POLITECNICO DI TORINO

POLITECNICO DI TORINO

Corso di Laurea in Ingegneria Informatica

Tesi di Laurea

Outlier Analysis con tecniche di Time Series Forecasting e algoritmi Unsupervised

Outlier detection in contesto aziendale

Relatore
Prof. Daniele Apiletti

Candidato
Domenico Cefalo
matricola: 267569

Supervisore aziendale
KPMG SpA
dott. Alberto Marocchino

Anno accademico 2020-2021

Sommario

Sono stati analizzati e migliorati, in contesto aziendale, un processo automatico di vendite ed acquisti strettamente correlati ad una soglia statica. L'obiettivo è quello di rendere il processo in uso il più intelligente possibile cercando di ridurre al minimo le correzioni degli ordini da parte dell'essere umano (in questo contesto li chiameremo Metal Manager). L'implementazione di un sistema capace di predirre, o meglio suggerire, una soglia non più statica ma dinamica mira proprio a questo scopo, facendo si che essa possa essere aggiornata giorno per giorno rendendo il modello sempre più preciso e puntuale. Sono stati analizzati diversi metodi partendo da un'analisi preliminare utilizzando lo strumento intuitivo del boxplot, passando per tecniche più sofisticate ed innovative di Time Series Forecasting come il modello Prophet (oggi largamente utilizzato in tantissime applicazioni di e-commerce) terminando poi all'analisi di algoritmi unsupervised per l'outlier detection come l'Isolation Forest.
Con grafici e risultati numerici e grazie ad un diverso numero di KPI si è dimostrato che i modelli risultano essere altamente efficienti riducendo al minimo l'intervento umano ed essere altamente flessibile a seconda dell'asse temporale in cui è richiesta la predizione.

Ringraziamenti

Con il presente lavoro che segna la chiusura del mio percorso di studi, desidero ringraziare in primis chi ha creduto in questo lavoro e mi ha sostenuto durante tutto il percorso di tesi, in particolare il Prof. Daniele Apiletti, il tutor aziendale Dott. Alberto Marocchino e la mia ex collega Giulia con la quale ho condiviso difficoltà, successi e risate.

Un sentimento di riconoscenza e di sincero affetto lo rivolgo alla città di Torino che mi ha accolto da ragazzo quando mi sentivo una persona spaesata ed estranea rispetto ad un ambiente per me nuovo. Mi ha saputo includere e mi ha fatto diventare una persona più matura, sicuramente una città che farà sempre parte della mia vita.

Un ringraziamento sincero a mio padre, mia madre e mia sorella Maria Pia perché mi hanno sostenuto ed ascoltato, con equilibrio ed attenzione, durante tutto il periodo accademico.

Ringrazio Valentina, con cui ho condiviso tutte le mie giornate affrontando le difficoltà e le paure del complesso percorso di studi ma sopratutto le gioie dei sudati esami superati appoggiandomi con forza e determinazione.

In fine, voglio ringraziare il collegio ONAOSI e il Politecnico perché mi hanno dato la possibilità di crescere e di conoscere persone fantastiche che mi hanno accompagnato ed aiutato per tutta la durata dell'Università e con cui ho instaurato rapporti di vera e sincera amicizia.

Success is not final, failure is not fatal: it is the courage to continue that counts. (W. Churcill)

Indice

Elenco delle tabelle

Elenco delle figure

Parte I

Descrizione del problema

Capitolo 1

Introduzione all'Outlier Analysis

1.1 Introduzione

L'uso del Machine Learning, sia con tecniche Supervisionate sia Non-Supervisionate, si sta ampliando in ambiti sempre maggiori quali ad esempio la verifica 'audit' di ordini in real-time. I processi di verifica ordini, per le grandi aziende, sono un processo costoso in quanto i controlli non riescono, per volumi, ad essere sufficientemente capillari. La possibilità di usufruire di tecniche di Machine Learning permetterebbe un controllo puntuale, rigoroso e al contempo continuo. In un processo di approvazione o blocco di ordine, la soglia fissa, è sempre risultata, per quanto limitata, chiara a tutti i livelli aziendali permettendo un controllo coerente con la normativa così come dai controller. L'applicazione di soglie fisse non elimina però errori, comportamenti che si discostano da un comportamento virtuoso. È dunque necessario sviluppare algoritmi di Unsupervised Machine Learning in grado di identificare tutte e soli gli outlier in spazi n-dimensionali in grado di limitare l'intervento umano a verifiche specifiche ed efficaci. La parte di unsupervised deve, in questa fase, servire come step per capire come il data-flagging possa essere composto per una fase di supervised.

La tesi è divisa nel seguente modo:

- **Capitolo 1**: Introduzione all'outlier analysis e dei diversi modelli di detection
- **Capitolo 2**: Descrizione ed obiettivi del lavoro di tesi in contesto aziendale

- **Capitolo 3**: Strumenti coinvolti nella fase di sviluppo
- **Capitolo 4**: Introduzione alle tecniche di Outlier Detection
- **Capitolo 5**: Analisi preliminare con utilizzo di boxplot
- **Capitolo 6**: Analisi con algoritmi di Time Series Forecasting
- **Capitolo 7**: Analisi con algoritmi di tipo Unsupervised
- **Capitolo 8**: Conclusioni con analisi tramite KPI e tempo di esecuzione
- **Capitolo 9**: Lavori futuri

1.2 Anomaly Detection

Ci sono diverse definizioni che nel corso degli anni sono state attribuite ai valori anomali (outliers), ma in generale si può affermare che un valore anomalo è un punto dati significativamente diverso dai restanti dati. Hawkins ne diede una definizione analoga, ovvero: *"un outlier è un'osservazione che si discosta così tanto dalle altre osservazioni da far sospettare che sia stata generata da un meccanismo diverso"* [1].
Un outlier spesso fornisce una indicazione e contiene informazioni utili sulle caratteristiche anomale dei sistemi e delle entità che influiscono sul processo di generazione dei dati. Alcuni esempi possono essere i seguenti:

1. **Intrusion Detection**: in molti sistemi informatici vengono raccolti diversi tipi di dati sul sistema operativo o sul traffico di rete. Questi dati possono mostrare un comportamento insolito a causa di attività dannose. Il riconoscimento di tale attività è indicato come rilevamento delle intrusioni

2. **Diagnosi medica**: in molte applicazioni mediche, i dati vengono raccolti da una varietà di dispositivi producendo ad esempio radiografie, risonanza magnetica, tomografia, elettrocardiogramma etc. Sintomi inusuali o risultati di test anomali possono riflettere la condizione di una possibile malattia

3. **Fraud Detection**: si può rilevare un evento illegale analizzando quali sono le transizioni oppure i movimenti bancari di una persona. Queste operazioni possono dare vita a valori anomali identificando così un possibile evento illegale

1.3 I diversi tipi di Anomalie

In alcune applicazioni come quello del rilevamento di frodi, i valori anomali corrispondono a sequenze di più punti dati anziché a singoli punti dati. Ad esempio, un evento di frode può spesso riflettere le azioni di un individuo in una particolare sequenza temporale. La specificità della sequenza è rilevante per identificare l'evento anomalo. Tali anomalie sono anche denominate **anomalie collettive**, poiché possono essere dedotte solo collettivamente ovvero osservando un insieme. Tali anomalie spesso sono il risultato di eventi insoliti che generano modelli di attività anomale.
In generale possiamo dire che le anomalie sono di tre tipi:

1. **Anomalie puntiformi**: se una singola istanza dei dati può essere considerata anomala rispetto al resto dei dati (es. pagamento con carta di credito di una somma molto superiore rispetto alle restanti transizioni);

2. **Anomalie contestuali**: se un gruppo di dati è anomalo in un contesto specifico, ma non altrimenti (anomalia se si verifica in un determinato momento o in una determinata regione, ad esempio un forte picco nel cuore della notte);

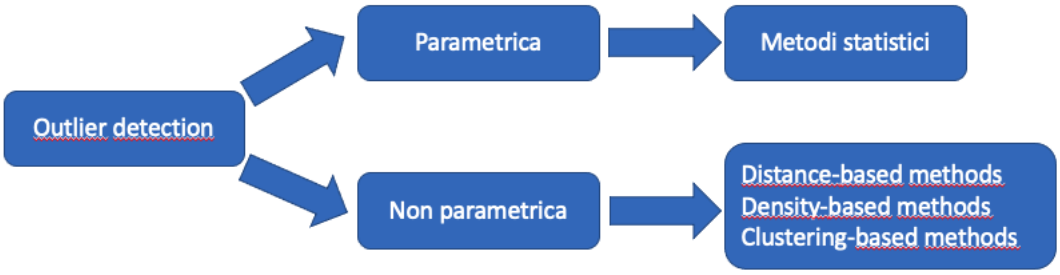
3. **Anomalie collettive**: Se una raccolta di istanze di dati correlati è anomala rispetto all'intero set di dati, ma non ai singoli valori. Hanno due varianti: **eventi in ordine imprevisto** (ordinati, ad es. interruzione del ritmo nell'ECG); **combinazioni di valori imprevisti** come l'acquisto di un numero elevato di articoli costosi;

1.4 Metodi di Outlier Detection

Sono stati ideati numerosi metodi algoritmici per rilevare valori anomali nei dati nel tempo. Tuttavia, nessun singolo metodo rappresenta l'applicabilità universale a causa della variabilità nel contesto dell'applicazione. Quindi, a seconda del requisito specifico dell'applicazione, viene considerata una strategia di rilevamento adeguata tra le opzioni disponibili. Le tecniche di rilevamento degli outlier possono essere ampiamente suddivise in varietà parametriche e non parametriche, come mostrato in Fig. 1.1.

I metodi basati sulla statistica che assumono un modello per i dati appartengono alla varietà parametrica. Tipicamente, è necessario modellare i dati utilizzando una distribuzione statistica e gli oggetti dati sono determinati

Figura 1.1: Tassonomia dei metodi di rilevamento degli outlier



come valori anomali a seconda di come appaiono in relazione al modello postulato. D'altra parte, la maggior parte dei metodi non parametrici si basa su una nozione ben definita di distanza per misurare la separazione tra una coppia di oggetti dati. La varietà non parametrica include:

1. **Distance-based**: la distanza di un punto dati dal suo vicino più vicino (o altra variante) viene utilizzata per definire la prossimità. I punti dati con grandi distanze k-vicini sono definiti come valori anomali. Gli algoritmi basati sulla distanza in genere eseguono l'analisi con una granularità molto più dettagliata rispetto agli altri due metodi. D'altra parte, questa maggiore granularità ha spesso un costo computazionale significativo.

2. **Density-based**: il numero di altri punti all'interno di una regione locale specificata (regione della griglia o regione basata sulla distanza) di un punto dati, viene utilizzato per definire la densità locale. Questi valori di densità locale possono essere convertiti in punteggi anomali. Possono essere utilizzati anche altri metodi basati sul kernel o metodi statistici per la stima della densità. La principale differenza tra i metodi di clustering e quelli basati sulla densità è che i metodi di clustering suddividono i punti dati, mentre i metodi basati sulla densità suddividono lo spazio dati. Un esempio potrebbe essere l'algoritmo di DBSCAN.

3. **Clustering-based**: la non appartenenza di un punto dati in uno qualsiasi dei cluster, la sua distanza da altri cluster, la dimensione del cluster più vicino o una combinazione di questi fattori vengono utilizzati per quantificare il punteggio outlier. Il problema del clustering ha una relazione complementare al problema di rilevamento degli outlier in cui i punti appartengono a cluster o dovrebbero essere considerati outlier.

Tra i metodi statistici, invece, possiamo citare il modello dei boxplot molto semplice ed intuitivo oppure modelli che si basano sulla statistica e che vengono annoverati tra gli algoritmi di Time Series Forecasting come gli ARIMA oppure il modello Prophet (che vedremo nei capitoli successivi).

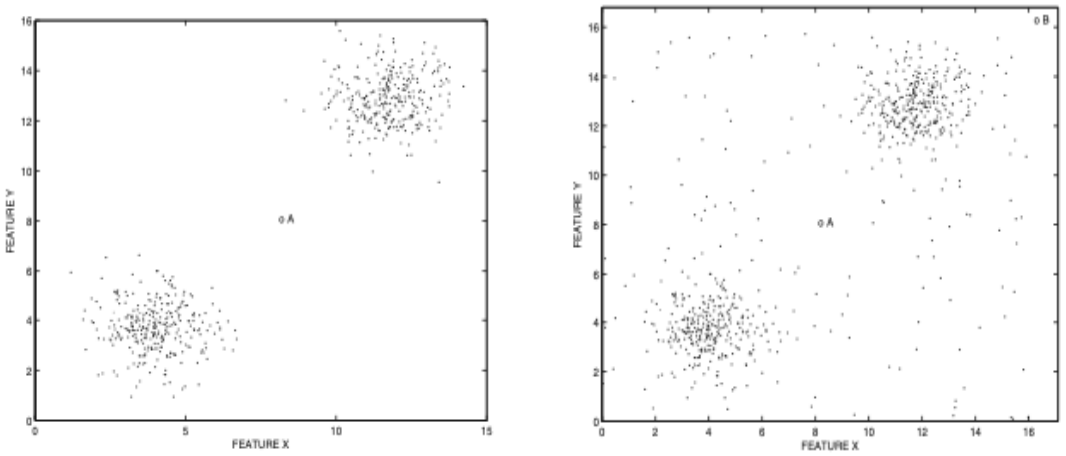
1.5 Cause delle anomalie

In generale, i motivi per i quali in un set di dati ci possono essere delle anomalie sono diversi. Come prima cosa si può pensare ad un **errore di misurazione** e come esempio si può pensare a un errore umano oppure un errore dovuto ad un dispositivo (es. sensore della temperatura). Un'altra causa molto frequente è quando i dati appartengono a **classi differenti**. Un esempio classico, riportato anche in precedenza, è quando un truffatore ruba o clona una carta di credito. In questo caso la persona che compie l'atto fraudolento eseguirà degli acquisti di importi differenti e in tempi differenti da quelli del legittimo proprietario (una ogni cinque minuti piuttosto che una ogni cinque ore).
I due esempi riportati precedentemente possono avere, nel contesto di studio e analisi dei dati, due obiettivi differenti: il primo è quello di non considerare i valori anomali, ovvero gli errori commessi da un umano perché non apportano valore significativo all'analisi o all'algoritmo di predizione. Questo è il cosiddetto **rumore** che desideriamo rimuovere. Il secondo approccio invece potrebbe essere quello di usare e ricercare i valori anomali per finalità di studio, capirne le cause e predirre quando si potrebbero ripresentare in correlazione con il tempo. In questo contesto la ricerca di outlier è l'obiettivo finale del lavoro.

1.6 Differenza tra anomalia e rumore

Un concetto che in questo campo è fondamentale è quello di **rumore**. Se consideriamo le immagini di 1.2 notiamo che i dati grossomodo sono gli stessi ed i cluster più evidenti pure, però ci sono delle differenze abbastanza importanti all'esterno di questi due cluster.

Nel caso della 1.2a, un singolo punto di dati (contrassegnato da 'A') sembra essere molto diverso dai restanti dati, ed è quindi molto chiaramente un'anomalia. Mentre il corrispondente punto di dati 'A' nella figura 1.2b si trova anch'esso in una regione rada dei dati, è molto più difficile affermare con sicurezza che esso rappresenta una vera deviazione dal resto dei dati. È



(a) No rumore (b) Si rumore

Figura 1.2: Differenza tra rumore e non rumore

molto probabile che questo punto di dati rappresenti un rumore distribuito casualmente nei dati. Questo perché il punto 'A' sembra adattarsi a un modello rappresentato da altri punti distribuiti casualmente. Nello scenario non supervisionato, che è quello di nostro interesse in questo lavoro, dove non sono disponibili esempi precedenti di anomalie interessanti, il rumore rappresenta il confine semantico tra i dati normali e le vere anomalie - il rumore è spesso modellato come una forma debole di outlier che non sempre soddisfa i criteri forti necessari perché un punto dati sia considerato abbastanza interessante o anomalo.

1.7 Confronto tra modelli supervised e unsupervised

Si può considerare il problema del rilevamento degli outlier come una variante del problema della classificazione in cui l'etichetta della classe ("normale" o "anomalia") non è osservata. Quindi, in virtù del fatto che gli esempi normali superano di gran lunga gli esempi anomali, si può "fingere" che l'intero set di dati contenga la classe normale e creare un modello (eventualmente rumoroso) dei dati normali. Le deviazioni dal modello normale sono trattate come punteggi outlier. Questa connessione tra classificazione e individuazione degli outlier è importante perché gran parte della teoria e dei metodi di classificazione si generalizzano all'individuazione degli outlier.

Le tecniche supervisionate di rilevamento degli outlier sono in genere molto più efficaci in molti scenari specifici dell'applicazione, perché le caratteristiche degli esempi precedenti possono essere utilizzate per affinare il processo di ricerca verso outlier più rilevanti.
In generale, i metodi non supervisionati possono essere utilizzati sia per la rimozione del rumore che per il rilevamento delle anomalie, mentre i metodi supervisionati sono progettati per il rilevamento di anomalie specifiche dell'applicazione. I metodi non supervisionati sono spesso utilizzati in un contesto esplorativo, dove i valori anomali scoperti sono forniti all'analista per un ulteriore esame della loro importanza specifica per l'applicazione.

Capitolo 2

Descrizione del problema

2.1 Inquadramento generale

Il problema di Outlier Detection in questione viene svolto all'interno di un contesto aziendale che si occupa di compravendita di metalli. Essi di cui si discuterà sono tre: alluminio (AL), piombo (PB) e rame (CU).
Gli ordini per questi metalli, in generale, possono essere di tre tipologie:

- **Ordine di acquisto**: l'azienda con codice identificativo dato dalla Legal Entity: 1234 intende acquistare una certa quantità (DAILY_QTY) di metallo (NF_KEY)
- **Ordine di vendita**: l'azienda con codice identificativo dato dalla Legal Entity: 6789 intende vendere una certa quantità (DAILY_QTY) di metallo (NF_KEY)
- **Ordine di coverage**

Ogni ordine è composto da una moltitudine di line-item e ognuno di essi contiene la quantità che una certa azienda intende acquistare o vendere per un certo metallo, l'identificativo dell'ordine, la data di elaborazione, etc. L'azienda per valutare se l'ordine che giunge quotidianamente è valido o meno si affida ad un processo piuttosto complesso che si andrà a declinare di seguito. In dettaglio, ogni mattina l'azienda riceve un **Metal Book** da un sistema automatizzato (un robot Appian). Questo Metal Book è composto da una serie di ordini, quindi di line-item, che possono essere di acquisto o di vendita per una determinata azienda con una specifica quantità (DAILY_QTY). Tutti questi ordini passano attraverso dei check che mettono insieme una serie di line-item sommando opportunamente di volta in volta la quantità del

metallo DAILY_QTY. Questo totale sarà confrontato con una soglia (ora impostata statica) dove sono possibili due risultati: soglia maggiore della quantità complessiva e in questo caso l'ordine viene accettato perché rientra nel valore stabilito; viceversa l'ordine viene rifiutato perché fuori soglia.
Come si è già accennato di sopra, la soglia allo stato originale era calcolata da personale specializzato (che in questo contesto chiameremo Metals Manager) dell'azienda in maniera statistica partendo da ipotesi elementari e delle volte poco efficienti. I calcoli effettuati dai Metals Manager risultavano essere, sin dai primi studi, molto approssimativi in quanto la soglia, molto di frequente, risultava essere di molto sovrastimata o di molto sottostimata. Questo comportava un dispendioso lavoro a valle del personale perché manualmente gli ordini dovevano poi essere corretti e opportunamente sbloccati o bloccati.
Queste soglie, come si è già accennato poco fa, vengono tuttora stimate sulla base del dato DAILY_QTY che rappresenta le quantità in tonnellate di metallo che ogni giorno può essere venduto o acquistato.
In generale, quindi, si può affermare che la grossa limitazione di tale processo era rappresentata dal fatto che la soglia essendo statica aveva bisogno quotidianamente dell'intervento umano per sbloccare o bloccare eventuali ordini che venivano erroneamente classificati dal robot.
A tal proposito, l'obiettivo è stato quello di rendere questa soglia **dinamica** per ridurre al minimo l'intervento umano identificando quelli che sono gli ordini fuori soglia (outlier) e anche suggerendo ai Metals Manager, di settimana in settimana, quali potevano essere le soglie che si potevano aspettare in modo da arrivare preparati per la valutazione degli ordini. In Figura 2.1 si può osservare in maggiore dettaglio il processo appena descritto.

2.2 Dataset utilizzati

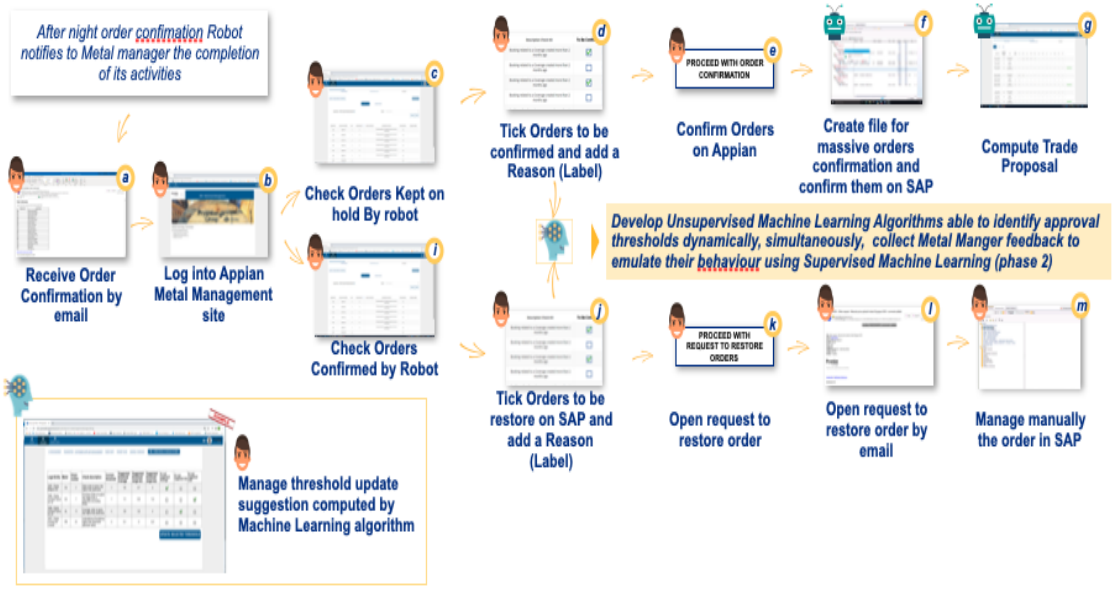
I dataset che sono stati utilizzati per l'analisi e la risoluzione del problema sono sostanzialmente due: il primo che contiene tutti gli ordini; il secondo che è formato da tutti gli ordini bloccati dal robot.

Il primo dataset, che qui chiameremo *ordini_totali*, è caratterizzato dai campi presenti in Tabella 2.1.

Invece, il secondo dataset, che chiameremo *totale_ordini_bloccati*, sarà composto dagli attributi presenti in Tabella 2.2.

Infine si ha a disposizione un'ulteriore tabella che è stata utile quando si

Figura 2.1: Processo di valutazione ordini



Attributo	Descrizione
ID	Identificativo univoco incrementale di ogni line item
NAME	Nome
BOOKING_FR	Data di prenotazione dell'ordine
ORDER_DATE	Data effettiva dell'ordine
DAILY_QTY	Quantità dell'ordine
COUNTRY_NAME	Nome dell'azienda che fa vendita/acquisto/coverage
NF_KEY	Nome del metallo
DOC_NUM	Nome univoco dell'ordine
COMPANY_CODE	Identificativo univoco della compagnia
WORKED_DATE	Data di lavorazione

Tabella 2.1: Dataset ordini totali

è voluto fare un confronto tra le soglie predette dall'algoritmo di intelligenza artificiale e quelle statiche impostate manualmente dai Metals Manager. Questa tabella prende il nome di *df_actual_thresholds* ed è composta dagli attributi visibili nella Tabella 2.3.

Attributo	Descrizione
ID	Identificativo univoco incrementale di ogni line item
CHECK_ID	Check ID che ha bloccato l'ordine in questione
ORDER_NUMBER	Numero di ordine
WORKED_DATE	Data di lavorazione

Tabella 2.2: Dataset ordini totali bloccati

Attributo	Descrizione
ID	Identificativo univoco incrementale di ogni line item
CHECK_ID	Check di riferimento
COMPANY_CODE	Identificativo della compagnia di riferimento
METAL	Nome del metallo
THRESHOLD	Soglia impostata dal Metal Manager (statica)

Tabella 2.3: Dataset soglie statiche

2.3 Verifiche del dominio applicativo

Il processo decisionale che porta il robot a scegliere se un ordine deve essere accettato o meno è costituito da una serie di check di verifica. In questo contesto, per ogni Check verrà valutata la soglia dinamica e anche identificati eventuali Outliers. Nella Tabella 2.4 si può vedere uno per uno la descrizione dei check.

Tabella 2.4: Descrizione dei Check

Descrizione dei Check	
ID	Descrizione
1	Nel caso in cui il volume complessivo degli ordini di vendita in una data giornata risulti maggiore di un certo parametro, tutti gli ordini di vendita della giornata rimangono on-hold all'interno del metal book
2	Nel caso in cui il volume di un ordine di vendita (stesso DOC NUM) risulti maggiore di un certo parametro, detto ordine rimane on-hold all'interno del metal book
3	Nel caso in cui il volume complessivo degli ordini di vendita in una data giornata risulti minore di un certo parametro, tutti gli ordini di vendita della giornata rimangono on-hold all'interno del metal book

5	Nel caso in cui gli ordini di vendita abbiano come controparte un cliente presente all'interno di un elenco clienti ad hoc e il volume di tali ordini è superiore ad una soglia predefinita, detti ordini di vendita rimangono on hold metal book
6	Nel caso in cui il volume complessivo degli ordini di acquisto in una data giornata risulti maggiore di un certo parametro, tutti gli ordini di acquisto della giornata rimangono on-hold all'interno del metal book
7	Nel caso in cui il volume di un ordine di acquisto (stesso "Doc. Num.") diverso da un ordine di acquisto di vergella ("Doc. Num.": 27xxxxxx-xx) risulti maggiore di un certo parametro, detto ordine rimane on-hold all'interno del metal book
8	Nel caso in cui il volume complessivo degli ordini di acquisto in una data giornata risulti minore di un certo parametro, tutti gli ordini di acquisto della giornata rimangon on-hold all'interno del metal book
10	Nel caso in cui gli ordini di vendita abbiano come controparte un fornitore presente all'interno di un elenco fornitori ad hoc e il volume di tali ordini è superiore ad una soglia predefinita, detti ordini di acquisto rimangono on hold metal book
12	Nel caso in cui il volume complessivo degli ordini di coverage in una data giornata risulti maggiore di un certo parametro, tutti gli ordini di coverage della giornata rimangono on-hold all'interno del metal book
13	Nel caso in cui il volume di un ordine di coverage (stesso "Doc. Num.") risulti maggiore di un certo parametro, detto ordine rimane on-hold all'interno del metal book
16	Nel caso in cui il volume complessivo degli ordini di coverage in una data giornata risulti minore di un certo parametro, tutti gli ordini di coverage della giornata rimangono on-hold all'interno del metal book
17	Nel caso di debooking di un ordine di acquisto maggiore rispetto ad una soglia di tolleranza (calcolata in rapporto al totale della quantità ordinata), detto ordine rimane on-hold all'interno del metal book
18	Nel caso di debooking di un ordine di vendita maggiore rispetto ad una quantità limite predefinita, detto ordine rimane on-hold all'interno del metal book
20	Nel caso in cui ad un ordine di acquisto è associato un RDK antecedente a 7 giorni e con una quantità superiore ad una data soglia, detto ordine di acquisto rimane on-hold all'interno del metal book

21	Nel caso in cui ad un ordine di vendita è associato un RDK antecedente a 7 giorni e con una quantità superiore ad una data soglia, detto ordine di vendita rimane on-hold all'interno del metal book
22	Nel caso in cui il volume di un singolo item di un ordine (acquisto/vendita/coverage) risulti in overdelivery o in underdelivery, rispetto ad un certo parametro in valore assoluto, detto ordine rimane on-hold all'interno del metal book
Fine descrizione dei Check	

Capitolo 3

Strumenti

3.1 Connessione macchina virtuale

Il lavoro è stato svolto da remoto utilizzando il software **VMWareHorizon**, un tool che è in grado di offrire desktop virtuali. Lavorare direttamente su macchina virtuale è stato necessario per vari motivi, primo tra tutti la possibilità di aggiornare ogni giorno i dati del dataset grazie a interrogazioni SQL su database Oracle. Questo ha consentito di poter effettivamente provare lo script quotidianamente con ordini sempre differenti e verificare effettivamente la dinamicità delle soglie.

3.2 Tool e librerie utilizzate

Per ciò che riguarda l'analisi e il lavoro vero e proprio, il principale tool utilizzato è stato **Anaconda**. All'interno di questo software si sono creati differenti environments (ambienti) in cui provare e installare le differenti librerie che si sono ritenute necessarie. Inoltre, proprio da questa interfaccia, si è utilizzato per la programmazione Python l'IDE **Spyder**.
Le librerie maggiormente utilizzate sono state:

- **cx_oracle**: libreria utilizzata per stabilire la connessione con il database. Viene istanziata una connessione con gli ambienti di "produzione" e "sviluppo". Tramite query SQL vengono importate tutte le tabelle utili per la manipolazione e l'analisi dei dati

- **seaborn**: libreria grafica

- **pandas**: manipolazione e creazione strutture dati
- **numpy**: manipolazioni di matrici e statistiche dei dati
- **JSON**: libreria usata per importare file in formato JSON (pre configurazione di alcune variabili statiche)
- **PyCaret**: si discuterà in maniera più approfondita nel Capitolo 7.1

Le sopracitate librerie rappresentano quelle che vengono usate in larga parte in tutto il codice di produzione ma non rappresentano la totalità. Specialmente nei capitoli a seguire, per ogni tecnica che è stata provata e testata sono state importate librerie ad hoc per ogni specifico caso.

3.3 Schedulazione dello script

Un aspetto molto importante al fine di mandare in produzione il codice e verificarne l'effettiva correttezza dei risultati è stata la schedulazione del processo finale. Grazie al software Microsoft **Task Scheduler** si è potuto runnare il codice impostando manualmente lo slot temporale e la data entro cui far eseguire lo script.

Parte II

Soluzioni proposte

Capitolo 4

Tecniche proposte di Outlier Detection

4.1 Introduzione

Esistono numerose tecniche algoritmiche che affrontano il problema del rilevamento dei valori anomali. Alla base troviamo i metodi statistici secondo cui gli outlier sono osservazioni che sembrano essere statisticamente incoerenti con il resto dei dati. In genere, i metodi statistici sono di natura parametrica e presuppongono una distribuzione sottostante nota o un modello statistico dei dati. Di conseguenza, gli outlier risultano essere gli oggetti che hanno una bassa probabilità di appartenere al modello sottostante.
Ci sono due considerazioni essenziali per selezionare una metodologia accettabile per il rilevamento di outlier:

- selezionare un algoritmo in grado di modellare accuratamente la distribuzione dei dati;
- selezionare un set di interesse adatto per caratterizzare un outlier.

I metodi per l'outlier detection possono essere suddivisi nelle seguenti tre categorie in base al tipo di tecniche di modellazione dei dati da essi utilizzate:

- **Metodi statistici**: questi metodi sono i primi ad essere utilizzati per il rilevamento dei valori anomali. Molte delle tecniche statistiche si basano su un modello monodimensionale o al massimo univariato dei dati, mentre la maggior parte delle applicazioni recenti richiedono la modellazione di dati multidimensionali per ottenere capacità di rilevamento anomalie;

- **Metodi basati su reti neurali**: sono generalmente non parametrici e generalizzano bene a modelli invisibili e sono in grado di apprendere complessi confini di classe. Tuttavia il set di dati deve essere attraversato numerose volte per consentire alla rete di convergere e modellare correttamente i dati;
- **Metodi basati sull'apprendimento automatico**: la maggior parte dei metodi statistici e neurali richiede il calcolo delle distanze vettoriali sui dati e non disponde di un meccanismo per elaborare i dati categorici. Gli alberi decisionali vengono utilizzati per rilevare valori anomali nei dati categoriali in quanto non richiedono alcuna conoscenza preliminare dei dati. Altri metodi invece sono quelli che si basano su sistemi di regole che sono simili agli alberi decisionali.

Capitolo 5

Analisi preliminare

In questo capitolo si va ad affrontare il problema in questione con una tecnica preliminare, semplice ed intuitiva come quella del boxplot.

5.1 Boxplot

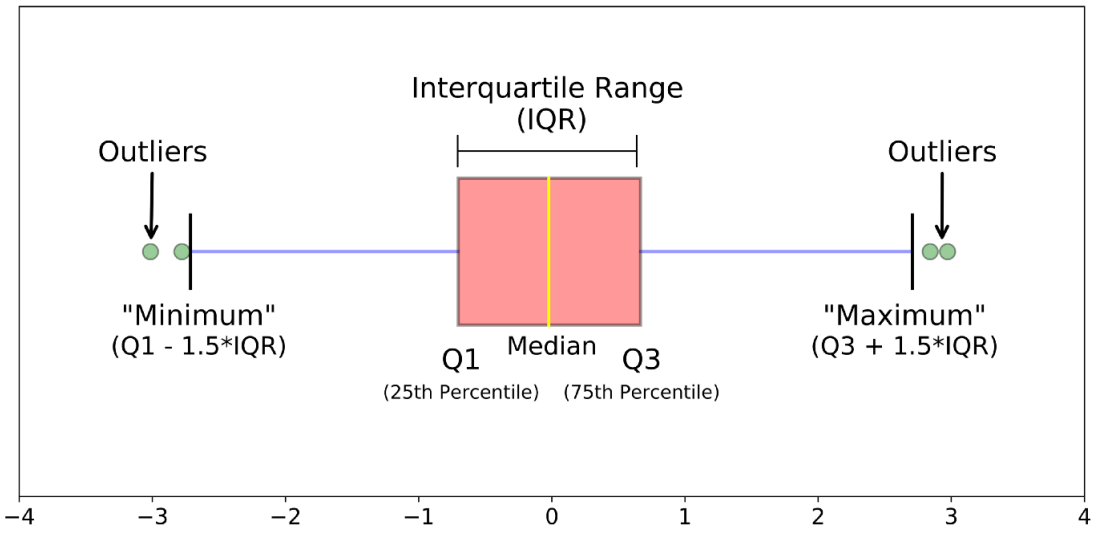
Il metodo dei boxplot consente di determinare gli outlier producendo un grafico (boxplot) che matematicamente è possibile interpretare per trarne utili informazioni.
Esso rappresenta i dati numerici attraverso la loro distribuzione in diversi quartili, come mostrato in Figura 5.1. Tale grafico è anche noto come diagramma **box and whisker**, poiché può avere linee che si estendono verticalmente dalle caselle (baffi) che indicano la possibile variabilità al di fuori dei quartili superiore e inferiore.

Attraverso un esame visivo dei dati utilizzando il boxplot è possibile determinare l'intervallo interquartile (IQR) e altri dati statistici i quartili o la mediana. In generale, in un diagramma a scatola, le parti inferiore e superiore corrispondono rispettivamente al primo e al terzo quartile e la banda di divisione all'interno del riquadro è il secondo quartile (la mediana).
Le estremità dei baffi possono indicare diverse possibilità tra le seguenti:

- valori minimo e massimo dei dati
- deviazione standard sopra e sotto la media dei dati

In base a ciò che abbiamo detto, quindi, qualsiasi elemento di dati non incluso nell'intervallo di valori coperto dai baffi viene identificato come outlier

Figura 5.1: Struttura di un boxplot. Fonte: https://towardsdatascience.com/understanding-boxplots-5e2df7bcbd51



e nel caso della Figura 5.1 è mostrata con il simbolo dei pallini verdi.

Si possono riassumere le varie dimensioni del boxplot come segue:

- **Mediana (Q2/50th Percentile)**: valore medio dei dati
- **Primo quartile (Q1/25th Percentile)**: il numero medio tra il numero più piccolo (non il "minimo") e la mediana del set di dati.
- **Terzo quartile (Q3/75th Percentile)**: il valore medio tra la mediana e il valore più alto (non il "massimo") del set di dati.
- **Range Interquartile (IQR)**: dal 25th al 75th percentile.
- **Whiskers (baffi)**: sono rappresentati in blu.
- **Maximum**: Q3 + 1.5*IQR.
- **Minimum**: Q1 - 1.5*IQR.

5.2 Applicazione dei boxplot sui dati

La soluzione descritta nella precedente sezione è molto utile in un contesto iniziale per "smussare" il problema di identificazione dei valori anomali e fornire una soluzione in tempi brevi ed allo stesso tempo efficiente.

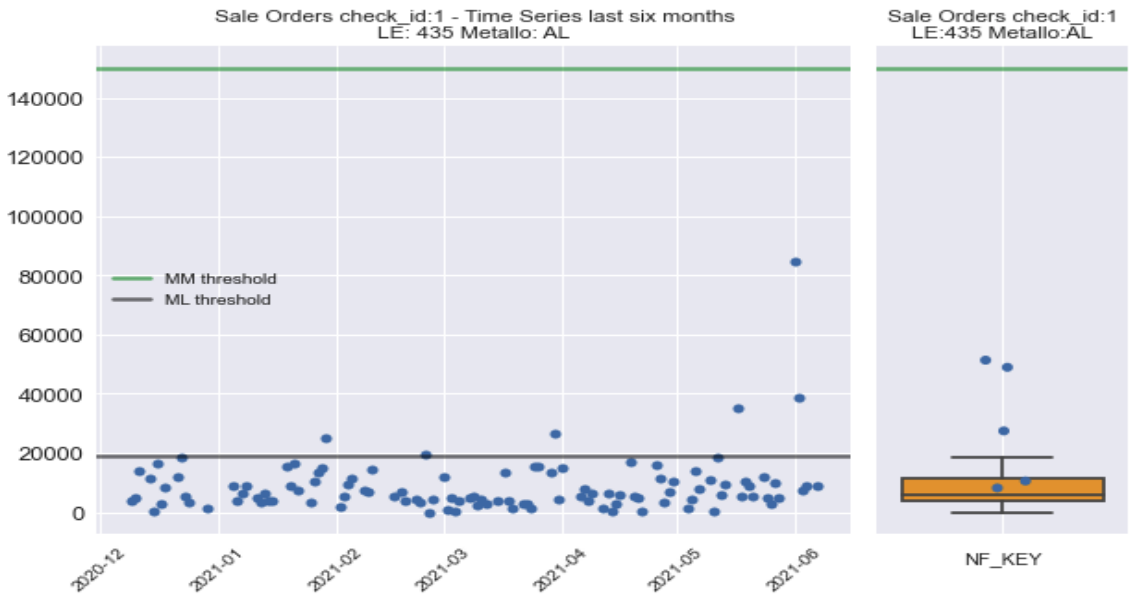
Nel caso in questione, come già si è anticipato nei capitoli iniziali, si hanno degli ordini identificati da diversi attributi (la data di creazione dell'ordine, la quantità, il metallo, etc.) nel corso del tempo e ciò che in questa prima fase è stato fatto è stato analizzare singolarmente ogni check per trovare una soglia dinamica da applicare a ciascun ordine.

In Figura 5.2 si è analizzato il Check 1 (per la definizione rimando alla Tabella 2.4), Legal Entity 435, metallo Alluminio e un ordine è di tipo "Sale" (vendita). Entrambi i grafici hanno sull'asse delle ascisse il tempo rappresentato nel formato *mese-anno* e poi sull'asse delle ordinate c'è la quantità predetta in quintali ($DAILY\_QTY$) del metallo.

A seguito di confronti, per i check che richiedono una soglia del tipo "non superiore a" (es. Check 1) si è deciso, di comune accordo, di scegliere il percentile 95-esimo mentre, per i check che richiedono una soglia del tipo "non inferiore a" (es. Check 3) si è deciso di scegliere il primo percentile.
Il grafico, da come è possibile vedere nella stessa figura, è diviso in due: sulla sinistra ci sono dei pallini blu che indicano gli ordini effettuati durante l'arco temporale di sei mesi (da Dicembre 2020 a Giugno 2021), poi ci sono due linee che sono di colore verde e nero che rispettivamente indicano la soglia impostata dal Metal Manager (soglia statica) e la soglia predetta dal boxplot (soglia dinamica); sulla destra invece troviamo il boxplot in cui all'interno ci sono dei pallini blu che indicano gli ordini di test, ovvero gli ordini nella settimana che intendiamo analizzare per predirre se sono ordini che vanno accettati oppure ordini che sono fuori soglia e quindi rigettati. Il baffo più in alto indica la soglia dinamica aggiornata settimana per settimana.

In Figura 5.2 è chiaro che la soglia impostata dai Metals Manager è molto al di sopra di quella che potrebbe essere una soglia realistica, da casi come questi nasce l'importanza e l'esigenza di analisi statistiche e predittive. In questo caso si può notare che il numero di ordini fuori soglia, che nella settimana di test non passerebbero è soltanto uno e la soglia è impostata poco al di sotto delle 20 tonnellate.
In Tabella 5.1 possiamo vedere un risultato parziale dei primi calcoli effettuati per semplicità solo per il check 1, per gli altri check ci si muove in maniera

Figura 5.2: Boxplot: Sale Orders Check ID: 1, LE: 0435, Metallo: AL percentile 95%

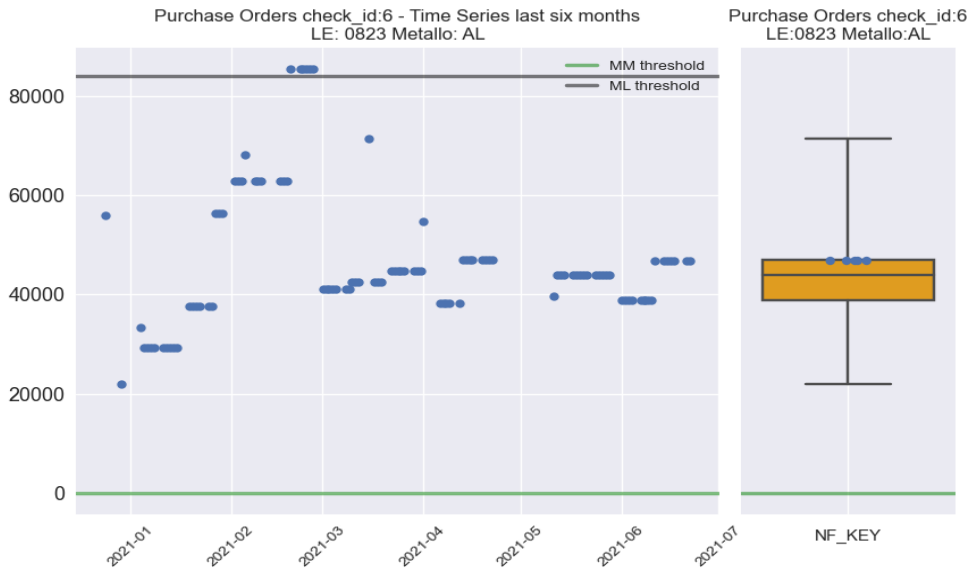


analoga.

Un ulteriore esempio è rappresentato dalla Figura 5.3 in cui questa volta si ha il Check numero 6 che recita: "nel caso in cui il volume complessivo degli ordini di acquisto in una data giornata risulti maggiore di un certo parametro, tutti gli ordini diacquisto della giornata rimangono on-hold all'interno del metal book". Il parametro citato dal testo del check, questa volta calcolato anche dinamicamente risulta essere pari a più di 80 tonnellate, precisamente 85.41 tonnellate, mentre quello precedentemente impostato in maniera statica era pari a 0. In questo caso, a differenza del precedente, tutti gli ordini che arrivano nella settimana di test sarebbero accettati perché rispettano la soglia dinamica (in quanto inferiori di quantità) mentre per i Metals Manager gli stessi ordini sarebbero bloccati perché tutti superiori alla soglia.

Altro esempio, questa volta con check di natura differente come ad esempio il Check numero 3 che recita: "nel caso in cui il volume complessivo degli ordini di vendita in una data giornata risulti minore di un certo parametro, tutti gli ordini divendita della giornata rimangono on-hold all'interno del metal book". Questa volta viene chiesto di trovare la soglia al contrario di quanto fatto ad esempio con i check 1 e 3. Un esempio lo possiamo vedere in Figura 5.4. Anche questa volta la soglia impostata statica è 0 tonnellate

Figura 5.3: Boxplot: Purchase Orders Check ID: 6, LE: 823, Metallo: AL percentile 95%



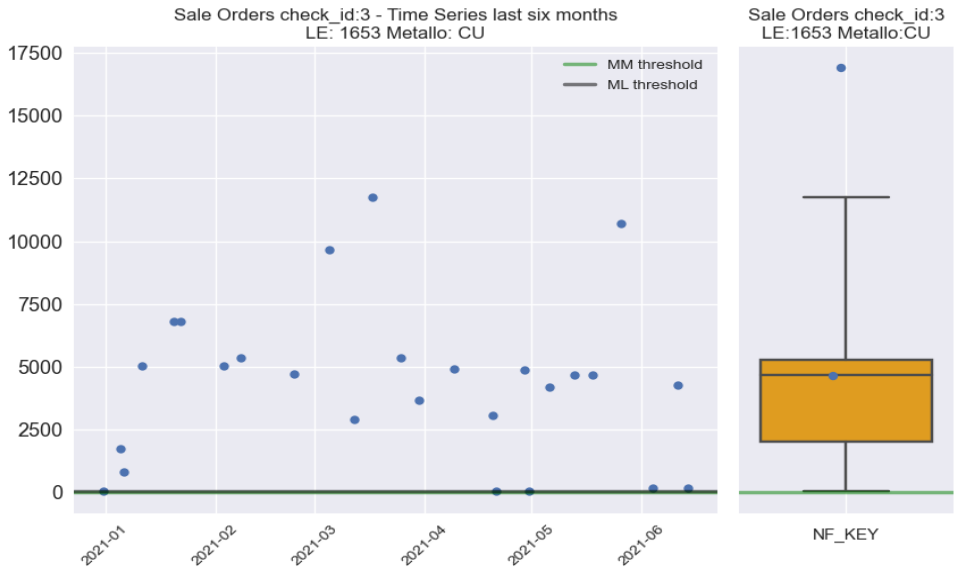
mentre quella ricavata dinamicamente risulta essere di 10.57 tonnellate.

ID_CHECK	COMPANY_CODE	METAL	TH_BOXPLOT	TH_MM
1	435	AL	19.08	150
1	435	CU	185.97	250
1	435	PB	19.41	100
1	1919	CU	73.67	200
1	1948	AL	27.52	100
6	823	AL	85.41	0
3	1653	CU	10.57	0
...	...	...	...	...

Tabella 5.1: Confronto soglie Metal Manager e soglie predette tramite boxplot

Come è facile constatare dalla parziale tabella riportata, le soglie sovrastimate e sottostimate (per gli altri check) sono molte e rappresentano un problema di efficienza e costi per l'azienda.

Figura 5.4: Boxplot: Purchase Orders Check ID: 3, LE: 1653, Metallo: CU percentile 1%



Capitolo 6

Analisi di Time Series Forecasting

6.1 Definizione di Time Series

Una serie temporale è un insieme sequenziale di punti dati misurati in genere su tempi successivi.
Una serie temporale contenente i valori di una singola variabile è definita **univariata**, ma, se si considerano i valori di più di una variabile, si parla di serie **multivariata**. Una serie temporale inoltre può essere di due tipi: continua e discreta. Una serie temporale si dice **continua** quando le osservazioni sono misurate ad ogni istante di tempo (ad esempio lettura della temperatura, misura della velocità di una automobile, etc.); una serie temporale invece si chiama **discreta** se le osservazioni sono misurate in punti temporali discreti (ad esempio popolazione di una città, tassi di cambio, valore economico di una azienda, etc.).
Di solito una buona rappresentazione di una serie storica prevede che sull'asse delle ascisse vi siano i giorni, quindi la serie temporale considerata, mentre sull'asse delle ordinate ci sono le quantità, o meglio i valori numerici che si andranno a valutare e prevedere nel periodo di interesse.

6.2 Casi di studio delle Time Series

Le Time Series analizzano e trovano applicazione in due grandi aree:

- Caratterizzano la natura del fenomeno rappresentato dalla sequenza di osservazioni

- Attività di classificazione rispetto al forecasting (previsione dei valori futuri). Per citare alcuni esempi e rendere il concetto maggiormente chiaro possiamo pensare ai problemi di classificazione per il riconoscimento vocale, oppure la previsione di un guasto, mentre per il forecasting possiamo pensare di fare previsioni sulla domanda di energia che un determinato quartiere di una città richiederà in un preciso periodo di tempo, oppure una previsione del traffico.

6.3 Componenti delle Time Series

Lo studio delle serie storiche ha un duplice obiettivo: il primo è quello di analizzare quantitativamente la richiesta; il secondo ha come obiettivo l'eliminazione dei valori anomali riscontrati durante tutta la serie storica e che quindi si discostano dagli altri valori.

Le serie storiche sono caratterizzate da quattro componenti prinicpali che sono:

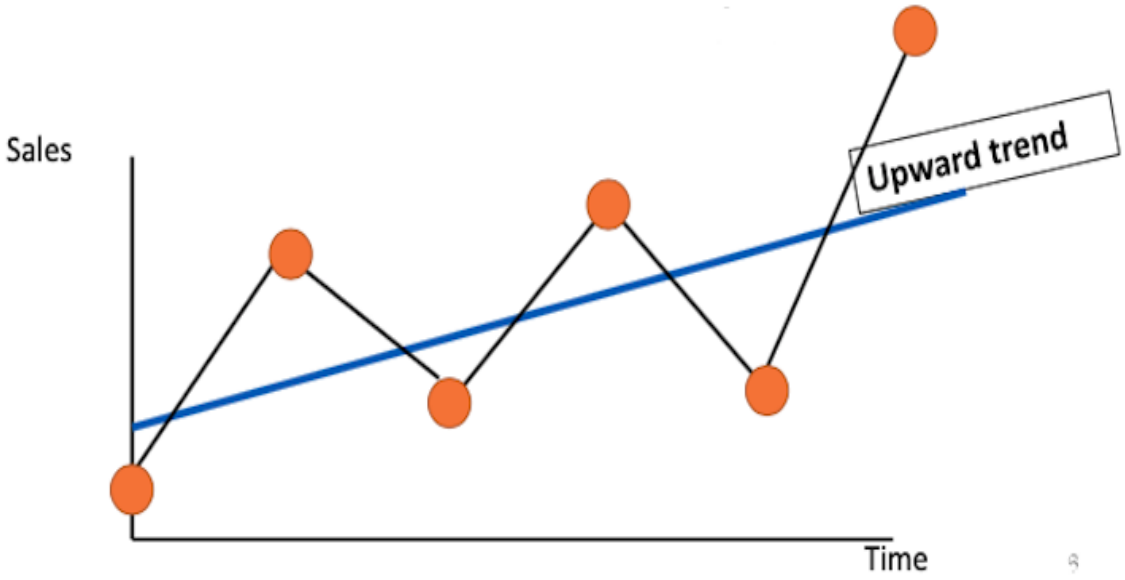
- **Trend**: Figura 6.1 - descrive la tendenza di fondo del fenomeno oggetto di studio, riferita ad un lungo periodo di tempo. Si è soliti ipotizzare che i valori del trend siano esprimibili tramite una funzione sufficientemente regolare e liscia (smooth), ad esempio funzioni polinomiali o esponenziali. Se si assume che la funzione da utilizzare sia monotona, allora il trend può essere una funzione crescente o decrescente del fenomeno nel periodo analizzato

- **Stagionalità**: Figura 6.2 - movimenti del fenomeno nel corso dell'anno che per effetto dell'influenza di fattori vari (ad esempio climatici, sociali, etc.), tendono a ripetersi in maniera pressoché analoga nel medesimo periodo (mese o trimestre) di anni successivi

- **Ciclicità**: Figura 6.3 - fluttuazioni ovvero movimenti alternati verso l'alto e verso il basso. La differenza rispetto alla stagionalità è che dal punto di vista numerico è uguale alla ciclicità, è il fatto che la prima è nell'ordine degli anni per i cicli di lungo periodo, mentre la seconda dell'ordine dei mesi o al più un anno

- **Componente Casuale (rumore)**: rappresenta il rumore di fondo della serie storica, cioè la componente di domanda non prevedibile, data dalla fluttuazione casuale dei valori di domanda attorno al valor medio della serie. La fluttuazione casuale viene rilevata dopo aver rimosso le

Figura 6.1: Componente Trend, Fonte: Corso di Laurea Magistrale Politecnico di Torino materia Data Science Lab A.A. 2020/2021

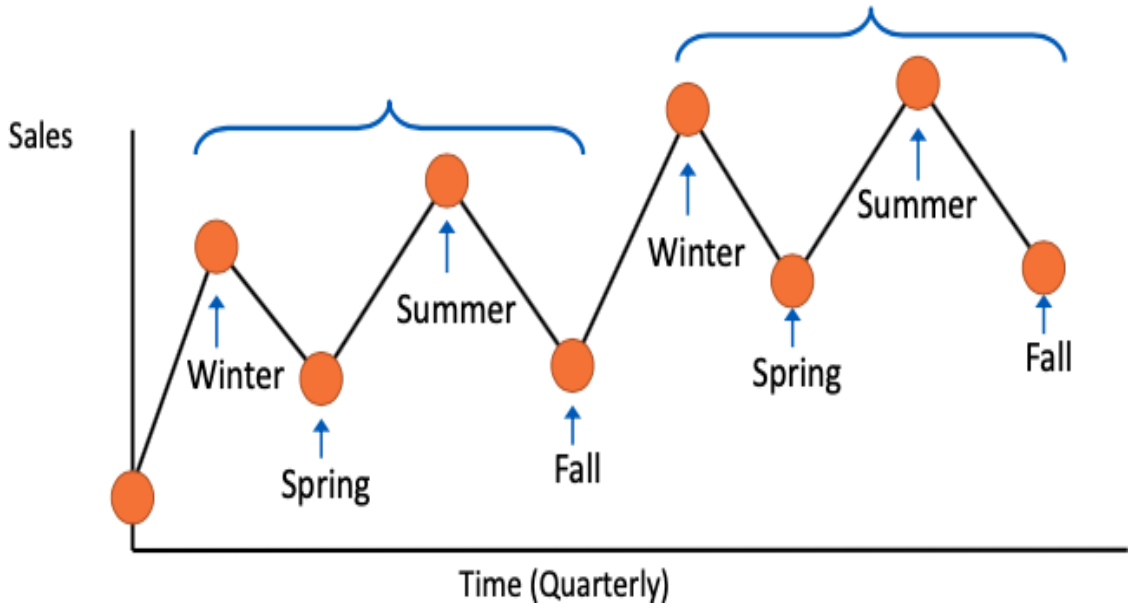


tre componenti regolari della serie storica, avendo cioè isolato la domanda media. Va evidenziato il fatto che la componente casuale non è statisticamente prevedibile; tuttavia se risulta essere numericamente rilevante, è possibile applicare modelli di regressione lineare nei quali vengono testate diverse variabili indipendenti sulla serie storica con la sola componente casuale. Questo è utile per scovare correlazioni tra componente casuale e variabili di input misurabili (per le quali è disponibile anche la previsione per i valori futuri).

6.4 ARIMA

ARIMA è l'abbreviazione di AutoRegressive Integrated Moving Average ed è un algoritmo di previsione basato sull'idea che l'informazione dei valori passati della serie temporale può essere usata per prevedere quelli futuri.
Il modello ARIMA è un modello che combina i blocchi di costruzione AR e MA, proprio come ARMA. L'aggiunta in ARIMA è il blocco "I", che sta per differenziazione automatica delle serie temporali non stazionarie. La stazionarietà è un concetto importante nelle serie temporali che sta ad inficare una

Figura 6.2: Componente Stagionalità, Fonte: Corso di Laurea Magistrale Politecnico di Torino materia Data Science Lab A.A. 2020/2021



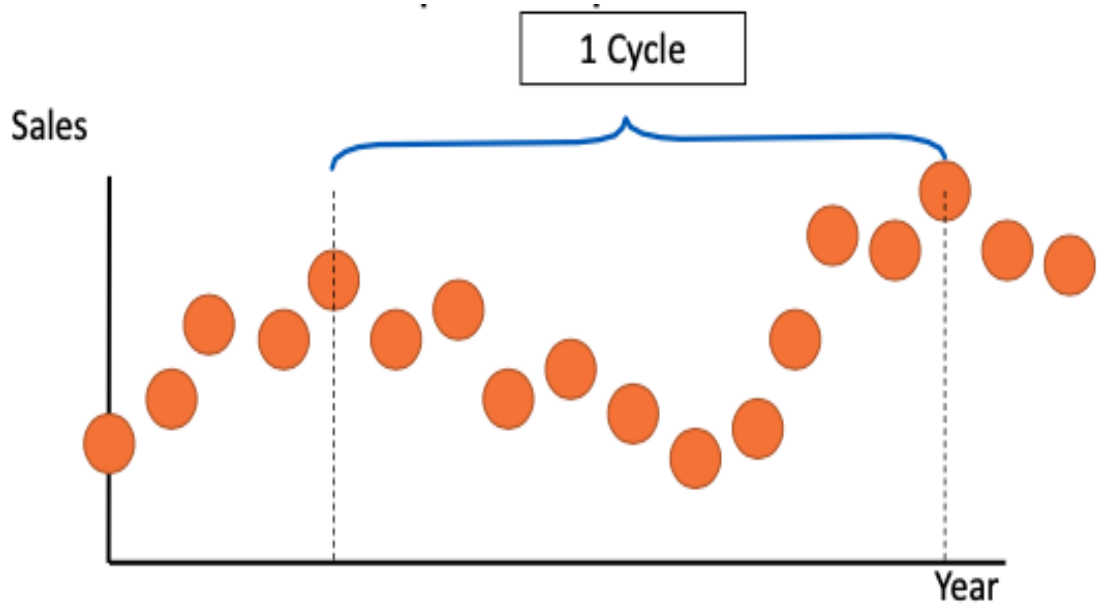
non una tendenza a lungo termine. Se una serie temporale non è stazionaria, è possibile renderla stazionaria applicando il differenziamento: sostituendo i valori attuali con la differenza tra il valore attuale e quello precedente. La "I" in ARIMA sta per integrare. Integrare è un sinonimo più matematico per differenziare una serie temporale non stazionaria. Nell'ARIMA, questo differenziamento non è più fatto prima della fase di modellazione, ma è fatto durante l'adattamento del modello.

6.5 Prophet

6.5.1 Introduzione

La previsione è un'attività molto comune nell'ambito dell'analisi dei dati perché fondamentale per molte attività all'interno di un'azienda. Queste organizzazioni necessitano di pianificare delle capacità per allocare in modo efficiente le risorse scarse e la definizione degli obiettivi al fine di misurare le prestazioni rispetto a una linea di base. Produrre previsioni di alta qualità non è un problema facile né per le macchine né per la maggior parte degli analisti. Abbiamo osservato due temi principali nella pratica della creazione

Figura 6.3: Componente Ciclicità, Fonte: Corso di Laurea Magistrale Politecnico di Torino materia Data Science Lab A.A. 2020/2021



di previsioni di business. In primo luogo, le tecniche di previsione completamente automatiche possono essere difficili da mettere a punto e spesso sono troppo poco flessibili per incorporare ipotesi o euristiche utili. In secondo luogo, gli analisti responsabili delle attività di data science in un'organizzazione in genere hanno una profonda esperienza di dominio sui prodotti o servizi specifici che supportano, ma spesso non hanno una formazione sulla previsione delle serie temporali. Gli analisti in grado di produrre previsioni di alta qualità sono quindi piuttosto rari perché la previsione è un'abilità specializzata che richiede una notevole esperienza. Il risultato è che la domanda di previsioni di alta qualità spesso supera di gran lunga il ritmo con cui possono essere prodotte. I primi due tipi di scala che affrontiamo sono che i metodi di previsione aziendale dovrebbero essere adatti a 1) un gran numero di persone che effettuano previsioni, possibilmente senza formazione sui metodi delle serie temporali; e 2) una grande varietà di problemi di previsione con caratteristiche potenzialmente idiosincratiche.

6.5.2 Funzionamento del modello

Il secondo modo per predirre la soglia è stata quella di usare la libreria open source Prophet ideata da Facebook per le serie temporali. Negli ultimi anni sta trovando applicazione anche su siti di ecommerce ad esempio Amazon.

Il modello Prophet si basa su tre funzioni del tempo e un termine di errore[3]:

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t$$

dove **g(t)** è la funzione di tendenza che modella i cambiamenti non periodici nel valore della serie storica, **s(t)** rappresenta i cambiamenti periodici (ad esempio, stagionalità settimanale e annuale), e **h(t)** rappresenta gli effetti delle vacanze o degli eventi rari che possono capitare in una serie storica.

6.5.3 Funzione di Trend g(t)

In generale sono stati implementati due modelli di trend che coprono molte delle applicazioni in Facebook: un **modello di crescita saturante** e un **modello lineare a tratti**.
Per quanto riguarda il primo modello che riguarda la crescita saturante, il componente principale del processo di generazione dei dati è un modello che analizza come la popolazione è cresciuta e come si prevede che continui a crescere nel futuro. Si può pensare che la modellazione della crescita degli utenti su Facebook è comminsurata crescita della popolazione negli ecosistemi naturali, dove c'è una crescita non lineare che satura alla capacità di carico. Ad esempio, la capacità di carico per il numero di utenti di Facebook in una particolare area potrebbe essere il numero di persone che hanno accesso a Internet. Questo tipo di crescita è tipicamente modellato utilizzando il modello di **crescita logistica**, che nella sua forma più elementare è

$$g(t) = \frac{C}{1 + exp(-k(t - m))} \tag{6.1}$$

dove C la è la **capacità di carico**, k è il **tasso di crescita** e m è un **parametro di offset**.
Ci sono due aspetti importanti della crescita che non vengono catturati nell'equazione 6.1. In primo luogo, la capacità di carico non è costante: con l'aumento del numero di persone nel mondo che hanno accesso a Internet, aumenta anche il tetto di crescita. Si può quindi sostituire la capacità fissa C con una capacità variabile nel tempo $C(t)$. In secondo luogo, il tasso di crescita non è costante. I nuovi prodotti possono alterare profondamente il

tasso di crescita in una regione, quindi il modello deve essere in grado di incorporare un tasso variabile per adattarsi ai dati storici.

$$g(t) = \frac{C(t)}{1 + exp(-(k + a(t)^T\delta)(t - (m + a(t)^T\delta)))} \tag{6.2}$$

L'equazione 6.2 viene ricavata partendo dal presupposto che il tasso di crescita può cambiare. Si suppone che ci siano S punti di changepoint agli istanti che vanno da $1, ..., S$. Si definisce per facilità un vettore, in questo caso δ, dove δ_j rappresenta la variazione di velocità che si verifica all'istante s_j. Il tasso in qualsiasi momento t è quindi il tasso di base k più tutti gli aggiustamenti che ci sono sino a quel momento: $k + \sum_{j:t>s_j} \delta_j$ Questo è rappresentato più chiaramente definendo un vettore $a(t)$ che vale 1 se $t > s_j$ altrimenti vale 0. Quindi la velocità al tempo t è allora data da $k + a(t)^T\delta$. Se si sostituisce questo valore all'equazione 6.1 si ottiene 6.2.

Possiamo riassumere dicendo che la funzione di crescita modella l'andamento dei dati e i punti di cambiamento sono momenti nei dati in cui i dati cambiano direzione. Usando i nuovi casi COVID-19 come esempio, potrebbe essere dovuto a nuovi casi che iniziano a diminuire dopo aver raggiunto un picco una volta introdotto un vaccino. Oppure potrebbe essere un improvviso aumento di casi quando un nuovo ceppo viene introdotto nella popolazione e così via. Prophet può rilevare automaticamente i punti di cambiamento o si possono impostare. È inoltre possibile regolare il potere che i punti di cambiamento hanno nel modificare la funzione di crescita e la quantità di dati presi in considerazione nel rilevamento automatico del punto di cambiamento.

La funzione di crescita ha tre opzioni principali:

- **Crescita lineare**: questa è l'impostazione predefinita per Prophet. Utilizza un insieme di equazioni lineari a tratti con pendenze diverse tra i punti di cambio. Quando viene utilizzata la crescita lineare, il termine di crescita sarà simile al classico $y = mx + b$, tranne che la pendenza (m) e l'offset (b) sono variabili e cambieranno valore ad ogni punto di cambio.

- **Crescita logistica**: questa impostazione è utile quando le serie temporali hanno un limite o un valore minimo in cui i valori che si stanno modellando diventano saturi e non possono superare un valore massimo o minimo. Quando viene utilizzata la crescita logistica, il termine di crescita sarà simile a una tipica equazione per una curva logistica (vedi

sotto), tranne per il fatto che la capacità di carico (C) varierà in funzione del tempo e il tasso di crescita (k) e l'offset (m) sono variabili e cambieranno il valore ad ogni punto di cambio.

$$g(t) = C(t)/(1 + x^{-k(t-m)}) \tag{6.3}$$

- **Crescita piatta**: si ha un trend piatto quando non c'è crescita nel tempo (ma potrebbe esserci ancora stagionalità). La funzione di crescita, in questo caso, sarà un valore costante.

6.5.4 Funzione di stagionalità s(t)

Le serie temporali aziendali hanno spesso una stagionalità multiperiodale a causa dei comportamenti umani che rappresentano. Ad esempio, una settimana lavorativa di 5 giorni può produrre effetti su una serie temporale che si ripete ogni settimana, mentre gli orari delle ferie e le vacanze scolastiche possono produrre effetti che si ripetono ogni anno. Per adattare e prevedere questi effetti dobbiamo specificare modelli di stagionalità che sono funzioni periodiche di t. Ci affidiamo alla serie di Fourier per fornire un modello flessibile di effetti periodici. Sia P il periodo regolare che ci aspettiamo che le serie temporali abbiano (ad esempio P = 365,25 per i dati annuali o P = 7 per i dati settimanali, quando scaliamo la nostra variabile temporale in giorni). Possiamo approssimare gli effetti stagionali regolari arbitrari con:

$$s(t) = \sum_{n=1}^{N} (a_n cos(\frac{2\pi nt}{P}) + b_n sin(\frac{2\pi nt}{P})) \tag{6.4}$$

La **funzione di stagionalità s(t)** è semplicemente una serie di Fourier in funzione del tempo (somma di seni e coseni successivi). Ogni termine seno e coseno viene moltiplicato per un coefficiente.

6.5.5 Funzioni Eventi e Holidays

La **funzione holidays (vacanza o evento) h(t)** consente al modello di regolare la predizione quando una vacanza o un evento importante possono modificare le previsioni. Richiede un elenco di date (nel caso in questione useremo le festività in Italia) e quando ciascuna data è presente nella previsione viene aggiunta o sottratta valore dalla previsione dai termini di crescita e stagionalità in base ai dati storici sulle date di vacanza identificate.

Prophet consente quindi all'analista di fornire un elenco personalizzato di eventi passati e futuri, identificati dall'evento o dal nome univoco della festività.

6.5.6 Applicazione di Prophet

In Tabella 6.1 si possono osservare le nuove soglie calcolate con il modello appena illustrato. Ci sono delle differenze rispetto alle soglie predette dagli altri algoritmi ma esse non sono sostanziali. Il vantaggio è che su un periodo temporale più vasto è possibile analizzare con maggior precisione i punti di cambiamento facendo uno studio approfondito sulla stagionalità dei dati, sui giorni della settimana o sui mesi. Sempre con un set di dati di maggiore storicità (ad esempio due anni) è possibile migliorare il tuning dei parametri del modello per renderlo ancor più preciso.

In Figura 6.4 è rappresentato lo stesso ordine della Figura 5.2. I puntini rappresentano la serie storica degli ordini con le relative quantità nel medesimo periodo temporale. I giorni predetti sono gli ultimi (ultima settimana di test) e l'andamento previsto è quello riportato in figura con la linea blu scura. Questa linea è racchiusa all'interno di una banda di colore più chiaro che ne da degli intervalli di confidenza. Quello che qui si è deciso di assumere come valore di riferimento è il valore massimo di confidenza. Questi valori si possono leggere nella tabella 6.1.
I pallini in rosso sono gli ordini valutati per il check 1 in questo caso nella settimana di test e si può vedere dove essi sono posizionati rispetto alla soglia. Nel caso in esame sono tre gli ordini che vanno fuori soglia e due invece che sono sotto soglia. Un altro esempio è rappresentato dalla figura 6.5 in cui i pallini sono tutti all'interno della soglia predetta.

Gli iperparametri configurati sono stati quelli dei giorni delle festività nella Nazione Italia, non ci sono stati punti di changepoing in cui il tasso possa cambiare ed in fine si ipotizza una crescita lineare.

Figura 6.4: Sale Orders Check ID: 1, LE: 0435, Metallo: AL

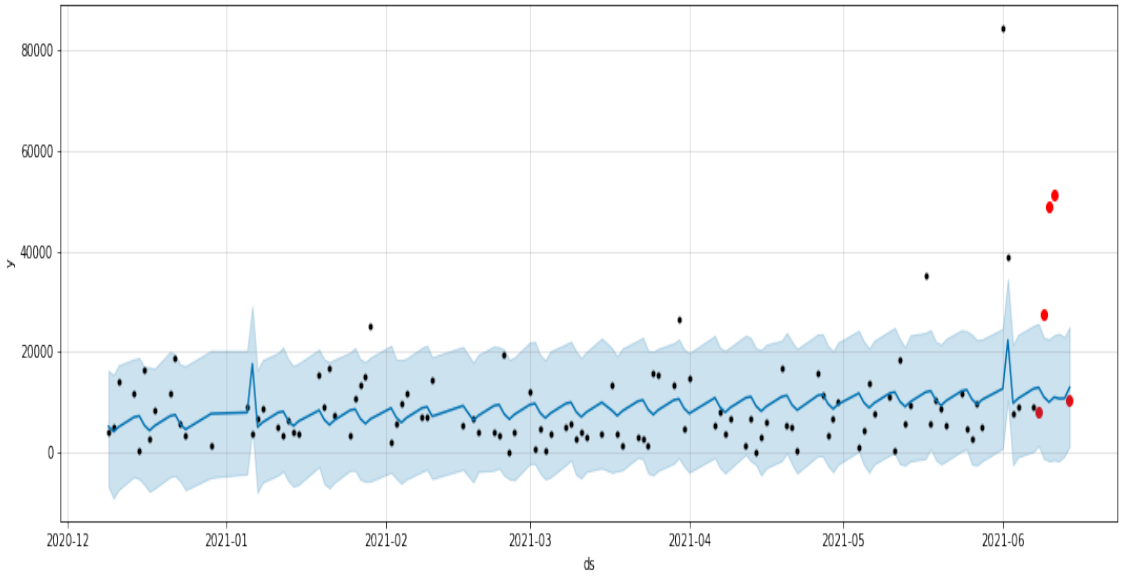
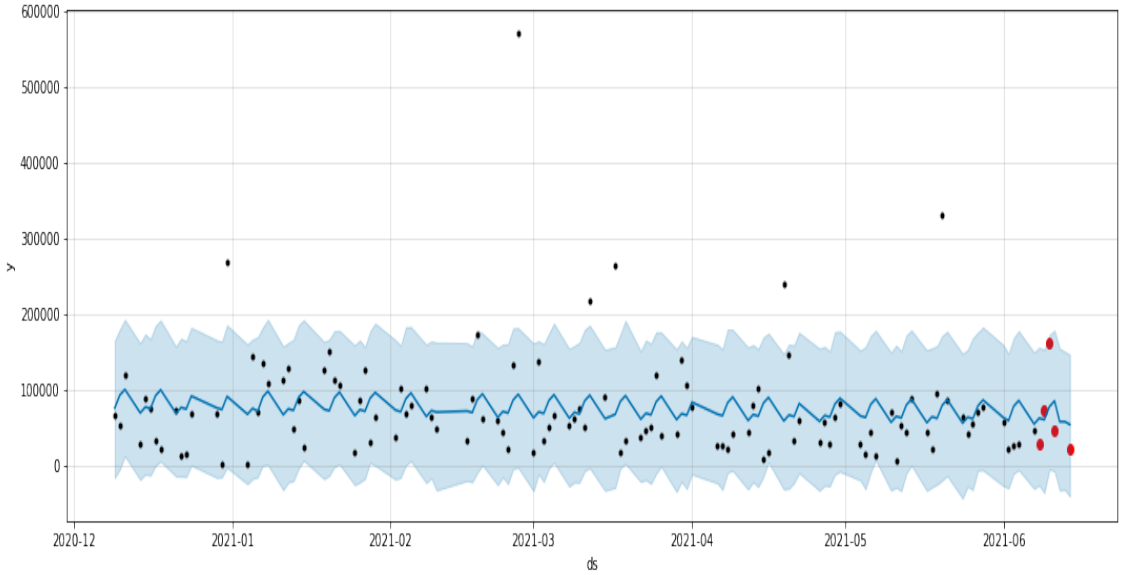


Figura 6.5: Sale Orders Check ID: 1, LE: 0435, Metallo: CU



ID_CHECK	COMPANY_CODE	METAL	TH_BOXPLOT	TH_MM
1	435	AL	21.35	150
1	435	CU	165.08	250
1	435	PB	10.31	100
1	1919	CU	23.91	200
1	1948	AL	5.88	100
...	...	...	...	...

Tabella 6.1: Confronto soglie Metal Manager e soglie predette tramite modello Prophet

Capitolo 7

Metodi unsupervised

Esiste una ben nota relazione complementare tra il clustering e il rilevamento dei valori anomali. Una visione semplicistica sarebbe che ogni punto dati sia un membro di un cluster o un valore anomalo. Nel clustering, l'obiettivo è di suddividere i punti in sottoinsiemi densi, mentre nel rilevamento dei valori anomali, l'obiettivo è identificare i punti che non sembrano adattarsi naturalmente a questi sottoinsiemi densi. Tuttavia, è importante capire che l'utilizzo solo della relazione complementare dei punti ai cluster per definire gli outlier porta alla scoperta di outlier deboli o rumore. Ad esempio, un punto dati che si trova ai margini di un grande cluster è molto diverso da uno completamente isolato da tutti gli altri cluster. Inoltre, tutti i punti dati in cluster molto piccoli possono talvolta essere considerati valori anomali. Pertanto, quando si utilizza il clustering per il rilevamento degli outlier, viene utilizzato un approccio più sfumato (rispetto a quello della non appartenenza al cluster) per calcolare i punteggi degli outlier.

7.1 PyCaret

Un tool molto utile che si andrà ad utilizzare in questa fase dello sviluppi è PyCaret.
PyCaret è una libreria di machine learning open source scritta in Python che automatizza i flussi di lavoro di machine learning. Essa viene molto utilizzata perché racchiude all'interno di se tantissime librerie evitando in questo modo l'utilizzo di migliaia di righe di codice. Dai programmatori ed esperti di Data Science e Machine Learning trova molta diffusione perché altamente efficiente ad affrontare svariati problemi tra i quali: classificazione, regressione, clustering, anomaly detection e natural language processing (NLP). In questo caso

particolare si andrà ad utilizzare il pacchetto riguardante l'anomaly detection che fornisce svariati algoritmi tra i quali si possono citare l'Isolation Forest, k-Nearest Neighbors Detector, Local Outlier Factor, Support Vector Machine e tanti altri [10].

7.2 Isolation Forest

L'Isolation Forest (IF) è un algoritmo di tipo unsupervised e si basa sull'algoritmo Decision Tree.
La maggior parte degli approcci esistenti basati su modelli per il rilevamento delle anomalie costruisce un profilo di istanze normali, quindi identifica come anomalie le istanze che non si conformano al profilo normale. L'algoritmo in questione propone un metodo basato su un modello diverso che isola esplicitamente le anomalie invece dei punti normali.
Il principio sul quale si basa questo algoritmo è il seguente: *le anomalie sono poche e diverse*.

L'uso dell'isolamento consente al metodo proposto, iForest, di sfruttare il sottocampionamento in misura non praticabile nei metodi esistenti, creando un algoritmo che ha una complessità temporale lineare con una bassa costante e un basso fabbisogno di memoria. La nostra valutazione empirica mostra che iForest si comporta favorevolmente rispetto metodi basati sulla distanza come il LOF e Random Forests sia in termini di AUC sia in termini di tempo di elaborazione. iForest funziona bene anche in problemi dimensionali elevati che hanno un gran numero di attributi irrilevanti e in situazioni in cui il training set non contiene anomalie.
Il metodo proposto, chiamato Isolation Forest o iForest, costruisce un insieme di iTrees per un dato set di dati, quindi le anomalie sono quelle istanze che hanno percorsi medi brevi sugli iTrees. Ci principalmente solo due iperparametri da configurare: **il numero di alberi da costruire** e la **dimensione del sottocampionamento**. Le prestazioni di rilevamento di iForest convergono rapidamente con un numero molto ridotto di alberi e richiede solo una piccola dimensione di sottocampionamento per ottenere elevate prestazioni di rilevamento con un'elevata efficienza.

7.2.1 Vantaggi e svantaggi dell'algoritmo iForest

La grande differenza che c'è rispetto ai metodi basati sulla densità e sulla distanza è che l'Isolation Forest risulta essere molto efficiente in quanto non

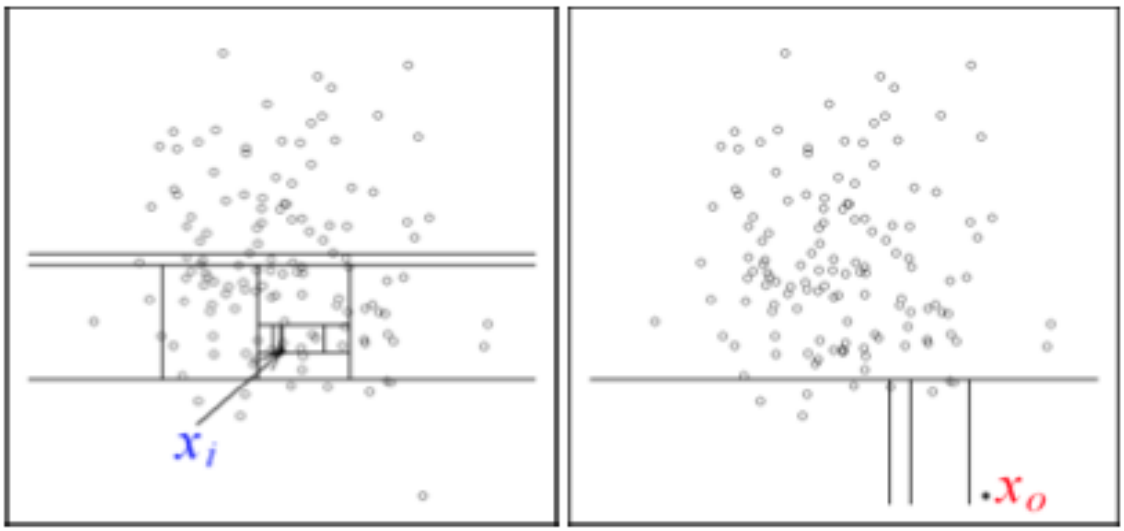
utilizza misure di distanza o densità per rilevare anomalie. Questo è molto importante in termini di costo computazionale dell'algoritmo in quanto diminuisce. Inoltre, un ulteriore vantaggio è rappresentato dal fatto di avere una bassa complessità [6].

7.2.2 Come lavora l'iForest?

In un albero decisionale classico (Decision Tree) le istanze dei dati vengono partizionate ricorsivamente fin quando tutte risultano isolate. Le ultime foglie vengono chiamati nodi foglia e l'altezza entro cui si arriva a questi nodi può essere sia lunga che corta. Generalmente percorsi più brevi indicano le anomalie poiché (a) il minor numero di istanze di anomalie risulta in un minor numero di partizioni e (b) è più probabile che le istanze con valori di attributo distinguibili siano separati nella partizione anticipata (questo significa che l'outlier è visibile subito). Quindi, quando una foresta di alberi casuali produce collettivamente percorsi più brevi per alcuni punti particolari, è molto probabile che si tratti di punti anomali.

Per dimostrare l'idea che le anomalie sono più suscettibili all'isolamento in caso di partizionamento casuale, illustriamo un esempio nella Figura 7.1 per visualizzare il partizionamento casuale di un punto normale rispetto a un'anomalia. Osserviamo che un punto normale, x_i, richiede generalmente più partizioni per essere isolato. È vero anche il contrario per il punto di anomalia, x_o, che generalmente richiede meno partizioni per essere isolato. In questo esempio, le partizioni vengono generate selezionando casualmente un attributo e quindi selezionando casualmente un valore diviso tra i valori massimo e minimo dell'attributo selezionato. Poiché il partizionamento ricorsivo può essere rappresentato da una struttura ad albero, il numero di partizioni necessarie per isolare un punto è equivalente alla lunghezza del percorso dal nodo radice a un nodo terminale (foglia). In questo esempio, la lunghezza del percorso di x_i è maggiore della lunghezza del percorso di x_o. Poiché ogni partizione viene generata casualmente, i singoli alberi vengono generati con diversi insiemi di partizioni. Per trovare una media delle lunghezze dei percorsi su un numero di alberi per trovare la lunghezza del percorso prevista. La Figura7.2 ha lo stesso significato della 7.1 però qui è rappresentato anche l'albero dal quale le partizioni hanno origine. Si vede che la prima divisione, quindi ad altezza 1 dell'albero a partire dal nodo radice si è in presenza di un punto anomalo.

Figura 7.1: Profondità dell'albero decisionale a seconda della posizione del punto di outlier



(a) Isolating x_i (b) Isolating x_o

Quindi una profondità media inferiore significa una maggiore probabilità di essere un outlier. Tuttavia, in pratica, non possiamo usare la profondità media, poiché la profondità di un albero dipende dal numero di campioni su cui è stato adattato. Per questo motivo abbiamo bisogno di una formula che tenga conto anche del numero totale di istanze. Questa è la formula proposta nel documento ufficiale [5]:

$$s(x, n) = 2^{-\frac{E(h(x))}{c(n)}}$$

dove

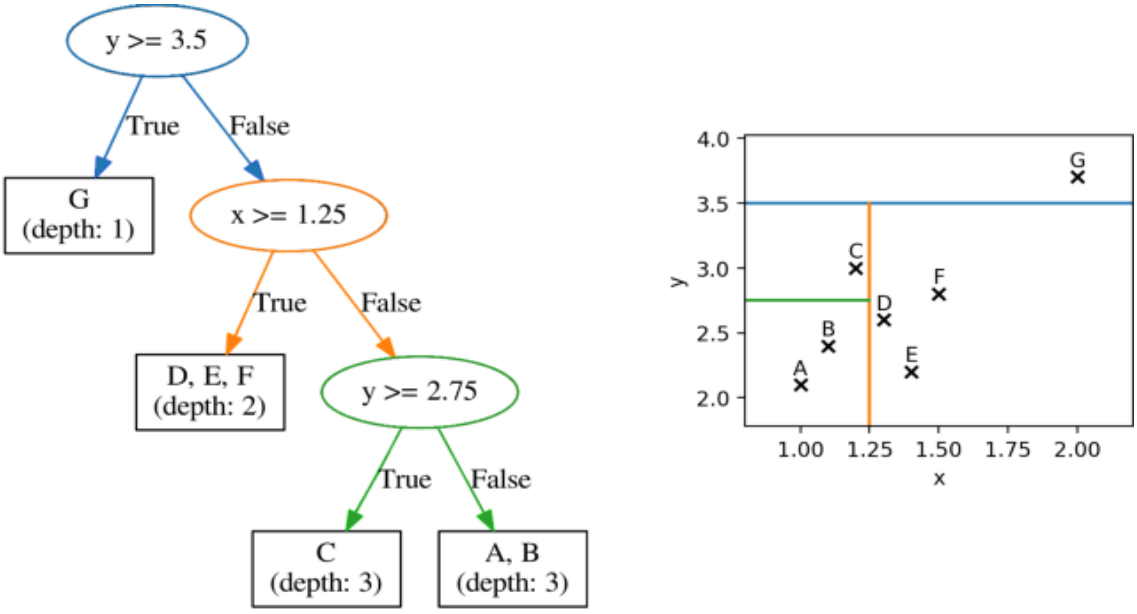
$$c(n) = 2H(n-1) - (2(n-1)/n)$$

dove n è il numero di istanze, $h(x)$ è la profondità alla quale si trova il punto dati in un particolare albero ($E(h(x))$ è la sua media su diversi alberi) e H è il numero armonico. $s(x, n)$ è un numero compreso tra 0 e 1, dove più alto è il punteggio più è probabile che sia un valore anomalo.

7.2.3 Anomaly Detection con iForest

L'algoritmo di Isolation Forest rileva le anomalie in due fasi: la prima è quella di addestramento in cui si costruiscono gli alberi di isolamento usando i

Figura 7.2: Punto anomalo ad altezza 1 rispetto al nodo radice

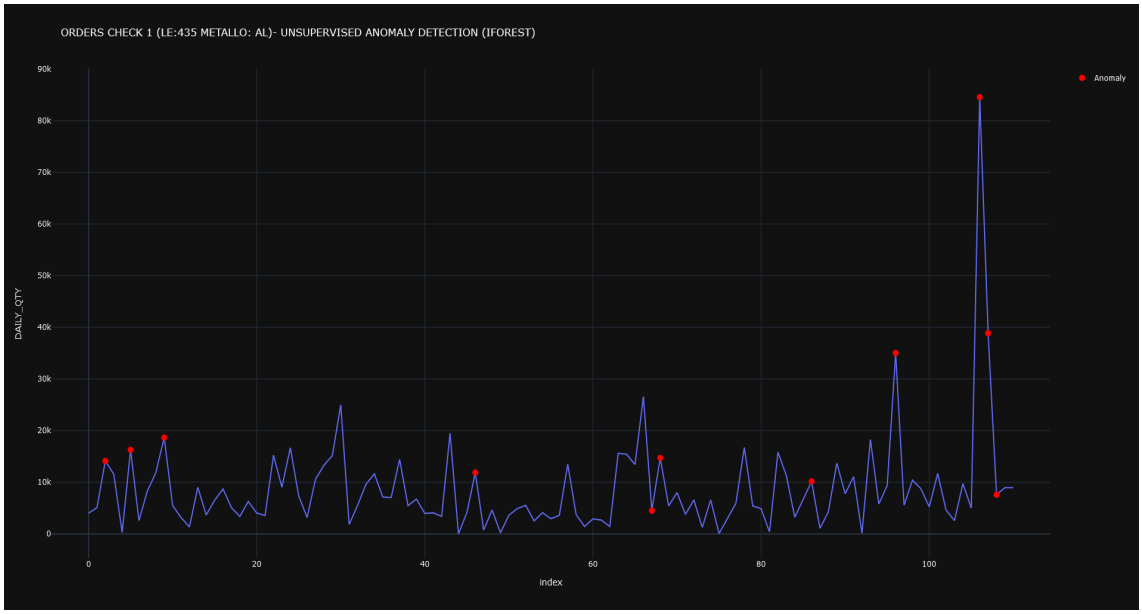


sottocampioni del set di train; la seconda fase (test) fa passare le istanze di test attraverso alberi di isolamento per ottenere un punteggio di anomalia per ciascuna istanza.

Ci sono due parametri di input per l'algoritmo iForest: **dimensione del sottocampionamento** e il **numero di alberi t**. Il parametro della dimensione sottocampionamento ψ controlla la dimensione dei dati di training. Quando questo parametro aumenta a un valore desiderato, iForest rileva in modo affidabile le anomalie e non è necessario aumentarlo ulteriormente perché aumenterebbe il tempo di elaborazione e la dimensione della memoria senza alcun ulteriore guadagno nelle prestazioni. Empiricamente, i documenti ufficiali [5] riportano che l'impostazione di ψ su 28 o 256 generalmente fornisce dettagli sufficienti per eseguire il rilevamento delle anomalie su un'ampia gamma di dati. In questo caso specifico si userà $\psi = 256$ come valore predefinito. Il numero dell'albero t controlla la dimensione dell'insieme. Si studia [5] che le lunghezze dei cammini di solito convergono ben prima di $t = 100$ ed in questo caso useremo questo valore come predefinito. La complessità dell'addestramento di un iForest è $O(t\psi log(\psi))$.

Nel caso in esame, sempre in riferimento al Check 1, Company Code 435

Figura 7.3: Isolation Forest: Company Code 435 - Metallo AL - Check 1



e Metallo alluminio, l'analisi risultante è riportata in Figura 7.3 in cui i puntini rossi identificano il valore anomalo lungo la serie storica. Nella Tabella 7.1 possiamo vedere tutti i punti rilevati come anomalie nel periodo di addestramento, mentre nella tabella 7.2 i valori dei punti anomali nella settimana di test. Come si può notare i punti identificati come anomalie sono questa volta tre su cinque dei totali a differenza del modello Prophet che identificava zero anomalie.

7.2.4 Valutazione dell'algoritmo

Nella fase di valutazione, viene valutato un punteggio di anomalia s dalla lunghezza del percorso prevista $E(h(x))$ per ciascuna istanza di test. $E(h(x))$ sono derivati passando istanze attraverso ogni iTree in un iForest. Usando la funzione *PathLength*, una singola lunghezza di percorso $h(x)$ viene derivata contando il numero di archi e dal nodo radice a un nodo terminale quando l'istanza x attraversa un iTree. Quando x termina in un nodo esterno, dove Dimensione > 1, il valore restituito è e più un aggiustamento c(Dimensione). La regolazione tiene conto di un sottoalbero non costruito oltre il limite di altezza dell'albero. Quando si ottiene $h(x)$ per ogni albero dell'insieme, si produce un punteggio di anomalia calcolando $s(x, \psi)$. La complessità del

COMPANY_CODE	METALLO	DATE	DAILY_QTY	ANOMALY_SCORE
435	AL	2020-12-11	14128.11	0.003644
435	AL	2020-12-16	16294.74	0.005291
435	AL	2020-12-22	18649.81	0.023908
435	AL	2021-03-01	11888.39	0.125668
435	AL	2021-03-31	4508.31	0.164575
435	AL	2021-04-01	14734.97	0.127728
435	AL	2021-04-30	10170.70	0.177986
435	AL	2021-05-17	35029.06	0.016006
435	AL	2021-06-01	84578.16	0.185302
435	AL	2021-06-02	38844.44	0.037880
435	AL	2021-06-03	7607.34	0.013373

Tabella 7.1: Isolation Forest - risultati periodo di train

COMPANY_CODE	METALLO	DATE	DAILY_QTY	ANOMALY_SCORE
435	AL	2021-06-09	27434.32.17.11	0.0.15815
435	AL	2021-06-10	48947.24	0.047756
435	AL	2021-06-11	51359.12	0.056649

Tabella 7.2: Isolation Forest - risultati periodo di test

processo di valutazione è $O(nt \log \psi)$, dove n è la dimensione dei dati di prova. I dettagli della funzione PathLength possono essere trovati in Algorithm 3. Per trovare le m anomalie superiori, si ordinano semplicemente i dati usando s in ordine decrescente. Le prime m istanze sono le m anomalie principali.

7.3 Local Outlier Factor (LOF)

L'algoritmo LOF appartiene alla classe di algoritmi unsupervised di tipo density based.
L'obiettivo di questo metodo è assegnare a ciascun oggetto una misura o grado di valore anomalo. Questo grado è chiamato **fattore di valore anomalo locale (LOF) di un oggetto**. È locale in quanto il grado dipende da quanto è isolato l'oggetto rispetto all'ambiente circostante.

7.3.1 Come lavora LOF?

[8, 9, 7]

La distanza tra i due punti dati p e o in uno spazio euclideo può essere calcolata utilizzando:

$$d(p,o) = \sqrt{\sum_{i=1}^{N}(p_i - o_i)^2}$$

Dopo aver calcolato tutte le distanze viene scelto l'iperparametro k che seleziona le k distanze vicine più prossime in ordine decrescente (dalla distanza più prossima a quella meno prossima), ovvero identifica il numero di osservazioni che dobbiamo osservare. Un esempio potrebbe essere $k = k_0$.
La scelta di k è molto importante perché un valore basso l'algoritmo diventa sensibile al rumore, al contrario, con un valore alto di k l'algoritmo potrebbe non riconoscere le anomalie locali.

La seconda fase che consideriamo si rifà al calcolo del **Local Reachability Density** in cui dobbiamo andare dal punto in cui ci troviamo per raggiungere il punto successivo o l'insieme di punti. Le distanze di raggiungibilità calcolate vengono utilizzate per calcolare la densità di accessibilità locale. Per tutti i k vicini più prossimi di a, vengono calcolate le distanze di raggiungibilità e i valori trovati vengono sommati e divisi per il valore di k. Quando viene preso l'inverso di questo valore, calcoliamo la densità che stiamo cercando.

Quindi dapprima si calcola:

$$AverageRD_A = \frac{1}{k}\sum_k max[(k^{th} distance of A's neighbor, distance(A, k^{th} neighbor))]$$

$$LRD_A = \frac{1}{RD_A}$$

Le densità di raggiungibilità locali trovate vengono confrontate con le densità di raggiungibilità locali dei k vicini più vicini di A. La densità di ciascun vicino viene sommata e divisa per la densità di A. Il valore trovato viene diviso per il numero di vicini, ad esempio k.

$$LOF_A = \frac{\frac{[(LRD(1st.neighbor)+LRD(2nd.neighbor)+...+LRD(kth.neighbor))}{LRD_A}}{k}$$

7.3.2 Applicazione del LOF

Nelle Figure 7.4 e 7.5 è possibile vedere una predizione dell'algoritmo di LOF per il Check numero 1, la Company Code 435 e il Metallo Alluminio. Gli iperparametri utilizzati e concordati sono stati *k=3* e la metrica utilizzata per la distanza è quella di *manhattan*. Riguardo la scelta di k il confine di decisione non è stato così grande perché spesso la settimana che si va a testare contiene pochi ordini, al più cinque, quindi inserendo un k>2 spesso si è notato di non riuscire a dare al Metals Manager un risultato in quanto l'algoritmo non riesce a trovare i k-esimi più vicini. Il compromesso è stato scegliere k=3 che nel periodo in cui lo script è andato in produzione ha mostrato un'accuratezza elevata.
In generale sono possibili più tipi di metriche per calcolare la distanza tra due punti ma le più famose sono la distanza **euclidea** e **manhattan**. Per quanto riguarda la prima, la distanza la si può calcolare con la seguente formula:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (y_i - x_i)^2}$$

Invece per la distanza di manhattan:

$$d(x, y) = \sum_{i=1}^{n} |x_2 - x_1| + |y_2 - y_1|$$

Anche in questo caso, così come per l'algoritmo esposto precedentemente, troviamo i pallini rossi che rappresentano gli ordini che vengono ritenuti anomali e quindi non devono essere accettati. Nel grafico 7.4 viene rappresentata la serie storica degli ordini e i puntini finali sono quelli che vengono valutati anomali nella settimana di test che si possono vedere poi meglio nella figura 7.5.
Nella Tabella 7.3 qui di seguito vengono rappresentati numericamente gli ordini che vengono ritenuti anomali nella settimana di test (pallini rossi nel grafico).

COMPANY_CODE	METALLO	DATE	DAILY_QTY
435	AL	2021-06-10	48947.24
435	AL	2021-06-11	51359.12

Tabella 7.3: LOF - risultati periodo di test

Figura 7.4: LOF: Company Code 435 - Metallo AL - Check 1 - SERIE STORICA

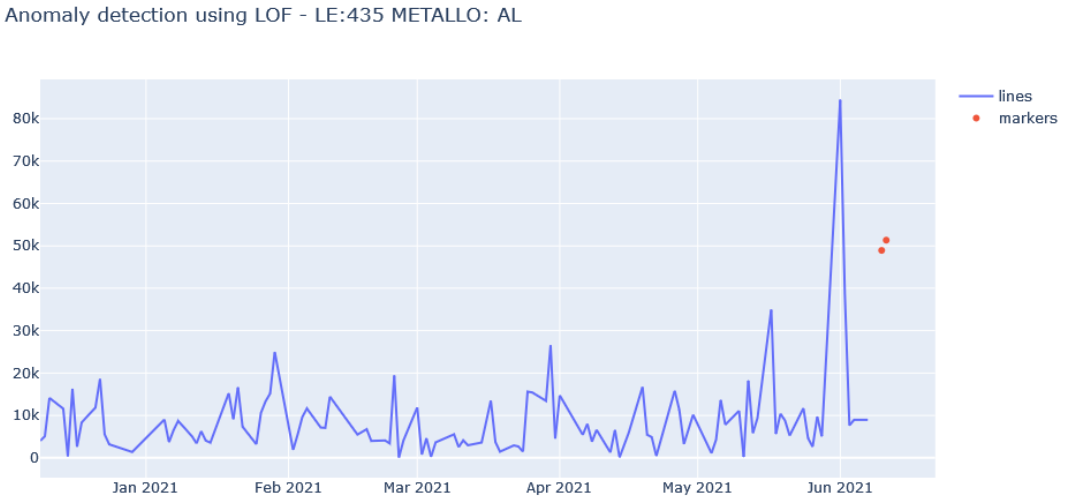
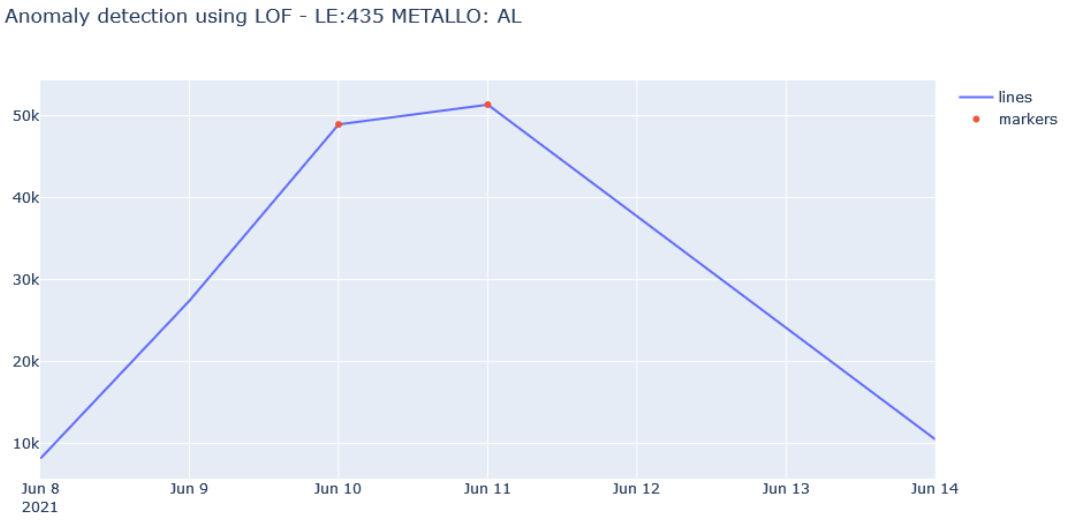


Figura 7.5: LOF: Company Code 435 - Metallo AL - Check 1 - ANOMALIE



Parte III

Conclusione e lavori futuri

Capitolo 8

Risultati sperimentali

In questo capitolo verranno illustrati i risultati sperimentali degli algoritmi utilizzati (Time Series Forecasting e Unsupervised anomalies detection) ed inoltre verranno giustificati tali risultati ricorrendo alla definizione di specifici KPI.

8.1 Confronto dei modelli usati

In generale, per il lavoro descritto non esistono dei risultati che sono migliori di altri o algoritmi che hanno performance più accurati di altri. Questo perchè l'azienda non aveva riportato quali ordini effettivamente venivano rifiutati nel periodo storico considerato e neppure quali aggiustamenti o modifiche erano state apportate alla soglia statica al fine di far passare gli ordini che altrimenti venivano rifiutati.
Le soglie statiche, come si è già avuto modo di dire nei capitoli precedenti, sono state impostate dai Metals Manager senza alcuna giustificazione matematica o statistica motivo per il quale risultavano essere sovra dimensionate o sotto dimensionate. L'unico termine di paragone e di efficacia di questi algoritmi è stata la "prova sul campo", ovvero la messa in produzione dello script e un confronto costante con il personale dell'azienda. Questo confronto avveniva mettendo in produzione lo script che di volta in volta veniva configurato con i giusti iperparametri per circa e lo si faceva girare per circa una settimana. Al termine della stessa si valutava l'andamento degli ordini che erano stati accettati o rigettati al fine di valutare l'effettiva efficienza della soglia. Questo processo ha portato a considerare un algoritmo migliore piuttosto di un altro non tanto da un punto di vista di accuratezza (perché opportunamente configurati gli algoritmi funzionavano quasi tutti bene) ma

da un punto di vista di tempi.
Il tempo entro cui un algoritmo finiva l'esecuzione e prediceva la soglia per ogni check è stato un fattore molto importante per l'azienda perché ci sono delle ore specifiche in cui il robot generava gli ordini (il metal book) e il calcolo della soglia non poteva occupare più di un'ora.

metti la tabella che sta ad indicare i tempi di LOF-IF e Prophet

8.2 Key Performance Indicators (KPI)

Le KPI acronimo di **Key Performance Indicators** sono delle metriche che misurano le prestazioni nel tempo di una azienda. In questo contesto i KPI vengono utilizzati per dimostrare all'azienda che i metodi di Time Series Forecasting piuttosto che i metodi di Anomaly Detection con gli algoritmi Unsupervised sono efficaci e producono notevoli vantaggi se applicati e sostituiti al processo attualmente in esecuzione.
I KPI che si andranno a valutare sono i seguenti:

1. Rapporto tra soglia fissata dal Metals Manager e quella prevista dal modello
2. Percentuale di ordini bloccati quando viene applicata la soglia del Metals Manager (statica)
3. Percentuale di ordini bloccati quando viene applicata la soglia del modello
4. Rapporto tra soglia stimata dal modello e quantità relativa all'ordine più consistente arrivato nell'ultima settimana (periodo di prova)
5. (per il caso del boxplot) Rapporto tra il 95° percentile calcolato sulla distribuzione delle quantità di ordini gestiti nel periodo di formazione e il 95° percentile calcolato sulla distribuzione delle quantità di ordini gestite nell'ultima settimana (test set)

I tempi degli algoritmi sono visualizzabili nella seguente tabella 8.2

Algoritmo	**Tempo di esecuzione**
Prophet	197 secondi
Boxplot	283 secondi
Local Outlier Factor (LOF)	243 secondi
Isolation Forest (IF)	173 secondi

Come già detto precedentemente, gli algoritmi di tipo unsupervised hanno uno svantaggio non indifferente ovvero quello di non riuscire a fornire un suggerimento di soglia ai Metals Manager che quindi non possono visualizzare e impostare manualmente un valore di soglia ma riescono solo ad identificare quali sono quegli ordini che sono nel cluster, quindi accettati, o fuori dal cluster, quindi rifiutati. Come spiegato e come ci aspettavamo, proprio per natura dell'Isolation Forest, esso risulta essere l'algoritmo più performante in termini di tempo e consuma anche meno risorse dal punto di vista computazionale. Per i primi esperimenti che sono stati mandati in produzione però sono stati scelti sia il modello Prophet che quello di Isolation Forest.

Capitolo 9

Lavori futuri

In base al lavoro che è stato svolto sarebbe molto interessante in futuro allenare delle persona a flaggare i dati per un periodo temporale di due/tre mesi al fine di poter fare una'analisi utilizzando algoritmi supervised al fine di trovare valori anomali. Questa modalità sicuramente avrà una accuratezza maggiore che sarà possibile valutare matematicamente. Ci sono tanti algoritmi che si usano per la Supervised Outlier Detection come per esempio k-Nearest neighbor, Centroid Classifier e Naive Bayes.
Un passo avanti sulla scia di quanto già fatto sarebbe quella di mantenere uno storico del cambiamento di soglie e anche delle date in cui ciò accade per cercare di leggere in modo più accurato la serie storica e individuare quelli che nel modello Prophet si era definiti changepoint. Inoltre per quanto riguarda gli algoritmi di tipo Unsupervised sarebbe molto importante come primo step cercare di trovare un modo per capire come trovare un suggerimento di soglia da poter fornire ai Metals Manager. Allo stato attuale infatti gli algoritmi di tipo Unsupervised sono capaci solo di prevedere quali ordini devono essere scartati e quali no però non riescono a trovare dov'è il confine che separa gli ordini dentro il cluster e gli ordini fuori cluster.
Come ultimo sviluppo si potrebbe sviluppare l'algoritmo LSTM che si basa sulle reti neurali. Essi cercano di rilevare delle anomalie sulle serie storiche come quelli appena descritti in maniera leggermente.

Bibliografia

[1] D. Hawkins, *Identification of Outliers*, Chapman and Hall, 1980.

[2] C.C. Aggarwal and S. Sathe, *Theoretical Foundations and Algorithms for Outlier Ensembles.*, ACM SIGKDD Explorations, 17(1), June 2015

[3] Sean J. Taylor, Benjamin Letham, *Forecasting at scale.*

[4] E. Torricelli e A. Vasari, in "Delle misure", *Atti Nuovo Cimento*, vol. III, n. 2 (feb. 1607), p. 27–31.

[5] Liu, Fei Tony; Ting, Kai Ming; Zhou, Zhi-Hua, *Isolation Forest*, 2008 Eighth IEEE International Conference on Data Mining

[6] M. Wu and C. Jermaine; *Outlier detection by sampling with accuracy guarantees*, In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 767–772, New York, NY, USA, 2006. ACM.

[7] Breunig, M. M.; Kriegel, H.-P.; Ng, R. T.; Sander, J. *LOF: Identifying Density-based Local Outliers*, Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data.

[8] Hanxiang Dinga , Kun Dinga, , Jingwei Zhangb , Yue Wanga , Lie Gaoa , Yuanliang Lia , Fudong Chena , Zhixiong Shaoc , Wanbin Laia, *Local outlier factor-based fault detection and evaluation of photovoltaic system*, 2018

[9] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, Jörg Sander, *LOF: Identifying Density-Based Local Outliers*, 2000

[10] Sito ufficiale della libreria Open Source di PyCaret: *https://pycaret.org/guide/*