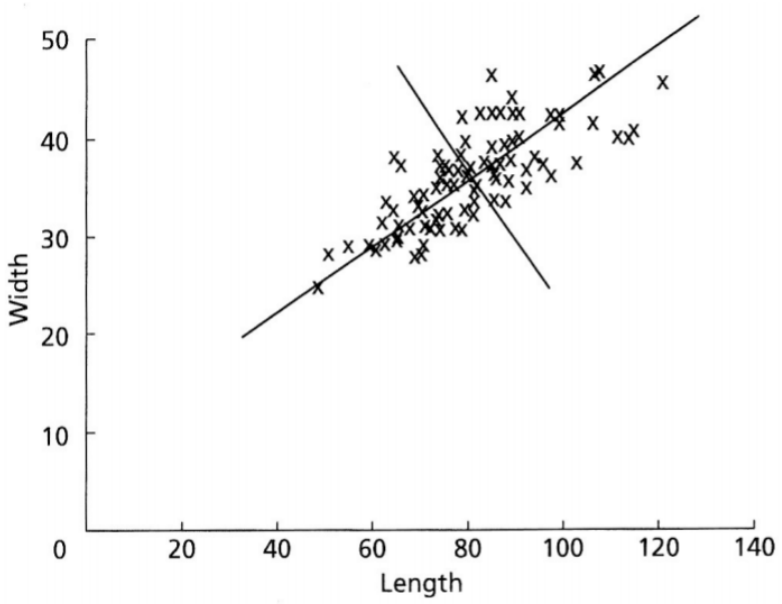


Figura 4.1: Esempio di utilizzo del PCA[22]

Il numero delle componenti principali è minore o uguale al numero di input (feature). Ogni componente principale può essere definita come un'asse di riferimento per la quale si ottiene la massima varianza, o in altre parole, che permetta la perdita minore di informazione. In pratica, il primo componente principale sarà quello che avrà la varianza massima, il secondo componente principale la seconda varianza più elevata e così via.

L'analisi dei componenti principali è quindi un'approssimazione dei dati di partenza. Le componenti principali sono le proiezioni dei dati su una nuova asse, operazione che comporta inevitabilmente una perdita di informazioni. Da notare però che calcolando il numero totale di componenti principali, non si avrà una perdita di informazioni, tuttavia non viene mai effettuato

in pratica in quanto non si ottengono informazioni aggiuntivi, vanificando lo scopo del PCA stesso.

I componenti principali ottenuti possono essere utilizzati al posto delle feature negli algoritmi di Machine Learning.

Selezione delle feature

A differenza dell'estrazione di feature nella quale i dati vengono trasformati, la selezione delle feature è appunto la selezione di un sottoinsieme dei dati di partenza attraverso diverse strategie[18].

- *Filter*: le variabili sono selezionate in fase di preprocessamento senza tenere in considerazione l'algoritmo di Machine Learning utilizzato.

- *Wrapper*: valuta il valore predittivo delle singole variabile e individua le possibili interazioni tra esse attraverso l'utilizzo di una macchina per l'apprendimento come blackbox.

- *Embedded*: le feature sono selezionate durante il processo di training.

Durante le sperimentazioni, l'algoritmo di selezione delle feature utilizzato appartiene alla tipologia wrapper. Confronta l'accuratezza generata delle singole feature e ritorna una percentuale di feature dei dati originali, composta solamente da quelle che hanno ottenuto uno score elevato.

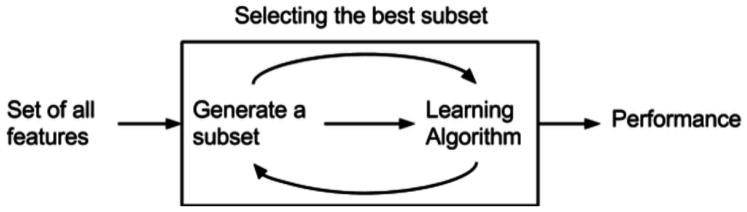


Figura 4.2: Funzionamento del metodo Wrapper [46]

4.2 Valutazione degli algoritmi e dei dati

Una volta effettuati tutti i preparativi, si è passato al testing dei modelli predittivi attraverso l'utilizzo delle varie feature ottenute. Quel che si vuol valutare è la capacita dei vari algoritmi di Machine Learning di predire le stime progettuali delle varie user story al variare dell'algoritmo, dei parametri, delle feature in input e del numero di elementi di testing.

4.2.1 Algoritmi utilizzati

Algoritmi di regressione

Essendo i valori da predire valori reali, gli algoritmi utilizzati fanno tutti parte della categoria "Regressione". Questi sono già stati implementati nella libreria Scikit-learn, il quale fornisce un utilizzo immediato di qualsiasi tipo di algoritmo. É sufficiente seguire il pattern seguente:

```
from sklearn import regression_algorithm
#creazione del modello
reg = regression_algorithm.Algorithm(some parameters)
#processo di training
reg.fit(feature_train, label_train)
#processo di testing
pred = reg.predict(feature_test)
```

Per le sperimentazioni sono stati utilizzati quattro diversi algoritmi di regressione:

- *Lineare*: senza nessun parametro specificato;
- *Vettori di supporto (SVR)*: con kernel lineare o gaussiana (rbf) e parametro C=10;
- *k-neighbourg*: numero di vicini presi in considerazione pari a 25 con peso uniforme;
- *Albero di decisione*: 3 campioni per essere considerato un nodo foglia.

Metriche

La principale metrica per la valutazione dei vari modelli predittivi è lo *score* R^2, la quale funzione è già stata implementata in Scikit-learn[41]. R^2 è il coefficiente di determinazione, cioè un numero che indica la proporzione tra il modello statistico utilizzato e la variabilità dei dati. I valori possibili per lo score R^2 vanno da 1.0, per uno score perfetto, a un numero negativo indefinito, in quanto l'algoritmo di regressione potrebbe fornire delle predizioni arbitrariamente inesatte.

$$R^2(eff, pred) = 1 - \frac{\sum_{i=0}^{n_{samples}-1}(eff_i - pred_i)^2}{\sum_{i=0}^{n_{samples}-1}(eff_i - med)^2}$$

$$med = \frac{1}{n_{samples} - 1} \cdot \sum_{i=0}^{n_{samples}-1} eff_i$$

dove eff è il valore reale, $pred$ la predizione fornita dal modello e med la media dei valori reali.

Oltre allo score R^2, si è tenuto in considerazione l'entità dell'errore nella predizione (valore assoluto della differenza tra la predizione e valore effettivo) e il valore medio di essi.

4.2.2 Approcci al problema

Valutazione della dimensione del training set

Nel Machine Learning, aumentando la quantità di dati forniti per il training si può ottenere un significativo miglioramento dell'accuratezza della predizione[12]. Tuttavia, una sproporzione tra la cardinalità del training set rispetto al testing set potrebbe risultare in una situazione di overfitting. Per questa ragione, la soluzione ottimale sarebbe trovare un compromesso. Generalmente i valori scelti corrispondono a un 75% di dati per il training e il restante 25% per il testing.

Ai fini di questa tesi, è utile verificare anche il grado di precisione dell'algoritmo in presenza di pochi elementi a propria disposizione. Infatti, in

un caso reale, questo processo di automatizzazione delle stime può fornire un elevato livello di supporto soprattutto per le fasi iniziali dello sviluppo. Durante le varie fasi delle sperimentazioni, si è utilizzato una forma manuale e rudimentale di *k-fold cross validation*, dividendo il dataset in un numero fissato e uguale di parti e comparando i risultati ottenuti incrementando di volta in volta la dimensione del training set.

Scelta del target e delle feature

Durante la fase di realizzazione del dataset, per ogni requisito, sono state effettuate delle stime progettuali (sezione 3.2.2) i quali possono essere ritenuti come i target dei modelli predittivi. Quasi la totalità degli algoritmi di regressione non permettono la predizione di più valori contemporaneamente; i pochi che prevedono questa opzione invece, predicono i singoli target per poi concatenarli in un unico vettore. Detto ciò, è banale osservare che predire più valori insieme non fornisce più o meno informazioni rispetto alla predizione dei target separatamente.

Per quanto riguarda le feature in input, nonostante l'algoritmo di feature selection fornisca già quelle con un impatto più elevato sui risultati, essendo esso un algoritmo non supervisionato, può avere comunque dei margini di errore. Durante i vari test si è quindi provato a verificare il comportamento del modello predittivo utilizzando o meno le varie informazioni fornite dal dataset o ricavate indirettamente da esse.

4.2.3 Risultati

I test sono stati effettuati in modo da determinare come le varie combinazioni di feature e parametri influenzino l'accuratezza delle predizioni del modello e se esso possa effettuare un buon lavoro anche nella situazione in cui si hanno poche informazioni.

In quasi la totalità dei test, l'algoritmo di regressione a macchine a vettori di supporto con kernel lineare è stato quello che ha fornito i risultati

migliori, per questa ragione se non diversamente specificato i risultati forniti si considerano come generati da SVR.

Intuitivamente, le stime da ottenere sono direttamente proporzionali alla difficoltà dell'implementazione della user story, quindi un modello lineare risulta sicuramente più semplice ed efficace. Un modello lineare e un modello SVR con kernel lineare si differenziano per il fatto che quest'ultimo tiene in considerazione anche un numero di elementi intorno alla retta ottenuta (parametro c) rendendola più efficace per le funzioni reali più complesse.

Dato che solo un algoritmo ha generato risultati significativi, si è cercato di valutare il suo comportamento utilizzando configurazioni diverse di input.

Configurazione ottimale

Per prima cosa si è cercato la configurazione ottimale di feature (con la quale si ottiene lo score più elevato) per ciascun target.

Per ognuno, le feature selezionate sono le seguenti:

- *LOC*: sforzo, entropia, servizi;
- *nuove classi*: classi modificate, entropia;
- *classi modificate*: LOC, nuove classi, entropia;
- *sforzo*: LOC, n.revisioni test, entropia, servizi, utilizzando feature selection al 4%;
- *n. revisioni test*: LOC, classi modificate, sforzo;
- *entropia*: LOC, nuove classi, classi modificate.

Come mostrato nella tabella 4.1 (score ottenuti selezionando le feature elencate sopra), le predizioni ottenute per il numero delle classi create e modificate, e il livello dell'entropia delle modifiche sono pressoché perfette. Buoni i risultati per il numero di LOC e lo sforzo mentre quelli relativi al numero di revisioni di test risultano essere i peggiori. Da notare come, nonostante la tendenza generale sia quella di ottenere un'accuratezza più elevata

n.training	LOC	n.class	c.class	effort	n.test	entropy
25	0.8799	0.9544	0.9522	0.8134	0.7093	0.7979
50	0.8580	0.9560	0.9709	0.7937	0.7035	0.9617
75	0.7191	0.9763	0.9868	0.8112	0.6887	0.9918
100	0.8228	0.9773	0.9865	0.9010	0.7189	0.9945
125	0.8132	0.9804	0.9851	0.8826	0.7260	0.9925
150	0.7198	0.9711	0.9926	0.8573	0.6736	0.9968
175	0.8060	0.9693	0.9965	N/A	0.6538	0.9985

Tabella 4.1: R^2 per le configurazioni ottimali

all'aumentare del numero di dati per il training (eccetto alcune eccezioni), questi miglioramenti risultino essere piuttosto minimi.

In realtà analizzando le singole predizioni, alcune user story potrebbero essere state sovrastimate o sottostimate in fase di realizzazione del dataset. Per questo i valori ottenuti, anche se con un errore elevato, non sono da considerarsi propriamente sbagliate.

Purtroppo, nonostante i risultati siano piuttosto buoni, non forniscono informazioni sufficientemente interessanti. Infatti, è piuttosto prevedibile che valori altamente correlati tra loro riescano a predirsi a vicenda, ad esempio come accennato nella sezione 3.2.2, l'entropia è stata calcolata come direttamente proporzionale alle classi e alle linee di codice necessarie per l'implementazione della user story. Inoltre, i risultati ottenuti non prendono in considerazione le informazioni ottenute dai campi testuali. Da ciò si evince che i valori numerici stimati hanno potere informativo nettamente superiore ai valori ottenuti tramite gli algoritmi di NLP.

Con solo dati forniti dal Product Backlog

Più interessante è quindi verificare il comportamento del modello predittivo fornendo solamente le informazioni ottenute dai campi testuali. Le feature utilizzate sono quindi state: la lunghezza del testo, il vettore fornito dal tf-idf

n.training	LOC	n.class	c.class	effort	n.test	entropy
25	0.01609	-0.6500	-1.5184	0.0853	-0.1258	-0.2320
50	0.1823	-0.1762	-0.9380	0.0560	-0.3015	0.0209
75	0.0902	-0.2045	-0.5927	0.1525	-0.1547	0.0198
100	0.1371	-0.5321	-0.8225	-0.0017	0.1222	0.2667
125	0.1911	-0.0505	-0.6251	-0.0960	0.0268	0.2398
150	0.1716	-0.3404	-0.0449	0.0310	0.2893	0.1494
175	0.0196	-0.3405	0.0403	0.1013	0.3111	0.3360

Tabella 4.2: R^2 utilizzando le informazioni testuali e feature selection

e quello generato dall'analisi semantica; elaborando queste informazioni con o senza feature selection o PCA.

Purtroppo i risultati ottenuti sono ben al di sotto delle aspettative. Gli score ottenuti sono bassi per tutte le combinazioni, la tabella 4.2 riporta i risultati ottenuti utilizzando tutte e tre le categorie di informazioni ricavate dal testo e applicando la feature selection.

Lo score R^2 è un intuitivo metodo di misurazione dell'accuratezza della predizione, infatti come accennato nella sezione 4.2.1, misura il livello di deviazione della predizione rispetto al valore effettivo. La figura 4.3 mostra il grafico generato dal modello predittivo formato dalle informazioni testuali (100 elementi di training e 100 di testing), inoltre, data l'impossibilità di rappresentare graficamente dati a elevata dimensionalità, si sono ridotte queste feature alla loro componente principale primaria. Come si può notare, i dati non forniscono una correlazione tra le informazioni testuali e l'entropia. I dati sono piuttosto sparsi e non sembrano seguire un andamento uniforme. La deviazione della predizione raggiunge livelli piuttosto elevati raggiungendo oltre le 200 unità di errore, abbassando notevolmente lo score.

Lo score R^2 stima quindi quanto sia forte la relazione tra il modello e la variabile di risposta, tuttavia non fornisce un ipotesi formale in merito a questa relazione[15].

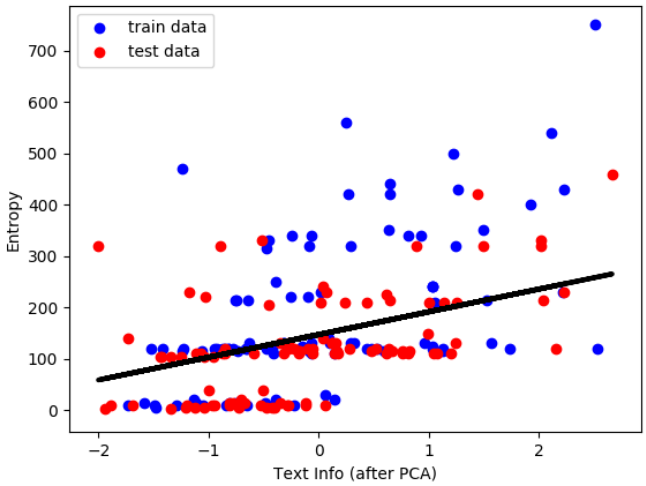
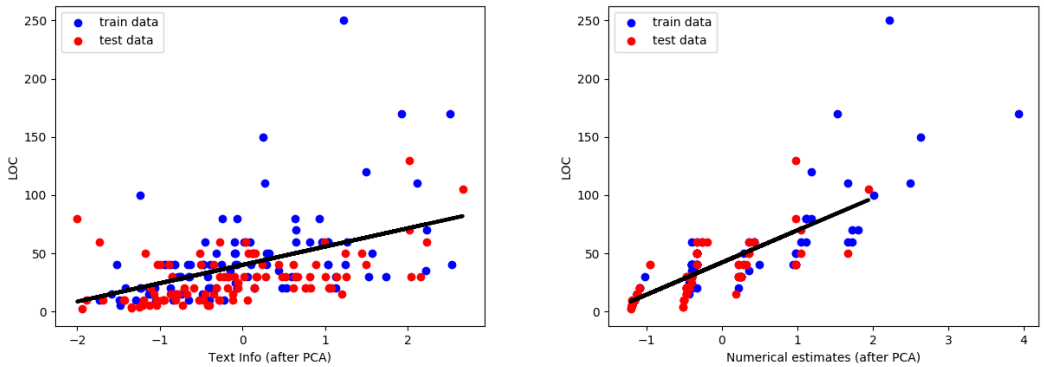


Figura 4.3: Grafico: predizione del livello di entropia con l'utilizzo di feature testuali



(a) con feature testuali (b) con feature numerici

Figura 4.4: Confronto grafici per LOC

Nella figura 4.4 invece, vengono messi a confronto i modelli con le feature testuali con quelle numeriche. Eccetto alcuni outlier, il grafico (b) ha chiaramente un pattern abbastanza lineare (la configurazione ottimale ha un errore inferiore a quello mostrato in figura ma questo non può essere rappresentato tramite un grafico), mentre nel grafico (a), come nella figura 4.3, le relazioni tra modello e target non sono molto evidenti, nonostante si possa notare una qualche lieve forma di linearità nella disposizione dei dati.

É necessario chiedersi quali siano le ragioni che hanno portato a questi risultati.

La prima considerazione è la non correlazione tra i valori ottenuti dai campi testuali e i target. Ciò non significa che il significato delle user story non abbia impatto su quelle che sono le stime progettuali, tuttavia, processare il linguaggio naturale non è un lavoro perfetto e assoluto. Gli algoritmi utilizzati possono essere considerati piuttosto semplici in quanto lo scopo di questa tesi non era incentrato su questo aspetto. Nell'estrazione delle feature si è ottenuto un vettore che riassumesse la semantica delle user story. Si è effettuato quindi un'analisi a campione su questi vettori utilizzando la funzione *similar_by_vector* la quale elenca le parole che assomiglino semanticamente di più al testo fornito in input. Nella maggior parte dei casi la prima e/o seconda posizione sono occupate dalla parola "user" o "users", da ciò si evince che tutte le user story hanno una parte rilevante in comune e che quindi non permette una distinzione marcata tra esse rispetto all'utilizzo delle feature numeriche. Senza contare che le user story sono molto brevi. Anche considerando il campo "user story elaboration" non si ha una visione completa della complessità dell'implementazione.

In secondo luogo il numero degli elementi utilizzati per questo esperimento possono essere considerati relativamente pochi in considerazione del tipo di risultati che si è cercato di ottenere. Come accennato nella sezione 2.3.2 più la funzione reale è complessa, maggiore dovrà essere la dimensione del training set. Nonostante gli score forniti nella tabella 4.2 non sembrino supportare questa tesi, andando a osservare l'errore medio effettivo generato

n.training	LOC	effort	entropy
25	18	4	99
50	18	3	80
75	17	3	77
100	15	3	65
125	14	3	60
150	12	3	63
175	13	3	54

Tabella 4.3: Unità di errore medio con feature selection e informazioni testuali

dalla predizione, in quasi tutti i casi si può osservare di una diminuzione di questo valore all'aumentare della dimensione dell'input come mostrato nella tabella 4.3.

Come ultima osservazione, alcune tipologie di dati contengono intrinsecamente un livello elevato di variabilità inspiegabile. Ad esempio, molti studi sulla psicologia hanno uno score R^2 inferiore al 50%[15]. I valori semantici non potranno mai essere perfetti, per questa ragione non è ragionevole aspettarsi score eccessivamente elevati.

Riduzione della difficoltà del problema

Effettuare delle stime progettuali può essere un problema piuttosto complesso, soprattutto quando non il livello di conoscenza non è elevato e non si hanno molte informazioni a propria disposizione. Può essere quindi troppo pretenzioso effettuare delle stime di questo genere attraverso l'utilizzo di tecniche superficiali. Si è quindi pensato di ridurre la complessità del problema da risolvere.

Come ulteriore parametro a ogni user story, si è aggiunto il "livello di difficoltà" dal punto di vista implementativo. Per semplicità si hanno solamente tre livelli: facile, medio, difficile. Per determinare il livello di ciascuna

n.training	FS & PCA	PCA	FS	None
25	43%	52%	45%	53%
50	48%	53%	49%	47%
75	55%	53%	44%	53%
100	56%	54%	40%	49%
125	54%	55%	45%	56%
150	58%	60%	50%	56%
175	68%	68%	48%	68%

Tabella 4.4: Percentuale di predizioni corrette con classificazione

user story si è deciso di far riferimento alle stime già effettuate quali LOC o effort, facendo una suddivisione che permettesse di avere una distribuzione piuttosto equa tra i vari livelli di difficoltà.

L'algoritmo utilizzato è un classificatore a macchine a vettori di supporto (SVC) con kernel "rbf" e parametro $c = 5$. La tabella 4.4 mostra i risultati ottenuti utilizzando solamente le tre categorie di informazioni ottenute dai campi testuali e applicando o meno gli algoritmi di feature selection e PCA.

Mediamente l'algoritmo riesce a classificare correttamente più del 50% degli elementi di test. Sicuramente un risultato ancora non ancora sufficiente. Ciò nonostante, eccetto nel test effettuato utilizzando solo la feature selection, l'accuratezza è aumentata considerevolmente all'aumentare del numero di elementi di training supportando quindi la tesi discussa nel paragrafo precedente.

La Figura 4.5 mostra un esempio dei risultati ottenuti per il test con feature selection e PCA. Dalla figura non si evincono delle suddivisioni nette tra le tre categorie, tuttavia si può notare un accentramento degli elementi in blu (user story "facili") il quale è correttamente individuato dall'algoritmo di classificazione. Da ciò si evince che in qualche modo si possono correlare le informazioni ottenute dal testo alla difficoltà implementativa. Al contrario, gli elementi rossi(user story "difficili") risultano essere sparsi per tutto il

grafico senza un effettivo nesso logico.

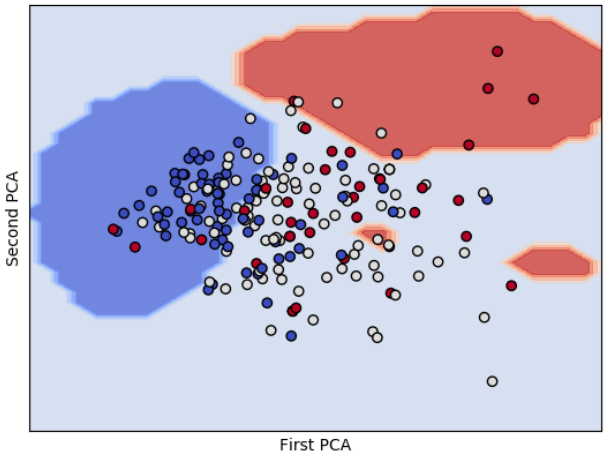


Figura 4.5: Grafico: Classificazione delle user story in base alla loro difficoltà

Conclusioni

In questa tesi si è proposto quello che può essere considerato un prototipo di un tool di supporto alla progettazione software in un processo agile. Lo script realizzato permette di effettuare un'analisi dei dati di un Product Backlog attraverso una combinazione di tecniche di NLP, feature selection e feature extraction, per poi realizzare un modello predittivo di regressione a partire dalle feature ottenute. Il modello sarà in grado di fornire le stime progettuali relative al costo, tempo e dimensione delle user story da implementare, in relazione ai parametri scelti.

Le precisioni delle predizioni ottenute varia in base alle informazioni in input e all'algoritmo utilizzato. Per quanto riguarda quest'ultimo, indipendentemente dalle feature utilizzate, l'algoritmo più performante è stato la regressione con macchine a vettori di supporto con kernel lineare. Ciò deriva dal fatto che i pattern nei dati sono prevalentemente lineari, ma rispetto a un algoritmo di regressione lineare, SVR è più flessibile in quanto tiene in considerazione un numero di fattori maggiori. In merito ai dati in input, come era prevedibile, più informazioni si forniscono al modello predittivo, maggiore sarà la sua accuratezza.

Allo stato attuale, lo script realizzato è ancora lontano da poter essere considerato di supporto in una situazione reale. Tuttavia, nonostante alcuni risultati non rispecchiassero esattamente quelli attesi, diversi sono i possibili margini di miglioramento. Eventuali sviluppi futuri si potrebbero concentrare su due fronti differenti:

- *I dati*: il dataset è stato realizzato a partire da un progetto fittizio, per

questa ragione sarebbe utile verificare il comportamento dello script realizzato utilizzando dati reali (ottenuti da un progetto reale). Inoltre, la quantità dei dati può essere considerato troppo esiguo per poter dare un giudizio definitivo in merito ai risultati ottenuti. Detto ciò, si potrebbe creare un dataset composto da più progetti con domini più o meno simili tra loro, ciascuno composto da un numero elevato di requisiti. Oltre alla qualità dei dati del dataset, si possono avere miglioramenti anche in merito alla qualità delle informazioni ottenute dal dataset. Gli algoritmi di NLP utilizzati sono risultati troppo generici, ad esempio, la semantica ottenuta dalle user story non permetteva una suddivisione netta in merito alla difficoltà dell'implementazione.

- *Il problema*: in questa tesi si è svolta una valutazione diretta dei dati attraverso un algoritmo di regressione. Come affermato nel corpo, è difficile trovare correlazioni ben definite a partire da informazioni vaghe come quelle ottenute dal linguaggio naturale. Una macchina non ha l'esperienza di un programmatore e non riesce a contestualizzare il requisito. L'ultimo test effettuato ha però mostrato che diminuendo il livello di difficoltà del problema da risolvere si possono ottenere dei risultati discreti. L'idea sarebbe quindi di raggiungere i nostri obiettivi con una metodologia top-down, riducendo ciò che si vuol ottenere nelle componenti più elementari, utilizzando i risultati stessi come input e determinando infine le stime cercate.

Appendice

Per questa tesi è stato realizzato un dataset in formato .xlsx, successivamente convertito nel formato .csv. Il dataset è composto da 200 requisiti Scrum. Ciascun record è formato da:

- cinque campi testuali: User Story, Business Value, User Story Elaboration, Definition of Done ed Expected Output;
- sei campi numerici: LOC, N. New Classes, N. Changed Classes, Effort e Changes Entropy;
- un campo in forma matriciale: Services Dependencies.

Inoltre, gli esperimenti sono stati svolti con uno script da me realizzato disponibile pubblicamente all'indirizzo:

https://github.com/GiulioZhou/ML_for_AgileRE

Il progetto è così composto:

- *textLearning.py*: effettua una manipolazione dei testi attraverso lo *stemming*, con l'utilizzo di *Snowball stemmer* in *NLTK*[1], e la rimozione delle *stopwords*.

  Converte il testo in un vettore di feature Tf-Idf attraverso l'algoritmo *TfidfVectorizer*[2] implementato in *Scikit-learn*.

  Carica un modello pre-addestrato nel formato *doc2vec*[3] e inferisce la semantica del testo.

[1] http://www.nltk.org/api/nltk.stem.html

[2] http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

[3] https://radimrehurek.com/gensim/models/doc2vec.html

- *dataExtractor.py*: carica il dataset in formato .csv e lo converte in una struttura adeguata all'elaborazione. Ottiene le informazioni dei testi chiamando le funzioni contenute nel file *textLearning.py*.

  Effettua una selezione delle feature con *SelectPercentile*[4] per poi fare l'analisi delle componenti principali[5].

  Infine, ritorna i valori suddivisi in training set e testing set.

- *getFeatures.py*: chiede all'utente quali feature utilizzare, il target da predire e la dimensione del training set.

- *regression.py*: realizza un modello di predizione[6]. I modelli possono essere regressioni (lineare, SVR, k-neighbors, Bayessiana o ad albero) o classificazioni (SVC).

- *script.py*: utilizza il modello realizzato in *regression.py* e i dati ottenuti in *dataExtractor.py* per effettuare le predizioni, salvare i risultati in un file .xlsx con il framework *XlsxWriter*[7] e realizzare un grafico in 2d con *Matplotlib*[8]

- *fast_test_script.py*: a differenza di *script.py*, le informazioni per il test non vengono ottenute chiamando *getFeatures.py*, ma sono inserite direttamente nel codice. Lo script effettua le predizioni per tutti i target e dimensioni del training set (aumentando, ad ogni iterazione, la dimensione di 25 record).

In totale, le linee di codice prodotte, inclusi i commenti, sono 638.

[4] http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectPercentile.html
[5] http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html
[6] http://scikit-learn.org/stable/supervised_learning.html#supervised-learning
[7] http://xlsxwriter.readthedocs.io/
[8] https://matplotlib.org/

Bibliografia

[1] 1&1. Live chat: una moderna soluzione di supporto, 2016. url: https://www.1and1.it/digitalguide/siti-web/creare-siti/live-chat-una-moderna-soluzione-di-supporto/.

[2] AAMS. Linee guida per la certificazione della piattaforma di gioco versione 1.2, 2014. url: https://www.agenziadoganemonopoli.gov.it/portale/monopoli/giochi/giochi_abilita/giochi_ab_cert.

[3] Aizawa, A. An information-theoretic perspective of tf–idf measures. *Information Processing & Management 39*, 1 (2003), 45–65.

[4] Beck, K. *Extreme programming explained: embrace change*. addison-wesley professional, 2000.

[5] Bird, S. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions* (2006), Association for Computational Linguistics, pp. 69–72.

[6] Black, A. W. Betting exchange system, 2010. US Patent 7,690,991.

[7] Brown, B., Chui, M., and Manyika, J. Are you ready for the era of 'big data'. *McKinsey Quarterly 4*, 1 (2011), 24–35.

[8] Capitanucci, D. Strategie di prevenzione del gioco d'azzardo patologico tra gli adolescenti in italia. l'utilizzo di strumenti evidence-based per distinguere tra promozione e prevenzione. *Italian Journal on Addiction 2*, 3-4 (2012).

[9] Cohn, M. *Agile estimating and planning.* Pearson Education, 2005.

[10] Davies, M., Pitt, L., Shapiro, D., and Watson, R. Betfair. com:: Five technology forces revolutionize worldwide wagering. *European Management Journal 23*, 5 (2005), 533–541.

[11] Dyba, T. Improvisation in small software organizations. *IEEE Software 17*, 5 (2000), 82–87.

[12] Foody, G., McCulloch, M., and Yates, W. The effect of training set size and composition on artificial neural network classification. *International Journal of Remote Sensing 16*, 9 (1995), 1707–1723.

[13] Forbes, D. S. Agile: The world's most popular innovation engine, 2015. url: `https://www.forbes.com/sites/stevedenning/2015/07/23/the-worlds-most-popular-innovation-engine/#70537a3c7c76`.

[14] Fowler, M., and Highsmith, J. The agile manifesto. *Software Development 9*, 8 (2001), 28–35.

[15] Frost, J. How to interpret a regression model with low r-squared and low p values, 2014. url: `https://goo.gl/qLztgv`.

[16] Geman, S., Bienenstock, E., and Doursat, R. Neural networks and the bias/variance dilemma. *Neural computation 4*, 1 (1992), 1–58.

[17] Griffiths, T. L., Steyvers, M., and Tenenbaum, J. B. Topics in semantic representation. *Psychological review 114*, 2 (2007), 211.

[18] Guyon, I., and Elisseeff, A. An introduction to variable and feature selection. *Journal of machine learning research 3*, Mar (2003), 1157–1182.

[19] Halevy, A., Norvig, P., and Pereira, F. The unreasonable effectiveness of data. *IEEE Intelligent Systems 24*, 2 (2009), 8–12.

[20] Hassan, A. E. Predicting faults using the complexity of code changes. In *Proceedings of the 31st International Conference on Software Engineering* (2009), IEEE Computer Society, pp. 78–88.

[21] Hofmann, H. F., and Lehner, F. Requirements engineering as a success factor in software projects. *IEEE software 18*, 4 (2001), 58.

[22] Holland, S. M. Principal components analysis (pca). *University of Georgia* (2008).

[23] IEEE. *Standard Glossary of Software Engineering Terminology*. IEEE Standard 610.12-1990, 1990.

[24] James, G. M. Variance and bias for general loss functions. *Machine learning 51*, 2 (2003), 115–135.

[25] Jones, C. Software project management practices: Failure versus success. *CrossTalk: The Journal of Defense Software Engineering 17*, 10 (2004), 5–9.

[26] Lau, J. H., and Baldwin, T. An empirical evaluation of doc2vec with practical insights into document embedding generation. *arXiv preprint arXiv:1607.05368* (2016).

[27] Le, Q., and Mikolov, T. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)* (2014), pp. 1188–1196.

[28] Leffingwell, D. *Agile software requirements: lean requirements practices for teams, programs, and the enterprise*. Addison-Wesley Professional, 2010.

[29] Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[30] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (2013), pp. 3111–3119.

[31] Monaghan, S. Responsible gambling strategies for internet gambling: The theoretical and empirical base of using pop-up messages to encourage self-awareness. *Computers in Human Behavior 25*, 1 (2009), 202–207.

[32] Murphy, K. P. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[33] Newman, S. *Building microservices*. " O'Reilly Media, Inc.", 2015.

[34] Paetsch, F., Eberlein, A., and Maurer, F. Requirements engineering and agile software development. In *Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003. WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on* (2003), IEEE, pp. 308–313.

[35] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research 12*, Oct (2011), 2825–2830.

[36] Perini, A., Susi, A., and Avesani, P. A machine learning approach to software requirements prioritization. *IEEE Transactions on Software Engineering 39*, 4 (2013), 445–461.

[37] Rehurek, R., and Sojka, P. Gensim–python framework for vector space modelling. *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic* (2011).

[38] Rossant, C. Introduction to machine learning in python with scikit-learn, 2014. url: `http://ipython-books.github.io/featured-04/`.

[39] Rubin, K. S. *Essential Scrum: A practical guide to the most popular Agile process.* Addison-Wesley, 2012.

[40] Schopper, M. D. Internet gambling, electonic cash & (and) money laundering: The unintended consequences of a monetary control scheme. *Chap. L. Rev. 5* (2002), 303.

[41] Scikit-learn. sklearn.metrics.r2score. url: `http://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2_score.html#sklearn.metrics.r2_score`.

[42] Sommerville, I. *Software engineering.* Pearson, 2011.

[43] Sommerville, I., and Sawyer, P. *Requirements engineering: a good practice guide.* John Wiley & Sons, Inc., 1997.

[44] Taylor, L. M., and Hillyard, P. Gambling awareness for youth: An analysis of the "don't gamble away our future^{TM}" program. *International Journal of Mental Health and Addiction 7*, 1 (2009), 250.

[45] Tipping, M. E., and Bishop, C. M. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 61*, 3 (1999), 611–622.

[46] Wikipedia. Feature selection. url: `https://en.wikipedia.org/wiki/Feature_selection`.

[47] Zhang, D., and Tsai, J. J. Machine learning and software engineering. In *Tools with Artificial Intelligence, 2002.(ICTAI 2002). Proceedings. 14th IEEE International Conference on* (2002), IEEE, pp. 22–29.