



Università degli Studi di Pisa

Dipartimento di Informatica

Master II livello in BigData Analytics & Artificial Intelligence for Society

“Proactive Theft: Sviluppo di un modello di Machine Learning per l'individuazione di pattern ricorrenti di viaggi relativi a furti”

Il Candidato

Giulia Segurini

Il Relatore

Mirco Nanni

A.A. 2023/ 2024

Ho trovato la mappa del labirinto: checkmate.

PROACTIVE_THEFT_REPORT_TECNICO

“Sviluppo di un modello di Machine Learning per l'individuazione di pattern ricorrenti di viaggi relativi a furti”

1. Introduzione.....	4
2. Obiettivi della Ricerca.....	5
3. Monitoraggio e Analisi dei Dati.....	5
4. Monitoraggio in Tempo Reale e Sistemi di Allarme.....	5
5. Analisi dei Pattern nei Viaggi.....	5
6. Sfide Tecniche.....	6
7. Soluzioni Esistenti e Vantaggi dell'Approccio Machine Learning.....	6
8. Conclusioni e Impatti.....	6
3. Metodologia.....	7
3.1 Descrizione dei Dati.....	7
Pulizia e Preparazione del Dataset.....	10
Rimozione di Colonne Non Necessarie.....	10
Gestione dei Valori Mancanti.....	10
Conversione delle Date e Creazione di Variabili Temporali.....	11
Calcolo della Durata del Viaggio.....	11
Eliminazione delle Colonne Ridondanti.....	11
Analisi della Correlazione.....	12
Calcolo della Matrice di Correlazione.....	12
Analisi degli Istogrammi.....	14
4. Sviluppo di un Modello di machine Learning per l' individuazione di pattern ricorrenti di viaggi relativi a furti.....	30
5. Conclusioni e Sviluppi Futuri.....	30
6. Riferimenti Bibliografici.....	30
7. Appendici.....	31

1. Introduzione

Contesto e Motivazione

Il furto di veicoli è un fenomeno in crescita, con implicazioni economiche rilevanti per i proprietari e le assicurazioni. Le modalità di furto sono sempre più sofisticate, rendendo difficile la prevenzione. Le compagnie assicurative e le forze dell'ordine sono alla ricerca di soluzioni innovative per contrastare i furti, e uno degli approcci promettenti è l'uso della telematica avanzata.

Tecnologia Telematica per la Prevenzione dei Furti

Octo Telematics, leader nel settore della telematica automobilistica, offre una soluzione basata sull'analisi dei dati raccolti da dispositivi GPS installati sui veicoli. Questi dati permettono di monitorare in tempo reale i comportamenti di guida e identificare pattern anomali che potrebbero indicare un furto imminente. Analizzare i dati telematici consente di prevedere e prevenire i furti con maggiore efficacia.

Impatto del Furto di Veicoli

I furti di veicoli comportano danni economici significativi per i proprietari e per le compagnie assicurative. Inoltre, le forze dell'ordine affrontano difficoltà nell'identificare e recuperare i veicoli rubati. La prevenzione tramite analisi dei pattern di viaggio è quindi una strategia chiave per ridurre il rischio di furti.

1.1 Obiettivi della Ricerca

La tesi si propone di analizzare i dati telematici dei veicoli per identificare caratteristiche che potrebbero suggerire comportamenti sospetti. Si intende sviluppare un modello di machine learning, basato sulle prime features estraibili dai dati anonimizzati ed un albero decisionale, per identificare pattern che precedono i furti e confrontare questo approccio con soluzioni tradizionali di monitoraggio GPS.

1.2. Monitoraggio e Analisi dei Dati

Dati Telemetrici e Machine Learning

Octo Telematics raccoglie dati in tempo reale relativi alla posizione, velocità e direzione dei veicoli. Questi dati vengono analizzati per identificare anomalie nei comportamenti di guida, come accelerazioni o fermate improvvise che potrebbero indicare un furto. Utilizzando tecniche

di machine learning, è possibile costruire modelli predittivi che identificano comportamenti anomali associati ai furti.

1.3 Tecniche di Analisi

L'uso di un modello di machine learning, come l'albero decisionale, consente di identificare le caratteristiche decisive che separano i viaggi normali da quelli sospetti. Queste caratteristiche includono la velocità, la geolocalizzazione e il comportamento del conducente. L'obiettivo è rilevare in tempo reale i furti e migliorare la prevenzione.

1.4 Monitoraggio in Tempo Reale e Sistemi di Allarme

Funzionalità Telematiche Avanzate

Octo Telematics offre un monitoraggio continuo dei veicoli e una serie di funzionalità che consentono di rilevare movimenti sospetti. Tra queste, il geofencing, che definisce aree di sicurezza e genera un allarme se il veicolo esce da queste aree, e la possibilità di immobilizzare il veicolo a distanza in caso di furto.

Notifiche e Interventi

Quando viene rilevato un movimento sospetto, il sistema invia notifiche in tempo reale al proprietario, alle forze dell'ordine e alle compagnie di assicurazione, consentendo un intervento rapido. Questo processo aiuta a ridurre il tempo necessario per il recupero del veicolo rubato.

1.5 Analisi dei Pattern nei Viaggi

Identificazione di Comportamenti Anomali

I pattern sospetti nei dati di viaggio possono includere deviazioni improvvise dal percorso, accelerazioni o frenate anomale, e fermate inattese in aree a rischio. Questi comportamenti sono spesso associati a tentativi di furto o a situazioni che indicano un furto imminente.

Approccio del Machine Learning

Per identificare questi comportamenti, si utilizzano modelli predittivi che analizzano variabili come la velocità, la durata del viaggio e la posizione geografica. Gli alberi decisionali sono utilizzati per separare i viaggi "normali" da quelli sospetti. Il modello è allenato utilizzando dati storici per identificare le caratteristiche più comuni nei furti.

1.6 Sfide Tecniche

Qualità dei Dati

Una delle sfide principali nell'analisi dei dati è garantire la qualità e la completezza delle informazioni raccolte dai dispositivi GPS. I dati devono essere precisi e aggiornati in tempo reale per essere utili nell'identificazione dei furti.

Scalabilità e Interpretabilità

Il sistema deve essere in grado di gestire grandi volumi di dati provenienti da milioni di veicoli. Inoltre, il modello di machine learning deve essere interpretabile, affinché gli operatori possano comprendere le decisioni del sistema e intervenire se necessario.

Elaborazione in Tempo Reale

Il sistema deve essere in grado di rilevare e rispondere a eventi sospetti in tempo reale, senza compromettere la velocità di elaborazione dei dati.

1.7 Soluzioni Esistenti e Vantaggi dell'Approccio Machine Learning

Le soluzioni esistenti di monitoraggio basato su GPS non sono sufficienti per rilevare autonomamente i pattern di furto. L'introduzione di un modello di machine learning consente di migliorare l'efficacia della rilevazione dei furti, identificando in modo più preciso i comportamenti sospetti e riducendo il rischio di furti non rilevati.

1.8 Conclusioni e Impatti

L'adozione di un modello di machine learning per la prevenzione dei furti basato su telematica avanzata rappresenta un significativo passo avanti rispetto alle soluzioni tradizionali. Non solo migliora la capacità di rilevare i furti prima che accadano, ma offre anche nuove opportunità per le compagnie assicurative e le forze dell'ordine di agire in tempo reale. L'analisi dei dati telematici non solo contribuisce alla prevenzione, ma offre anche una panoramica più completa e accurata dei comportamenti di guida, migliorando la sicurezza dei veicoli e riducendo i costi operativi per le assicurazioni.

L'approccio con machine learning, in particolare attraverso l'uso di alberi decisionali, fornisce un modello predittivo interpretabile che può essere utilizzato per affinare ulteriormente la prevenzione dei furti, creando una rete di sicurezza che va ben oltre il semplice recupero dei veicoli rubati.

2. Metodologia

2.1 Descrizione dei Dati

Fonti dei Dati: I dati utilizzati in questa ricerca sono stati forniti da Octo Telematics, azienda leader di settore nella raccolta e analisi di dati GPS. Octo Telematics ha messo a disposizione un dataset che include il monitoraggio di viaggi effettuati da veicoli in territorio italiano, comprensivo sia di viaggi associati a furti che non associati a furti. I dati sono stati correttamente anonimizzati al fine di proteggere la privacy degli utenti, nel pieno rispetto delle normative vigenti, per facilitare l'analisi dei pattern di viaggio legati a eventi di furto.

Caratteristiche dei Dati: Il dataset utilizzato è un file CSV denominato [`anonymised_complete.csv`](#), che contiene informazioni dettagliate sui viaggi dei veicoli monitorati. Il file include una serie di variabili che descrivono le caratteristiche dei viaggi, le coordinate GPS, la velocità, la qualità del segnale GPS e la presenza di eventi di furto. Le colonne del dataset originale, prima della pulizia, sono le seguenti:

- **Unnamed: 0.1, Unnamed: 0:** Colonne senza significato specifico, probabilmente generati durante la creazione del file.
- **id_terminal:** Identificativo univoco del terminale GPS installato nel veicolo.
- **position_number:** Numero progressivo della posizione rilevata durante il viaggio.
- **latitude, longitude:** Coordinate geografiche (latitudine e longitudine) della posizione rilevata.
- **speed:** Velocità del veicolo al momento della rilevazione.
- **quality:** Qualità del segnale GPS (indica precisione della rilevazione, in particolare: #3: precisione 10 m; #2 : fix 2 d(2 sfere) 100 m errore; #1, ti mostra l'unica posizione; #0, prima accensione).
- **io_status:** Indica lo stato del veicolo ed in particolare : #3 : Quadro off veicolo con spostamento #2 :Quadro on veicolo con spostamento #1 : Quadro off veicolo senza spostamento #0 : Quadro on veicolo senza spostamento
- **timestamp:** Data e ora della rilevazione della posizione.
- **data_furto:** Data in cui è stato registrato un furto, se applicabile.
- **data_apertura_dossier:** Data di apertura del dossier per la segnalazione del furto, se presente.
- **furto:** Variabile binaria che indica se il viaggio è associato a un furto (1) o meno (0).

Il dataset originale contiene un totale di **2.273.743 righe e 13 colonne**, con una dimensione di memoria di circa **225.5 MB**. Le colonne numeriche includono variabili di tipo intero e float, mentre alcune colonne (come le date e il timestamp) sono memorizzate come oggetti di tipo stringa.

Formato e Metodi di Raccolta: I dati sono stati raccolti attraverso dispositivi GPS installati sui veicoli, che hanno registrato in tempo reale la posizione, la velocità e altri parametri durante il viaggio. Ogni rilevazione è stata registrata con un timestamp preciso, e i dati sono stati successivamente esportati e forniti come file CSV. I dati includono anche informazioni relative agli incidenti di furto, identificando i viaggi che potrebbero essere associati a questi eventi.

Il dataset è stato anonimizzato per garantire la protezione della privacy dei soggetti monitorati, rimuovendo o mascherando eventuali informazioni identificabili. I dati sono stati poi utilizzati per analizzare i pattern di viaggio e per sviluppare modelli predittivi relativi agli eventi di furto, sfruttando tecniche di machine learning.

2.2 Preprocessing dei Dati

- Pulizia dei dati: gestione dei valori mancanti, outlier, errori nei dati.

Il preprocessing del dataset ha incluso i seguenti passaggi:

- Rimozione di colonne non rilevanti come indici e altre informazioni non utili per l'analisi.
- Creazione di una colonna '**datetime**' per gestire correttamente le informazioni temporali.

- Rimozione di una riga contenente 3 valori nulli, che non ha avuto un impatto significativo sul dataset.
- Eliminazione di un valore anomalo (valore '**9**' nella colonna '**quality**'), presumibilmente dovuto a un errore di battitura.
- Integrazione di diversi tipi di dati (dati temporali, spaziali, categorici).

Preparazione del dataset finalizzata all' analisi dei pattern di viaggi relativi a furti, descrizione dei poassaggi salienti:

Calcolo delle Pause e Creazione del Flag `long_pause`:

```
prova['time_diff'] = prova.groupby('id_terminal')['datetime'].diff()
# pause di almeno 15 min con velocità == 0

prova['long_pause'] = (prova["speed"] == 0) & (prova["time_diff"] >=
pd.Timedelta(minutes=15))
```

In questo passaggio, calcoliamo il tempo trascorso tra due rilevamenti consecutivi per ciascun veicolo (`id_terminal`) utilizzando il metodo `diff()` sulla colonna '`datetime`'.

Successivamente, creiamo una nuova colonna '`long_pause`' che identifica le pause significative, ovvero quelle in cui il veicolo si è fermato (`speed == 0`) per un periodo di almeno 15 minuti. Questo flag è utile per separare i periodi di sosta e identificare i viaggi (trips) in modo corretto.

Creazione dell'Identificativo del Viaggio (`trip_id`):

```
copia = prova.copy()

copia["trip_id"] = copia["long_pause"].cumsum()

copia
```

Creiamo una copia del dataframe originale per evitare modifiche non desiderate. Successivamente, utilizziamo la colonna '`long_pause`' per creare un identificativo univoco per ciascun viaggio. Il comando `cumsum()` calcola la somma cumulativa dei valori booleani nella colonna '`long_pause`', creando un nuovo '`trip_id`' che identifica i singoli viaggi, separando le pause significative (quando il veicolo rimane fermo per almeno 15 minuti).

Motivazione della Selezione dei Dati con `quality` inferiore a 3: In questa fase, il dataframe viene filtrato sulla base della **qualità dei dati** relativi alla geolocalizzazione (latitudine e longitudine). La **colonna 'quality'** indica l'affidabilità dei dati, con valori più bassi che

suggeriscono dati di scarsa qualità. La ragione per cui il filtro sulla qualità viene applicato **dopo** la definizione dei viaggi è che:

- Durante la creazione dei **trip** (basati sulle pause e sulla velocità), è importante considerare tutte le righe disponibili, anche quelle con '**quality**' < 3. Questo perché tali righe contengono informazioni temporali che sono necessarie per definire correttamente l'inizio e la fine di un viaggio. Escludere i dati con '**quality**' < 3 prima della definizione dei viaggi potrebbe portare a **incoerenze** nel calcolo della durata dei viaggi e nella segmentazione degli stessi.
- Solo una volta che i viaggi sono stati definiti e segmentati correttamente, è possibile applicare il filtro sulla **qualità** per eliminare le righe con dati di latitudine e longitudine non affidabili. Questo assicura che la qualità dei dati non influisca negativamente sulle caratteristiche geospaziali estratte dai viaggi.

Filtraggio Finale dei Dati sulla Base della Qualità:

```
prova = prova[prova["quality"] == 3]
```

Dopo aver estratto i **trip_id** e definito i viaggi, il filtro finale viene applicato per **escludere i dati con una qualità inferiore a 3**. Questo passaggio elimina le righe che contengono dati di latitudine e longitudine non affidabili, che potrebbero compromettere l'accuratezza delle analisi geospaziali e delle feature calcolate successivamente, come la distanza percorsa e la geolocalizzazione del veicolo durante i viaggi. Inoltre abbiamo controllato che il 60% dei dati ed oltre è costituito da valori di quality pari a 3 quindi l'eliminazione di valori inferiori non dovrebbe creare problemi di rappresentatività.

Si faccia riferimento alla funzione per creare i trips nel file Preprocessing.ipynb

2.3 Generazione di una mappe visive relative ai viaggi di furti e non furti e Heatmap

Attraverso la generazione di una mappa dei viaggi possiamo osservare come questi ultimi sono distribuiti sulla penisola italiana (isole comprese) ed individuare "ad occhio nudo" le aree principalmente osservate.

Grazie al codice colore è possibile identificare immediatamente le tipologie di evento target.

Generazione della mappa con percorsi GPS

- Utilizzando la libreria **folium**, viene creata una mappa interattiva.
- I percorsi vengono tracciati con colori differenti:
 - **Blu**: Viaggi senza furto (**furto=0**).
 - **Rosso**: Viaggi con furto (**furto=1**).
- La mappa viene salvata come file HTML (**map_furti.html**).

Generazione di mappe per singoli sispositivi con furto

- Viene impostato un limite massimo di dispositivi per cui generare mappe (`num_max_mappe_device=10`).
- Per ciascun dispositivo, viene creata una mappa centrata sulla mediana delle coordinate GPS.
- I percorsi sono tracciati separatamente in base alla presenza o meno di furti.
- Le mappe sono salvate come file HTML con il nome `{id_terminal}_map.html`.

Apertura della Mappa Generata

- Il codice salva la mappa e può essere aperta manualmente in un browser per l'analisi visiva dei percorsi.

```
#MAPPA SINGOLI TRIP per DEVICE soggetto a FURTO

num_max_mappe_device=10 #IMPORTANTE, NUMERO MASSIMO MAPPE DA GENERARE

colors=['deepskyblue','red']

lat_ct=new_dfmap['latitude'].quantile(0.5)

long_ct=new_dfmap['longitude'].quantile(0.5)

devices=new_dfmap['id_terminal'].unique()

for j in range(min(int(num_max_mappe_device),len(list(devices)))):
    m = folium.Map(location=[lat_ct,long_ct], zoom_start=8,tiles="OpenStreetMap")

    mapbounds=[[new_dfmap['latitude'].min(),new_dfmap['longitude'].min()],
    [new_dfmap['latitude'].max(),new_dfmap['longitude'].max()]]

    m.fit_bounds(mapbounds)

    for i in [0,1]:
        device=list(devices)[j]

        print(device)

        dfdevice=new_dfmap[new_dfmap['id_terminal']==device]

        dfdevice=dfdevice[dfdevice['furto']==i]

        dfdevice.reset_index(drop=True,inplace=True)
```

```

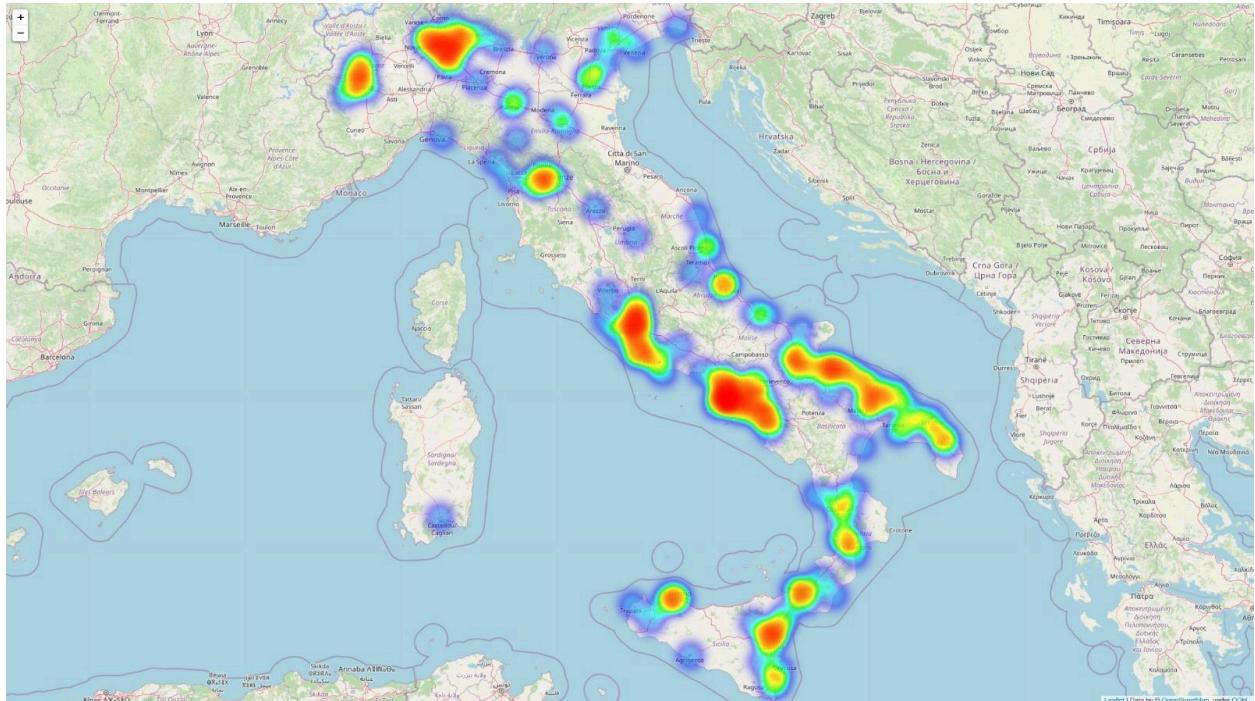
# dfdevice=dfdevice.loc[range(0,len(dfdevice),redux),:]
points = []
for n in range(min(int(1e6),len(dfdevice))):
    points.append([dfdevice.iloc[n]['latitude'],dfdevice.iloc[n]['longitude']])
if len(points)>1:
    folium.PolyLine(points, color=colors[i],
                    weight=5,opacity=0.8).add_to(m)
map_name=f'{device}_map.html'
m.save(map_name)
# webbrowser.open(r'file:///C:/Users/g.segurini/'+map_name)

```

Link Cliccabile per visualizzare e zoommare la mappa (per visualizzare, scaricare file e aprirlo in.html):

https://drive.google.com/file/d/1F6Xim5bEu10S7HOfqDwtXgczsZ-X_ZJc/view?usp=sharing

Viene riportata qui di seguito una mappa di calore furti ed il relativo link cliccabile:



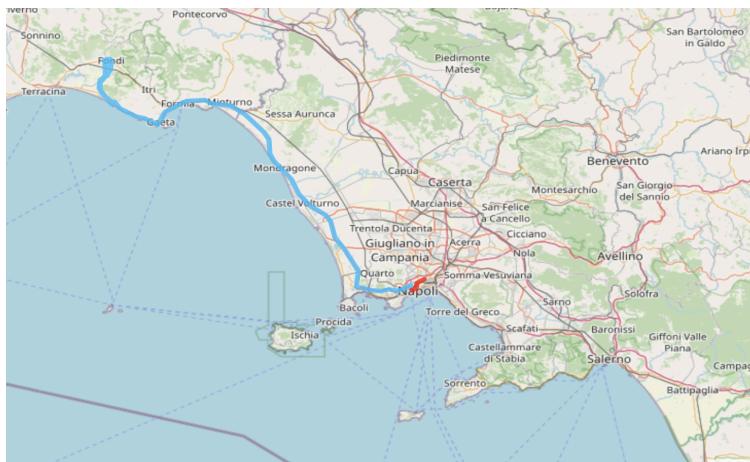
<https://drive.google.com/file/d/1nE-mGmjdG4TsJToaU4aSXWqL82PADRvl/view?usp=sharing>

L'intensità dei colori indica la densità degli eventi di furto registrati:

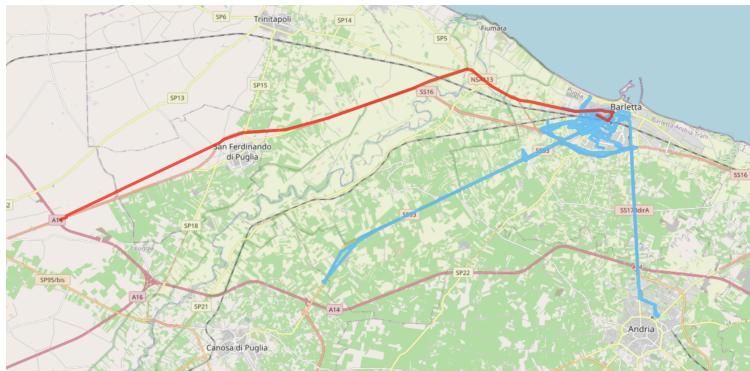
- Rosso e arancione → Aree con il maggior numero di furti segnalati
- Giallo e verde → Zone con una densità media di furti
- Blu → Zone con pochi furti registrati

La mappa mostra una distribuzione concentrata in alcune regioni specifiche, con punti caldi in Lombardia, Emilia-Romagna, Lazio, Campania e Sicilia. Questo suggerisce che i furti potrebbero essere più frequenti in determinate città o lungo specifici corridoi di transito.

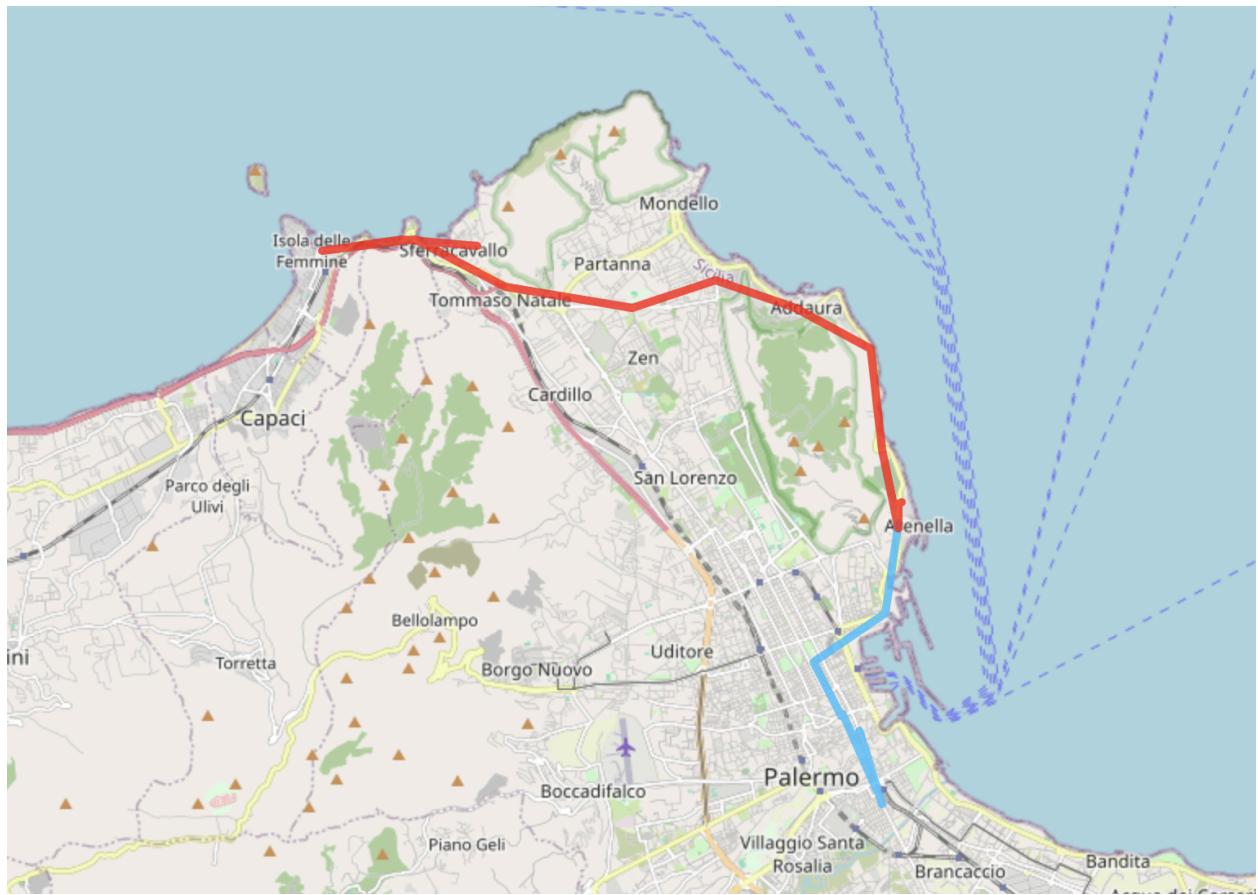
Qui invece immagini relativi a un esempi di trip per singoli veicoli soggetti a furto con relativo link alla mappa cliccabile:



<https://drive.google.com/file/d/1KdOjOsfwXalBrQyN5xwFl5mfZRvINeP/view?usp=sharing>



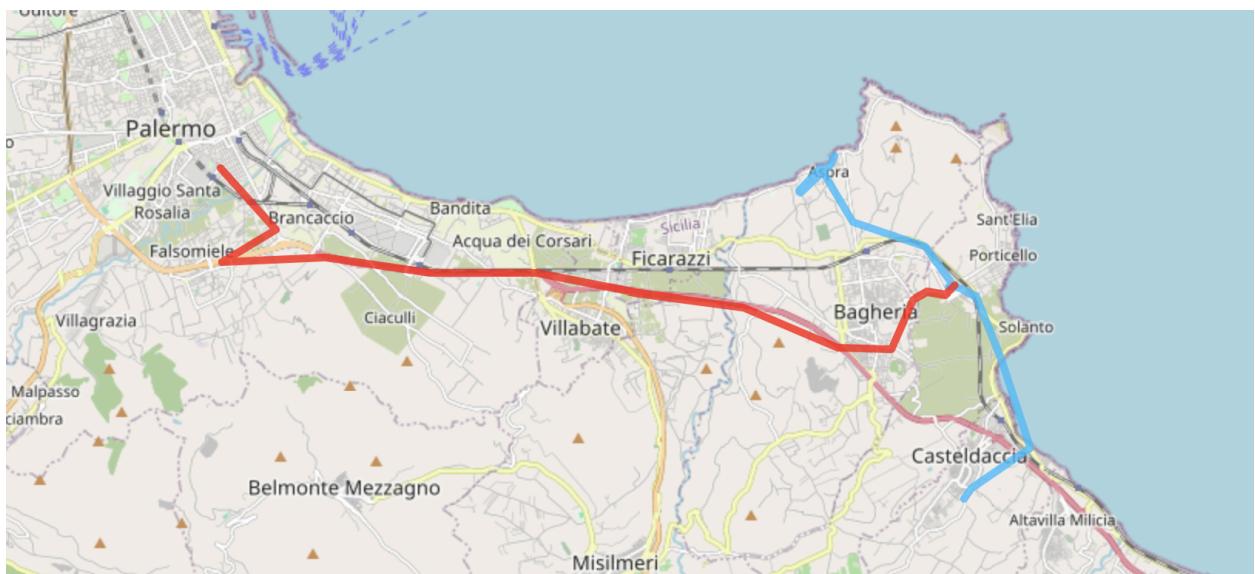
<https://drive.google.com/file/d/1GYZOjWsXgkg0zEbetcaAV23b5ETJmVCL/view?usp=sharing>



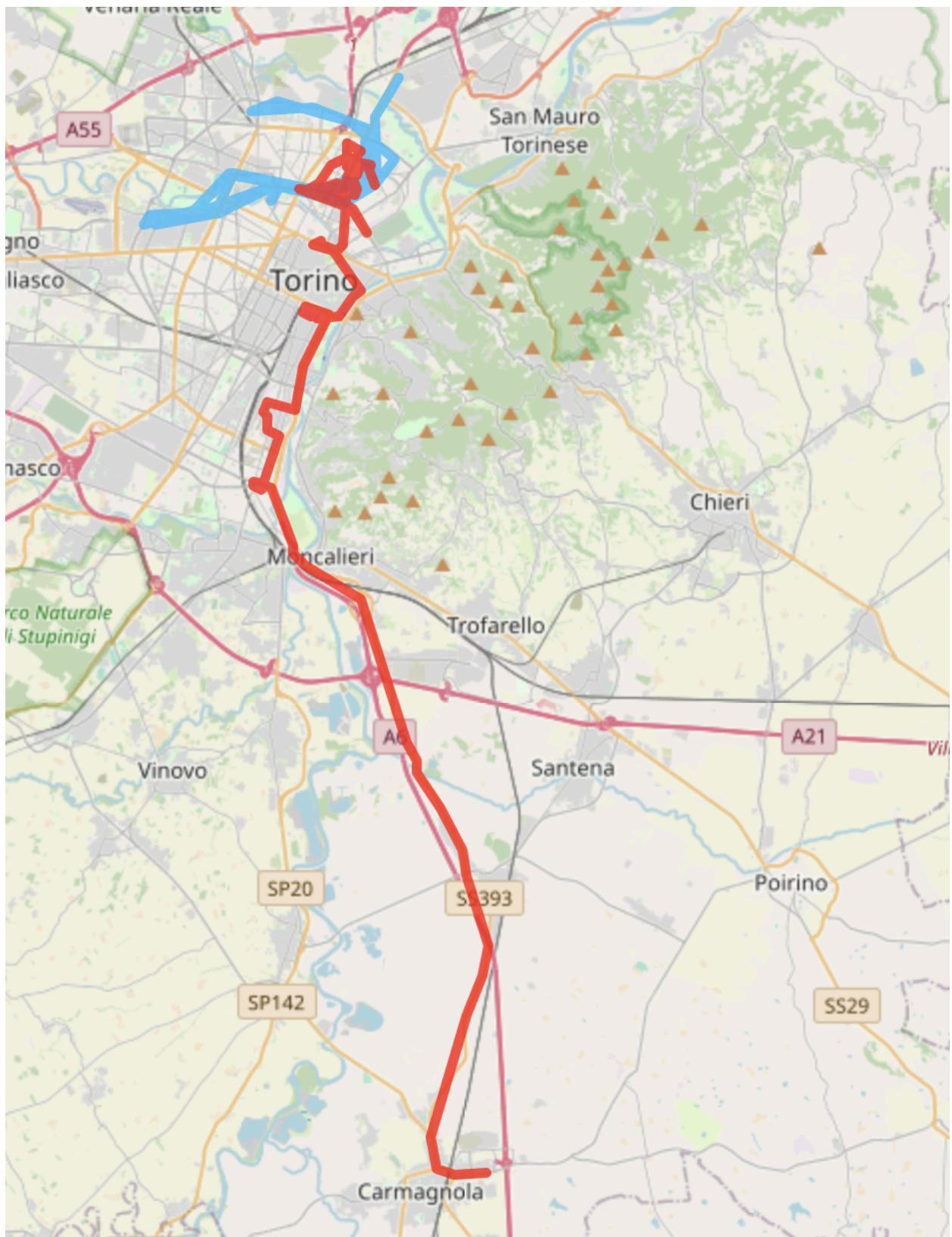
<https://drive.google.com/file/d/1sMnMs71M5FLZ7R10M4DjsKoI-9L9O2YR/view?usp=sharing>



<https://drive.google.com/file/d/1k-ivebme1rFuGgf7ndDTL8LqCE3wcVd/view?usp=sharing>



<https://drive.google.com/file/d/1PhbORAUHX5aEoaTeDDy1AxBlgi-RPIXo/view?usp=sharing>



<https://drive.google.com/file/d/1-cXqxU4omkBAtZOjmWkF8a-4b-pK6ec0/view?usp=sharing>

Nota: nelle mappe di esempi di trip non sono ovviamente riportati tutti i trip del veicolo ma solo quelli del furto (in rosso) e quelli precedenti nella giornata (in blu). Inoltre i furti sono un parziale di tutti i furti che avvengono in italia ma la distribuzione è rappresentativa perché i furti sono selezionati randomicamente quindi rappresentano un campione.

Per il procedimento di generazione delle mappe si faccia riferimento allo script qui :
https://docs.google.com/document/d/1aQeltQJqBGywKAHRxIIHbf4xofurKYV_3KIh6tpubzg/edit?usp=sharing

e questa integrazione :

https://drive.google.com/file/d/16qWC5fDB_noEwLOWysMH2bahSxUi5OJH/view?usp=sharing

2.4 Postprocessing, Matrice di Correlazione e catturabilità dell' effetto

Il presente report descrive il processo di preprocessing applicato a un dataset contenente informazioni sui viaggi di veicoli ottenuto dal preprocessing dei dati grezzi anonimizzati, al fine di prepararlo per un'analisi di machine learning. Il dataset originale includeva vari attributi relativi a tempo, posizione, velocità e un'indicazione di furto del veicolo.

Il dataset è stato pulito e arricchito con nuove feature temporali, migliorando la qualità dei dati per la fase successiva di analisi e modellazione di machine learning. La fase di analisi ha evidenziato relazioni deboli tra le variabili tramite la matrice di correlazione, ma un effetto misurabile nei furti in relazione all'ora del giorno, suggerendo la necessità di approfondimenti tramite ulteriori istogrammi.

Pulizia e Preparazione del Dataset

Rimozione di Colonne Non Necessarie

Le seguenti colonne sono state rimosse in quanto non necessarie per l'analisi:

- `id_terminal`: Identificativo del terminale, non rilevante per l'analisi.
- `Unnamed : 0`: Probabilmente un indice generato automaticamente.
- `quality`: Non specificato nel contesto, presumibilmente non utile.

Il `trip_id` è stato mantenuto per eventuali aggregazioni future.

Gestione dei Valori Mancanti

Il numero di valori mancanti è stato verificato utilizzando il metodo `df.isnull().sum()`. Questo passaggio aiuta a identificare eventuali problemi nei dati prima dell'analisi.

Conversione delle Date e Creazione di Variabili Temporali

Le colonne `start_time` e `end_time` sono state convertite in formato datetime per facilitarne l'uso:

```
df['start_time'] = pd.to_datetime(df['start_time'], errors='coerce')
```

```
df['end_time'] = pd.to_datetime(df['end_time'], errors='coerce')
```

Successivamente, sono state estratte diverse caratteristiche temporali:

- **Ora del giorno:** `start_hour, end_hour` (0-23)
- **Minuti:** `start_minute, end_minute` (0-59)
- **Giorno della settimana:** `start_weekday, end_weekday` (0=Lunedì, 6=Domenica)
- **Giorno del mese:** `start_day, end_day` (1-31)
- **Mese:** `start_month, end_month` (1-12)
- **Anno:** `start_year, end_year`

Calcolo della Durata del Viaggio

La colonna `trip_duration` è stata convertita in un formato timedelta per calcolare la durata del viaggio in minuti:

```
df['trip_duration'] = pd.to_timedelta(df['trip_duration'])
```

```
df['trip_duration_minutes'] = df['trip_duration'].dt.total_seconds() / 60
```

Eliminazione delle Colonne Ridondanti

Dopo aver estratto le informazioni temporali e calcolato la durata del viaggio, le colonne originali `start_time`, `end_time` e `trip_duration` sono state rimosse per ridurre la dimensionalità del dataset:

```
df = df.drop(columns=['start_time', 'end_time', 'trip_duration'])
```

Analisi della Correlazione

Calcolo della Matrice di Correlazione

Per valutare le relazioni tra le variabili, sono state calcolate due matrici di correlazione:

- **Correlazione di Pearson**, utile per identificare relazioni lineari tra le variabili.
- **Correlazione di Spearman**, più adatta a catturare relazioni monotone non lineari.

Le matrici di correlazione sono state visualizzate tramite heatmap:

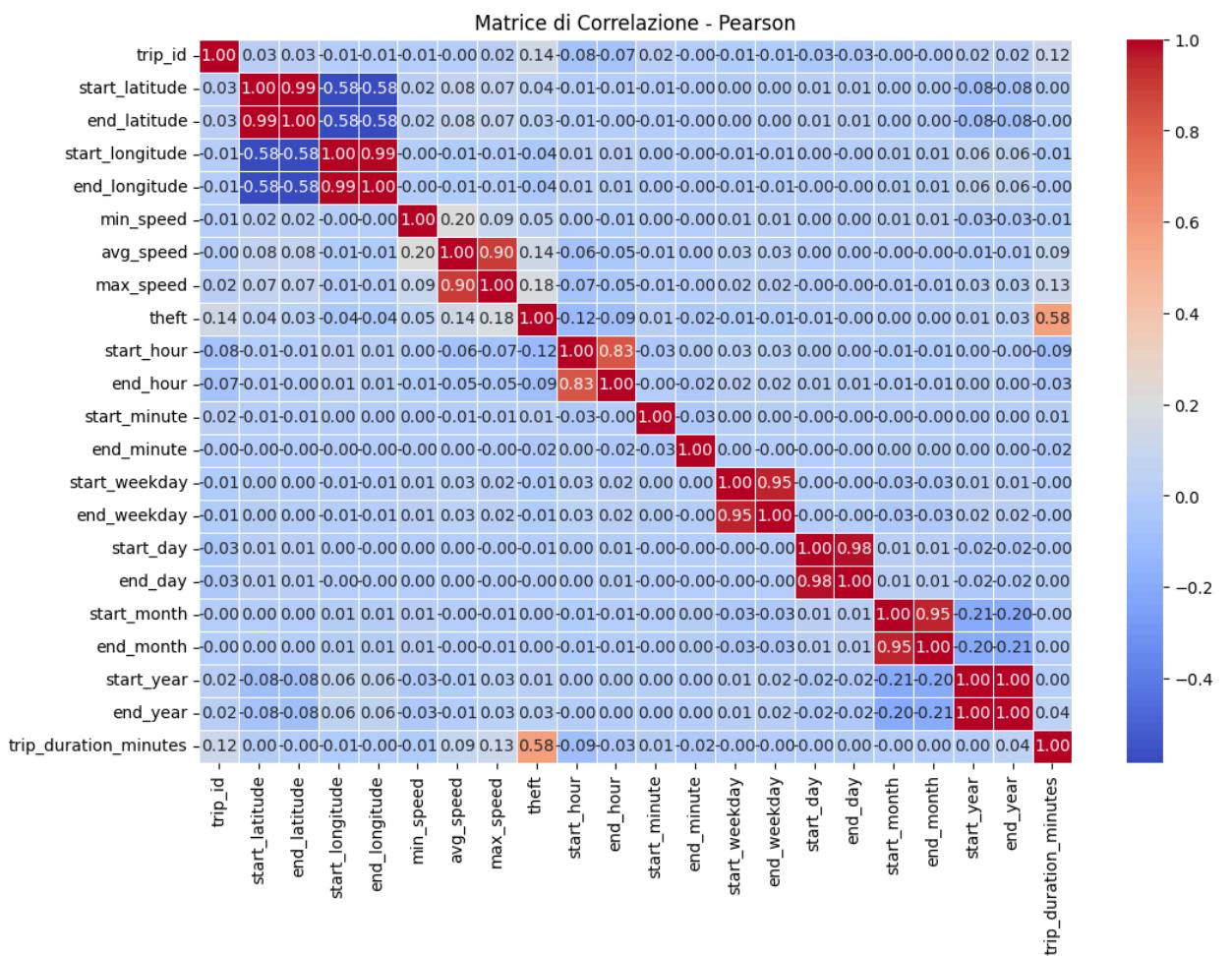
```
correlation_matrix = df.corr()

plt.figure(figsize=(12, 8))

sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)

plt.title('Matrice di Correlazione - Pearson')

plt.show()
```



```
spearman_corr = df.corr(method='spearman')
```

```
plt.figure(figsize=(12, 8))
```

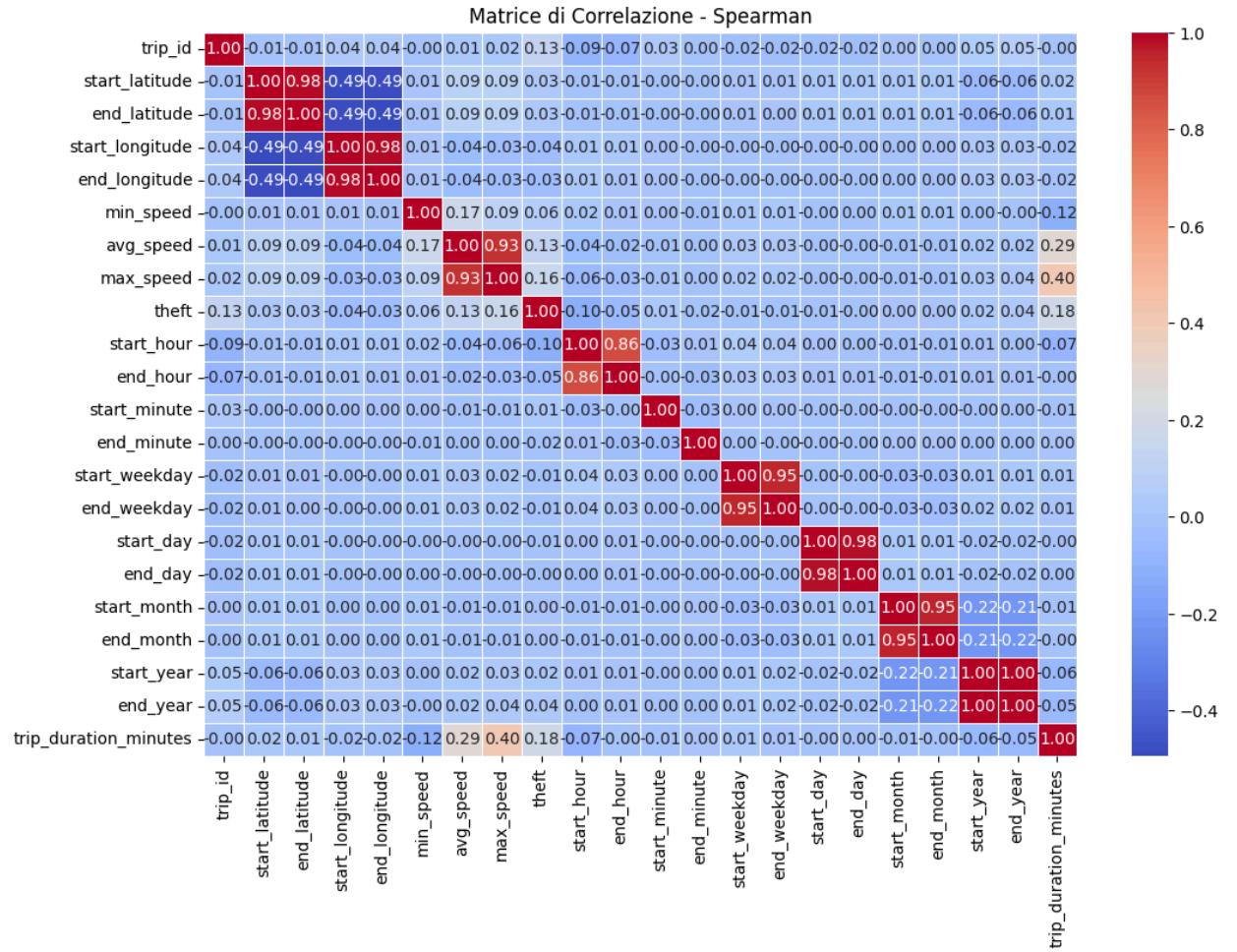
```

sns.heatmap(spearman_corr, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)

plt.title('Matrice di Correlazione - Spearman')

plt.show()

```



I risultati indicano una correlazione debole tra le variabili, ma suggeriscono la necessità di un'analisi più approfondita tramite istogrammi.

Analisi degli Iistogrammi

Per analizzare l'incidenza dei furti in funzione dell'ora del giorno, è stata eseguita un'aggregazione considerando solo la prima occorrenza di furto per ogni **trip_id** e ora di inizio (**start_hour**):

```
df = df.sort_values(by=['trip_id', 'start_hour'])
```

```

def count_first_theft(group):
    return (group['theft'] & ~group['theft'].shift(fill_value=False)).sum()

df_thefts_per_groupId_and_hour = df.groupby(['trip_id',
    'start_hour']).apply(count_first_theft).reset_index(name='theft_count')

final_result =
df_thefts_per_groupId_and_hour.groupby('start_hour')['theft_count'].sum().reset_index()

```

Il numero di furti per ora è stato visualizzato tramite un istogramma:

```

plt.figure(figsize=(10, 5))

plt.bar(final_result['start_hour'], final_result['theft_count'], color='red', alpha=0.7)

plt.xlabel("Ora di inizio (start_hour)")

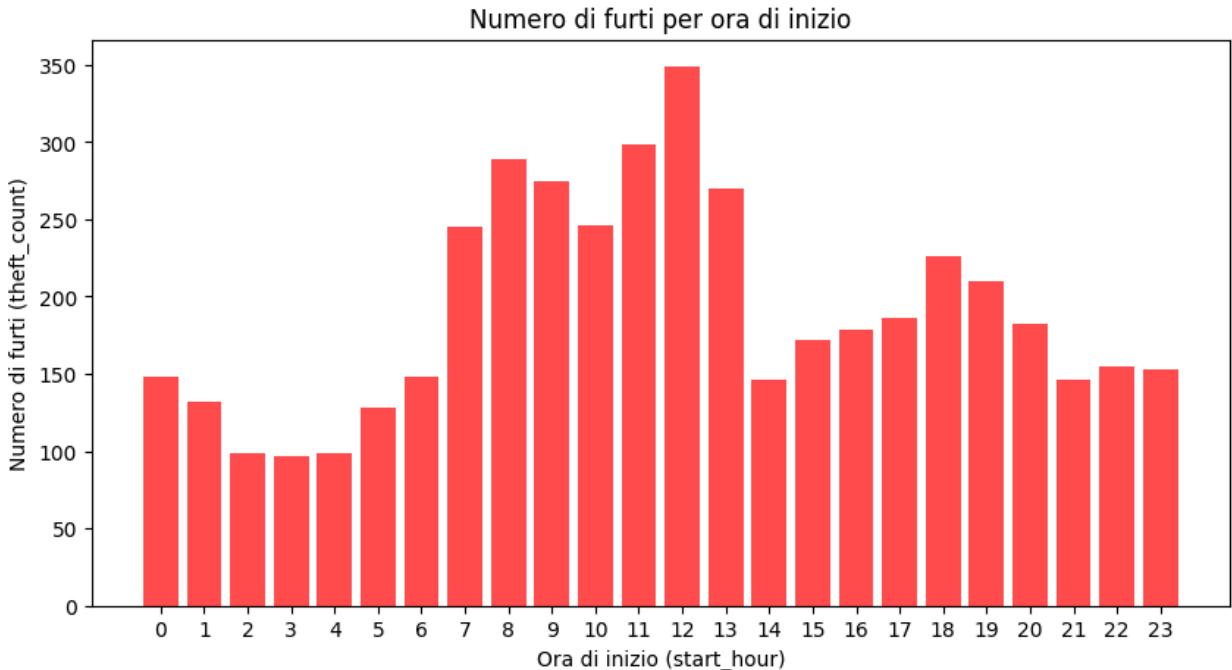
plt.ylabel("Numero di furti (theft_count)")

plt.title("Numero di furti per ora di inizio")

plt.xticks(final_result['start_hour'])

plt.show()

```



I risultati mostrano che, sebbene la correlazione tra le variabili sia debole, esiste un effetto rilevabile nei dati che giustifica ulteriori analisi attraverso la generazione di istogrammi più specifici.

2.5 Analisi Esplorativa e Statistiche relative ai Pattern (altri istogrammi, scrivere cosa significa ogni grafico)

L'obiettivo di questa analisi è esplorare e visualizzare la relazione tra gli orari di inizio dei viaggi e il tasso di furti, nonché esaminare i viaggi con velocità media elevata per determinare se ci sono correlazioni con i furti.

Preparazione dei Dati: Il dataset utilizzato contiene diverse colonne tra cui:

- `start_hour`: l'ora di inizio del viaggio.
- `theft`: un indicatore binario che segnala se il viaggio è stato un furto (1 per furto, 0 per non furto).
- `average_speed`: la velocità media del viaggio in km/h.

Analisi del Tasso di Furti per Ora del Giorno:

Il primo step è stato calcolare il tasso di furti per ogni ora del giorno. Questo è stato ottenuto raggruppando i dati per la colonna `start_hour` e calcolando la media della colonna `theft`, che restituisce una percentuale di furti per ogni ora. Il codice per questa operazione è il seguente:

```
hourly_theft_rate = df.groupby('start_hour')['theft'].mean() * 100
```

Questo produce una serie che mostra la percentuale di furti per ogni ora del giorno, con valori che variano tra 0% e 100%. Successivamente, è stato calcolato il tasso medio di furti su tutto il dataset, che è risultato essere circa il 9%:

```
average_theft_rate = df['theft'].mean() * 100
```

La visualizzazione del tasso di furti per ora è stata realizzata utilizzando un grafico a linee, con una linea rossa orizzontale che rappresenta la media dei furti. Il codice per la visualizzazione è il seguente:

```
plt.plot(hourly_theft_rate.index, hourly_theft_rate.values,
marker='o')

plt.axhline(y=average_theft_rate, color='r', linestyle='--',
label=f"Media: {average_theft_rate:.2f}%)
```

Questo grafico ha mostrato chiaramente che tra le 3:00 e le 5:00 del mattino, il tasso di furti è significativamente più alto della media, superando il 40%, mentre la media generale dei furti è stata di circa il 9%.

Identificazione delle Ore con Tasso di Furti Elevato:

Il codice ha anche permesso di identificare specificamente le ore in cui il tasso di furti è maggiore del 40%. Questo è stato fatto utilizzando il seguente filtro:

```
high_theft_hours = hourly_theft_rate[(hourly_theft_rate > 40)]
```

I risultati di questa analisi sono stati utilizzati per evidenziare le ore in cui il rischio di furto è particolarmente elevato.

Analisi delle Velocità per i Furti con Alta Velocità:

Per indagare ulteriormente sul comportamento dei furti, è stata effettuata un'analisi delle velocità medie dei viaggi. In particolare, sono stati esaminati i viaggi con velocità media superiore a 150 km/h per determinare se ci fosse una connessione con i furti. Il filtro per identificare questi viaggi è stato:

```
high_speed_thefts = df[df['average_speed'] > 150]

high_speed_thefts_furti = high_speed_thefts[high_speed_thefts['theft'] == 1]
```

Questo ha restituito i viaggi con velocità superiore a 150 km/h che sono anche stati classificati come furti. Il codice ha quindi determinato che tutti e quattro i viaggi con velocità media superiore a 150 km/h erano furti.

Un grafico a dispersione è stato utilizzato per visualizzare questi viaggi, mostrando l'orario di inizio del viaggio e la velocità media per ciascuno di essi:

```
plt.scatter(high_speed_thefts_furti['start_hour'],
            high_speed_thefts_furti['average_speed'], color='red')
```

Questo grafico ha permesso di evidenziare la correlazione tra velocità elevata e furto, confermando che tutti i viaggi con velocità superiore a 150 km/h sono stati furti.

Dalla prima analisi esplorativa, sono emersi i seguenti risultati:

- Il tasso di furti varia significativamente in base all'orario, con picchi tra le 3:00 e le 5:00 del mattino, dove il tasso di furti supera il 40%, contro una media di circa il 9%.
- È stata identificata una correlazione tra velocità elevata e furti: i soli quattro viaggi con velocità media superiore a 150 km/h sono tutti associati a furti.

Questi risultati suggeriscono che ci potrebbero essere fattori temporali e comportamentali (come l'alta velocità) che influenzano la probabilità di furto, e potrebbero essere utili per ulteriori analisi o per la costruzione di modelli predittivi.

Analisi della Distribuzione dei Furti per Ora del Giorno

Obiettivo: L'obiettivo di questa analisi è esaminare la distribuzione dei furti durante le diverse ore del giorno, confrontando le occorrenze di furti veri (con `theft=True`) e false (con `theft=False`) e calcolando la percentuale di furti per ogni ora.

Descrizione del Dataset: Il dataset in esame contiene le seguenti colonne:

- `trip_id`: identificatore univoco del viaggio.
- `start_latitude`, `end_latitude`, `start_longitude`, `end_longitude`: coordinate geografiche di inizio e fine viaggio.

- `min_speed`, `avg_speed`, `max_speed`: velocità minima, media e massima durante il viaggio.
- `theft`: indicatore binario che segnala se il viaggio è stato un furto (1 per furto, 0 per non furto).
- `start_hour`, `end_hour`: ore di inizio e fine del viaggio.
- Altre colonne relative ai giorni, mesi, anni, durata del viaggio, ecc.

Analisi delle Occorrenze di Furto per Ora del Giorno:

L'analisi si concentra sul calcolo delle occorrenze di furto per ogni ora del giorno, suddivise in due categorie: furti veri e furti falsi.

- **Furti veri (`theft=True`)**: Si raggruppano i dati per `trip_id` e `start_hour`, e si contano solo le prime occorrenze di `theft=True` per ogni gruppo, utilizzando la funzione `count_first_theft` che è definita come segue:

```
def count_first_theft(group):
    return (group['theft'].shift(fill_value=True) &
~group['theft']).sum()
```

Questa funzione conta le transizioni da `False` a `True` in modo da catturare solo il primo furto in ogni viaggio. I risultati vengono aggregati per ogni ora utilizzando il metodo `groupby`:

```
df_thefts_per_groupId_and_hour = df.groupby(['trip_id',
'start_hour']).apply(count_first_theft).reset_index(name='theft_count_true')
thefts_per_hour =
df_thefts_per_groupId_and_hour.groupby(['start_hour']).sum().reset_index()
```

Si calcola quindi il totale complessivo dei furti veri (`total_thefts_count`) e la percentuale di furti veri per ogni ora del giorno, come segue:

```
thefts_per_hour['thefts_percentage_true'] =
(thefts_per_hour['theft_count_true'] / total_thefts_count *
100).round(2)
```

- **Furti falsi (`theft=False`)**: In modo analogo, si conta il numero di transizioni da `True` a `False` per ciascun viaggio, utilizzando la funzione `count_first_false_2`:

```

def count_first_false_2(group):
    return (group['theft'].shift(fill_value=True) &
~group['theft']).sum()

```

I dati vengono poi aggregati per `start_hour` in maniera simile a quanto fatto per i furti veri:

```

df_thefts_per_groupId_and_hour_false = df.groupby(['trip_id',
'start_hour']).apply(count_first_false_2).reset_index(name='theft_count_false')
thefts_per_hour_false =
df_thefts_per_groupId_and_hour_false.groupby(['start_hour']).sum().reset_index()

```

Viene calcolata anche la percentuale di furti falsi per ogni ora:

```

thefts_per_hour_false['thefts_percentage_false'] =
(thefts_per_hour_false['theft_count_false'] / total_thefts_count_false
* 100).round(2)

```

Unione dei Dati e Preparazione per la Visualizzazione:

I dati relativi ai furti veri e falsi vengono uniti utilizzando un merge sulla colonna `start_hour`:

```

merged_df = thefts_per_hour.merge(thefts_per_hour_false,
on='start_hour', suffixes=('', '_drop'))
merged_df = merged_df.drop(columns=['start_hour_drop'],
errors='ignore')

```

Visualizzazione dei Risultati:

Per rappresentare graficamente la distribuzione dei furti per ora, è stato creato un grafico a barre affiancate che mostra la percentuale di furti veri (in rosso) e falsi (in blu) per ogni ora del giorno. Il grafico è stato creato utilizzando `matplotlib` con il seguente codice:

```

fig, ax = plt.subplots(figsize=(10, 6))

# Barre per True (Rosso)

```

```

bar_true = ax.bar(x - bar_width / 2,
merged_df['thefts_percentage_true'], width=bar_width, color='red',
label='True')

# Barre per False (Blu)
bar_false = ax.bar(x + bar_width / 2,
merged_df['thefts_percentage_false'], width=bar_width, color='blue',
label='False')

ax.set_xlabel('Ore', fontsize=12)
ax.set_ylabel('Percentuale di Furti', fontsize=12)
ax.set_title('Distribuzione dei Furti per Ora', fontsize=14)
ax.set_xticks(x) # Posizione delle etichette sulle ascisse
ax.set_xticklabels(merged_df['start_hour']) # Imposta le ore sulle
ascisse
ax.legend(title="Tipo di Furto", loc="upper left", fontsize=10)

plt.show()

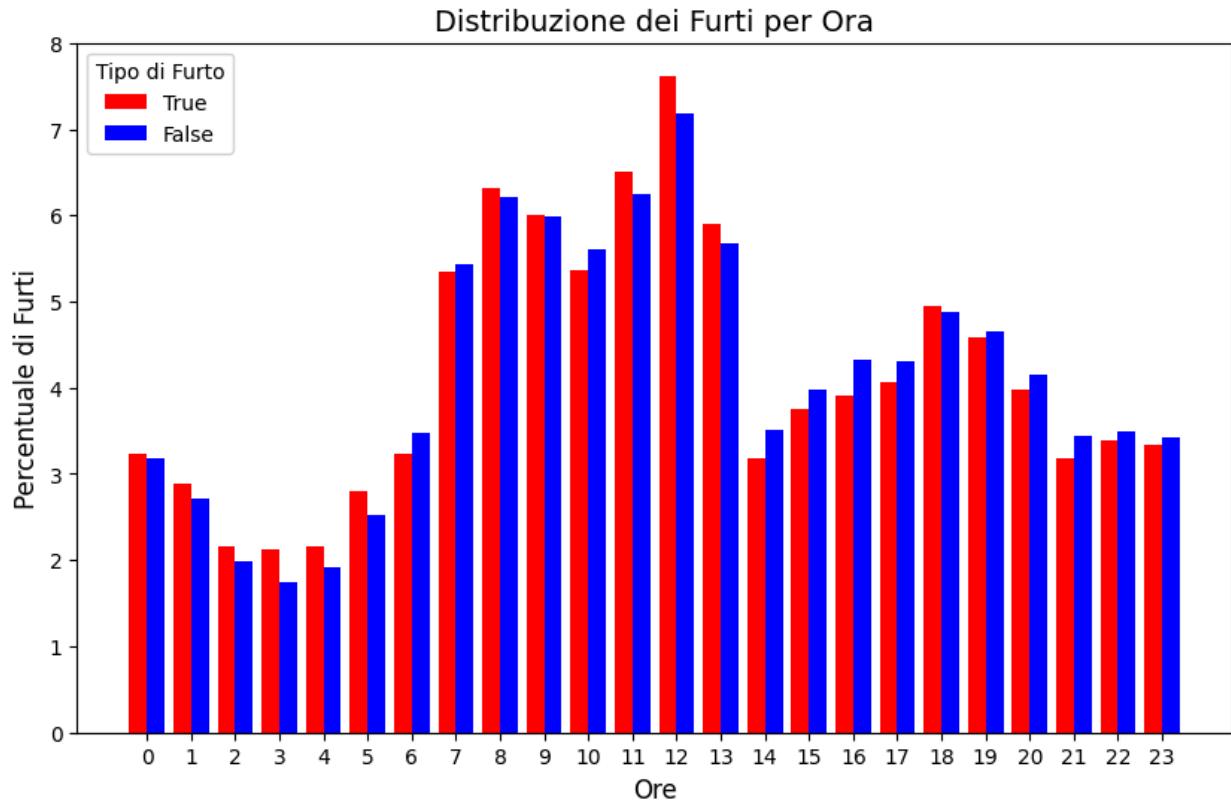
```

L'analisi ha prodotto un quadro chiaro della distribuzione dei furti per ora del giorno, distinguendo tra furti veri e falsi. I risultati mostrano le percentuali di furti veri e falsi per ogni ora, evidenziando eventuali tendenze temporali.

In particolare, l'analisi consente di:

- Identificare le ore del giorno con un'alta percentuale di furti veri.
- Confrontare la frequenza dei furti falsi in relazione a quella dei furti veri.

Il grafico risultante è utile per visualizzare le tendenze temporali e può essere utilizzato per fare previsioni o ottimizzare le strategie di sicurezza in base agli orari a maggiore rischio.



A questo punto ho iterato per tutte le colonne `cols_to_hist`:

Il codice definito nella funzione `hist_plots()` segue una struttura ripetitiva per ciascuna variabile di interesse, che viene fornita nella lista `cols_to_hist`. Per ogni variabile, vengono eseguiti i seguenti passaggi:

Furti veri (`theft=True`): Viene raggruppato il dataset per `trip_id` e per la variabile corrente (ad esempio, `start_hour`, `start_day`, ecc.), per applicare la funzione `count_first_theft` che conta solo la prima occorrenza di un furto vero per ciascun viaggio. Successivamente, il totale dei furti per quella variabile viene calcolato, e la percentuale di furti per ciascun valore della variabile viene aggiunta come nuova colonna.

```
df_thefts_per_groupId_and_hour = df.groupby(['trip_id',
                                             col]).apply(count_first_theft).reset_index(name='theft_count_true')
thefts_per_hour =
df_thefts_per_groupId_and_hour.groupby([col]).sum().reset_index()
total_thefts_count = thefts_per_hour['theft_count_true'].sum()
thefts_per_hour['thefts_percentage_true'] =
(thefts_per_hour['theft_count_true'] / total_thefts_count *
100).round(2)
```

Furti falsi (theft=False): Un'analoga operazione viene effettuata per i furti falsi, utilizzando la funzione `count_first_false_2`, che conta le transizioni da `True` a `False`. Anche in questo caso, si calcola il totale dei furti falsi per la variabile e si calcola la percentuale di furti falsi per ciascun valore della variabile.

```
df_thefts_per_groupId_and_hour_false = df.groupby(['trip_id', col]).apply(count_first_false_2).reset_index(name='theft_count_false')
thefts_per_hour_false =
df_thefts_per_groupId_and_hour_false.groupby([col]).sum().reset_index()
total_thefts_count_false =
thefts_per_hour_false['theft_count_false'].sum()
thefts_per_hour_false['thefts_percentage_false'] =
(thefts_per_hour_false['theft_count_false'] / total_thefts_count_false * 100).round(2)
```

Unione dei Dati: I dati relativi ai furti veri e falsi per ciascuna variabile vengono uniti in un singolo DataFrame utilizzando un `merge` sulla variabile specifica. Viene successivamente rimossa la colonna duplicata se presente.

```
merged_df = thefts_per_hour.merge(thefts_per_hour_false, on=col, suffixes=('', '_drop'))
merged_df = merged_df.drop(columns=[str(col) + '_drop'], errors='ignore')
```

Visualizzazione dei Dati: Per ciascuna variabile, vengono generati istogrammi con barre affiancate che mostrano la percentuale di furti veri (in rosso) e furti falsi (in blu). Il grafico include etichette per le ascisse, le ordinate, e una legenda per distinguere tra i due tipi di furto.

```
bar_width = 0.4
x = np.arange(len(merged_df[col])) # Indici delle ore

fig, ax = plt.subplots(figsize=(18, 15))

bar_true = ax.bar(x - bar_width/2,
merged_df['thefts_percentage_true'], width=bar_width, color='red',
label='True')
```

```

bar_false = ax.bar(x + bar_width/2,
merged_df['thefts_percentage_false'], width=bar_width, color='blue',
label='False')

ax.set_xlabel(col, fontsize=12)
ax.set_ylabel('Percentuale di Furti', fontsize=12)
ax.set_title(f'Distribuzione dei Furti per {col}', fontsize=14)
ax.set_xticks(x) # Posizione delle etichette sulle ascisse
ax.set_xticklabels(merged_df[col]) # Imposta le ore sulle ascisse
ax.legend(title="Tipo di Furto", loc="upper left", fontsize=10)

plt.show()

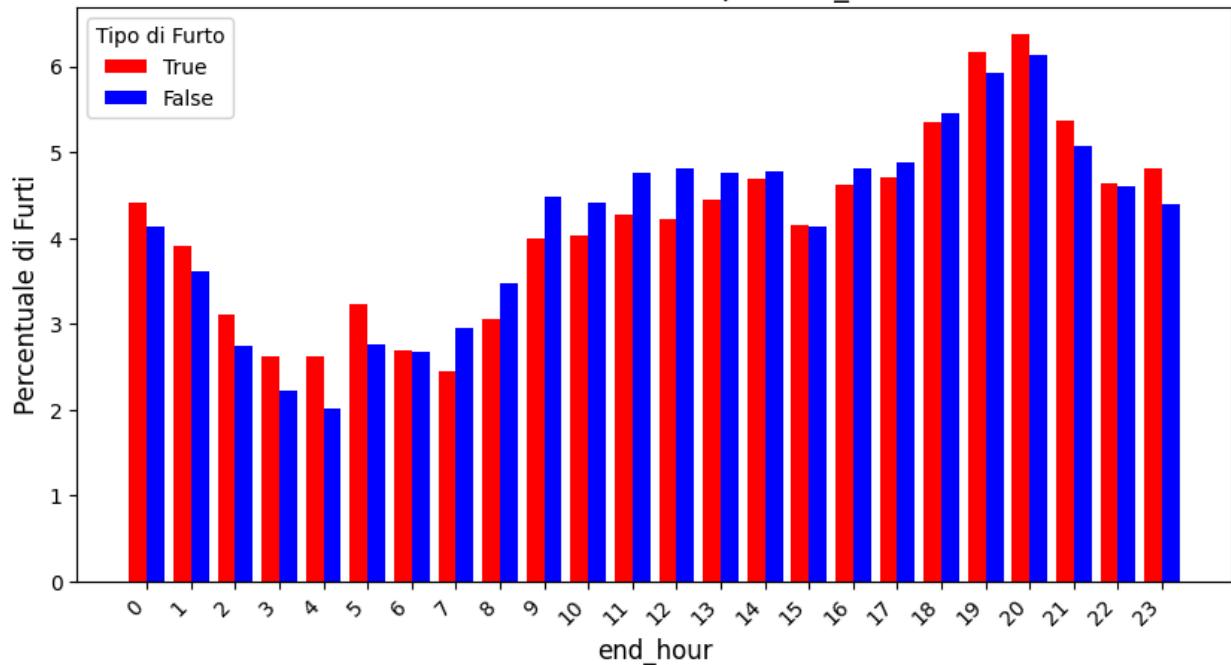
```

Il codice ha prodotto istogrammi per ciascuna variabile presente in `cols_to_hist`, `cols_to_hist = ['avg_speed', 'trip_duration_minutes', 'start_hour', 'end_hour']`, mostrando le percentuali di furti veri e falsi. Questi grafici sono stati utili per visualizzare le tendenze temporali o comportamentali legate ai furti in base alle variabili scelte, come ad esempio l'ora del giorno, il giorno della settimana, o la durata del viaggio.

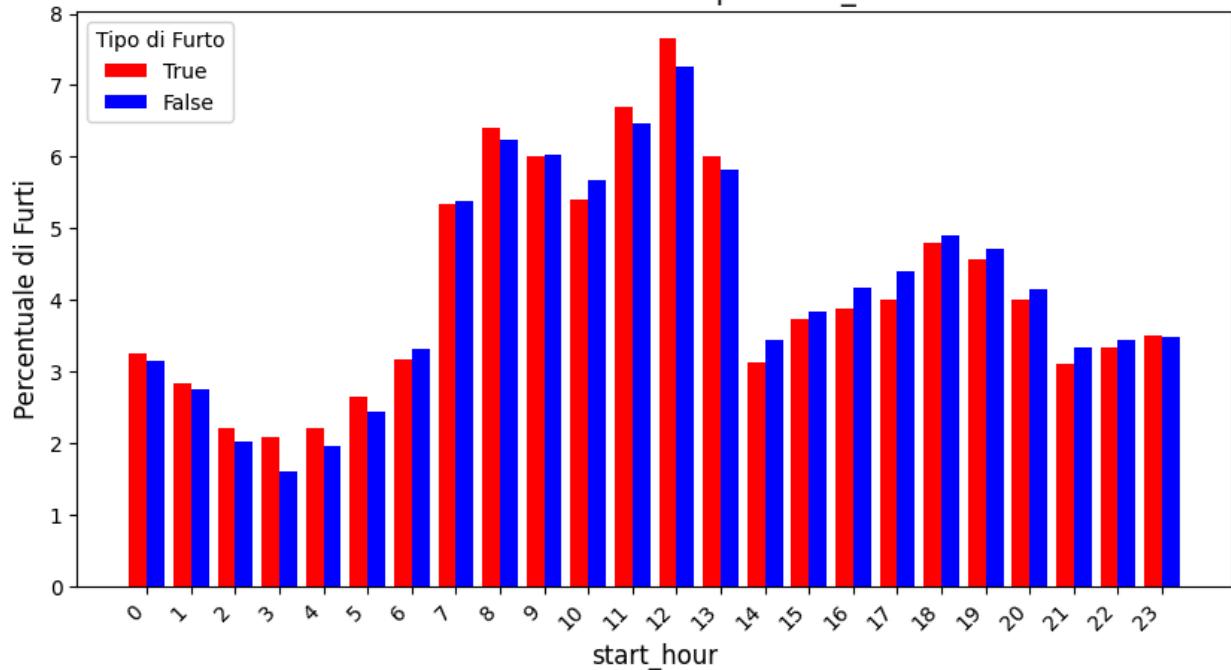
Questi istogrammi sono strumenti utili per esplorare e comprendere la distribuzione dei furti in base a vari fattori. Forniscono una visione chiara su come i furti veri e falsi si distribuiscono tra diverse categorie, e possono essere utilizzati per identificare pattern significativi che potrebbero influire sulle strategie di sicurezza.

Di seguito gli istogrammi ottenuti prima di aver trattato gli outliers.

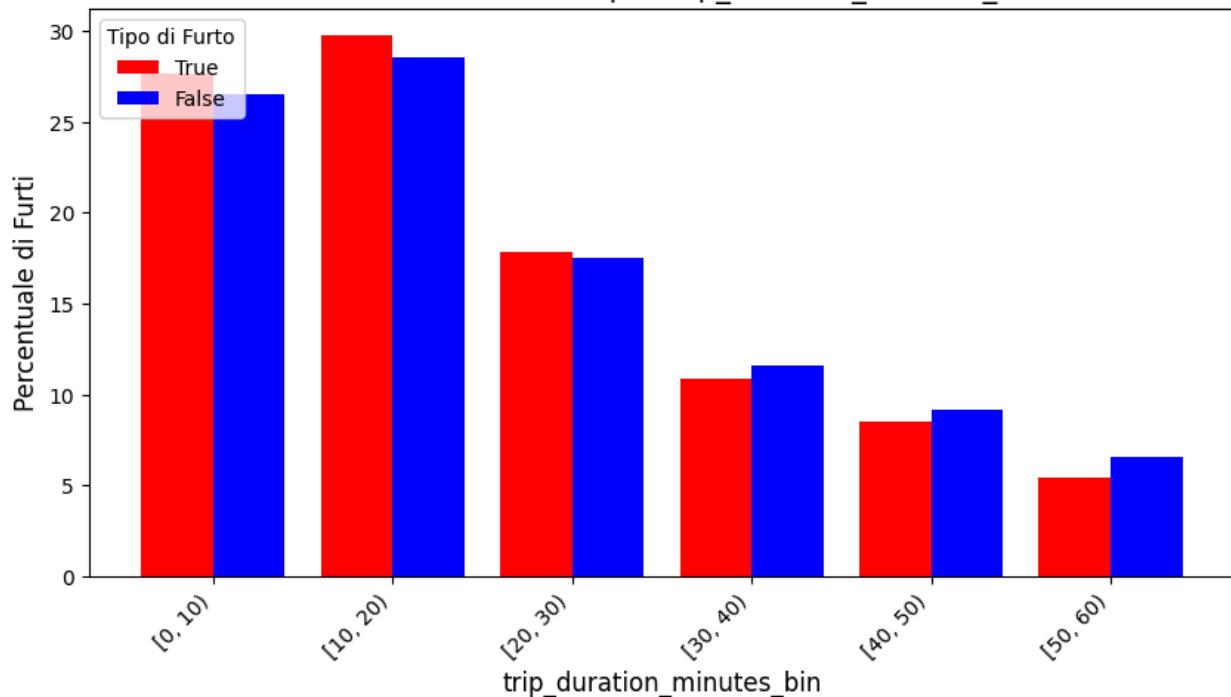
Distribuzione dei Furti per end_hour



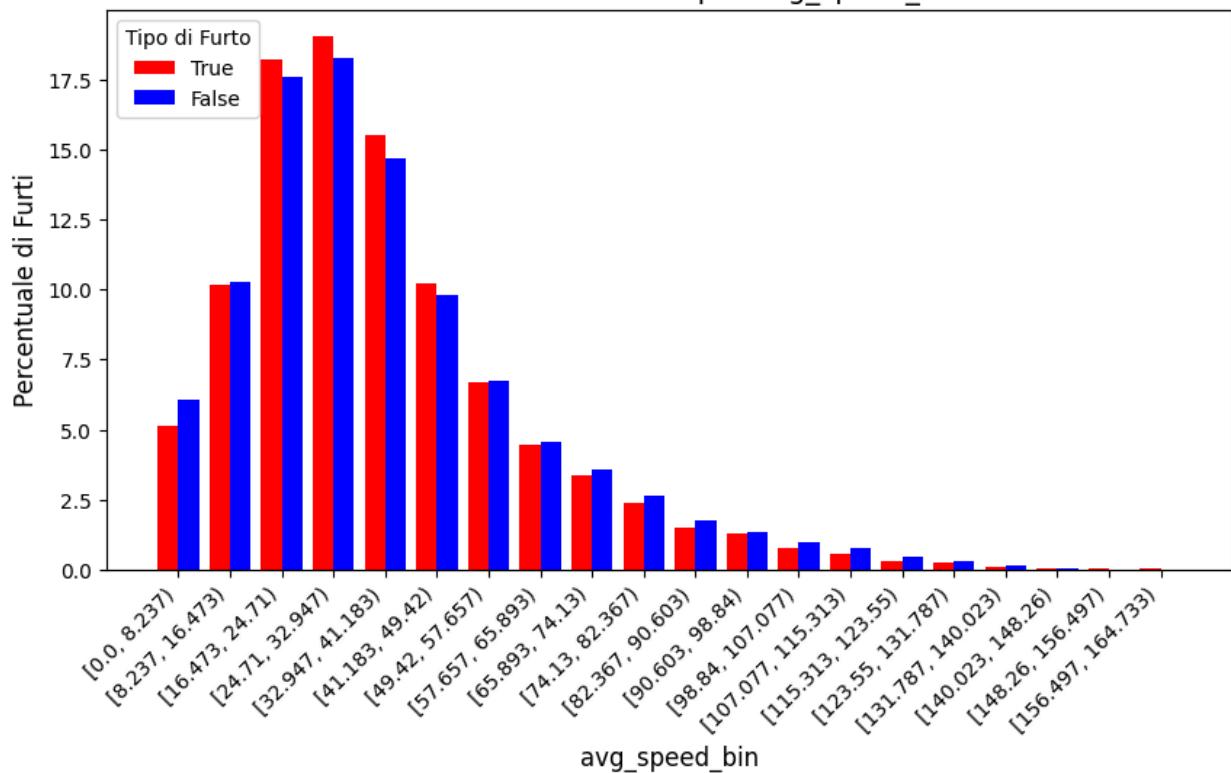
Distribuzione dei Furti per start_hour



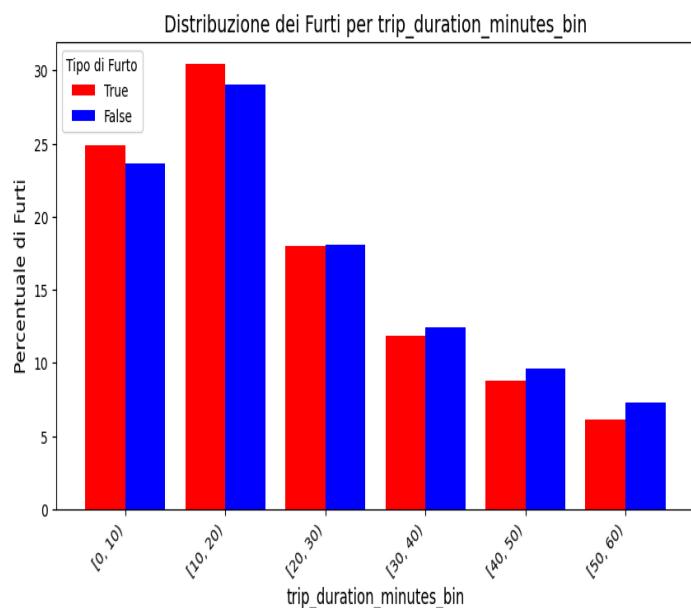
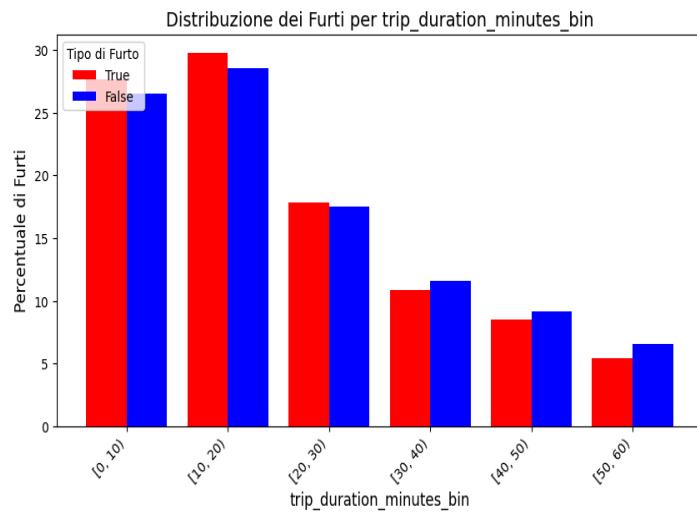
Distribuzione dei Furti per trip_duration_minutes_bin



Distribuzione dei Furti per avg_speed_bin



Qui di seguito si prosegue al confronto e commento tecnico tra coppie di grafici della stessa tipologia prima e dopo trattamento degli outliers con metodo IQR.



Interpretazione del primo grafico (senza trattamento degli outlier)

- La distribuzione dei furti (True e False) segue un andamento simile, con una maggiore concentrazione nei primi intervalli.
- Circa il 60% dei furti avviene nei primi 20 minuti di viaggio.
- Le percentuali di furti veri e falsi si mantengono molto vicine, con una leggera preponderanza di furti veri nelle prime due fasce.
- Per viaggi più lunghi, la percentuale di furti diminuisce progressivamente.

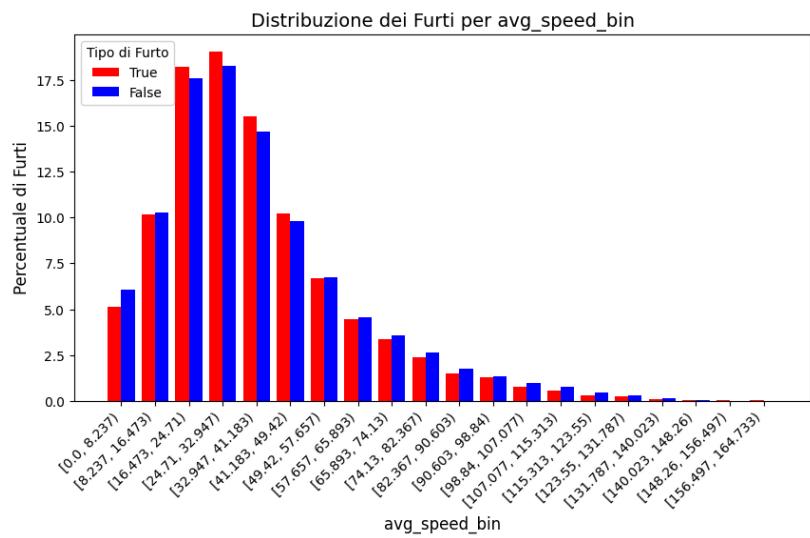
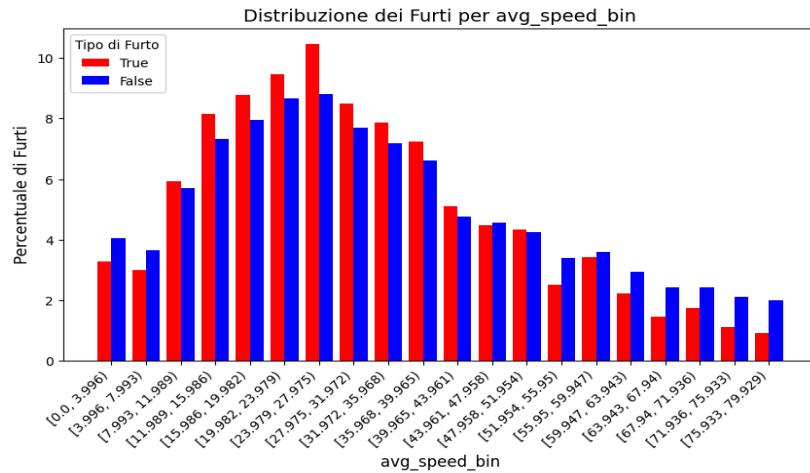
Interpretazione del secondo grafico (dopo rimozione outlier con IQR)

- Se l'algoritmo IQR ha rimosso valori estremi, è probabile che i viaggi molto lunghi (oltre 60 minuti) siano stati eliminati.
- Dovremmo osservare una maggiore concentrazione nei bin iniziali (0-30 min), perché i viaggi più brevi sono meno soggetti a essere considerati outlier.
- Se la distribuzione cambia rispetto al primo grafico, potremmo notare una variazione delle percentuali, soprattutto nei bin più lunghi.

Confronto tra i due grafici

- Se il secondo grafico mostra una maggiore concentrazione nei primi bin, significa che i viaggi molto lunghi erano influenzati da dati anomali.
- Se le proporzioni furti veri vs. falsi cambiano, allora gli outlier potrebbero aver influenzato di più un tipo di viaggio rispetto all'altro.

Quindi, analizzando anche i viaggi senza furto, posso confermare che la breve durata dei viaggi non è una caratteristica esclusiva dei furti, ma è comune anche nei viaggi normali. Questo significa che, per individuare furti in base alla durata, servirebbero altre variabili a supporto.



- La distribuzione delle velocità è più ampia, con valori che raggiungono oltre 160 km/h.
- La percentuale di viaggi per furti e non furti segue una distribuzione simile, con un picco tra 40-50 km/h.
- Tuttavia, ci sono code più lunghe verso velocità elevate, suggerendo la presenza di outlier a velocità molto alte.
- Il numero di eventi si riduce rapidamente sopra i 70-80 km/h, ma ci sono ancora dati presenti oltre i 100 km/h, indicando la necessità di un trattamento per gli outlier.

Distribuzione dopo il trattamento IQR

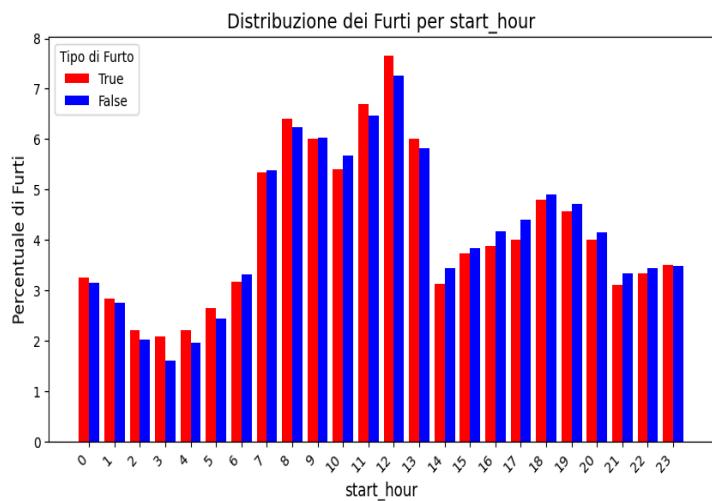
- La gamma delle velocità è più ristretta, con un taglio evidente intorno a 80 km/h.
- Il comportamento generale della distribuzione rimane simile, ma i valori estremi sono stati eliminati.
- Il picco della distribuzione è ancora tra 40-50 km/h, il che indica che la pulizia degli outlier non ha alterato il comportamento principale dei dati.
- La distinzione tra viaggi legati ai furti e non furti resta evidente, ma con una riduzione del "rumore" generato da velocità estreme.

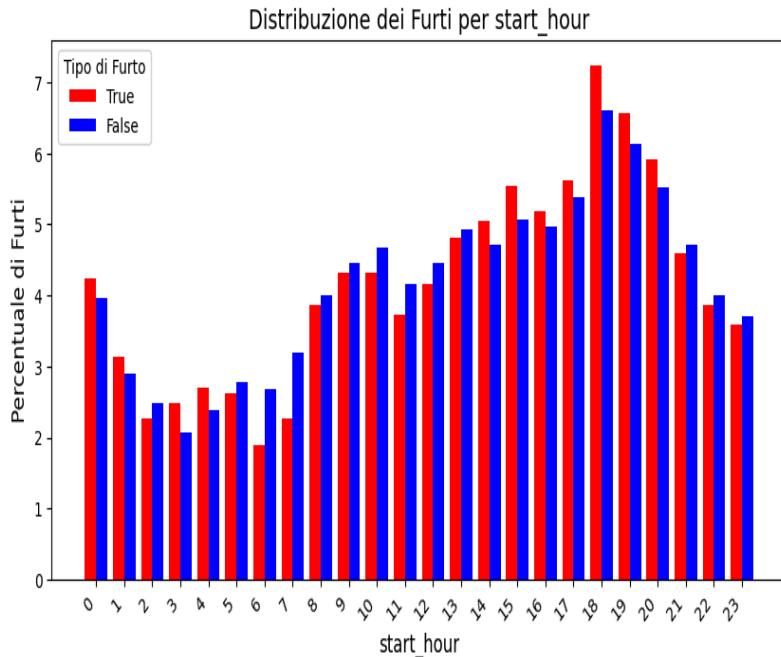
Confronto tra le due situazioni

Il trattamento degli outlier con IQR ha rimosso velocità estreme oltre 80 km/h, riducendo la variabilità dei dati.

Il picco della distribuzione è rimasto invariato, confermando che il pattern principale dei viaggi è robusto.

La proporzione tra viaggi con furti e non furti non ha subito grandi variazioni, il che suggerisce che gli outlier non erano determinanti per la differenziazione.





Distribuzione prima del trattamento IQR

- La distribuzione mostra due picchi principali:
 - Mattina (7-12) con un massimo attorno alle 12:00.
 - Pomeriggio-sera (17-20) con una crescita progressiva.
- Le percentuali di furti (rosso) e non furti (blu) sono molto simili, suggerendo che il comportamento orario è omogeneo.
- La notte e le prime ore del mattino (0-6) hanno una percentuale più bassa di eventi.
- Il calo più marcato si nota tra 13-16, periodo di transizione tra le due fasi.

Distribuzione dopo il trattamento IQR

- La struttura della distribuzione rimane simile, ma i valori sembrano più uniformi.
- Il picco della mattina (7-12) è rimasto, ma appare leggermente ridotto rispetto alla distribuzione originale.
- Il picco serale (17-20) è ancora presente, con un massimo più evidente alle 18:00.
- La fascia notturna (0-6) resta poco significativa, con una riduzione minima nelle prime ore.

Confronto

Il trattamento degli outlier non ha modificato significativamente la forma della distribuzione, indicando che non c'erano valori estremi dominanti.

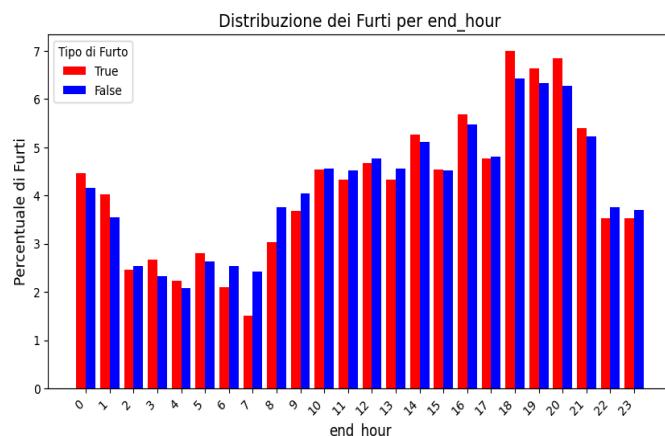
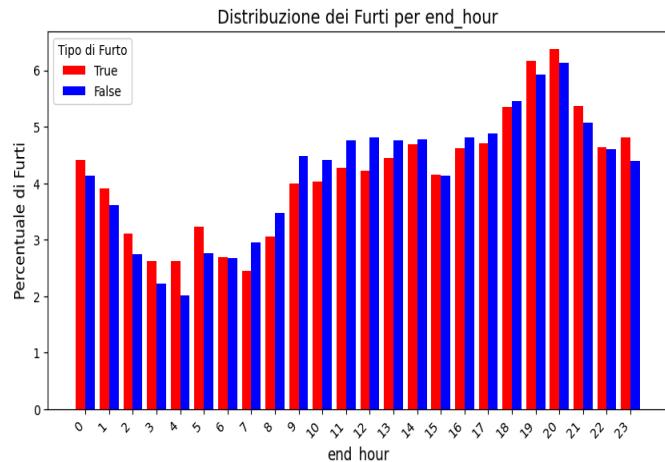
Il picco delle 12:00 è leggermente attenuato dopo il trattamento, segnalando possibili valori anomali rimossi in quella fascia.

La fascia serale (18-20) è più pronunciata dopo il trattamento, suggerendo che l'effetto degli

outlier fosse più presente nella mattina.

La distinzione tra furti e non furti rimane coerente, senza alterazioni drastiche nei rapporti.

Conclusione: Il trattamento con IQR ha reso la distribuzione più uniforme, attenuando alcune variazioni estreme senza alterare i pattern principali. I picchi nelle ore centrali della giornata restano confermati, suggerendo che i furti e i viaggi non furto seguano dinamiche temporali simili.



Analisi delle distribuzioni:

Prima del trattamento outlier :

- La distribuzione mostra una percentuale relativamente stabile di furti durante le ore del giorno.
- Si osservano picchi significativi nelle ore serali (19-21), con il valore massimo attorno alle 20.
- I furti avvengono in modo simile tra le due categorie (True/False), con leggere differenze in alcune ore.

Dopo il trattamento outlier con IQR :

- La distribuzione generale è simile, ma con variazioni più marcate nelle frequenze.
- La rimozione degli outlier ha ridotto alcuni valori nelle ore notturne e al mattino presto.
- Il picco serale (19-21) sembra ancora più accentuato, suggerendo che gli outlier eliminati fossero più distribuiti nelle altre ore.
- Gli intervalli tra le frequenze di True e False sembrano più regolari e meno fluttuanti.

Confronto:

- La pulizia con IQR ha ridotto le variazioni anomale nei dati, enfatizzando pattern più chiari.
- L'orario di picco per i furti è rimasto invariato (intorno alle 20).
- Alcune fasce orarie (ad esempio le prime ore del mattino) mostrano una leggera riduzione della percentuale.

Successivamente sono stati trattati gli outliers con diversi approcci. L'approccio più indicato è risultato essere quello che utilizza il metodo della differenza interquartile. Dopodichè è stato ripetuto il procedimento di visualizzazione istogrammi per fare valutazione relative all'impattato di IQR sulla distribuzione.

Ecco la funzione utilizzata per IQR:

```
def detect_outliers_iqr(df, columns, iqr_multiplier=1.5):  
    """  
    Identifica gli outlier in un DataFrame utilizzando il metodo IQR su più colonne.  
  
    :param df: DataFrame contenente i dati  
  
    :param columns: Lista delle colonne su cui calcolare gli outlier  
  
    :param iqr_multiplier: Moltiplicatore per l'intervallo interquartile (default: 1.5)  
  
    :return: DataFrame con una colonna aggiuntiva 'is_outlier' che indica gli outlier  
    """  
  
    # Seleziona solo colonne numeriche per evitare errori  
  
    numeric_columns = df[columns].select_dtypes(include=["number"]).columns
```

```

outlier_mask = np.zeros(df.shape[0], dtype=bool)

for column in numeric_columns:

    Q1 = df[column].quantile(0.25)

    Q3 = df[column].quantile(0.75)

    IQR = Q3 - Q1

    lower_bound = Q1 - iqr_multiplier * IQR

    upper_bound = Q3 + iqr_multiplier * IQR

    outlier_mask |= (df[column] < lower_bound) | (df[column] > upper_bound)

    df["is_outlier"] = outlier_mask

return df

```

La funzione esegue i seguenti passi:

1. Filtra le colonne numeriche per evitare errori dovuti a colonne non numeriche.
2. Inizializza una maschera booleana (`outlier_mask`) per contrassegnare gli outlier.
3. Per ogni colonna numerica:
 - o Calcola il primo quartile (Q1) e il terzo quartile (Q3).
 - o Determina l'IQR come la differenza tra Q3 e Q1.
 - o Calcola i limiti inferiore e superiore per l'individuazione degli outlier.
 - o Aggiorna la maschera booleana per segnare i valori che eccedono tali limiti.
4. Aggiunge la colonna `is_outlier` al DataFrame e lo restituisce.

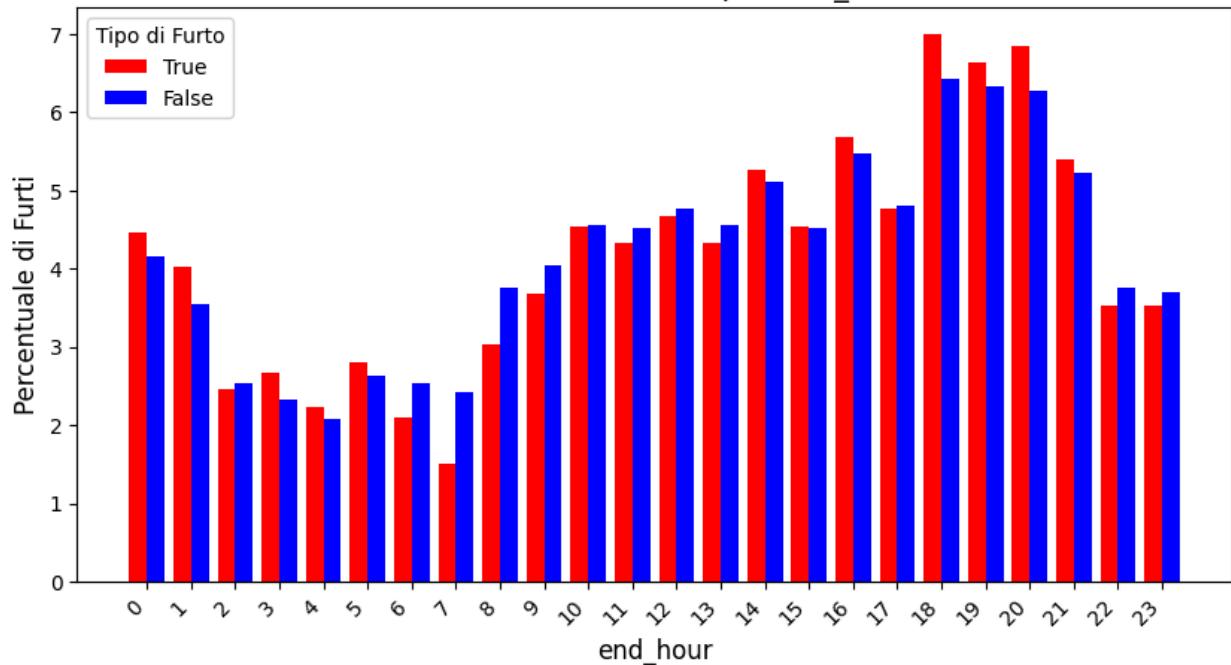
Essendo che:

```
df_no_outliers.shape #riduce la shape del df restituisce un df di dimensioni 39199x25
```

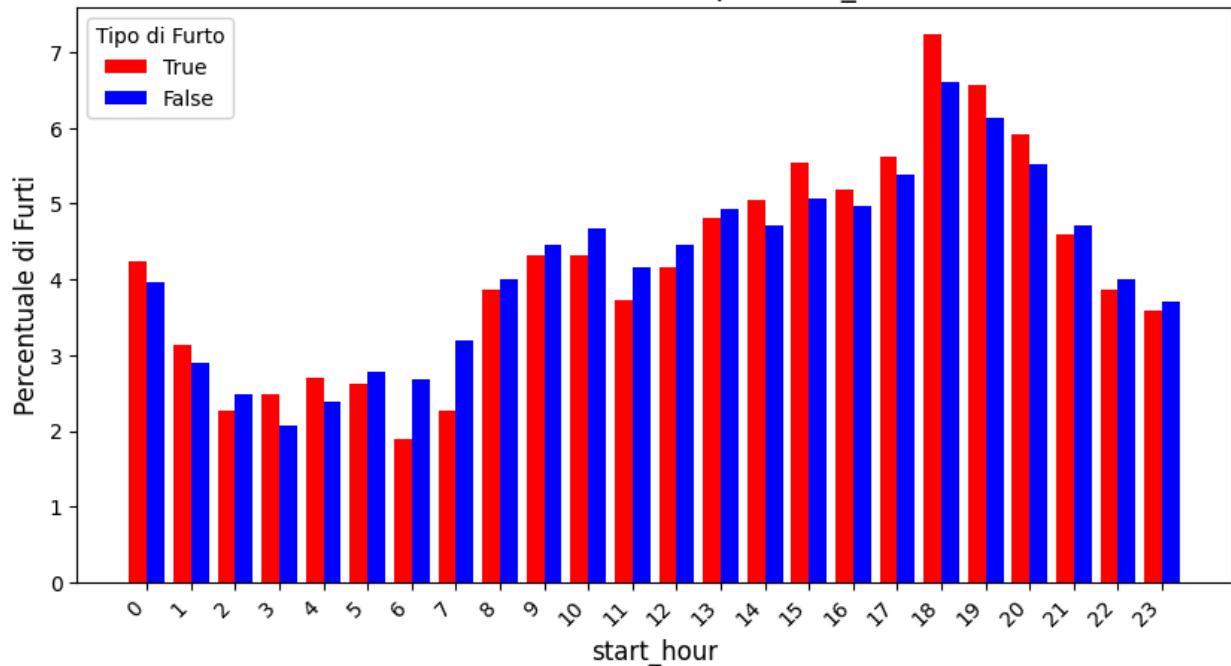
mentre quello con gli outliers era di 66917x25 si deduce che la distribuzione viene modificata significativamente. Per questo tipo di dati, comunque, trattare gli outliers potrebbe non essere una operazione da farsi in quanto proprio i dati relativi a furti potrebbero avere il comportamento anomalo tipico degli outliers e quindi, eliminando gli outliers, queste informazioni relative ai furti potrebbero andare erroneamente perse.

Qui di seguito riportate gli stessi istogrammi già generati e riportati sopra ai fini del confronto visivo(quelli ottenuti sulla base dei dati filtrati per IQR).

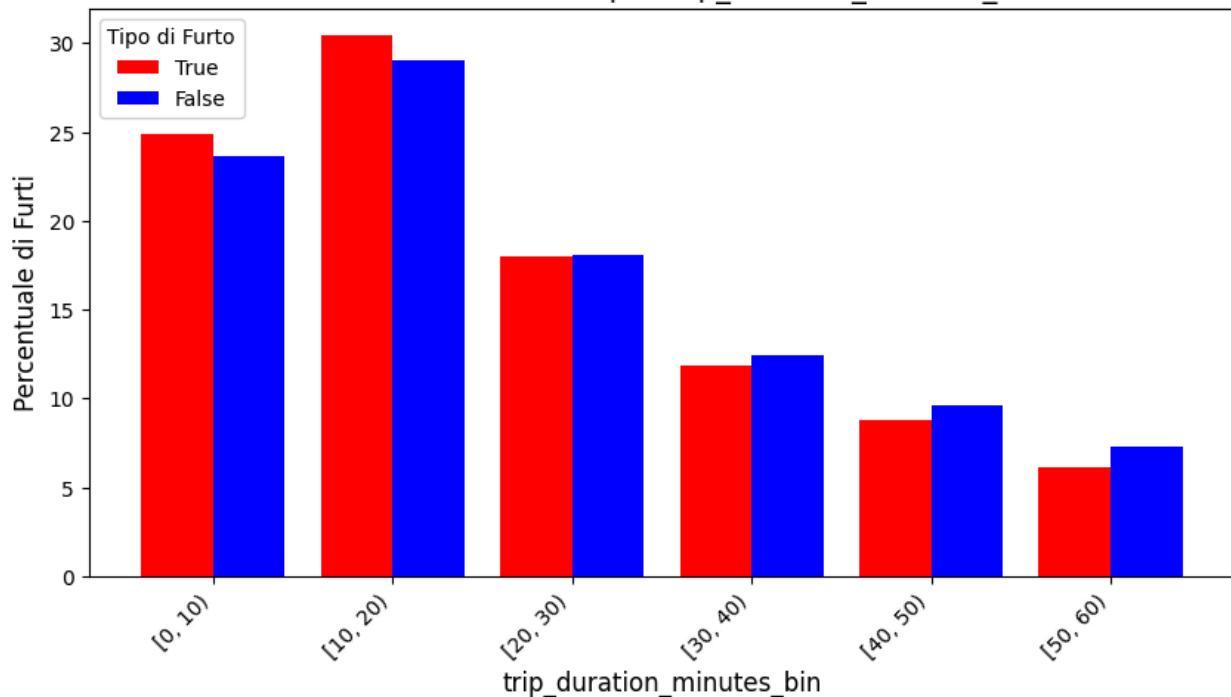
Distribuzione dei Furti per end_hour



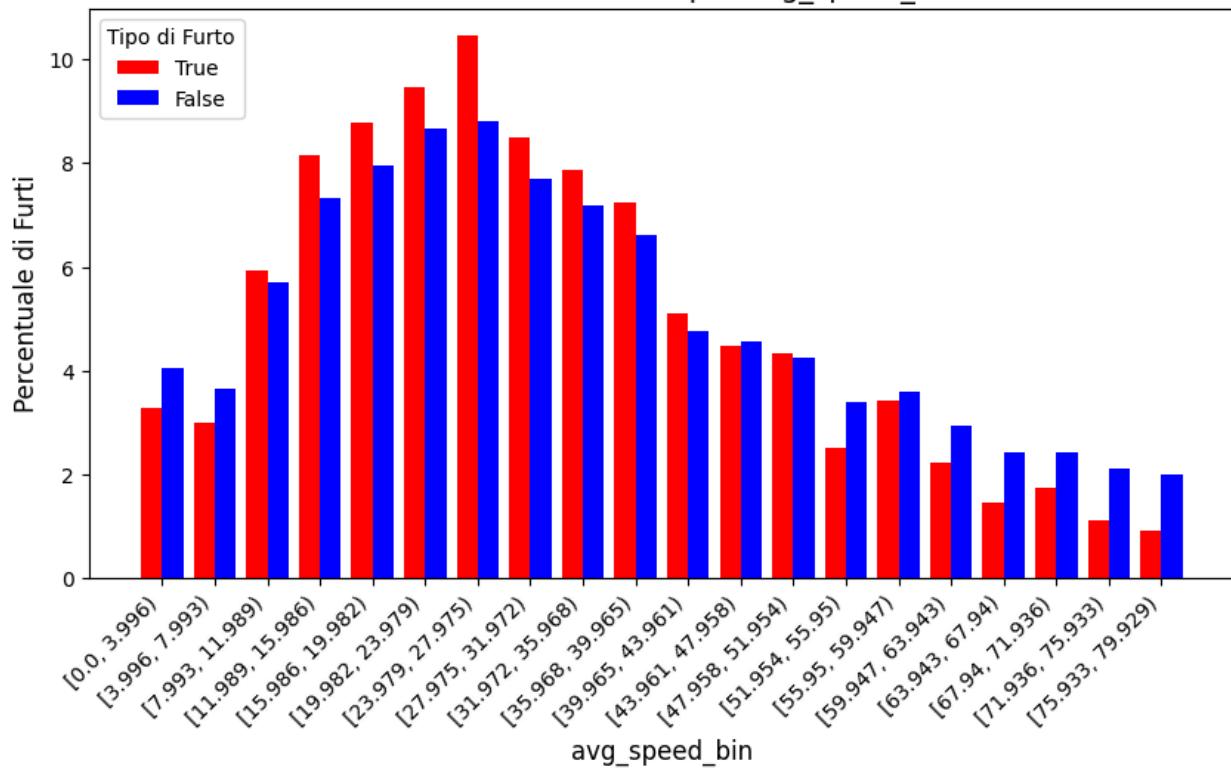
Distribuzione dei Furti per start_hour



Distribuzione dei Furti per trip_duration_minutes_bin



Distribuzione dei Furti per avg_speed_bin



```
Infine df_no_outliers.to_csv('df_without_outliers.csv')
```

rende disponibile il dataset filtrato per eventuale e successive ulteriori analisi.

```
Infine salvo anche il df per il modello df.to_csv('df_randomforest.csv')
```

3. Sviluppo di un Modello di machine Learning per l' individuazione di pattern ricorrenti di viaggi relativi a furti

Modello Random Forest per la Predizione del Furto

Questa parte del report descrive il processo di costruzione, addestramento e valutazione di un modello di apprendimento automatico basato su Random Forest per la predizione del furto ('theft').

Pre-elaborazione dei Dati

Ordinamento e Copia del Dataset

Il dataset iniziale viene ordinato in base ai seguenti campi:

- `trip_id, start_year, start_month, start_day, start_weekday, start_hour, start_minute`

Successivamente, viene creata una copia di backup (`df_initial`) del dataset originale prima di effettuare modifiche.

Feature Selection

Le seguenti colonne vengono rimosse:

- `Unnamed: 0`: colonna indicizzata senza valore informativo.
- `trip_id`: identificativo del viaggio, non utile per la predizione.

Le feature rimanenti vengono utilizzate per la creazione delle matrici di features (`X`) e target (`y`):

- `X`: Tutte le colonne eccetto `theft`.
- `y`: Colonna `theft` come variabile target.

3.1 Suddivisione del Dataset

Il dataset viene diviso in due parti:

- **Train set (80%)**: Utilizzato per l'addestramento del modello.
- **Test set (20%)**: Utilizzato per la valutazione del modello.

La suddivisione è effettuata con **stratificazione** sulla variabile target (`theft`) per garantire una distribuzione bilanciata.

3.2 Selezione del Modello e Iperparametri

Viene utilizzata una **Random Forest Classifier**, con una ricerca esaustiva dei migliori iperparametri tramite `GridSearchCV`.

Gli iperparametri testati sono:

- Numero di alberi (`n_estimators`): [50, 100, 200]
- Profondità massima (`max_depth`): [10, 20, 30, None]
- Minimo numero di campioni per split (`min_samples_split`): [2, 5, 10]
- Minimo numero di campioni per foglia (`min_samples_leaf`): [1, 2, 4]
- Numero di feature considerate per split (`max_features`): ['sqrt', 'log2']
- Utilizzo del bootstrap (`bootstrap`): [True, False]

La valutazione viene eseguita utilizzando la metrica **F1-score** con una validazione incrociata a 3 fold. La model selection è stata effettuata sul validation set, perché il suo scopo è selezionare l'insieme di iperparametri che ottimizzano le prestazioni del modello su dati non visti, senza "guardare" direttamente i dati di test. Se scegliessimo gli iperparametri sulla base delle prestazioni sul training set, il modello si adatterebbe troppo ai dati di addestramento, rischiando di overfittare. Questo significa che funzionerebbe bene sui dati con cui è stato allenato, ma non generalizzerebbe bene su nuovi dati.

3.3 Risultati della Selezione del Modello

Dopo il tuning degli iperparametri, il miglior modello di Random Forest ha ottenuto:

- **Miglior punteggio F1: 0.6601**
- **Migliori parametri:**
 - `bootstrap`: False
 - `max_depth`: 30
 - `max_features`: sqrt
 - `min_samples_leaf`: 1
 - `min_samples_split`: 2
 - `n_estimators`: 200

3.4 Valutazione del Modello

Il modello ottimizzato viene riaddestrato e testato su un set di test indipendente, ottenendo le seguenti metriche:

- **Accuratezza:** 96.02%

- **Precisione:** 96.04%
- **Recall:** 96.02%
- **F1-score:** 95.50%

Matrice di Confusione

La matrice di confusione mostra:

- **12236 veri negativi (TN)**
- **22 falsi positivi (FP)**
- **511 falsi negativi (FN)**
- **615 veri positivi (TP)**

Il modello ha un'alta accuratezza ma presenta alcune difficoltà nel rilevare correttamente i casi positivi (furti).

Feature Importance

L'importanza delle feature è stata analizzata con il miglior modello Random Forest. Le feature più influenti sono:

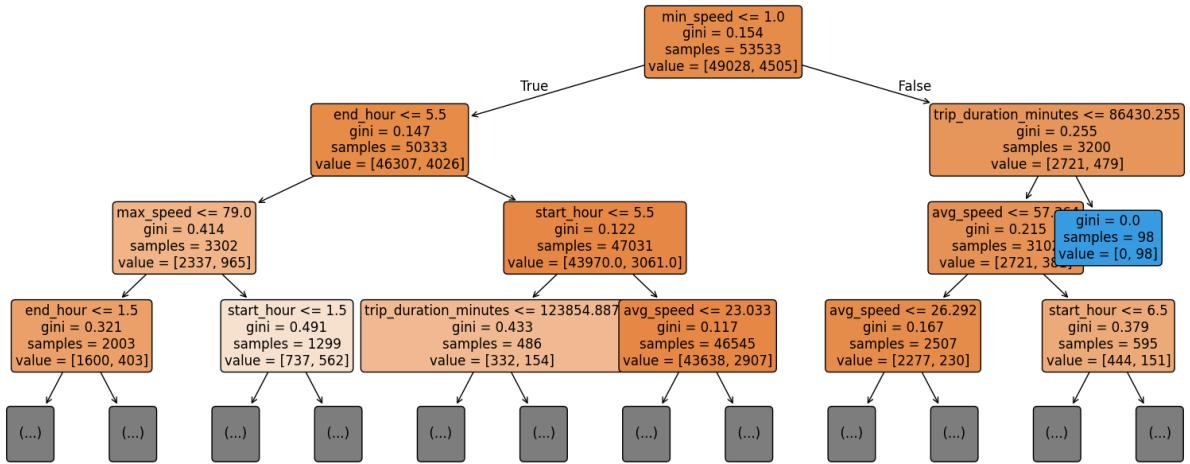
1. `trip_duration_minutes` (35.64%)
2. `end_hour` (7.05%)
3. `avg_speed` (5.19%)
4. `max_speed` (4.98%)
5. `end_latitude` (4.53%)
6. `start_longitude` (4.18%)
7. `start_hour` (4.29%)
8. `start_latitude` (4.21%)

Le feature temporali e di velocità sembrano avere il maggiore impatto sulla previsione dei furti.

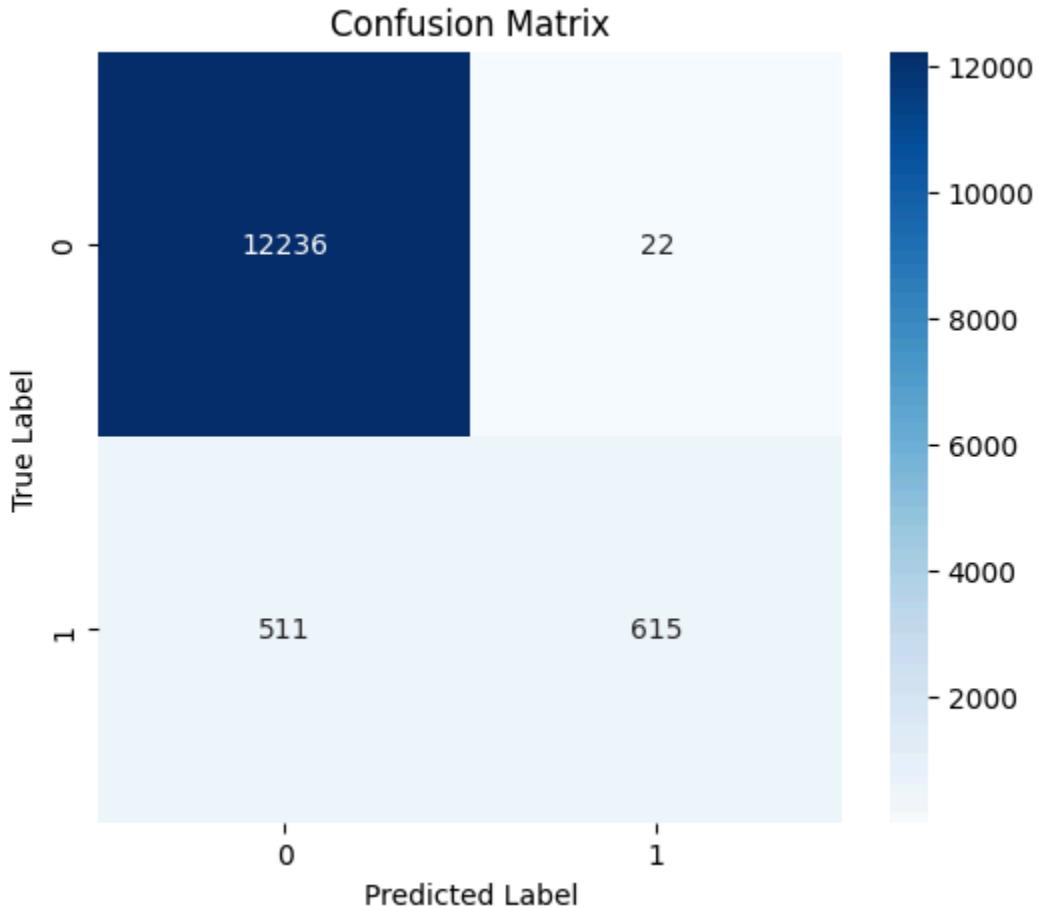
3.5 Visualizzazioni

Sono state create diverse visualizzazioni:

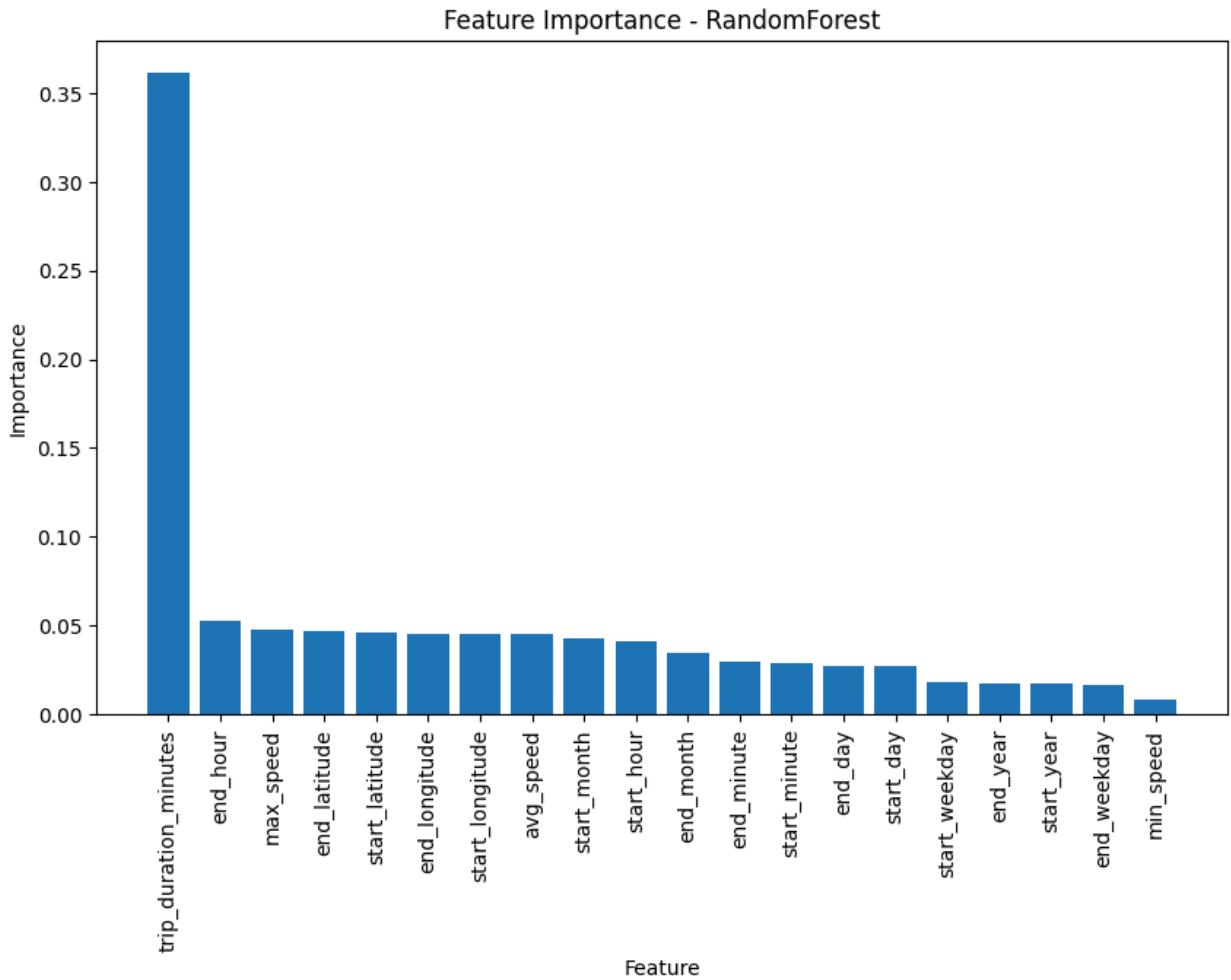
- **Albero di decisione:** Visualizzazione di uno degli alberi della foresta per comprendere meglio le decisioni del modello.



- **Matrice di confusione:** Heatmap per analizzare gli errori del modello.



- **Feature Importance:** Grafico a barre per evidenziare le feature più influenti.



4. Conclusioni e Sviluppi Futuri

4.1 Conclusioni

Il modello Random Forest ha dimostrato di essere efficace per la predizione del furto con un'accuratezza del 96%. Tuttavia, il recall del modello per la classe "furto" può essere migliorato per ridurre il numero di falsi negativi. Possibili strategie di miglioramento includono:

- **Uso di modelli ensemble** (XGBoost, Gradient Boosting).
- **Bilanciamento del dataset** (SMOTE o tecniche di data augmentation per la classe minoritaria).
- **Aggiunta di feature ingegnerizzate**, come variazioni di velocità o indicatori di anomalie.

Con ulteriori ottimizzazioni, il modello può diventare uno strumento ancora più affidabile per la rilevazione di furti nei dati GPS analizzati.

4.2 Sviluppi Futuri

Tenendo presente che, come specificato dall' azienda, il presente modello è pensato per lavorare all' unisono con altri modelli, per migliorare ulteriormente il modello e la sua applicabilità pratica, è possibile individuare i seguenti punti per apportare migliorie e approfondimenti futuri:

- Integrazione con tecniche di deep learning, come reti neurali convoluzionali o ricorrenti, per catturare pattern più complessi nei dati.
 - Implementazione di un sistema di allerta in tempo reale, che utilizzi il modello per segnalare potenziali furti mentre i veicoli sono in movimento.
 - Analisi geospaziale avanzata, sfruttando tecniche di clustering per identificare aree ad alto rischio di furto.
 - Ottimizzazione della pipeline di preprocessing, per migliorare la gestione dei dati mancanti e la selezione delle feature più informative.
 - Test su dati reali e integrazione con sistemi di sicurezza esistenti, per valutare l'efficacia del modello in scenari operativi concreti.
-

5. Riferimenti Bibliografici

- Dati resi disponibili da Octo Telematics SPA
 - Documentazione disponibile su BigData Research Infrastructure
-

6. Appendici

- Link cliccabile al progetto di tesi nel mio Github:
<https://github.com/GiuliaSegurini/Proactive-Theft.git>