UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

Progettazione di un prototipo per il confronto in termini di efficacia tra modelli di serie storiche di grandi dimensioni

Relatore: Prof. Massimo Melucci

Laureando: Gabriele Fugagnoli

ANNO ACCADEMICO 2022 – 2023

Data di laurea: 21 novembre 2023

Indice

1. Introduzione

1.1 Scelta del tirocinio

Devo la mia scelta di iscrivermi alla facoltà di Ingegneria Informatica ad un interesse nato in giovane età verso i computer ed il mondo del digitale, influenzato anche dalla passione per i videogiochi.
Durante gli anni scolastici non misi mai in discussione l'idea della mia predisposizione per la logica e la programmazione che invece si rafforzò grazie ai buoni risultati ottenuti in informatica alle superiori.

Nel corso della triennale ho realizzato che l'idea che mi ero fatto dell'informatica, ossia un mezzo con il quale dare vita alle proprie idee, in realtà nel mondo del lavoro si limita ad uno strumento per la traduzione di concetti e metodi in istruzioni compatibili con la rigida infrastruttura del computer.

Ho quindi cominciato ad esplorare possibilità vicine al mondo dell'informatica cercando un ambito che potesse offrire prospettive lavorative per me più appaganti e stimolanti.
Nel Data Science ho individuato una disciplina con la quale sfruttare le competenze acquisite durante la triennale aggiungendo un elemento di investigazione e interpretazione della realtà, argomento che da sempre ha suscitato il mio interesse.

È conseguita quindi la decisione di passare al curriculum applicativo del corso di Ingegneria Informatica per svolgere un tirocinio che mi permettesse di avvicinarmi al mondo del Data Science e acquisire una comprensione più approfondita delle competenze necessarie per questa figura professionale, oltre a confrontarmi con problemi reali.

L'azienda scelta, Logos Technologies S.r.l. e nello specifico la divisione Humco S.r.l, è una piccola impresa di consulenza che conta circa trenta dipendenti. Storicamente si è occupata principalmente di sistemistica ma negli ultimi anni sta sviluppando una presenza nel mondo dell'Intelligenza Artificiale e del Data Science.

1.2 Obiettivi del tirocinio

Lo scopo dell'azienda è quello di predire efficacemente la vendita dei prodotti di un cliente ad intervalli regolari. Per fare ciò è necessario selezionare il modello predittivo migliore con l'eventuale possibilità di creare gruppi di serie storiche di prodotti con andamenti di vendite simili per selezionare in maniera più mirata i modelli da utilizzare.

Iniziando con un set di tremila serie storiche di test che contengono osservazioni relative al passato recente, con voci valide per un periodo di due anni, e successivamente un secondo set aggiornato alla data attuale comprendente novemila serie storiche con voci parziali rispetto alla finestra temporale considerata, l'obiettivo era quello di ottenere una matrice per comparare la performance di diversi modelli predittivi tramite l'utilizzo di metriche di errore.

Nel corso del progetto è emersa la necessità di passaggi intermedi come un'analisi esplorativa dei dati e possibili trasformazioni delle serie. Tra queste trasformazioni rientrano, ad esempio, l'eliminazione dei fine settimana che contengono prevalentemente entry nulle e la conversione delle serie storiche da giornaliere a settimanali.

A causa della diversità tra le serie, prima di applicare i modelli di previsione si è considerata anche la possibilità di clusterizzare le serie storiche in gruppi contenenti serie simili tra di loro. In questo modo sarebbe stato possibile confrontare la performance di predizione dei modelli su serie simili ed evitare di dover applicare un singolo modello ad un grande numero di serie con caratteristiche diverse.

2. Lettura dei dati e analisi esplorativa

Il primo set di tremila serie storiche composte da 880 osservazioni giornaliere contiene una media di 477 osservazioni nulle ed un minimo di 232 osservazioni nulle per serie. Non sono invece presenti valori NaN.

Convertendo le date in formato *datetime* è stato possibile utilizzare la sua funzione membro *weekday()* che restituisce il giorno della settimana a cui corrisponde la data per eliminare i weekend dalle serie storiche e spostare le eventuali osservazioni positive al venerdì. Così facendo è stata osservata una riduzione arrivando ad una nuova media di 240 zeri per serie storica e un nuovo minimo di 25 zeri.

Un'altra elaborazione che si è rivelata utile per future operazioni è stata quella di creare una lista degli zeri consecutivi per ogni serie storica. La lista consiste in numeri che rappresentano il numero di zeri consecutivi: ad esempio la lista [18, 4, 7] descrive una serie in cui sono presenti diciotto zeri consecutivi, seguiti da osservazioni positive a cui seguono quattro zeri consecutivi e dopo altre entry non nulle ancora sette zeri di fila.

Sono anche state scritte funzioni per applicare trasformazioni alle serie come il Simple Moving Average e la divisione di serie in array per il reinforcement learning.

Il file csv contenente il secondo set da novemila serie storiche comprendeva dati mal rappresentati e organizzati, c'è stato quindi bisogno di:

- trasformare le date da stringhe di testo a formato *pandas.datetime*
- eliminare spazi e simboli non necessari dalle entry di ID prodotto e quantità vendute
- convertire le quantità vendute da stringhe a float
- ordinare cronologicamente le vendite

Molte serie del secondo set contengono troppe poche entry perché sia possibile una previsione, è stato quindi necessario selezionare un set di serie con abbastanza dati senza però scartare serie storiche di prodotti con forte stagionalità che contengono vendite solo in certi periodi dell'anno.
Come primo passo sono state eliminate le serie con numero di zeri negli ultimi due anni sopra una certa soglia parametrizzabile, successivamente è stata associata ogni serie al numero massimo di vendite mensili registrate negli ultimi 12 mesi.
Si eliminano quindi le serie che hanno il valore massimo di vendite mensili nell'ultimo anno sotto una certa soglia parametrizzabile (ad esempio quindici giorni di vendite) o che hanno la media di vendite mensili nell'ultimo anno sotto un'altra soglia (dieci vendite).
Ciò facendo le serie finali selezionate sono state poco meno di quattromila.

3. Clustering

La prima strategia utilizzata per dividere le serie in gruppi contenenti serie storiche simili tra di loro è stata quella di utilizzare la distanza Dynamic Time Warping ed il metodo di clusterizzazione K-means.

3.1 Dynamic Time Warping e K-means

Il **Dynamic Time Warping** è un algoritmo che estende il concetto di Distanza Euclidea.
La Distanza Euclidea consiste nel sommare la differenza tra le coppie di entry delle due serie appartenenti allo stesso istante temporale.
Il Dynamic Time Warping prima di computare le differenze dei punti crea un allineamento tra le due serie facendo un'associazione tra coppie di punti in modo da considerare espansioni e contrazioni temporali.

La matrice delle distanze calcolate con il Dynamic Time Warping è stata elaborata iterando su tutte le combinazioni di coppie del set e computando la distanza utilizzando direttamente la funzione *TimeSeriesKmeans* della libreria tslearn, impostando il parametro della metrica a "*dtw*".

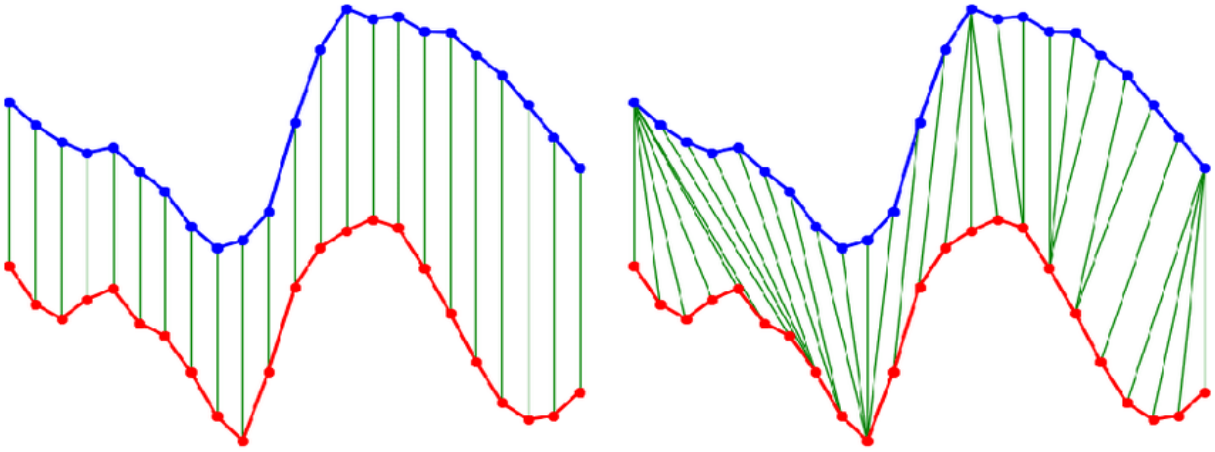

Figura 3.1: A sinistra l'algoritmo di accoppiamento dei punti della Distanza Euclidea, a destra quello del Dynamic Time Warping.

Il **K-means** è un algoritmo di clustering non gerarchico che inizia la prima iterazione con dei centri creati tramite un processo stocastico e ad ogni passo ne aggiorna la posizione in base ad una metrica parametrizzabile.

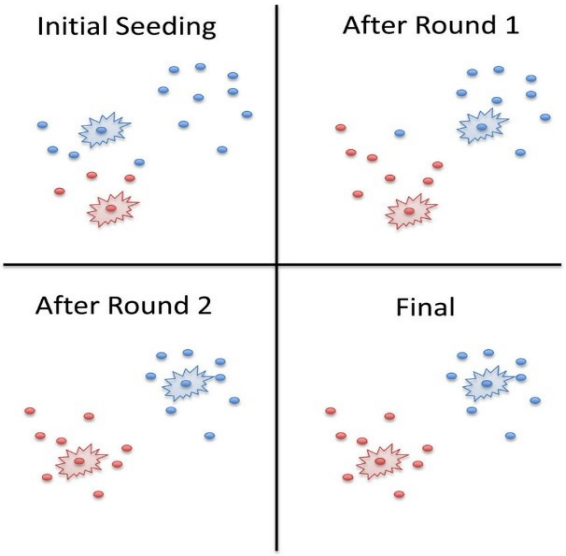

Figura 3.2: Algoritmo K-means

È stato quindi eseguito il clustering K-means con metrica DTW su un subset di esempio formato da sei serie storiche (scalate e lisciate con Simple Moving Average).

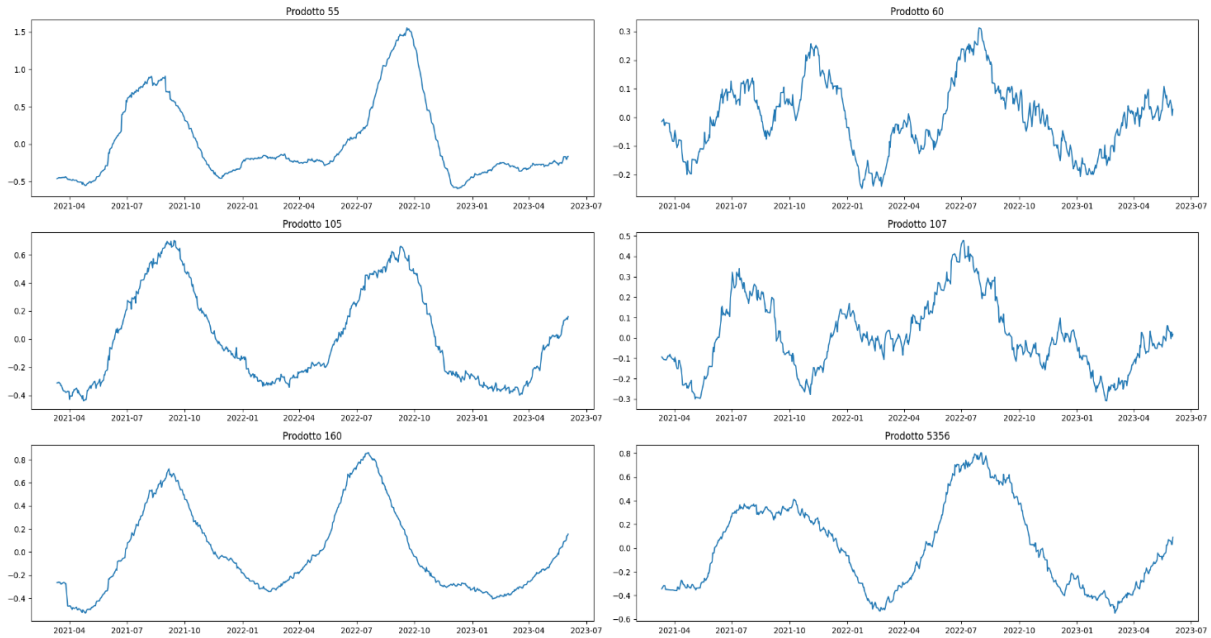

Figura 3.3: Grafici delle serie su cui è stato fatto il clustering con DTW e K-means

La clusterizzazione in due cluster ha prodotto i seguenti gruppi:

- cluster 1: serie 55, 105, 160, 5356
- cluster 2: serie 60, 107

Il clustering di queste sei serie ha impiegato 30 secondi, un tempo troppo elevato considerando l'andamento esponenziale dell'algoritmo che deve calcolare la distanza di ogni coppia possibile di serie.

A causa della complessità computazionale dell'algoritmo è stato quindi tentato un altro metodo di clusterizzazione che impiega la Compression Based Distance.

3.2 Symbolic Aggregate Approximation e Compression Based Distance

La **Compression Based Distance** è un metodo per calcolare la dissimilarità tra due stringhe di testo attraverso l'utilizzo di un algoritmo di compressione. L'algoritmo si basa sull'assunzione che, se due stringhe di testo X e Y sono simili, il file derivato dalla compressione della stringa X + Y sia più piccolo della somma dei file della compressione separata di X e Y dato che l'algoritmo può usare gli stessi pattern presenti in entrambe le serie per eseguire la compressione. Nel caso invece in cui le serie siano completamente diverse, ci si aspetta che la dimensione della compressione di X+Y sia simile alla somma delle due compressioni separate.

La tecnica che ci permetterà di applicare la CBD alle nostre serie storiche è la Symbolic Aggregate Approximation.

```
function  dist = CDM(A, B)
 save A.txt A -ASCII                          % Save variable A as A.txt
 zip('A.zip', 'A.txt');                       % Compress A.txt
 A_file = dir('A.zip');                       % Get file information

 save B.txt B -ASCII                          % Save variable B as B.txt
 zip('B.zip', 'B.txt');                       % Compress B.txt
 B_file = dir('B.zip');                       % Get file information

 A_n_B = [A; B];                              % Concatenate A and B
 save A_n_B.txt A_n_B -ASCII                  % Save A_n_B.txt
 zip('A_n_B.zip', 'A_n_B.txt');               % Compress A_n_B.txt
 A_n_B_file = dir('A_n_B.zip');               % Get file information
                                               % Return CDM dissimilarity
dist =  A_n_B_file.bytes / (A_file.bytes + B_file.bytes);
```

Figura 3.4: Pseudo codice della CBD estratto dal paper "Towards Parameter-free Data Mining" di Keogh, Leonardi e Ratanamahatana

La **Symbolic Aggregate Approximation** è una procedura che divide il dominio dei valori a cui appartengono le osservazioni delle serie storiche in intervalli, associando ciascun intervallo ad una lettera.

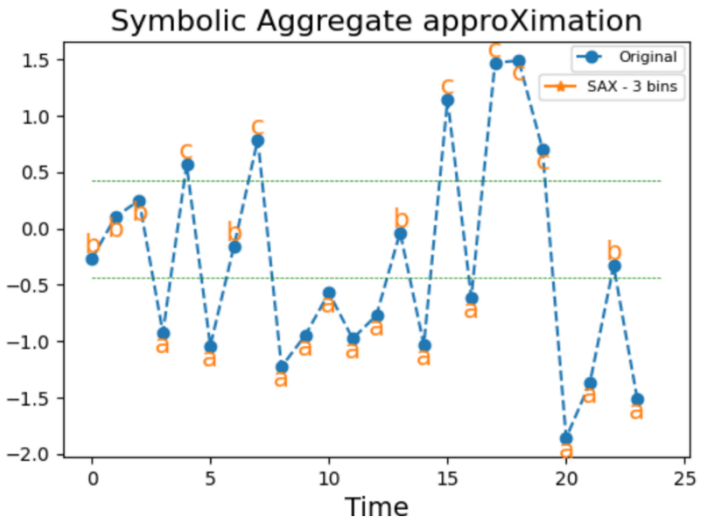

Figura 3.5: Rappresentazione della Symbolic Aggregate Approximation

Una volta ottenuta la matrice delle distanze con la Compression Based Distance, è stato usato l'algoritmo Agglomerative Clustering, scelta motivata dall'impossibilità di utilizzare come input della funzione *TimeSeriesKmeans()* di tslearn una matrice precalcolata.

L'Agglomerative Clustering è un algoritmo di clusterizzazione gerarchico, crea quindi una struttura ad albero contenente cluster che diventano più piccoli avvicinandosi alle foglie.

"Tagliando" l'albero lungo i cammini (nella figura 3.6 il taglio si presenterebbe come una linea orizzontale) si ottiene quindi una divisione in gruppi degli elementi originali. Sono stati usati come materiale di riferimento i seguenti paper: "*Experiencing SAX: a Novel Symbolic Representation of Time Series*" di J. Lin, E. Keogh, L. Wei e S. Lonardi, "*A Symbolic Representation of Time Series, with Implications for Streaming Algorithms*" di J. Lin, E. Keogh, S. Lonardi e B. Chiu, "*Towards Parameter-Free Data Mining*" di E. Keogh, S. Lonardi e C.A. Ratanamahatana, "*Clustering by Compression*" di R. Cilibrasi e P.M.B. Vitànyi.

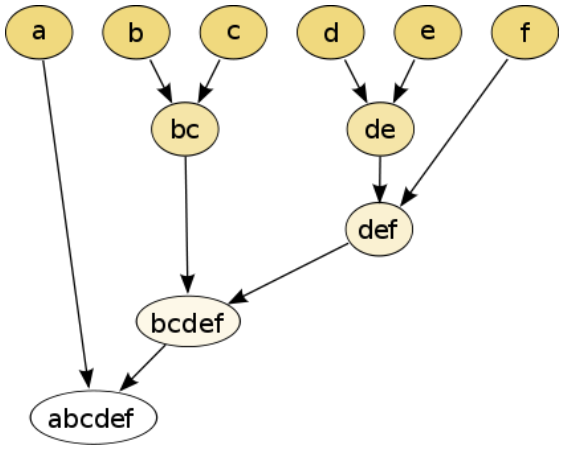

Figura 3.6 Algoritmo Agglomerative Clustering, le lettere rappresentano gli oggetti da clusterizzare.

Applicando questo metodo alle serie d'esempio presentate in figura 3.3 otteniamo lo stesso risultato di clustering impiegando una frazione del tempo impiegato dal metodo con DTW e K-means.

Scalando il numero di serie però i risultati degenerano e otteniamo cluster contenenti serie molto diverse.

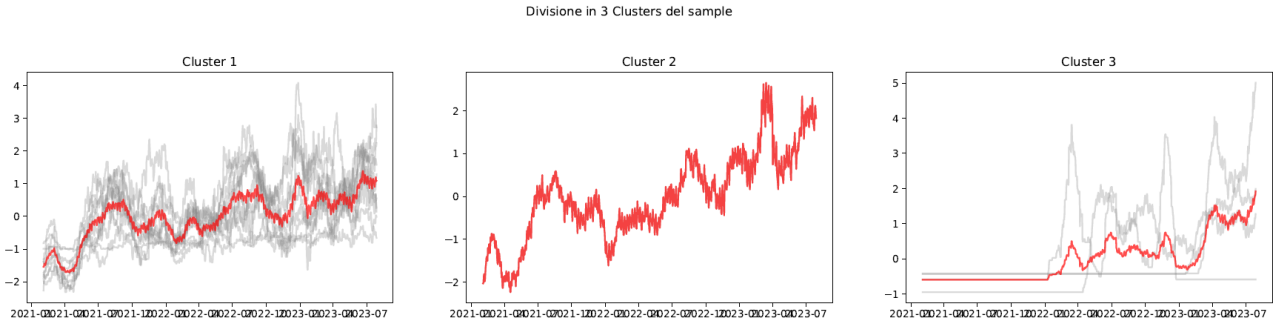

Figura 3.7: Clustering di quindici serie con SAX e CBD, le linee in grigio rappresentano le singole serie storiche mentre quelle in rosso la media delle serie presenti nel cluster.

4. Forecasting

4.1 Introduzione dei modelli

Per la previsione delle vendite future sono stati considerati quattro modelli: il Prophet, una Rete Neurale Convoluzionale (CNN), il DeepAr e il Non Parametric Time Series (NPTS).

Il modello **Prophet** è un metodo di previsione di serie storiche che considera stagionalità di diverse frequenze e "holiday effects" sviluppato da Facebook (ora Meta) per essere facilmente utilizzato da utenti senza approfondite conoscenze tecniche. Prophet esegue il fit su una singola serie storica ed è stato pensato per funzionare principalmente con serie giornaliere.

Una **CNN** è una rete neurale che, nei primi layer, applica una convoluzione ai dati in input. Ciò permette di creare feature che tengono conto della relazione tra input adiacenti (più è grande il kernel più input adiacenti vengono elaborati assieme). L'utilizzo più comune per le CNN è nel campo del computer vision dato che la convoluzione permette di elaborare le immagini mantenendo le relazioni tra pixel adiacenti. Per utilizzare una CNN con delle serie storiche basterà utilizzare kernel unidimensionali a differenza di quelli più comunemente usati per l'elaborazione di immagini che sono bidimensionali.
Essendo una rete neurale, ci sarà bisogno di modificare i dati in modo che siano processabili, gli altri modelli invece essendo progettati per l'elaborazione di serie storiche potranno ricevere input simili ai dati in nostro possesso. Per l'implementazione mi sono basato sulle guide scritte da J. Brownlee PhD nel sito "machinelearningmastery.com".

Il modello **DeepAr** sviluppato da Amazon utilizza un modello di rete neurale ricorsiva per fare predizioni su serie storiche univariate. A differenza del Prophet, DeepAr è stato implementato per fare training e predizioni su un gruppo di serie storiche generate dallo stesso processo stocastico.

Il modello **NPTS** sviluppato anch'esso da Amazon è un modello non parametrico, ossia non fa assunzioni sulla struttura dei processi stocastici che producono la serie storica. Utilizza pesi con decadimento esponenziale per dare più importanza alle osservazioni più vicine alla previsione. Esegue il training e previsioni sulle singole serie.

4.2 Tuning degli iperparametri

Per il tuning degli iperparametri è stato utilizzato il framework Optuna. Optuna permette di eseguire un tuning automatico degli iperparametri con la stessa metodologia per qualsiasi modello.

La ricerca dei migliori iperparametri in Optuna viene eseguita durante uno studio che consiste in diverse prove, ognuna delle quali sperimenta un diverso set di iperparametri.
Per iniziare uno studio è necessario definire una funzione Objective che accetta come parametro un oggetto trial definito nel framework Optuna. All'interno della funzione Objective bisogna:

- creare il modello desiderato utilizzando i parametri contenuti nell'oggetto trial
- fare training e predizione con il modello creato
- ritornare la metrica da ottimizzare indicata all'oggetto study

4.3 Preparazione dei dati per tuning e forecast

Per tutti i modelli i dati disponibili sono stati divisi in set di train e test. Il set di train è stato ulteriormente diviso in set di train e validation da utilizzare nel tuning.
Il set di train viene utilizzato per addestrare i modelli che ritorneranno una previsione, la previsione andrà quindi confrontata con il set di validation per computare la metrica con la quale confrontare diversi set di iperparametri. Una volta scelto il migliore set di iperparametri verranno utilizzati tutti i dati disponibili (tranne il set di test) per l'addestramento, la previsione finale verrà quindi confrontata con il set di test. Nel caso delle serie storiche la divisione deve essere effettuata in modo da mantenere la sequenzialità dei dati.
Poiché i modelli Prophet, DeepAr e NPTS gestiscono autonomamente i dati ricevuti in input per fare fit e predizione, è stato sufficiente dividere le serie nei set sopra citati. Per la CNN è stato invece necessario dividere le serie storiche in set di dati adatti al supervised learning, ossia coppie di dati in formato input/output atteso.

5. Metriche d'errore e risultati

5.1 Introduzione delle metriche d'errore

Le metriche d'errore utilizzate sono il MASE, MAPE, RMSE, WAPE.

Partendo dal semplice errore:

$$e_t = Y_t - \hat{Y}_t$$

dove Y_t e $\hat{Y}_t$ sono rispettivamente l'osservazione e la predizione relativi all'instante *t*, il **MASE** (Mean Absolute Scaled Error) è una misura di accuratezza basata sull'errore scalare che è definito come:

$$q_t = \frac{e_t}{\frac{1}{n-1}\sum_{i=2}^{n} |Y_i - Y_{i-1}|}$$

che ha la proprietà di essere indipendente dalla scala dei dati. Una volta definito l'errore si può calcolare il MASE come:

$$MASE = \frac{1}{n}\sum_{t=1}^{n}|q_t|$$

Il MASE non diverge ad infinito a meno che tutti i valori della serie storica siano uguali, il che lo rende una metrica di facile utilizzo in molteplici applicazioni.

Il **RMSE** (Root Mean Squared Error) si basa sul Mean Squared Error:

$$MSE = \frac{1}{n}\sum_{t=1}^{n}e_t^2$$

$$RMSE = \sqrt{MSE}$$

L'RMSE è una misura dipendente dalla scala dei dati. A differenza dell'MSE, l'RMSE permette di ottenere un errore con la stessa scala delle osservazioni della serie storica.

Il **MAPE** (*Mean Absolute Percentage Error*) invece si basa sull'errore percentuale p_t definito come:

$$p_t = \frac{100e_t}{Y_t}$$

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}|p_t|$$

Essendo l'errore utilizzato percentuale, anche il MAPE è una misura indipendente dalla scala dei dati utilizzati.

Anche il **WAPE** è un errore percentuale che però scala l'errore utilizzando la somma di tutte le osservazioni:

$$WAPE = \frac{\sum_{t=1}^{n}|e_t|}{\sum_{t=1}^{n}|Y_i|}$$

Come materiale di riferimento è stata consultata la tesi "*La valutazione dell'accuratezza delle previsioni: indici classici e il MASE*" di V. Zancan.

5.2 Risultati delle predizioni

Per la previsione è stato scelto un subset di undici serie simili tra loro (per non penalizzare il modello DeepAr che esegue il training su gruppi di serie che si assume siano state prodotte dallo stesso processo) e contenenti un'alta percentuale di entry maggiori di zero.

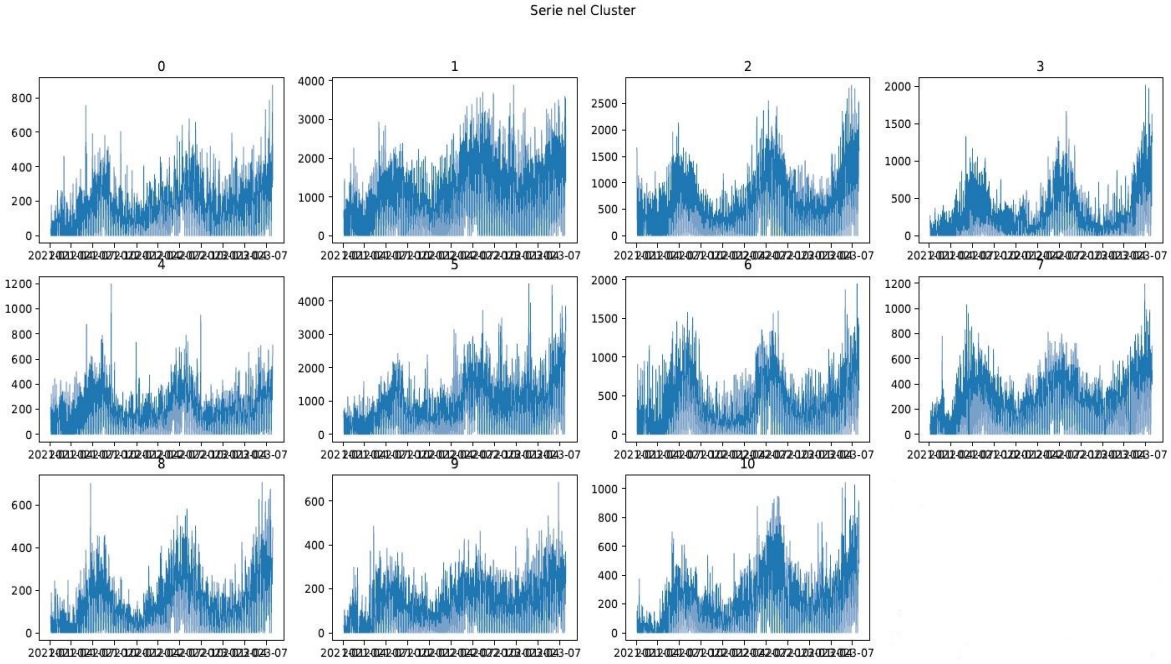

Figura 5.1 Serie storiche selezionate per la predizione

È stato usato un orizzonte di previsione di una settimana e le serie sono state mantenute giornaliere. Per utilizzare i dati delle serie con il modello CNN c'è stato bisogno di scalare le serie e differenziarle.

Metriche per ogni serie

0

	CNN	Prophet	DeepAr	NPTS
mase	0.4204	0.4351	0.3593	0.6804
mape	inf	inf	inf	inf
wape	0.3544	0.3669	0.303	0.5737
rmse	176.7035	198.5764	198.5616	295.8097

1

	CNN	Prophet	DeepAr	NPTS
mase	0.3343	0.2926	0.3771	0.4119
mape	inf	inf	inf	inf
wape	0.3217	0.2816	0.3628	0.3963
rmse	826.4483	624.5701	850.1217	934.6444

2

	CNN	Prophet	DeepAr	NPTS
mase	0.3295	0.2629	0.278	0.5543
mape	inf	inf	inf	inf
wape	0.2946	0.2351	0.2485	0.4957
rmse	449.2877	356.6051	416.1317	807.2564

3

	CNN	Prophet	DeepAr	NPTS
mase	0.5922	0.2975	0.3528	0.6729
mape	inf	inf	inf	inf
wape	0.5228	0.2626	0.3115	0.5941
rmse	541.1453	265.7962	338.1156	625.1893

4

	CNN	Prophet	DeepAr	NPTS
mase	0.3906	0.4437	0.2852	0.6238
mape	inf	inf	inf	inf
wape	0.2756	0.3131	0.2012	0.4402
rmse	99.4038	144.2043	104.0508	213.2603

5

	CNN	Prophet	DeepAr	NPTS
mase	0.4773	0.4518	0.3378	1.14
mape	inf	inf	inf	inf
wape	0.2391	0.2263	0.1692	0.5712
rmse	697.3765	606.1152	583.5992	1443.7656

6

	CNN	Prophet	DeepAr	NPTS
mase	0.4872	0.3673	0.3647	0.4959
mape	inf	inf	inf	inf
wape	0.4948	0.373	0.3704	0.5037
rmse	437.869	350.8657	374.0376	519.9901

7

	CNN	Prophet	DeepAr	NPTS
mase	0.476	0.3738	0.3692	0.5452
mape	inf	inf	inf	inf
wape	0.3316	0.2604	0.2573	0.3799
rmse	181.6208	124.8683	155.5182	192.9

8

	CNN	Prophet	DeepAr	NPTS
mase	0.8557	0.8145	0.7329	0.8988
mape	inf	inf	inf	inf
wape	0.4754	0.4525	0.4072	0.4994
rmse	160.7325	150.771	145.5077	177.5046

9

	CNN	Prophet	DeepAr	NPTS
mase	0.8419	0.4218	0.3527	0.3881
mape	inf	inf	inf	inf
wape	0.7061	0.3537	0.2958	0.3255
rmse	177.7761	81.4206	81.7764	86.1652

10

	CNN	Prophet	DeepAr	NPTS
mase	0.649	0.5206	0.5803	0.561
mape	inf	inf	inf	inf
wape	0.4834	0.3878	0.4323	0.4179
rmse	208.9637	181.8267	202.4912	238.171

Figura 5.2 Risultati delle previsioni per ogni serie storica

Lo scalamento dei dati in un range [-1, 1] è necessario per ottenere dati compatibili con le funzioni di attivazione della rete neuronale, la differenziazione invece è necessaria per evitare di avere una predizione attesa fuori scala rispetto ai dati su cui è stato eseguito il training del modello. Una volta ottenuta, la predizione è stata ritrasformata nel formato originale per poterla confrontare con i valori attesi.

Dato che le serie del set includono zeri, come atteso il MAPE risulta sempre infinito.
Dai risultati si evince che i modelli Prophet e DeepAR hanno prodotto le previsioni migliori.

6. Conclusioni

Pur non essendo riuscito ad ottenere i risultati prefissati, durante il tirocinio ho avuto occasione di affrontare problematiche specifiche della gestione ed elaborazione di dati. Colloco tra le più importanti la capacità di motivare le scelte fatte e prendere decisioni in mancanza di procedure già conosciute, di ottenere una comprensione approfondita dei metodi utilizzati e di pianificare il lavoro in modo da ottimizzare il tempo.

Avendo scritto molto codice Python ho avuto anche la possibilità di acquisire esperienza pratica, della quale ho sempre sentito la mancanza durante il mio percorso scolastico. Ciò ha aumentato la mia sicurezza nello scrivere codice funzionante, nell'utilizzare librerie e nella comprensione della loro documentazione.

Avendo svolto il lavoro principalmente in autonomia, ho realizzato anche l'importanza di ricevere feedback sul lavoro svolto da parte di un mentore con esperienza, oltre che alla possibilità di confrontarsi con colleghi e condividere idee. Questo aspetto sarà sicuramente un elemento che terrò in considerazione nella mia futura ricerca di lavoro.

Appendice

Setup ambiente di sviluppo

Per lo sviluppo del progetto il linguaggio di programmazione scelto è stato Python e, come ambiente di sviluppo integrato, Visual Studio Code.
Il progetto utilizza diversi metodi di sviluppo: Notebook Jupyter per sperimentare agevolmente nuovi metodi, moduli e pacchetti contenenti classi e funzioni utilizzate in diversi ambiti del progetto e script per l'esecuzione di codice che non rientra nelle precedenti categorie.

Ambiente virtuale

Una buona pratica dello sviluppo di codice Python consiste nell'uso di ambienti virtuali.
Un ambiente virtuale permette di creare ambienti isolati per installare e gestire pacchetti software e dipendenze, favorendo la divisione dell'ambiente di sviluppo tra diversi progetti. Ciò alleggerisce l'ambiente che conterrà solo i pacchetti necessari al progetto e previene conflitti tra diverse librerie.
Questo isolamento favorisce anche la riproducibilità del progetto che potrà essere eseguito facilmente su diversi sistemi, i quali dovranno installare solo i pacchetti essenziali per il progetto.

Un ambiente virtuale si presenta come una directory contenente l'interprete Python ed i pacchetti installati, oltre ad un insieme di metadati e script. Per poter compilare i file Python bisognerà indicare all'IDE l'ambiente virtuale da utilizzare.

Esistono diversi software per la creazione ed il mantenimento di ambienti virtuali, i più utilizzati sono Venv e Conda. Inizialmente per il progetto fu scelto Conda ma dopo problemi con l'installazione di alcuni pacchetti si passati a Venv.

Controllo di versione

Uno degli strumenti più diffusi nello sviluppo software è il sistema di controllo versione distribuito.
Il sistema di controllo versione distribuito è uno strumento che traccia e gestisce cambiamenti all'interno dei file selezionati, in genere contenenti codice sorgente, oltre a mantenere una versione di backup su una macchina remota. Una feature chiave dei VCS (Version Control System) è che i cambiamenti ai file vengono memorizzati con modalità additiva, ossia vengono esclusivamente aggiunte informazioni al repository contenente il progetto senza modificare le

versioni precedenti dei file, permettendo di tornare ad una versione precedente in caso di problemi.

I Distributed CVS vengono utilizzati principalmente per la collaborazione tra diversi programmatori ma in questo caso, dato che il tirocinio non prevedeva collaborazioni di alcun tipo nella scrittura del codice, è stato utilizzato solamente come sistema di backup e di controllo versione.

Lettura e gestione dei dati

Per la gestione dei dati contenuti nei file csv è stata scelta la libreria Pandas. Una prima lettura dei file è stata fatta con la funzione *pandas.read_csv(),* successivamente le serie storiche sono state organizzate in un dizionario, dove ogni chiave rappresenta il codice identificativo della serie e il valore associato è il dataframe corrispondente alla serie stessa.
Un dataframe è un tipo di rappresentazione di dati tabellari utilizzato dalla libreria Pandas.
I dizionari ottenuti sono stati poi salvati su disco utilizzando il modulo Pickle di Python.
Il modulo Pickle permette di serializzare e deserializzare oggetti Python in modo da salvarli sotto forma di file con estensione *pkl*, metodo utile quando ripetere le trasformazioni su dati ogni volta che ne è richiesto l'utilizzo sarebbe troppo oneroso dal punto di vista computazionale. È stato così possibile salvare come dizionari diverse versioni delle stesse serie storiche: giornaliere, settimanali, senza fine settimana e sottoinsiemi di serie scelte con varie regole.