Alma Mater Studiorum · Università di Bologna

CAMPUS DI CESENA
SCUOLA DI SCIENZE
CORSO DI LAUREA IN INGEGNERIA E SCIENZE INFORMATICHE

APPLICAZIONE DEL MACHINE LEARNING A SCENARI DI INQUINAMENTO AMBIENTALE

Tesi di Laurea in:
SISTEMI OPERATIVI

Relatore:
Chiar.mo Prof. VITTORIO GHINI

Presentata da:
STEFANO RIGHINI

Prima Sessione di Laurea
Anno Accademico 2017-2018

Sommario

L'inquinamento ambientale è un argomento sempre più presente e importante nell'era moderna. Molte volte lo trascuriamo ma non ci accorgiamo che stiamo trascurando anche la nostra salute. Sono quindi necessari alcuni sistemi per tenerne monitorato l'andamento e poter intervenire di conseguenza.

Indice

1 Introduzione **4**
- 1.1 Inquinamento ambientale . . . 4
- 1.2 Livelli di inquinamento in Italia . . . 4
- 1.3 Cosa si vuole realizzare . . . 4
  - 1.3.1 Attribuire un valore mancante . . . 6
  - 1.3.2 Previsione dei valori . . . 7
- 1.4 Descrizione del problema . . . 7
- 1.5 Generalizzazione del problema . . . 7

2 Analisi **8**
- 2.1 Analisi dei dati . . . 8
  - 2.1.1 Particolato di materia 2.5 . . . 8
  - 2.1.2 Dati correlati al PM 2,5 . . . 8
- 2.2 Analisi di correlazione . . . 10
  - 2.2.1 Miglioramento della correlazione . . . 11

3 Imputazione dei valori mancanti **14**
- 3.1 Valori mancanti . . . 14
  - 3.1.1 Tipologie . . . 15
  - 3.1.2 In quale tipologia ricadiamo? . . . 15
- 3.2 Imputazione . . . 16
  - 3.2.1 Imputazione singola . . . 17
    - 3.2.1.1 Metodi di corrispondenza . . . 17
    - 3.2.1.2 Imputazione media . . . 17
    - 3.2.1.3 Imputazione di regressione . . . 17
  - 3.2.2 Imputazione multipla . . . 18

4 Modelli applicabili al problema **19**
- 4.1 Introduzione . . . 19
- 4.2 Machine learning . . . 19
  - 4.2.1 Tecniche di apprendimento . . . 20

4.2.1.1 Apprendimento supervisionato 20
4.2.1.2 Apprendimento non supervisionato 20
4.2.1.3 Apprendimento semi supervisionato 20
4.2.1.4 Apprendimento rinforzato 20
4.2.1.5 Quale apprendimento va utilizzato? 21
4.2.2 Regressione 21
4.2.3 Support Vector Machine 21
4.2.4 Random Forest 22
4.2.5 Reti neurali 23
4.2.5.1 Allenamento di una rete 23
4.2.6 Reti neurali per la regressione 24
4.2.7 Deep neural network 24
4.3 Discussione generale sui modelli 24

5 Progettazione del sistema **26**
5.1 Progettazione dei dati 26
5.1.1 Raccolta dei dati 26
5.1.2 Preparazione dei dati 26
5.1.3 Ruolo dei dati 28
5.1.3.1 Variabili Indipendenti 28
5.1.3.2 Variabili Dipendenti 28
5.2 Librerie di sviluppo 28
5.2.1 Scikit-Learn 28
5.2.2 Tensorflow 29
5.2.3 Keras 29
5.3 Progettazione del Software 30
5.3.1 Modello iterativo 30

6 Processo di sviluppo **31**
6.1 Implementazione di più modelli 31
6.2 Test dei modelli 31
6.3 Reti neurali ricorrenti 32
6.4 Reti neurali a serie temporali 32
6.5 Ottimizzare il modello 32
6.5.1 Numero dei neuroni 33
6.5.2 Batch size 36
6.5.3 Quante rilevazioni precedenti osservare? 37
6.5.4 Quanti esempi per l'allenamento? 39
6.5.5 Funzione di ottimizzazione 40
6.5.6 Numero di epoche di allenamento 40
6.6 Previsione a più ore di distanza 43

6.7 Risultati dello sviluppo 43
6.8 Organizzazione del software 45

7 Implementazione **46**
7.1 Modulo di allenamento 46
7.2 Modulo per i valori mancanti 47
7.3 Modulo per fare previsioni future 48

8 Sviluppi Futuri **50**
8.1 Miglioramento della precisione 50
8.2 Sviluppo di un'interfaccia grafica 50
8.3 Ricerca di nuovi modelli 50

9 Conclusione **51**
9.1 L'inquinamento 51
9.2 Conoscenze acquisite 51

Capitolo 1

Introduzione

1.1 Inquinamento ambientale

Nell'era moderna l'inquinamento ambientale sta assumendo un ruolo sempre più importante. Negli ultimi anni il fenomeno sta raggiungendo livelli molto elevati e pericolosi, per questo sarebbe una buona prassi tenerlo monitorato per poi averne una visione completa nel tempo.

1.2 Livelli di inquinamento in Italia

La classifica europea mette sotto accusa le nostre città, che presentano concentrazioni record di polveri sottili; questo inquinante prende il nome di particolato di materia 2.5 (con diametro inferiore a 2.5 µm). Come scritto da Roberto Giovannini sul quotidiano La Stampa [1] il 28/09/2017 l'aria nelle città è migliorata negli ultimi anni, ma nonostante le nuove regole l'Italia primeggia nella triste classifica europea dei morti per inquinamento. La quota annuale di morti, come scritto nell'articolo "Smog, l'Italia maglia nera in Europa: 90mila morti l'anno" nel quotidiano La Repubblica [7], ammonta circa a 90 000 , circa 1500 per milione di abitanti all'anno.

1.3 Cosa si vuole realizzare

Tra i vari inquinanti si farà riferimento al Particolato di materia con diametro inferiore o uguale a 2.5 µm (PM 2.5) ovvero le polveri sottili. Il sistema avrà due compiti: il primo sarà quello andare ad attribuire un valore sensato al PM 2.5 se per caso non è stato possibile raccogliere il dato con sensori e quindi averne uno reale. Il secondo sarà quello di predire i valori del PM 2.5 nelle

24 ore successive. In altre parole il primo compito sarà quello di completare un dataset in cui è possibile che ci siano valori mancanti. Per fare chiarezza andiamo ad analizzare il contesto in cui il sistema dovrà operare. I dati saranno raccolti nella città di Bologna e l'obiettivo sarebbe quello di valutare e avere una visione completa dell'inquinamento nelle diverse zone della città. Per farlo Bologna sarà suddivisa in zone tramite una griglia virtuale, in un modo analogo a quello mostrato in figura. I dati raccolti saranno organizzati

Figura 1.1: Possibile suddivisione in zone della città di bologna tramite una griglia.

e relativi ad una zona, in un certo giorno e ad una certa fascia oraria. Per quanto riguarda le fasce orarie si è deciso di suddividere una giornata in 24 fasce in modo da ottenere una discrezione oraria. Quindi il valore del PM 2.5 in una certa zona, in un certo giorno e in una certa fascia oraria sarà uguale alla media delle rilevazioni in quella zona, in quel giorno e in quella fascia oraria. Si ottiene quindi come risultato un dataset che mostra per ogni zona, per ogni giorno e per ogni fascia oraria un valore medio di PM 2.5. Può capitare che in una certa zona, in un certo giorno e in una certa fascia oraria non si disponga di rilevazioni, in questo caso abbiamo un dato mancante. Se ci sono dati mancanti il sistema dovrà essere in grado come già detto di andare ad attribuire un valore il più coerente e corretto possibile. Per realizzare e

testare questo sistema abbiamo bisogno di un dataset descritto come sopra. Non avendo a disposizione però questi dati sono state effettuate delle ricerche per trovare un dataset con delle rilevazioni del PM 2.5. Il dataset che è stato trovato e sarà utilizzato per testare il sistema si riferisce alla città di Beijing (Pechino) in Cina. Il dataset è stato reperito da un repository dedicato al machine learning dell'Università della California, Irvine (UCI). In questo dataset i dati non sono suddivisi in zone ma rappresentano una media del valore nella città. Per questo motivo si è deciso di concentrarsi solo su un punto senza considerare le diverse zone della griglia; in altre parole i dati saranno considerati essere tutti relativi alla stessa zona. Questo permette di capire e concentrarsi su un singolo punto e valutare la potenza del sistema nel caso più semplice possibile, anche se non è da escludere che avendo solo un punto si potrebbero ridurre le informazioni a disposizione del sistema. Oltre a quanto già detto, concentrandosi su un solo punto, si vanno a ridurre il tempo e la difficoltà dei test. Tutti questi aspetti hanno portato a modificare in parte il problema e quindi, di conseguenza, anche i compiti che il sistema dovrà svolgere. Infatti adesso non si terrà più conto della zona in cui sono state fatte le rilevazioni, ma si avrà solo una media relativa ad un giorno e ad un'ora del valore del PM 2.5. Nel dataset oltre al giorno, alla fascia oraria e al PM 2.5 saranno presenti anche altri dati di tipo atmosferico come il vento, la temperatura, la pressione e le precipitazioni.

1.3.1 Attribuire un valore mancante

Il primo compito del sistema sarà appunto quello di attribuire un valore coerente e preciso qualora non si disponga di rilevazioni. Per farlo il sistema usufruirà delle rilevazioni ottenute nelle 12 ore precedenti in modo da poter osservare l'andamento e fornire un'accurata e coerente previsione. La scelta di utilizzare 12 ore non è casuale ma, è frutto di un processo di sviluppo che verrà illustrato e argomentato nell'apposito capitolo. In questo modo se al termine di una fascia oraria non si sono registrate rilevazioni è possibile attribuire un valore in modo da non avere valori mancanti. Se il dataset presenta valori mancanti in un tempo passato sarà sufficiente prendere le rilevazioni inerenti alle 12 ore precedenti e attribuire un valore completando così il dato mancante. Se ci troviamo al tempo t, di cui non abbiamo rilevazioni, sarà sufficiente prendere i dati dal tempo $t - 12$ a $t - 1$ per attribuire un valore al tempo t.

1.3.2 Previsione dei valori

Il secondo compito del sistema riguarda la previsione dei valori nelle 24 fasce orarie successive a quella attuale. Per farlo si andranno ad utilizzare i valori relativi alle ultime 12 fasce orarie fino a quella corrente. Il principio è molto simile ma quando si vuole prevedere a più ore di distanza, la precisione della previsione inevitabilmente diminuisce. Quindi la previsione a 24 fasce orarie di distanza sarà molto meno precisa di quella relativa all'ora successiva. Con lo scorrere del tempo la fascia oraria corrente cambia, quindi, appena ci sono rilevazioni, la previsione non viene più considerata, ma viene sostituita dalla media dei valori realmente rilevati. Assumendo di trovarci al tempo t e che i valori da $t - 11$ a t siano completi e disponibili, possiamo utilizzarli per fare previsioni dal tempo $t + 1$ al tempo $t + 24$.

1.4 Descrizione del problema

La parte di raccolta dei dati non viene considerata un obiettivo della tesi, quindi i problemi da risolvere sono due: il primo è attribuire un valore ai dati mancanti mentre il secondo è di effettuare previsioni in un futuro prossimo.

1.5 Generalizzazione del problema

Come nella maggior parte dei problemi, eseguendo la giusta analisi se ne riesce ad ottenere una forma più generale che, nella maggior parte delle volte, è più semplice, già risolta e che sicuramente sarà già stata ben argomentata e trattata. Analizzando il problema di sostituire i valori mancanti relativi al PM 2.5 con un valore coerente, si può generalizzare in un problema in cui esistono valori mancanti e dove sia necessario attribuirgli un valore. Questo problema viene definito "imputation of missing values" e verrà trattato in un capitolo a parte. Per quanto riguarda la previsione del valore del PM 2.5 nelle fasce orarie future possiamo generalizzare dicendo che è un problema in cui si hanno dei dati relativi al presente e al passato e si vuole predire come cambieranno nel futuro. Di questo non ne esiste una forma generale riconosciuta ma può essere semplicemente definito come previsione del futuro in base al presente e al passato.

Capitolo 2

Analisi

2.1 Analisi dei dati

Uno tra i punti più importanti della nostra elaborazione sono sicuramente i dati, infatti è necessario fare un'analisi per capire quali dati sono necessari e quali no. Oltre a questo sarà vitale capire in che modo utilizzare questi dati per ottenere più informazioni possibili e in modo più accurato possibile. In primo luogo analizziamo il PM 2,5 in quanto dobbiamo capire quali dati influenzano l'andamento nel tempo di questo valore.

2.1.1 Particolato di materia 2.5

Il particolato di materia sottile PM 2,5 è un inquinante presente nell'aria che se raggiunge livelli alti può essere molto dannoso per le persone. Queste piccolissime particelle quando presenti nell'aria a livelli elevati possono ridurre la visibilità.

2.1.2 Dati correlati al PM 2,5

Tra i dati correlati al PM 2,5 troviamo le precipitazioni piovose e nevose, il vento, il giorno dell'anno, l'ora, la pressione atmosferica, la temperatura e il punto di rugiada. Il punto di rugiada è la temperatura alla quale l'aria deve essere raffreddata per diventare satura di vapore acqueo. Per il vento saranno necessarie la provenienza e la velocità. Per predire un valore a partire dall'osservazione di altri valori è necessario che i valori in osservazione siano correlati al valore da predire.

Colonna	Descrizione
A	Anno
M	Mese
G	Giorno
O	Ora
R	Punto di rugiada
T	Temperatura
P	Pressione atmosferica
D	Direzione del vento
V	Velocità del vento
N	Ore di neve
P	Ore di pioggia

Tabella 2.1: Legenda per le colonne abbreviate.

A	M	G	O	PM 2.5	R	T	P	D	V	N	P
2010	1	2	0	129	-16	-4	1020	SE	1.79	0	0
2010	1	2	1	148	-15	-4	1020	SE	2.68	0	0
2010	1	2	2	159	-11	-5	1021	SE	3.57	0	0
2010	1	2	3	181	-7	-5	1022	SE	5.36	1	0
2010	1	2	4	138	-7	-5	1022	SE	6.25	2	0
2010	1	2	5	109	-7	-6	1022	SE	7.14	3	0
2010	1	2	6	105	-7	-6	1023	SE	8.93	4	0
2010	1	2	7	124	-7	-5	1024	SE	10.72	0	0
2010	1	2	8	120	-8	-6	1024	SE	12.51	0	0
2010	1	2	9	132	-7	-5	1025	SE	14.3	0	0
2010	1	2	10	140	-7	-5	1026	SE	17.43	1	0
2010	1	2	11	152	-8	-5	1026	SE	20.56	0	0
2010	1	2	12	148	-8	-5	1026	SE	23.69	0	0

Tabella 2.2: Dati relativi al 2 gennaio 2010 nelle prime 12 ore nella città di Pechino.

2.2 Analisi di correlazione

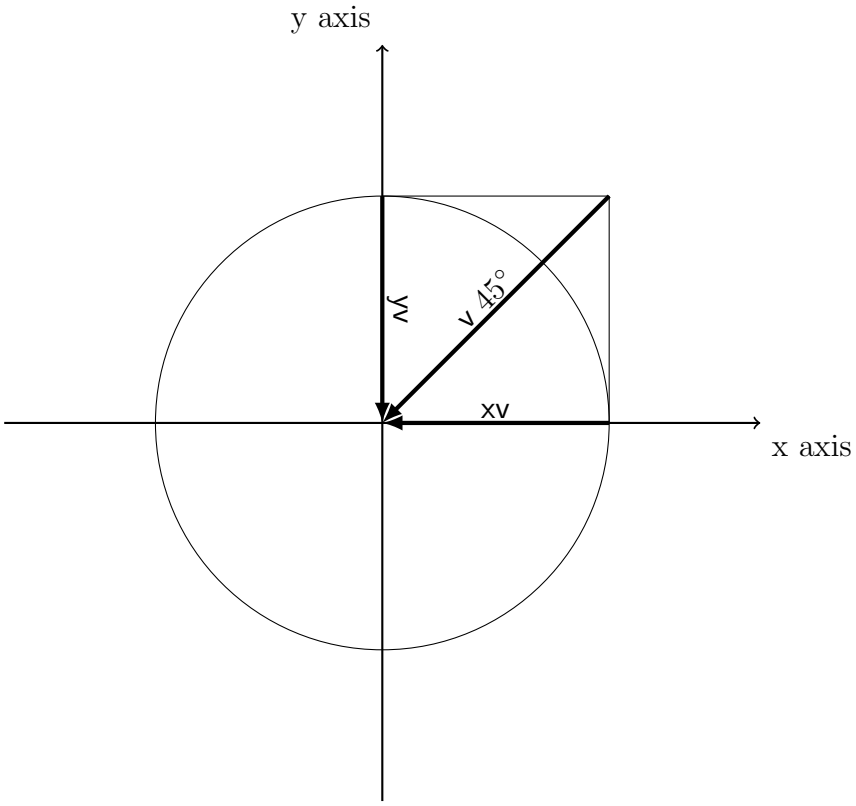
Come detto in precedenza oltre a capire quali dati sono correlati al valore di cui si vuole effettuare una previsione è necessario capire quanto questi dati siano correlati al suddetto valore. Per farlo si valuta la correlazione più o meno grande, positiva o negativa tra i dati. Nella tabella seguente vengono mostrate le relazioni tra il PM 2.5 e gli altri dati. I valori di correlazione variano nell'intervallo [-1;1] e mostrano le correlazioni positive o negative tra i dati. Più ci si avvicina allo 0 e meno i dati sono correlati mentre più ci si allontana più i dati sono positivamente, se il numero è positivo, o negativamente, se il numero è negativo, correlati. Si nota come non ci sia una grande correlazione tra i dati, sarà quindi necessario effettuare uno studio di correlazione per ottenere dei dati più correlati con il valore da prevedere in modo da ottenere risultati migliori. Questo è indipendente dal sistema che si andrà ad utilizzare.

Variabile	Correlazione con PM 2.5
Anno	-0.014690
Mese	-0.024069
Giorno	0.082788
Ora	-0.023116
PM 2.5	1.000000
Punto di rugiada	0.171423
Temperatura	-0.090534
Pressione atmosferica	-0.047282
Velocità del vento	-0.247784
Ore di neve	0.019266
Ore di pioggia	-0.051369

Tabella 2.3: Correlazione tra i dati.

2.2.1 Miglioramento della correlazione

In questo processo si utilizzano i dati a disposizione per crearne di nuovi cercando una correlazione migliore. Con i dati a disposizione si è cercato di organizzare meglio i dati relativi al vento, e al tempo inteso come giorno mese e anno. Partendo dal vento era necessario trovare un metodo o un sistema che permettesse di differenziarlo mediante un dato numerico rispetto alla sua direzione e alla sua velocità. In una prima analisi si è pensato di suddividere la colonna delle direzioni in tante colonne quante le direzioni e porre un valore di uno nella colonna corrispondente alla direzione giusta e zero nelle altre. Questa prima soluzione non ha però portato a buoni risultati né in termini di correlazione né in termini di risultati ottenuti dai test. Il motivo probabilmente è che i sistemi utilizzati, lavorando con valori normalizzati e continui in un certo intervallo, non riuscivano a gestire bene un salto così netto da 0 a 1. Scartata questa soluzione per trovare un modo per esprimere la direzione, si sono utilizzati i gradi e quindi l'angolazione del vento (es. SE indica la direzione sud est che corrisponde a 135°). I risultati sono però stati i medesimi o quasi in quanto con questa rappresentazione il sistema non conoscendo cosa in realtà i numeri siano considera un angolo di 0° completamente diverso da uno di 359° quando in realtà sono uno successivo all'altro e molto vicini. In queste due soluzioni la velocità resta espressa in metri al secondo. Fatte queste considerazioni è quindi necessario trovare una rappresentazione numerica del vento che riesca anche ad esprimere una corretta provenienza angolare e la sua corretta velocità. Si è quindi pensato di calcolare con seno e coseno due vettori direzione che indicassero la sua angolazione dandogli un senso di circolarità. A questi si è poi deciso di moltiplicare la velocità per avere due vettori che rappresentassero direzione e velocità.



v: rappresenta la direzione del vento
xv: il vettore direzione sull'asse x
yv: il vettore direzione sull'asse y

Assumendo che v abbia una lunghezza pari alla velocità (vel), espressa in metri al secondo, del vento in quella direzione e che l'angolo θ sia uguale a 45°, possiamo calcolare xv e yv come:

$$xv \rightarrow \sin\theta * vel$$
$$yv \rightarrow \cos\theta * vel$$

Questa soluzione permette di dare una direzione numericamente corretta in un range tra 0 e 1 e, moltiplicate le due direzioni, xv e yv, per la velocità, otteniamo due vettori direzione pesati con la velocità che permettono di avere una rappresentazione del vento migliore e con una correlazione migliore. Un altro dato che necessita una modellazione è il tempo; il dato relativo all'anno ad esempio è il meno correlato tra tutti i dati, infatti non è importante conoscere l'anno ma il periodo all'interno dell'anno. Riutilizzando i ragionamenti

già effettuati per il vento possiamo assumere l'anno come un periodo circolare in cui il primo giorno dell'anno sia immediatamente successivo all'ultimo. Possiamo quindi considerare un giorno come punto in un anno eliminando anche il concetto di mese, successivamente consideriamo i giorni come angoli normalizzandoli da 0° a 360° in modo che ogni giorno sia un vettore con un'angolazione θ precisa nel cerchio che rappresenta l'anno. A questo punto come con il vento calcoliamo i vettori direzione per gli assi x e y nel seguente modo:

$$xd \to \sin\theta$$
$$yd \to \cos\theta$$

Così facendo possiamo individuare un giorno all'interno di un anno in un modo corretto e comprensibile dal sistema di previsione. I nuovi dati ottenuti mostrano che la correlazione non è variata di molto, ma ora abbiamo una rappresentazione più comprensibile e corretta per il sistema sottostante. La correlazione maggiore è sempre imputata, come prevedibile, al vento, ma a differenza di prima adesso abbiamo trovato un sistema per avere una buona correlazione anche con la direzione e non solo con la velocità. Per quanto riguarda il tempo (giorno e ora) si possono notare dei piccoli miglioramenti che forniscono dei dati "migliori" per le fasi successive. Questo è il risultato emerso da questa analisi, anche se resta la possibilità di trovare delle rappresentazioni migliori.

Variabile	Correlazione con PM 2.5
Giorno x	−0.006 411
Giorno y	0.104 234
Ora x	0.034 771
Ora y	0.089 905
PM 2.5	1.000 000
Punto di rugiada	0.171 423
Temperatura	−0.090 534
Pressione atmosferica	−0.047 282
Vento x	0.218 454
Vento y	−0.241 087
Ore di neve	0.019 266
Ore di pioggia	−0.051 369

Tabella 2.4: Correlazione tra i dati elaborati.

Capitolo 3

Imputazione dei valori mancanti

3.1 Valori mancanti

In statistica i dati o valori mancanti [6] si presentano quando un dato o un valore non sono disponibili, quindi quando non è presente un valore per una variabile in osservazione. I dati mancanti possono presentarsi in molti ambiti e stadi della ricerca, per differenti motivi e in differenti forme. I dati mancanti possono avere conseguenze significative sulle conclusioni che si ottengono dall'analisi.

Anno	Mese	Giorno	Ora	PM 2.5
2010	1	1	20	NA
2010	1	1	21	NA
2010	1	1	22	NA
2010	1	1	23	NA
2010	1	2	0	129
2010	1	2	1	148
2010	1	2	2	159
2010	1	2	3	181

Tabella 3.1: Esempio di valori mancanti, indicati con NA, in una porzione di dataset.

3.1.1 Tipologie

Quando si lavora con un dataset con dati mancanti è molto importante capire la tipologia di dati mancanti e quindi il motivo per cui mancano. Esistono tre tipologie di dati mancanti MCAR (Missing Completely At Random), MAR (Missing At Random) e MNAR (Missing Not At Random).

- MCAR (Missing Completely At Random): quando gli eventi che portano a un dato mancante sono indipendenti sia da variabili osservabili (presenti nel database) che da parametri di interesse non osservabili (esistenti ma non presenti nel database). La mancanza non è quindi correlata a nessuna variabile studiata.

- MAR (Missing At Random): il motivo per cui mancano i dati deve poter essere spiegato dai dati di cui si ha una visione completa. Per esempio in un sondaggio sul reddito, chi lavora inserirà lo stipendio chi non lavora non lo inserirà perché non ce l'ha. Quindi reddito è mancante ma non dipende dal reddito, dipende dalla variabile lavoro (di cui per ipotesi si ha visione completa).

- MNAR (Missing Not At Random): quando i valori mancanti dipendono da altri valori mancanti e quindi non possono essere usati per approssimare i valori mancanti. La probabilità che una variabile y manchi dipende da altre variabili mancanti. Per esempio, estendendo quello sul reddito, assumendo per ipotesi che le persone con un reddito più alto tendono a non inserire il proprio reddito, la mancanza di reddito dipende dal valore del reddito che è mancante.

3.1.2 In quale tipologia ricadiamo?

Il punto focale di questa analisi è capire il perché nel dataset sono presenti dati mancanti. Esistono due metodi principali per determinare in quale categoria di dati mancanti ci si trova. Il primo, e quello che andremo ad usare, viene definito "buon senso". Il secondo si basa su studi statistici eseguiti sul dataset, ma non può essere applicata in questo caso perché non disponiamo di un dataset con i dati reali di dove dovremmo andare ad applicare il sistema. Per capire in quale tipologia ricadono i dati mancanti si analizzeranno le altre variabili in osservazione e le eventuali relazioni con il dato mancante. I dati presenti nel dataset sono raccolti dalle e-bike presenti nella città, sia quando sono parcheggiate negli appositi porta bici che quando sono utilizzate dalle persone; da questo si deduce che i dati saranno raccolti nei luoghi in cui passano le bici. Assumendo che nei luoghi dove depositare le bici dopo l'utilizzo

ci sia sempre un sistema che raccoglie i dati, il momento in cui potremmo avere dati mancanti è quando per un'ora in un area, dove non sono presenti parcheggi per le bici, non passa nessuna persona con una e-bike. Analizzando ci si accorge che è possibile determinare una relazione tra il luogo e l'ora e i rispettivi dati mancanti, infatti potremmo trovare dei momenti ricorrenti in cui in alcuni luoghi non passano bici. Da questa considerazione possiamo dedurre che l'evento di mancanza di un dato potrebbe dipendere dal luogo e dall'ora. In quanto abbiamo trovato una dipendenza con variabili osservabili, possiamo escludere il caso MCAR (Missing Coompletely At Random). Continuando con l'assunzione fatta in precedenza, possiamo pensare che se piove o nevica le persone non useranno le bici; con un incidenza minore anche il vento potrebbe influire, in quanto con un vento eccessivamente forte le persone potrebbero non voler usare la bici. Anche la temperatura può essere un valore importante in quanto in casi di eccessivo caldo o freddo è possibile che le bici non vengano utilizzate. Un ultimo dato, ricavabile da quelli in osservazione, è che l'utilizzo o meno delle bici potrebbe dipendere dal giorno della settimana. Avendo fatto tutte queste considerazioni e analizzato tutte le possibili variabili, ci accorgiamo che l'evento di mancanza di un dato dipende completamente da variabili di cui si ha un'osservazione completa. Nel dataset l'unico, possibile, dato mancante è relativo alla variabile che misura il particolato di materia 2.5 e sapendo che non ci sono variabili non in osservazione che influenzano la mancanza di un dato, possiamo escludere il caso MNAR(Missing Not At Random). Andando per esclusione si deduce i dati mancanti ricadono nella categoria MAR (Missing At Random) infatti il motivo per cui possono esistere dati mancanti all'interno del dataset in osservazione è spiegato completamente dalle variabili di cui si ha visione completa.

3.2 Imputazione

In statistica l'imputazione [2] è il processo con cui si vanno a rimpiazzare i dati mancanti con dei dati sostituiti. Quando si sostituisce un punto dati si parla di imputazione di un'unità, mentre quando si sostituisce un componente di un punto dati si parla di imputazione di oggetto. I tre principali problemi causati dai valori mancanti sono i seguenti:

- possono introdurre un sostanzioso aumento di pregiudizio;
- rendono la manipolazione e l'analisi dei dati più complicata;
- riducono l'efficienza.

Considerando che i dati mancanti possono causare problemi nell'analisi, l'imputazione è vista come un modo per evitare di non considerarli che spesso causa altri problemi. Per ovviare a questo esistono due principali metodi: il primo, che prende il nome di listwise, consiste nell'ignorarli e quindi eliminare i record contenenti dati mancanti; il secondo è l'imputazione, che è un modo per rimpiazzare i valori mancanti utilizzando le altre informazioni disponibili. Una volta che tutti i valori mancanti sono stati rimpiazzati è possibile utilizzare i dati per fare analisi.

3.2.1 Imputazione singola

Nell'imputazione singola i dati sono rimpiazzati da un valore. Non si può avere la certezza che questo sia esattamente quello mancante, ma è probabile che ci sia un errore anche se minimo.

3.2.1.1 Metodi di corrispondenza

Anche definiti Hot-Deck, questi metodi abbinano i valori non corrispondenti con i valori simili corrispondenti e i valori mancanti sono imputati con il valore del corrispondente simile. Esistono due approcci, il primo detto anche "Approccio del vicino più prossimo" imputa il valore mancante con il punteggio del caso con la statistica della distanza al quadrato più piccola al caso con il valore mancante. Il secondo chiamato "Metodo del modello di corrispondenza" invece stratifica il campione in gruppi omogenei separati; il valore imputato per il caso mancante viene estratto dai casi nello stesso gruppo. In generale questi metodi rimpiazzano il valore mancante con un valore che preserva la distribuzione delle variabili.

3.2.1.2 Imputazione media

In questo caso i valori mancanti di una certa variabile sono rimpiazzati con la media dei casi disponibili; così facendo si mantiene la dimensione del campione, ma si riduce la variabilità dei dati e di conseguenza la deviazione standard e la varianza tendono ad essere sottostimate.

3.2.1.3 Imputazione di regressione

I valori imputati sono predetti da un'equazione di regressione. Con questo metodo le informazioni con tutte le osservazioni complete sono usate per predire i valori delle osservazioni mancanti. La regressione assume che i valori imputati cadano esattamente su una linea di regressione con pendenza diversa da 0. Questo implica una correlazione di 1 tra le variabili predittrici

e la variabile di risultato mancante. L'imputazione di regressione sovrastima la correlazione tra le variabili; tuttavia, la varianza e la covarianza sono sottostimate.

3.2.2 Imputazione multipla

Nell'imputazione multipla, il processo di imputazione viene ripetuto più volte dando come risultato più set di dati imputati. Si suddivide in tre fasi: imputazione, analisi, raggruppamento. La fase di imputazione inizia con il creare varie copie del dataset per poter imputare i dati più volte e avere quindi dataset con diversi dati imputati. I dati imputati sono stimati con la media e la covarianza dei dati osservati. Le equazioni di regressione vengono utilizzate per prevedere i valori incompleti dai valori completi e un termine residuo normalmente distribuito viene aggiunto a ciascun valore per ripristinare la variabilità. Questo processo viene iterato più volte, aggiornando i parametri di regressione dopo ogni iterazione, per ottenere valori imputati differenti dopo ogni iterazione. Dopo ogni serie di iterazioni, un dataset è salvato fino a quando non viene raggiunto il numero di dataset richiesti. Oltre a questo è molto importante scegliere bene e correttamente le variabili da includere nel processo di imputazione. Possono essere aggiunte variabili ausiliarie per migliorare la stima dei valori imputati. Una volta scelte le variabili, è importante avere un modello di imputazione che si adatti alle ipotesi di distribuzione dei dati. Una volta terminata la prima fase si passa alla seconda dove si effettua l'analisi statistica che viene effettuata su tutti i dataset come se i dati fossero stati completi. Nell'ultima fase i dati sono raggruppati in un unico dataset. Vengono quindi raggruppate le stime e le stime dei parametri, queste ultime con una media tra tutte le stime di tutti i dataset imputati. Gli errori sono raggruppati combinando la varianza di imputazione e la varianza di imputazione interna. In questo modo si ottengono i parametri migliori ottenibili con i dataset a disposizione.

Capitolo 4

Modelli applicabili al problema

4.1 Introduzione

Secondo le analisi e gli studi precedentemente esplicati, ci si riduce ad avere due principali problemi da risolvere. Il primo è appunto andare a attribuire un valore ragionevole ai dati mancanti all'interno del dataset; il secondo invece ha l'obiettivo di fornire una previsione oraria, il più accurata possibile, delle 24 ore successive. Per fare questo esistono molteplici vie, per imputare i valori mancanti nel capitolo precedente sono state elencate numerose tecniche, ognuna più o meno valida in base alla situazione. In questo caso la precisione è molto importante quindi, si potrebbe pensare di utilizzare un imputazione di regressione singola o multipla. Entrambe sono valide, ma con la regressione singola si evita di dover creare tanti dataset diversi e si può eseguire l'operazione usandone uno solo. Per poter fare regressione esiste uno strumento molto potente che prende il nome di machine learning. Questo strumento è funzionale per entrambi i problemi da risolvere.

4.2 Machine learning

Rientra nella disciplina dell'intelligenza artificiale. Si identifica in un sistema costituito di due fasi, la fase di allenamento in cui apprende a partire da esempi. La fase di funzionamento o generalizzazione, successiva alla prima, permette di generalizzare e gestire nuovi dati nello stesso dominio applicativo. Riassumendo, il machine learning è un sistema che impara dagli esempi a migliorare la sua conoscenza sui dati e a gestirli in modo migliore e più efficiente. Tra le sue applicazioni ne osserviamo due che potrebbero risolvere il nostro problema: la regressione e le reti neurali applicate alla regressione.

4.2.1 Tecniche di apprendimento

4.2.1.1 Apprendimento supervisionato

Questo tipo di allenamento presenta i dati in un set classificato ed etichettato, quindi ad ogni classe di valori è associata una label comune nota. La rete viene allenata fornendo in input dei dati associati ad una label in modo che il sistema si alleni a capire che un certo tipo di dato è associato ad una certa label. Una volta che la rete è stata sottoposta ad un numero sufficiente di casi si può testare e osservare quanto sia precisa. Per testarla viene presentato un caso dove la label è mancante e la rete dovrà trovarla. Questo sembra essere applicabile per il problema dei valori mancanti, infatti, si hanno casi completi che si utilizzano per l'allenamento e casi incompleti per la verifica che la rete dovrà completare.

4.2.1.2 Apprendimento non supervisionato

In questa forma di allenamento il set dei dati non è etichettato. Si lascia quindi al sistema, la rete neurale, il compito di cercare relazioni tra i vari dati ed eventualmente suddividerli in categorie. Una volta che il sistema trova le relazione e suddivide le categorie sta a l'uomo, al programmatore, capire come usarle o dargli un'etichetta appropriata. Questo approccio viene usato quando si ha una forte convinzione (quasi una certezza) che ci siano relazioni interne tra i dati.

4.2.1.3 Apprendimento semi supervisionato

Qui il set di dati è parzialmente etichettato e viene prevalentemente utilizzato per aiutare a ottimizzare la regola di classificazione.

4.2.1.4 Apprendimento rinforzato

In questo caso l'obiettivo è apprendere un comportamento ottimale a partire dalle esperienze passate. Un agente esegue azioni che modificano l'ambiente, provocando passaggi da uno stato a l'altro. Quando l'agente ottiene risultati positivi riceve una ricompensa (reward) che però può essere temporalmente ritardata rispetto all'azione, o alla sequenza di azioni, che l'hanno determinata. L'obiettivo è apprendere l'azione ottimale in ciascun stato, in modo da massimizzare la somma dei reward ottenuti nel lungo periodo.

4.2.1.5 Quale apprendimento va utilizzato?

Il nostro set di dati si compone di dati sempre presenti come luogo, orario e data e di dati, relativi alle rilevazioni sull'inquinamento, che possono essere mancanti. Dovendo risolvere il problema dei valori mancanti il sistema più utilizzato è l'addestramento supervisionato. Con questo approccio possiamo allenare il sistema con i casi completi e una volta allenato un numero esauriente di volte possiamo testarlo con alcuni casi (di cui noi abbiamo visione completa) nascondendogli alcuni dati, che quindi diventano mancanti, per vedere con che precisione risponde. Gli altri approcci sono più o meno orientati alla ricerca o al miglioramento delle relazioni nel set di dati. Nel prossimo paragrafo si parlerà di regressione, la quale, utilizza un apprendimento supervisionato.

4.2.2 Regressione

Viene utilizzata per la predizione di valori continui. Consiste nell'apprendere una funzione approssimante delle coppie $\langle input, output \rangle$ date. La funzione è della forma $f(x) = y$ dove y è un valore numerico e continuo in R. La variabile indipendente x si assume sia esatta mentre la variabile dipendente y si assume che sia affetta da errore. Esistono più tipi di regressione che può essere lineare o non lineare. La regressione è il metodo che utilizzeremo sia per attribuire un valore quando ne incontriamo uno mancate, sia per fare previsioni sul futuro. I modelli che verranno descritti da qui in avanti sono tutti utilizzabili per fare regressione

4.2.3 Support Vector Machine

Le macchine a vettori di supporto (SVM), o macchine kernel, sono delle metodologie di apprendimento supervisionato per la regressione e la classificazione di pattern. Questi vettori sono nati per la classificazione inizialmente in due categorie ma poi anche di più; per farlo si attribuiscono gli esempi alle categorie in modo che per ogni categoria ci siano un numero notevole di esempi. Per l'addestramento crea un modello che assegna i nuovi esempi ad una tra le categorie. Esiste una versione per effettuare regressione chiamata Support Vector Regression (SVR) dove l'idea principale è sempre quella di minimizzare l'errore. Nella regressione, dovendo prevedere un valore continuo, si decide un margine di tolleranza dell'errore. La particolarità di questi vettori è che nel caso della regressione è possibile utilizzare una funzione non lineare per minimizzare l'errore ottenendo quindi una funzione curva che me-

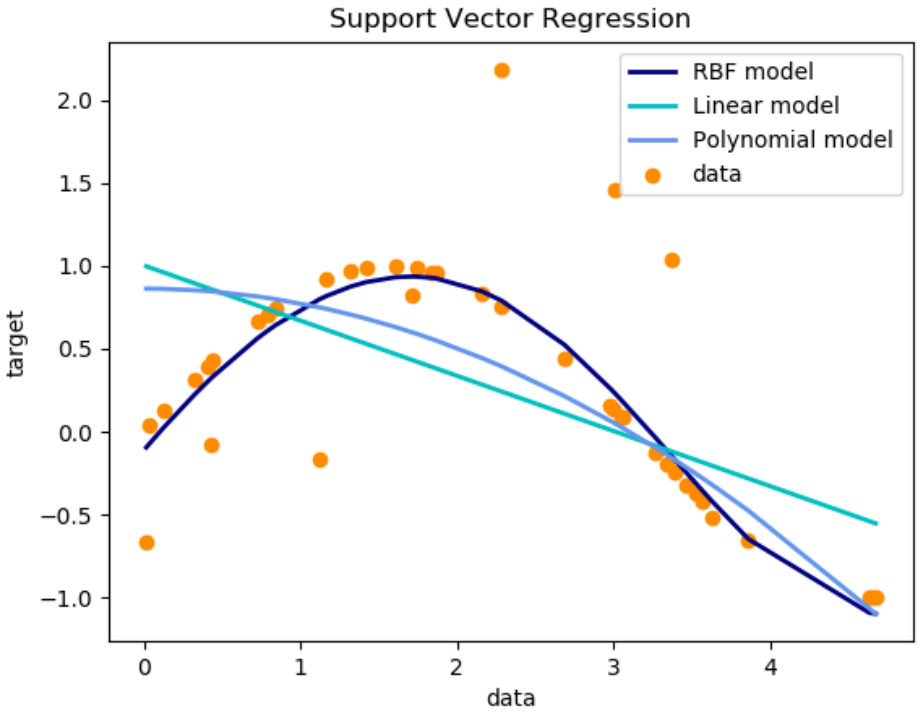


Figura 4.1: Esempio di regressione con i support vector machine utilizzando diverse funzioni: lineare, polinomiale e Radial Basis Function (RBF).

glio approssima i dati. A differenza dei modelli lineari dove si approssimano i valori con una retta che risulta molto meno precisa.

4.2.4 Random Forest

Le random forest (foreste casuali) sono uno strumento molto potente e possono essere usate sia per classificazione che per regressione. Per quanto riguarda la regressione si costruiscono una moltitudine di alberi decisionali e come output si ottiene la previsione di ogni albero. Come negli altri casi di regressione anche le random forest si allenano con un apprendimento supervisionato fornendo in output un valore basato sui valori di input e allenandosi con una moltitudine di esempi. Quando si parla di regressione prendono il nome di regression forest (foreste di regressione).

4.2.5 Reti neurali

Le reti neurali negli ultimi anni hanno acquisito un notevole successo. Come si evince dal nome, il loro funzionamento è simile al cervello umano: sono infatti composte da neuroni e archi che collegano i neuroni tra loro. Nelle sinapsi del cervello umano un neurone invia un segnale su un arco che si attiva avvisando un secondo neurone che deciderà se propagare il segnale su un altro arco oppure no, in altre parole attiva una sua funzione che in base al segnale compirà se necessario un'azione. Le reti neurali sono solitamente composte da più livelli ognuno dei quali è a sua volta composto da uno o più neuroni. Nel modello più semplice di una rete neurale sono presenti un livello di input e un livello di output e tutti i neuroni del primo livello sono connessi, tramite degli archi, a tutti quelli del secondo livello. Quando i valori di input sono dati alla rete passano attraverso gli archi pesati e si mutano di conseguenza; quando i neuroni del livello di output ricevono i valori applicano una funzione che genera un nuovo valore. Il processo appena descritto, come preannunciato, è molto simile al funzionamento del cervello umano. Il modello della rete può essere complicato aggiungendo dei livelli intermedi chiamati livelli nascosti, che a loro volta avranno un certo numero di neuroni e saranno collegati ai neuroni del livello precedente e a quelli del livello successivo tramite degli archi. Il funzionamento di una rete neurale, con un modello semplice, può essere facilmente descritto in pochi passi. Inizialmente vengono forniti alla rete i valori di input ponendo nel primo livello un numero di neuroni pari al numero di valori in input. A questo punto si moltiplicano per i pesi degli archi. Come detto prima da ogni neurone partono archi per ogni neurone del livello successivo. Una volta che i neuroni del livello nascosto hanno ottenuto un valore, lo trasformano con una funzione chiamata "funzione di attivazione". Esistono più funzioni di attivazione ad esempio Sigmoid e Relù. Adesso si applica lo stesso procedimento dal livello nascosto al livello di output, nel caso della regressione il livello di output avrà solo un neurone e non applicherà nessuna funzione di attivazione. Il valore ottenuto sarà la previsione che la rete neurale ha prodotto per i valori in input.

4.2.5.1 Allenamento di una rete

Nella fase di inizializzazione i pesi degli archi di una rete sono impostati a valori predefiniti o casuali; durante l'allenamento questi valori vengono modificati per trovare la configurazione migliore. Ma come avviene questo processo? L'allenamento di una rete si compone di due fasi principali: forward propagation e back propagation. Nella prima fase vengono forniti degli input e calcolato l'output, ripetendolo per un certo numero di volte (questo

numero viene definito batch size e verrà spiegato nel capitolo del processo di sviluppo). Effettuato questo passaggio avviene la back propagation che confronta il valore ottenuto con quello corretto e aggiorna i pesi degli archi a ritroso in modo da minimizzare l'errore. Questo procedimento composto dalle due fasi viene ripetuto molteplici volte durante l'allenamento per ottenere un buon risultato finale. In particolare viene ripetuto per un numero di volte pari agli esempi forniti in input e per un certo numero di epoche. Il modo in cui definire questi valori verrà affrontato nel processo di sviluppo.

4.2.6 Reti neurali per la regressione

Le reti neurali possono essere utilizzate per la regressione, infatti ponendo un unico neurone nel livello di output è possibile allenare una rete neurale a predire valori continui. Nell'ambito della regressione si utilizza l'apprendimento supervisionato; per farlo è necessario un dataset etichettato dove l'etichetta è rappresentata dal valore di output e cioè dalla variabile di cui si vogliono fare previsioni. Per misurare l'errore della rete si utilizza lo scarto quadratico medio. L'obiettivo sarà quindi quello di minimizzarlo in modo da ridurre al minimo l'errore.

4.2.7 Deep neural network

Rappresenta un tipo specifico di machine learning ma anche una sua "evoluzione". Può essere definito evoluzione infatti propone reti neurali multilivello costituite da un livello di input, uno di output e da due o più livelli nascosti organizzati gerarchicamente. Si parla quindi di "deep neural network" (DDN).

4.3 Discussione generale sui modelli

Il machine learning è uno strumento molto potente ma come ogni cosa va utilizzata nella giusta maniera e misura altrimenti non si ottiene progresso ma l'esatto opposto. Questo tipo di intelligenza artificiale può essere utilizzata per tanti scopi, noi ci concentreremo sulla regressione. Per poter fare regressione come già visto esistono molteplici modelli che a seconda del problema si adattano più o meno bene. La regressione serve come già detto per prevedere un valore continuo in R, quindi nel caso in cui si disponga una correlazione molto buona tra la variabile da predire e un'altra variabile nel dataset la regressione lineare, potrebbe essere un buon modello. Se esistono più variabili abbastanza correlate si potrebbe utilizzare una regressione

lineare. Modelli più potenti sono i super vector machine e le random forest. Oltre questi sistemi esistono le reti neurali che permettono di fare previsioni molto accurate costruendosi un modello composto da neuroni e archi pesati che vanno a creare una fitta rete in grado di predire con buona precisione. In seguito verrà discusso e testato quale tra i modelli potrebbe essere il più adatto. La scelta sarà effettuata basandosi sui risultati ottenuti nei test, quindi resta la possibilità che ne esista uno migliore.

Capitolo 5

Progettazione del sistema

5.1 Progettazione dei dati

5.1.1 Raccolta dei dati

La fase di raccolta dei dati e quindi di campionamento verrà effettuata con sensori appositi inseriti all'interno delle biciclette elettriche messe a disposizione dal comune di Bologna con il bike sharing. Come spiegato nel capitolo di analisi la città verrà suddivisa in una griglia virtuale in modo da poter avere una visione più completa e valutare la concentrazione delle polveri nelle varie aree della città. I dati verranno quindi raccolti dalle bici ma come è facile intuire è possibile che alcune volte nessuna bici passi in un determinato punto in una certa ora. Questo sistema di raccolta dati porta quindi inevitabilmente ad avere dati mancanti.

5.1.2 Preparazione dei dati

Questa fase è molto importante e se fatta male porta a risultati molto negativi. I dati sono stati strutturati in modo da avere per ogni dato un valore orario in modo che ad ogni ora si abbia una visone completa di tutto. Questa preparazione è figlia dell'analisi come descritta al capitolo due e conferma quanto già analizzato e deciso precedentemente aggiungendo però alcune informazioni sulle unità di misura e su cosa il dato realmente significa per eliminare ogni possibile ambiguità rimasta. Per quanto riguarda il dato relativo al PM 2.5 si è deciso di salvare il valore in $\mu g/m^3$. Per le precipitazioni sarà presente un numero intero che indica se e da quante ore sta nevicando o piovendo, quindi se è 0 non ci saranno quel tipo di precipitazioni, mentre se è un numero n intero positivo diverso da 0 vorrà dire che ci sono precipitazioni da n ore. La pressione atmosferica è espressa in bar mentre la temperatura e

Colonna	Descrizione
A	Anno
M	Mese
G	Giorno
O	Ora
R	Punto di rugiada
T	Temperatura
P	Pressione atmosferica
D	Direzione del vento
V	Velocità del vento
N	Ore di neve
P	Ore di pioggia

Tabella 5.1: Legenda per le colonne abbreviate.

il punto di rugiada sono espressi in gradi centigradi. Per il vento si è deciso di combinare l'angolazione e la velocità calcolando i vettori direzione. Un principio analogo è stato utilizzato per i giorni dell'anno in modo da descrivere l'anno come periodo circolare. Anche per le ore si è adottato la stessa rappresentazione. Come già detto nel capitolo due, non è stato possibile utilizzare dati reali della città di Bologna, quindi sono stati utilizzati dei dati relativi alla città di Pechino in Cina.

A	M	G	O	PM 2.5	R	T	P	D	V	N	P
2010	1	2	0	129	-16	-4	1020	SE	1.79	0	0
2010	1	2	1	148	-15	-4	1020	SE	2.68	0	0
2010	1	2	2	159	-11	-5	1021	SE	3.57	0	0
2010	1	2	3	181	-7	-5	1022	SE	5.36	1	0
2010	1	2	4	138	-7	-5	1022	SE	6.25	2	0
2010	1	2	5	109	-7	-6	1022	SE	7.14	3	0
2010	1	2	6	105	-7	-6	1023	SE	8.93	4	0
2010	1	2	7	124	-7	-5	1024	SE	10.72	0	0
2010	1	2	8	120	-8	-6	1024	SE	12.51	0	0
2010	1	2	9	132	-7	-5	1025	SE	14.3	0	0
2010	1	2	10	140	-7	-5	1026	SE	17.43	1	0
2010	1	2	11	152	-8	-5	1026	SE	20.56	0	0
2010	1	2	12	148	-8	-5	1026	SE	23.69	0	0

Tabella 5.2: Dati relativi al 2 gennaio 2010 nelle prime 12 ore nella città di Pechino.

5.1.3 Ruolo dei dati

I dati possono assumere diversi ruoli all'interno di un'analisi statistica, ed è molto importante capire il ruolo che ogni dato dovrà avere. Bisogna quindi fare uno studio per capire quali dati dipendono da altri dati. Quindi quali sono le variabili dipendenti che dipendono dal valore delle variabili indipendenti. I modelli e gli esperimenti testano o determinano gli effetti che le variabili indipendenti hanno sulle variabili dipendenti. In un esperimento, una variabile, manipolata dallo sperimentatore, è chiamata variabile indipendente X. La variabile dipendente Y è l'evento che si prevede cambierà quando le variabili indipendenti sono manipolate.

5.1.3.1 Variabili Indipendenti

Le variabili indipendenti, anche conosciute in un contesto statistico come regressori, rappresentano input o cause, cioè potenziali ragioni di variazione o, in una sperimentazione, la variabile controllata dallo sperimentatore. A volte le variabili indipendenti potrebbero essere incluse per altre ragioni, come il loro potenziale effetto confusionario, senza la volontà o il desiderio di testare direttamente il loro effetto.

5.1.3.2 Variabili Dipendenti

Le variabili dipendenti rappresentano l'output o il risultato di cui si sta studiando la variazione. Le variabili dipendenti sono quelle di cui si osserva il cambiamento sulla base di come variano le variabili indipendenti. In statistica sono le variabili di cui si vuole prevedere il valore in base al cambiamento delle variabili indipendenti.

5.2 Librerie di sviluppo

L'ambiente di sviluppo scelto è Python per la sua completezza in quanto a librerie e supporto per l'analisi dei dati e l'utilizzo di strumenti come il machine learning. Di seguito sono elencate alcune librerie utilizzate per lo sviluppo.

5.2.1 Scikit-Learn

Questa libreria [4], chiamata anche sk-learn, fornisce strumenti di sviluppo per data mining e analisi dei dati; mette a disposizione quasi tutti gli strumenti per il machine learning come la classificazione, regressione, clustering

e altre. La libreria è open source e può essere utilizzata liberamente. Per quanto riguarda la regressione mette a disposizioni delle implementazione di semplice utilizzo per la regressione lineare e la regressione polinomiale. Esistono anche classi per i Super Vector Machine (SVM) e, una loro variante per la regressione, i Super Vector Regressor (SVR). A questi sono applicabili più modelli di regressione tra cui lineari, polinomiali e gli RBF model (Radial Basis Function). Sono presenti anche implementazioni per Nearest Neighbors applicati alla regressione e implementazioni per gli alberi decisionali. Uno strumento importante che mette a disposizione sono le random forest che creano una moltitudine di alberi decisionali. Infine ci sono strumenti per le reti neurali e una classe apposita per la regressione con reti neurali. Questa libreria è costruita e basata su altre 3 librerie di Python: NumPy, SciPy e matplotlib.

5.2.2 Tensorflow

TensorFlow [5] è una libreria software open-source, sviluppata da Google, per la programmazione di flussi di dati in una vasta gamma di attività. È una libreria matematica simbolica ed è anche utilizzata per applicazioni di machine learning come le reti neurali. Viene utilizzato sia per la ricerca che per la produzione a Google. Il progetto è iniziato nel 2015 e l'ultima versione stabile è stata rilasciata il 27 aprile 2018. Fornisce vari livelli di astrazione per permettere un utilizzo semplice con classi già pronte ed un utilizzo più complicato e personalizzate con strumenti di più basso livello. La libreria è implementata in Python. Questa libreria mette a disposizione un'implementazione per allenare le reti neurali sfruttando l'ausilio delle GPU.

5.2.3 Keras

Keras [3] è una libreria per reti neurali scritta in Python. Può operare sopra TensorFlow, Microsoft Cognitive Toolkit, Theano o MXNet. Questa libreria infatti non fornisce delle interfaccia di alto livello per utilizzare in modo semplice le librerie suddette. Si focalizza appunto nell'essere user-friendly, modulare ed estendibile. Dal 2017 Google ha deciso di supportare Keras nella libreria principale di TensorFlow. Keras contiene numerose implementazioni dei blocchi comunemente usati nelle reti neurali come livelli, oggetti, funzioni di attivazione, ottimizzatori. Come TensorFlow permette di allenare le reti neurali sfruttando le GPU.

5.3 Progettazione del Software

La progettazione del software grazie all'ausilio delle librerie suddette si è rivelata abbastanza semplice. Il software sarà in grado di svolgere i due obiettivi identificati in fase di analisi e sarà composto da tre moduli principali. Il primo modulo si occuperà di allenare e salvare il risultato di più modelli in grado di svolgere i compiti preposti. Il secondo modulo avrà il compito di attribuire un valore coerente nel caso in cui un dato sia mancante, verrà attribuito un valore solo al PM 2.5 in quanto l'obiettivo è appunto quello di avere un dataset completo per questo valore in modo da poterlo studiare. Per fare questo si utilizzerà uno dei modelli prodotti dal primo modulo e adatto a questo compito. Il terzo modulo, infine, andrà ad effettuare le previsioni del PM 2.5 nelle 24 fasce orarie successive. Si utilizzeranno anche qui i modelli allenati dal primo modulo; in generale per ogni fascia diversa si utilizzerà un modulo allenato diversamente. In realtà la tipologia dei moduli e la struttura sarà la stessa ma cambieranno i dati utilizzati per l'allenamento e di conseguenza i pesi e i parametri che saranno salvati diversamente per ogni modulo. Il software è stato progettato con un modello basato sui test, infatti si aveva l'obiettivo di minimizzare lo scarto quadratico medio di errore tra il valore previsto e quello reale. Sono stati quindi prodotte più versioni del software che hanno raggiunto risultati sempre migliori fino ad ottenerne uno accettabile. Questo modello è basato sui test ma è anche iterativo infatti si producono versioni successive del software andando a migliorare sempre di più le funzionalità del software.

5.3.1 Modello iterativo

Nell'ingegneria del software il modello iterativo, a parte per la prima versione, presenta sempre una versione del software in esercizio e una in fase di sviluppo. La versione in esercizio fornisce tutte o quasi le funzionalità più o meno ottimizzate. La versione in sviluppo andrà a produrre un software più ottimizzato e completo. Nel caso specifico si è deciso di usare lo scarto quadratico medio tra i valori reali e i valori predetti come parametro per valutare la precisione del software. Ogni versione successiva del software avrà quindi l'obiettivo di ridurre lo scarto quadratico medio rispetto alla versione precedente. In questo caso ci si lega molto al risultato del test.

Capitolo 6

Processo di sviluppo

6.1 Implementazione di più modelli

In questa fase si sono implementati più modelli di regressione al fine di valutare e prendere decisioni in base ai risultati dei test e osservando come valore più importante lo scarto quadratico medio di errore sulle previsioni. Tutte le implementazioni forniscono un risultato; la differenza sta nella precisione di quest'ultimo. Si sono implementati questi sistemi: support vector regression (SVR), random forest, rete neurale e rete neurale a serie temporali. Grazie alle librerie l'implementazione si è ridotta al capire come utilizzare le classi messe a disposizione.

6.2 Test dei modelli

Il primo test effettuato aveva l'obiettivo di valutare quale tra i modelli fosse il più adatto e per farlo si sono eseguiti un numero significativo di test per ogni modello. In base ai test effettuati, come mostrato nella tabella, i risultati migliori sono stati ottenuti con le reti neurali a serie temporali. Per quanto riguarda le reti neurali sono stati eseguiti numerosi test modificando i parametri di configurazione, come il numero di livelli e di neuroni per ogni

Modello	Scarto quadratico medio di errore
Support Vector Regression	0.108949
Random Forest	0.120707
Rete neurale	0.108949
Rete neurale a serie temporali	0.001021

Tabella 6.1: Mostra i risultati ottenuti con i diversi modelli.

livello, ma questo non ha portato comunque a buoni risultati. La ricerca di un nuovo modello ha quindi portato alle reti neurali a serie temporali che secondo i test effettuati si sono dimostrate molto più precise.

6.3 Reti neurali ricorrenti

Normalmente le reti neurali assumono che tutti i valori di input siano indipendenti dagli altri; le reti neurali ricorrenti (RNN), invece, sfruttano la sequenza naturale dei loro input, come ad esempio una serie temporale di eventi. Sono utilizzate quando un evento è dipendente dagli eventi avvenuti in precedenza. Una RNN può essere vista come un grafo di celle, dove ogni cella compie la stessa operazione per ogni elemento nella sequenza. Il risultato quindi non dipende più da un singolo input ma da una sequenza di input. Per incorporare questa dipendenza si utilizzano stati nascosti o si memorizzano quelli che mantengono l'essenza di ciò che è stato visto finora; in questo modo si costruiscono una sorta di serie temporali di eventi che portano al risultato di output. Il valore di uno stato nascosto è calcolato in funzione del valore allo stato precedente e del valore in input. Come nelle altre reti neurali i parametri sono contenuti in una matrice di pesi che in questo caso è definita su input, output, e stati nascosti. Un altro aspetto è il processo di back propagation che deve tener conto che in questo caso i parametri sono condivisi tra tutti i passi temporali, quindi il risultato di ogni output non dipende solo dal passo corrente ma anche da quello precedente. Questo processo è definito back propagation through time (BPTT).

6.4 Reti neurali a serie temporali

Sono un tipo di reti neurali ricorrenti capaci di apprendere dipendenze temporali a breve e lungo termine, infatti sono chiamate anche long short-term mermory neural netowrk (LSTM) che si traduce in reti neurali con memoria a breve e lungo termine. Con questo tipo di rete neurale possiamo sfruttare quindi più rilevazioni per predirne una successiva. La rete neurale è stata implementata con un livello di input, uno di tipo LSTM e uno di output.

6.5 Ottimizzare il modello

Per i parametri di configurazione si è deciso di effettuare numerosi test e decidere in base al loro risultato. La decisione è stata presa principalmente in base allo scarto quadratico medio di errore, ma sono stati considerati anche

molti altri parametri come il tempo totale di allenamento. Per eseguire i test è stata utilizzata una macchina virtuale con un processore intel core i7 4790k, 8 GB di ram e il sistema operativo Lubuntu 16.04.3.

6.5.1 Numero dei neuroni

Per quanto riguarda i neuroni sono stati effettuati dei test aggiungendo neuroni al livello LSTM per valutare la variazione dello scarto quadratico medio all'aumentare del numero di neuroni. Si è seguito un approccio generale in cui non si va a complicare un modello (aggiungendo livelli e neuroni) senza che ci sia un effettivo miglioramento delle prestazioni.

Numero di neuroni	Scarto quadratico medio di errore
1	0.001073
12	0.001042
24	0.001071
36	0.001093
48	0.001210
96	0.001027

Tabella 6.2: Mostra i risultati ottenuti all'incrementare del numero di neuroni.

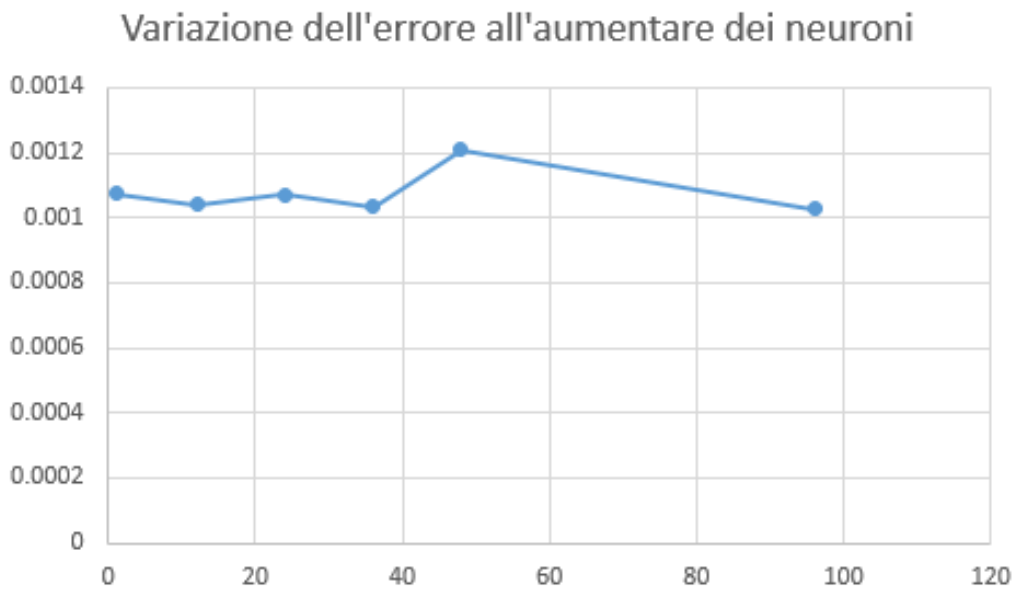


Figura 6.1: Variazione dello scarto quadratico medio d'errore all'aumentare del numero di neuroni.

I dati mostrati nella tabella e riportati nel grafico mostrano come all'aumentare del numero di neuroni lo scarto quadratico medio non diminuisca ma resti più o meno costante subendo variazioni minime. Rispettando quanto detto precedentemente e visto che non ci sono miglioramenti, si è deciso di non complicare il modello e mantenere un numero basso di neuroni. Quindi si è fatta una nuova analisi con un numero di neuroni tra 1 e 12.

Numero di neuroni	Scarto quadratico medio di errore
1	0.001072720
2	0.001109684
3	0.001040296
4	0.001020702
5	0.001054367
6	0.001039755
7	0.001037202
8	0.001084918
9	0.001106523
10	0.001106471
11	0.001050382
12	0.001041532

Tabella 6.3: Mostra i risultati ottenuti all'incrementare del numero di neuroni.

Dai risultati ottenuti possiamo notare che la differenza non sia molta ma che il miglior risultato corrisponda al modello con quattro neuroni. Quindi in base a questo test e a questi risultati si è deciso di utilizzare un livello LSTM con quattro neuroni.

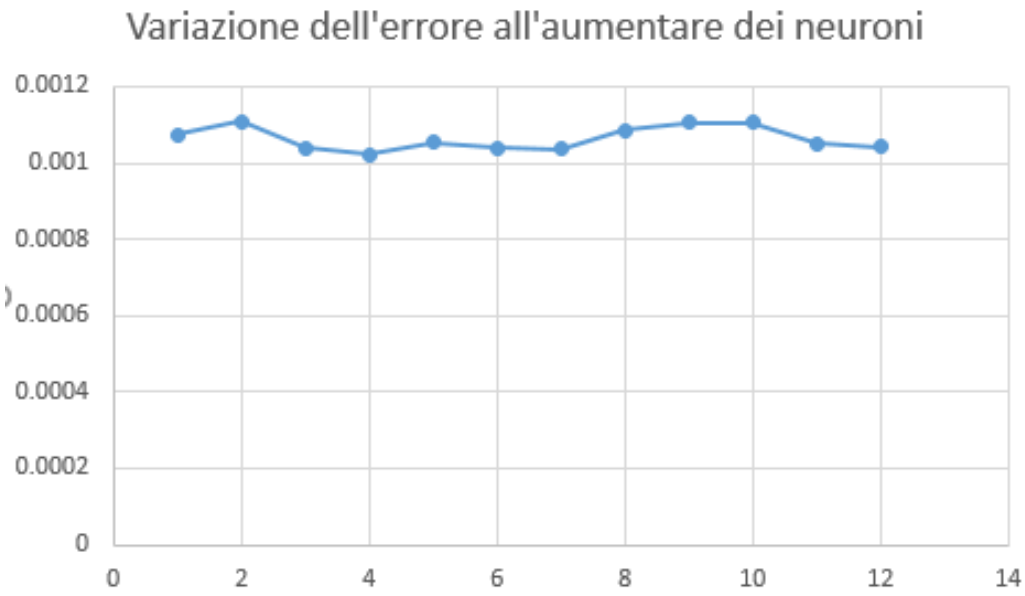


Figura 6.2: Variazione dello scarto quadratico medio d'errore all'aumentare del numero di neuroni.

6.5.2 Batch size

Un altro parametro molto importante è il batch size che indica, nella fase di allenamento, ogni quanti input si compie l'operazione di backpropagation. Questi esempi sono dati in input contemporaneamente, quindi all'aumentare di questo parametro più input di allenamento saranno sottoposti contemporaneamente, meno volte si fa backpropagation e quindi il tempo di allenamento si riduce. In altre parole indica ogni quanti esempi sottoposti alla rete si aggiornano i pesi degli archi in funzione dei risultati ottenuti. Per prendere una decisione si sono effettuati alcuni test. Analizzando i risultati ottenuti

Batch size	Scarto quadratico medio di errore
12	0.00105289
24	0.00104169
48	0.00108867
72	0.00106892
168	0.00116964

Tabella 6.4: Mostra i risultati ottenuti con diversi numeri di batch size.

si può notare come il migliore, in base ai test, sia in corrispondenza di 24. Questo potrebbe voler dire che la rete funziona in modo migliore, per il caso

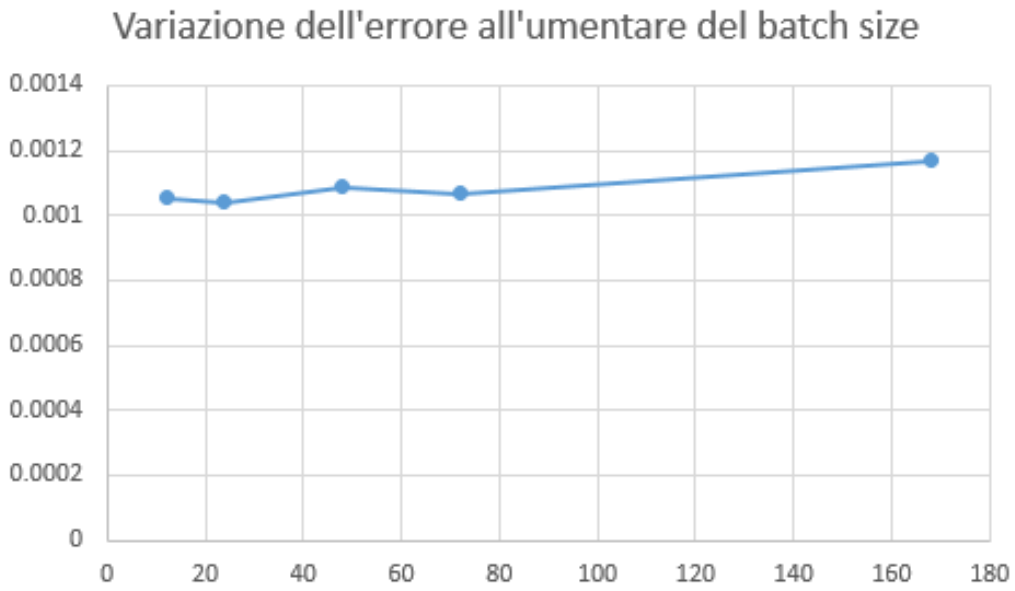


Figura 6.3: Variazione dello scarto quadratico medio d'errore all'aumento del batch size.

in questione, con un aggiornamento dei pesi ogni 24 campionamenti e quindi ogni 24 ore di osservazioni. A seguito del test si sceglie di utilizzare un batch size di 24.

6.5.3 Quante rilevazioni precedenti osservare?

Questo parametro sta ad indicare quanto la serie temporale andrà indietro nel tempo per fornire una previsione accurata. In altre parole bisogna capire di quante ore di rilevazione ha bisogno la rete per prevedere l'ora successiva in modo preciso. Secondo i test effettuati la rete ha un rendimento migliore e quindi un risultato più preciso fornendo in input le 12 rilevazioni precedenti e quindi i dati relativi alle 12 ore precedenti per la previsione dell'ora successiva.

Quantità di rilevazioni in input	Scarto quadratico medio di errore
1	0.00206530
2	0.00454622
3	0.00283056
4	0.00115235
6	0.00128039
12	0.00104171
18	0.00115179
24	0.00111532
48	0.00120345
168	0.00166787

Tabella 6.5: Mostra i risultati ottenuti al variare delle rilevazioni date in input.

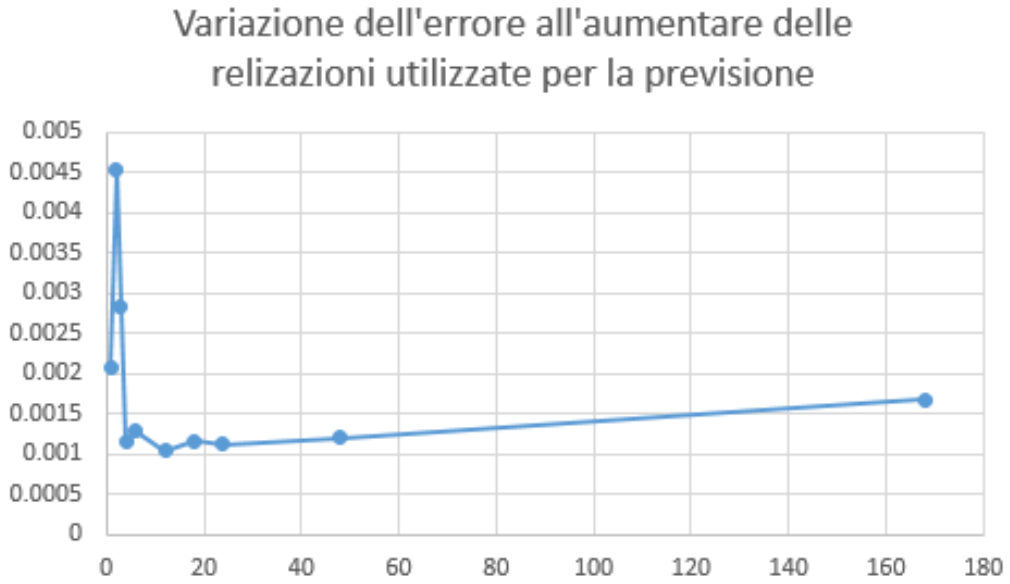


Figura 6.4: Variazione dello scarto quadratico medio d'errore all'aumentare del numero di rilevazioni utilizzate per effettuare la previsione.

6.5.4 Quanti esempi per l'allenamento?

Si potrebbe pensare che aumentando il numero esempi forniti ad una rete per allenarsi si aumenti di conseguenza anche la precisione e in questo caso si riduca lo scarto quadratico medio d'errore. I test effettuati mostrano come questa cosa non sia vera: infatti, facendo allenare la rete su troppi esempi, quest'ultima potrebbe incorrere in uno stato chiamato overfitting in cui la rete ha imparato troppo e sa rispondere quasi perfettamente a ciò che ha imparato ma in modo non corretto con valori con cui non si è allenata. Questo concetto testimonia ancora una volta come le reti neurali siano ispirate al cervello umano; infatti, l'overfitting si può verificare anche con un cervello umano quando studia ed impara qualcosa molto bene, a memoria, ed entra nella condizione di sapere solo quello. Il numero di esempi identifica il

Quantità di esempi in input	Scarto quadratico medio di errore
4175	0.001852966
8351	0.000825996
12527	0.000871604
16702	0.000725477
20878	0.000753251
25054	0.001177193
29229	0.001019238
33405	0.001147155

Tabella 6.6: Mostra i risultati ottenuti all'aumento del numero di esempi forniti nel training.

numero di rilevazioni orarie. Osservando i risultati l'errore più basso corrisponde a 16702 osservazioni, che corrispondono a poco meno di due anni, infatti $16702/(24 * 365) = 1.906621005$ anni che si arrotonda a due anni di osservazioni. Non sono stati utilizzati numeri di esempi corrispondenti a un tempo preciso come un anno e sei mesi, perché, per fare i test, si è utilizzata una percentuale che andava ad identificare una parte di dataset usato per l'allenamento e una parte di dataset utilizzata per i test.

Figura 6.5: Variazione dello scarto quadratico medio d'errore all'aumentare del numero di esempi utilizzati per l'allenamento.

6.5.5 Funzione di ottimizzazione

Gli ottimizzatori definiscono il modo in cui una rete neurale è allenata, ossia sono gli algoritmi con cui si allena una rete neurale. Stochastic gradient descent (SGD) è un metodo iterativo per ottimizzare diverse funzioni obiettivo e produce un'approssimazione stocastica dell'ottimizzazione della pendenza del gradiente. Adam è un'estensione di SGD. Si nota che Adam raggiunge

Ottimizzatore	Scarto quadratico medio di errore
Adam	0.000682403
Stochastic gradient descent	0.004079786

Tabella 6.7: Mostra i risultati ottenuti con Adam e SGD.

prestazioni, secondo i test effettuati, nettamente migliori di SGD. Quindi in accordo con il risultato del test, l'algoritmo di ottimizzazione scelto è Adam.

6.5.6 Numero di epoche di allenamento

L'allenamento di una rete neurale si sviluppa in più epoche, in ognuna delle quali la rete si allena su tutti gli esempi forniti. In altre parole nella sezione

Epoca	Errore training	Errore Validazione	Aumento/Decremento
1	0.015458556	0.005492814	0.000000000
2	0.002779242	0.002436485	-0.003056328
3	0.001476326	0.001876632	-0.000559854
4	0.001190052	0.001599097	-0.000277535
5	0.001026282	0.001453485	-0.000145611
6	0.000905517	0.001239395	-0.000214090
7	0.000799214	0.001159341	-8.00539*E-05
8	0.000728553	0.000995135	-0.000164207
9	0.000689719	0.000949235	-4.58994*E-05
10	0.000666253	0.000924730	-2.45051*E-05
11	0.000653363	0.000913073	-1.16573*E-05
12	0.000646025	0.000885796	-2.72766*E-05
13	0.000640121	0.000879230	-6.56635*E-06
14	0.000632459	0.000903152	2.39222*E-05
15	0.000628257	0.000923519	2.03673*E-05
16	0.000625153	0.000947666	2.4147*E-05
17	0.000622299	0.000852917	-9.47493*E-05
18	0.000623033	0.000891020	3.81029*E-05
19	0.000616123	0.000851674	-3.93458*E-05
20	0.000613663	0.000848345	-3.32937*E-06

Tabella 6.8: Variazione dell'errore all'aumentare delle epoche di allenamento.

precedente abbiamo individuato quanti esempi fornire alla rete per l'allenamento mentre qui andiamo a definire quante volte allenare la rete su questi esempi. Al termine di ogni epoca si calcola lo scarto quadratico medio sui dati di allenamento e su una porzione di dati riservati alla validazione (non usati per l'allenamento) in modo da ottenere due risultati che mostrano la precisione della rete su dati già visti e su cui si è allenata e sui dati su cui invece non si è allenata. Normalmente la precisione in validazione è inferiore a quella sui dati di allenamento, ma epoca dopo epoca, l'errore cala e la precisione cresce. Si arriva ad una certa epoca in cui l'errore sui dati di allenamento continua a diminuire ma quello sui dati di validazione aumenta rispetto all'epoca precedente. Questo sta ad indicare che la rete è entrata nello stato di overfitting (già discusso in precedenza). Per non finire in questo problema bisogna trovare il giusto equilibrio tra il numero di dati per l'allenamento e il numero di epoche. Infatti se si continua ad allenare la rete per troppe epoche si otterrà che la rete saprà rispondere correttamente solo ai dati su cui si è allenata e non su nuovi dati. La tabella mostra gli errori in allenamento e validazione delle prime 20 epoche. Possiamo notare come ad

un certo punto, alla quattordicesima epoca, l'errore della validazione aumenti invece di diminuire. Da qui in poi il valore inizia a salire e scendere in modo altalenante quindi, per quanto risultato dai test, il numero di epoche che si andranno ad eseguire nell'allenamento sarà di 13. Questo, perché continuando ad allenare la rete, il risultato, come si vede nel grafico, rimane circa lo stesso, ma aumenta inevitabilmente il tempo richiesto. Quindi a parità di risultato ottenuto si è deciso di ridurre il tempo di esecuzione per quanto possibile.

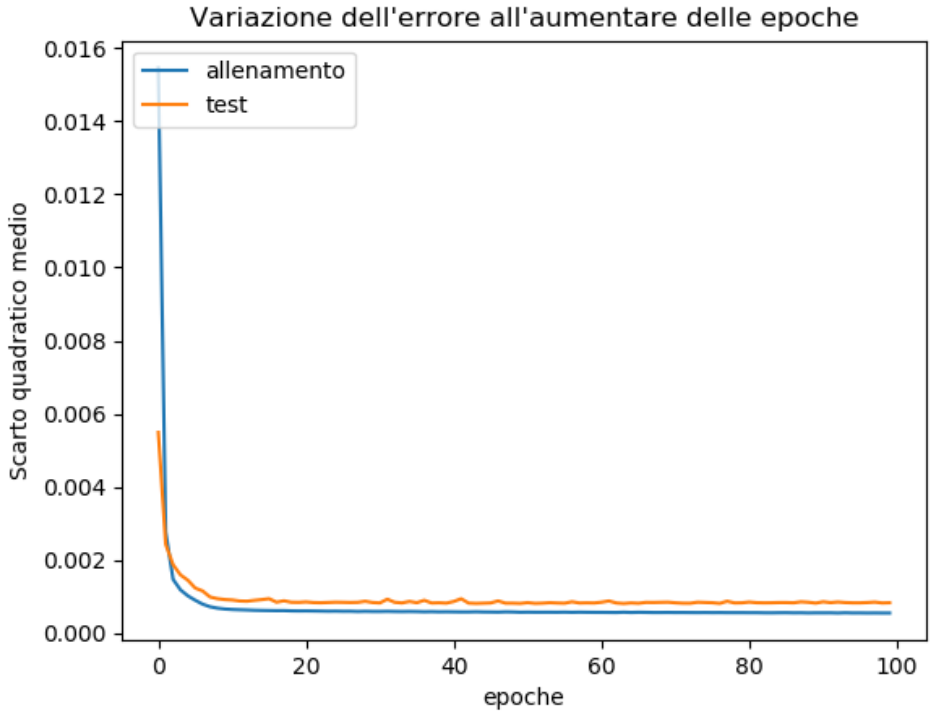


Figura 6.6: Variazione dell'errore all'aumentare delle epoche valutata sui dati di allenamento e sui dati di validazione.

6.6 Previsione a più ore di distanza

Con il modulo ottenuto possiamo fare previsioni del valore del PM 2.5. I valori relativi allo scarto quadratico medio mostrati precedentemente sono riferiti a previsioni dell'ora successiva. Si vuole però costruire un sistema in grado di prevedere anche a più ore di distanza. Per fare quanto appena detto la rete è allenata con un input rappresentante un certo numero di rilevazioni, mentre l'output è il valore del PM 2.5 ad un certo numero di ore di distanza.

6.7 Risultati dello sviluppo

Come risultato otteniamo più modelli in grado di risolvere i problemi iniziali. Ovviamente all'aumentare delle ore di distanza, diminuirà la precisione e aumenterà l'errore della previsione effettuata. L'errore calcolato ad un'ora di distanza è lo stesso che si avrà mediamente nella fase di attribuzione di un valore in caso di dati mancanti. Sapendo che i valori utilizzati sono stati normalizzati tra 0 e 1, l'errore percentuale sarà l'errore medio trovato moltiplicato per 100.

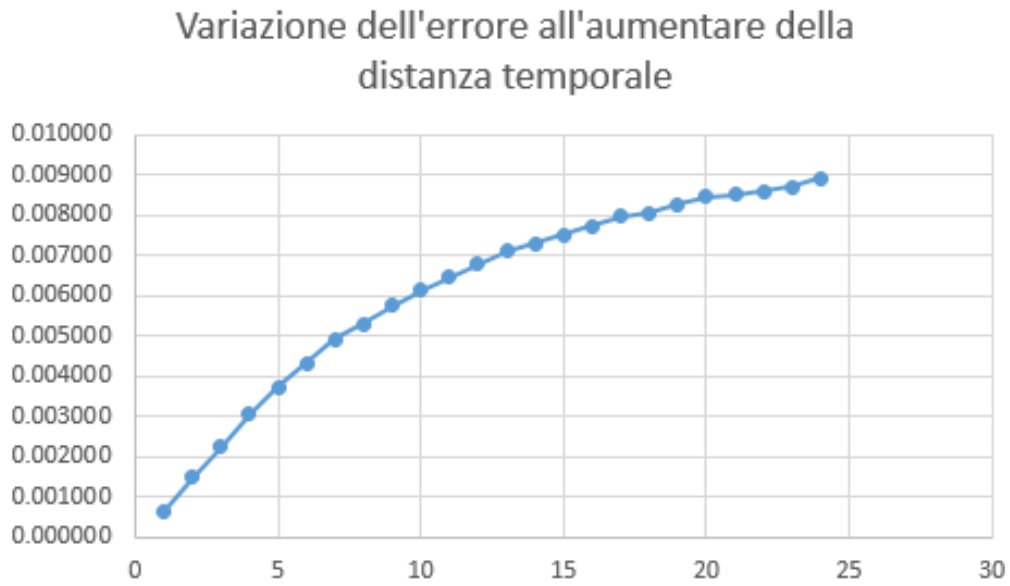


Figura 6.7: Variazione dell'errore all'aumentare della distanza temporale espressa in ore.

Ore di distanza	Scarto quadratico medio d'errore	Errore percentuale
1	0.000651	0.0650999
2	0.001470	0.1469958
3	0.002246	0.2246314
4	0.003050	0.3050432
5	0.003713	0.3713118
6	0.004328	0.4328007
7	0.004924	0.4923831
8	0.005285	0.5284639
9	0.005763	0.5762977
10	0.006137	0.6136655
11	0.006455	0.6454711
12	0.006787	0.6786696
13	0.007100	0.7099718
14	0.007305	0.7304726
15	0.007519	0.7519410
16	0.007720	0.7719712
17	0.007969	0.7969365
18	0.008039	0.8038682
19	0.008272	0.8271704
20	0.008446	0.8445536
21	0.008499	0.8499375
22	0.008589	0.8589117
23	0.008705	0.8705072
24	0.008930	0.8929500

Tabella 6.9: Variazione dell'errore all'aumentare della distanza temporale della previsione.

6.8 Organizzazione del software

Il software è stato suddiviso in tre moduli, esattamente come descritto in precedenza.

Capitolo 7

Implementazione

7.1 Modulo di allenamento

Questo modulo ha l'obiettivo di allenare i modelli al fine di renderli pronti ad effettuare le previsioni. Il modulo è stato implementato come libreria utilizzabile dagli altri due moduli per adempire ai loro compiti. Mette a disposizione una funzione principale chiamata `get_model` che permette, in base alla quantità di ore di distanza a cui si deve prevedere, di ottenere un modello allenato a svolgere quel determinato compito. Se il modulo esiste già lo carica, altrimenti lo produce, lo allena e lo salva. I entrambi i casi lo restituisce come output della funzione. Per produrre un modello viene richiamata la funzione `produce_model`. Quest'ultima al suo interno richiama a sua volta la funzione `neural_network_model`, la quale genera la struttura della rete neurale a serie temporali. Grazie alle funzioni di libreria di keras il tutto è possibile in poche linee codice.

```
38 def get_model(look_forward=1):
39     filename        =   'model_' + str(look_forward)
40     directory       =   'models'
41     if os.path.exists('./' + directory + '/' + filename + '.json') and os.path.exists('./' + directory + '/' + filename + '.h5'):
42         return load_and_compile_model(filename=filename, directory=directory, optimizer='adam', loss_function='mean_squared_error')
43     else:
44         model = produce_model(look_forward=look_forward)
45         save_model(model=model, filename=filename, directory= directory)
46         return model
```

Figura 7.1: Implementazione della funzione `get_model`.

```
33     model = neural_network_model(input_size=INPUT_SIZE, neurons=neurons, look_back=look_back, optimizer=optimizer, loss_function=loss_function)
34     history = model.fit(trainX, trainY, epochs=epochs, batch_size=batch_size, validation_split=validation_split)
```

Figura 7.2: Mostra come viene creato e allenato il modello in Python.

```python
def neural_network_model(input_size, neurons=4, look_back=12, optimizer='adam', loss_function='mean_squared_error'):
    model = Sequential()
    model.add(LSTM(neurons, input_shape=(input_size, look_back)))
    model.add(Dense(1))
    model.compile(loss=loss_function, optimizer=optimizer)
    return model
```

Figura 7.3: Implementazione della funzione `neural_network_model`.

7.2 Modulo per i valori mancanti

Inizialmente si vanno a leggere i dati dal dataset per individuare se e dove sono i valori mancanti. Individuati i valori mancanti, si creano gli input da fornire alla rete per fare la previsione prendendo le ore precedenti di rilevazioni necessarie. Infine vengono effettuate le previsioni e imputati i valori mancanti. I risultati sono slavati in un nuovo dataset. Il codice mostrato

```python
df   = pd.read_csv(TEST)

PM_COL      = 'pm2.5'
INPUT_SIZE  = len(df.columns)
testX, indexes = create_dataset(df=df, col=PM_COL, look_back=LOOK_BACK)
testX = np.reshape(testX, (int(testX.shape[0]/LOOK_BACK), INPUT_SIZE, testX.shape[1]))

model = get_model(look_forward=LOOK_FORWARD)
testY = denormalize(df.dropna()[PM_COL].values.astype('float32'), model.predict(testX))

for i in range(len(testY)):
    df.at[indexes[i],PM_COL] = testY[i]

df.to_csv(RESULT_PATH, index=False)
```

Figura 7.4: Codice che realizza l'imputazione di valori mancanti in un dataset.

nella figura 7.4 mostra come è stata implementata l'imputazione. Risulta molto evidente che la scelta delle librerie ha reso il tutto molto più semplice. La funzione `create_dataset` individua gli indici dei valori mancanti nel dataset e reperisce le rilevazioni necessarie per fare le attribuzioni. Queste ultime sono poi riorganizzate in modo che il modello possa utilizzarle per fare previsioni. La funzione `create_dataset` normalizza anche i dati in un intervallo compreso tra 0 e 1. A questo punto si richiama la funzione `get_model`, implementata nel modulo di allenamento, che restituisce un modello adatto a prevedere ad un certo quantitativo di ore di distanza; questo viene passato come input con il parametro `look_forward`. Ottenuto il modello, grazie alla libreria Keras, possiamo semplicemente richiamare il metodo predict e passare come input le rilevazioni recuperate in precedenza per ottenere le

previsioni. La funzione denormalize riporta i valori predetti alla loro scala originale. Continuando si vanno a sostituire i valori mancanti nel dataset con le previsioni effettuate, infine, si salva il dataset completo in un nuovo file csv.

7.3 Modulo per fare previsioni future

Per quanto riguarda le previsioni future si estraggono dal dataset i dati relativi alle ultime 24 ore che verranno utilizzate come input per tutte le previsioni. Successivamente vengono fatte le previsioni utilizzando modelli diversi in base all'ora che si vuole prevedere. Nella figura 7.5 viene mostrata l'im-

```
11  LOOK_BACK = 12
12  LOOK_FORWARD = 24
13  models = []
14  for i in range(LOOK_FORWARD):
15      models.append(get_model(look_forward=i+1))
16
17  df   = pd.read_csv(TEST)
18  PM_COL      = 'pm2.5'
19  INPUT_SIZE  = len(df.columns)
20  testX = df.tail(LOOK_BACK).values.astype('float32')
21  testX = np.reshape(testX, (int(testX.shape[0]/LOOK_BACK), INPUT_SIZE, testX.shape[1]))
22  predictions = []
23  offsets = []
24  original = df.dropna()[PM_COL].values.astype('float32')
25  for i in range(LOOK_FORWARD):
26      model = models[i]
27      offsets.append(i+1)
28      predictions.append(denormalize(original, model.predict(testX))[0][0])
29
30  result = pd.DataFrame(data={'offset' : offsets, 'PM2.5': predictions})
31  result.to_csv(RESULT, index=False)
```

Figura 7.5: Codice che realizza l'imputazione di valori mancanti in un dataset.

plementazione di questo modulo; anche qui si nota come con poche righe di codice si riesca ad ottenere il risultato. In alto sono definite il quantitativo di rilevazioni necessarie per effettuare una previsione e fino a quante ore di distanza si effettueranno previsioni. Questi due valori sono salvati rispettivamente nei parametri `LOOK_BACK` e `LOOK_FORWARD`. Successivamente si vanno a recuperare i modelli necessari con la funzione `get_model` precedentemente illustrata. Si vanno a reperire, scalare ed organizzare le rilevazioni necessarie

ad effettuare le previsioni. A questo punto si effettuano le previsioni utilizzando per ogni ora di distanza il modello appropriato. Infine, i valori sono riportati alla loro scala reale e, si salvazo i risultati in un dataset in formato csv.

Capitolo 8

Sviluppi Futuri

8.1 Miglioramento della precisione

Sicuramente è possibile e, quindi, bisogna capire come migliorare ulteriormente la precisione delle previsioni in modo da renderle ancora più accurate. Questo comporta uno studio più approfondito del modello che si sta utilizzando per capire come renderlo ancora più efficiente.

8.2 Sviluppo di un'interfaccia grafica

Sarebbe utile avere un'interfaccia grafica per controllare in modo più semplice il tutto. Potrebbe essere una buona idea creare un sito web o un applicativo che mostri i valori del PM 2.5 in modo da fornire una visione del suo andamento e il suo valore attuale.

8.3 Ricerca di nuovi modelli

Oltre a migliorare il modello che si sta utilizzando è possibile che ne esistano altri in grado di fornire prestazioni migliori e previsioni più accurate.

Capitolo 9

Conclusione

9.1 L'inquinamento

Data l'urgenza generalmente percepita del problema dell'inquinamento, quest'ultimo va affrontato poiché danneggia la qualità dell'aria e, quindi, anche la salute di chi la respira. La realizzazione di sistemi che tengano traccia del suo andamento per poter intervenire di conseguenza si rende indispensabile.

9.2 Conoscenze acquisite

Sviluppando la tesi ho potuto studiare una branca dell'informatica che fino a questo momento non avevo studiato molto: i dati e come utilizzarli. Un altro argomento che ovviamente è stato oggetto di studio è il machine learning che si è dimostrato, come da aspettative, uno strumento molto potente e in grado di risolvere problemi di diverso genere sfruttando un approccio innovativo. Infatti a differenza di altri approcci, dove si scrive un algoritmo in grado di risolvere un problema, qui il modello impara dai dati forniti come esempio per costruire un suo algoritmo partendo dai dati. L'algoritmo quindi non viene scritto dal programmatore ma ricavato dai dati. Oltre a questo mi sono imbattuto in Python: un linguaggio di programmazione a me nuovo. Python è un linguaggio che si può definire un po' "pazzo", ma in quanto a dati e machine learning, mette a disposizione degli strumenti e dei costrutti davvero potenti.

Bibliografia

[1] Roberto Giovannini. "Italia, qui l'inquinamento uccide di più". In: *La Stampa* (set. 2017). DOI: http://www.lastampa.it/2017/09/28/scienza/italia-qui-linquinamento-uccide-di-pi-CO7uY1aEC5EdcEwDbVNiIK/pagina.html.

[2] *Imputation (statistics)*. URL: https://en.wikipedia.org/wiki/Imputation_(statistics).

[3] *Libreria Keras*. URL: https://keras.io/.

[4] *Libreria scikit-learn*. URL: http://scikit-learn.org/stable/index.html.

[5] *Libreria TensorFlow*. URL: https://www.tensorflow.org/.

[6] *Missing data*. URL: https://www.iriseekhout.com/missing-data/.

[7] "Smog, l'Italia maglia nera in Europa: 90mila morti l'anno". In: *La Repubblica* (set. 2017). DOI: http://www.repubblica.it/ambiente/2017/09/29/news/smog_l_italia_maglia_nera_in_europa_90mila_morti_l_anno-176866776/.