UNICA

UNIVERSITÀ
DEGLI STUDI
DI CAGLIARI

TRAFFIC CRASH ANALYSIS IN CHICAGO (2017-2024)

CORSO DI LABORATORIO DI BIG DATA

MATTEO GHISU

PROF. GIULIANO ARMANO

INDICE

INTRODUZIONE

Nel seguente elaborato è stata effettuata un'analisi su un dataset di incidenti stradali a Chicago con dati raccolti dai distretti di polizia da settembre 2017 ad aprile 2025.

Il lavoro di analisi è stato svolto per mezzo dell'infrastruttura Spark con l'utilizzo della libreria Pyspark e altre librerie Python.

Dapprima è stata effettuata una procedura di pulizia del dataset originale, successivamente un'analisi esplorativa per mezzo delle librerie Pandas, Matplotlib e Seaborn e infine sono state effettuate delle elaborazioni con Spark con le quali sono stati ottenuti dei risultati attraverso dei modelli di apprendimento statistico supervisionati.

Il lavoro è stato suddiviso in diversi script python, i primi due (datacleaning.py, analisi_esplorativa.ipynb) dedicati alla pulizia del dataset e all'analisi esplorativa, i restanti tre (spark_models.py, project_runner.ipynb,ecoc_method.py) sono dedicati all' analisi statistica vera e propria.

SCELTA DEL DATASET

Il dataset scelto è stato scaricato da Traffic Crashes - Crashes | City of Chicago | Data Portal[1] e presenta 932400 osservazioni con 48 colonne.

Questo è aggiornato in tempo reale, infatti, il numero di osservazioni riportato potrebbe attualmente differire da quello riportato nel report.

Ogni osservazione nel dataset riguarda un incidente specifico avvenuto nelle strade di Chicago (riconosciuto tramite il CRASH_ID), le altre colonne riguardano altre informazioni come ad esempio la data dell'incidente, la presenza o meno di segnaletica stradale, le condizioni ambientali e del manto stradale al momento dell'incidente, il tipo di incidente, il numero di unità coinvolte nell'incidente, le possibili cause dell'incidente ecc.

Una descrizione più dettagliata delle features con la specificazione anche del tipo è presentata nel file 'struttura_dataset' all'interno della cartella del progetto.

DATA CLEANING

Per evitare uno sbilanciamento eccessivo dei dati è stata effettuata un'approfondita procedura di Data Cleaning nello script 'datacleaning.py' con l'ausilio della libreria 'Pandas'.

La procedura di datacleaning è stata eseguita all'interno della classe 'Preprocessor',questa classe è utilizzabile solo per questo dataset specifico infatti sono presenti dei metodi specifici 'add_year_column','replace_values' etc.. che riguardano modifiche relative solo al dataset in questione.

N.B è presente, per completezza, anche una versione del data cleaning effettuata con la libreria 'Pyspark'; tuttavia, presenta alcuni problemi di funzionamento visti i problemi di

compatibilità tra Spark e Windows che purtroppo non sono riuscito a risolvere.

I problemi di sbilanciamento si osservano già dalle prime features come ‘POSTED SPEED LIMIT’ che presenta 46 diversi valori numerici, molti presenti solo nell’ordine delle decine;

quindi, si è deciso di creare delle colonne categoriche per raggruppare meglio le osservazioni in un minor numero di labels.

I raggruppamenti più sostanziali sono stati inseriti nel file “group_confi.json” per garantire maggior ordine e pulizia dello codice nello script in cui è stato eseguito il data cleaning.

Un raggruppamento che è stato effettuato è, per esempio, quello sulla colonna ‘PRIM_CONTRIBUTORY_CAUSE’ che indica le principali cause degli incidenti. Si è passato da avere 40 categorie a 11, grazie a un raggruppamento semantico.

Oppure ancora, si può osservare che dopo questa operazione, il label ‘Distraction’ contiene infatti molteplici labels con significato similare ‘"DISTRACTION - FROM INSIDE VEHICLE", "DISTRACTION - FROM OUTSIDE VEHICLE"," TEXTING”, ecc…

Questa procedura è stata ripetuta per molteplici colonne.

In questo modo il dataset risulta essere più bilanciato per le analisi che verranno eseguite successivamente, senza causare la perdita di informazioni importanti.

Un ulteriore problema è l’eccessiva presenza di valori mancanti, in molte colonne come ad esempio ‘LANE_CNT’ sono assenti circa l’80% delle osservazioni.

Per tale motivo si è deciso di rimuovere dal dataset queste variabili poiché avrebbero portato a delle analisi statistiche eccessivamente distorte.

Inoltre, sono state effettuate delle sostituzioni come, ad esempio, sostituire i numeri con i giorni della settimana e aggiungere separatamente una colonna ‘YEAR’

Il dataset pulito quindi è stato salvato come ‘Traffic_Crashes_cleaned.csv’

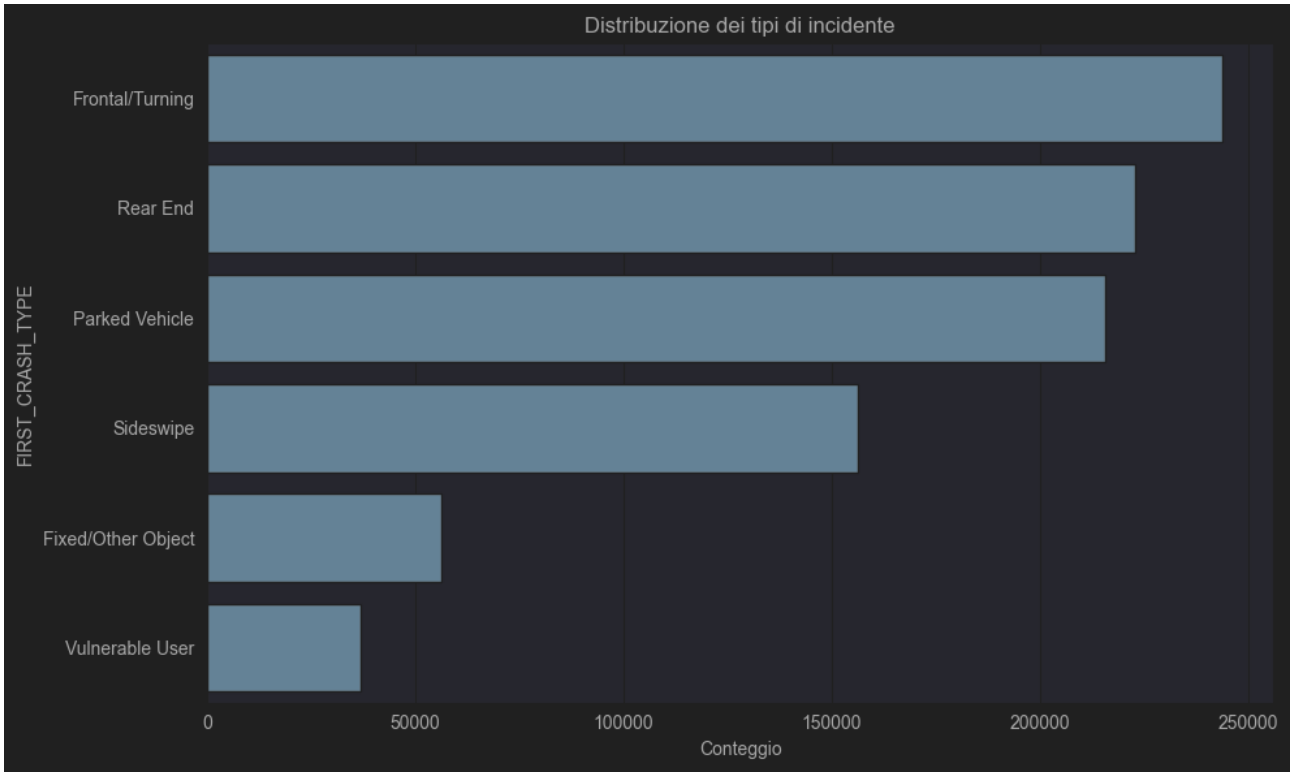
ANALISI ESPLORATIVA

Nell’analisi esplorativa si è deciso di analizzare alcune relazioni tra la variabile di risposta scelta ‘FIRST_CRASH_TYPE’ (che indica la tipologia di incidente) e altre variabili.

Le classi nella variabile di risposta sono:

1) **Frontal/Turning** (impatto frontale tra due veicoli o dopo una svolta)
2) **Rear End** (tamponamento)
3) **Parked Vehicle** (veicolo in movimento colpisce un veicolo parcheggiato)
4) **Sideswipe** (strisciata laterale tra due veicoli)
5) **Fixed/Other object** (impatto con un oggetto fisso come un albero o altri tipi di oggetti come ad esempio un treno)
6) **Vulnerable User** (scontro dei veicoli con pedoni o ciclisti)

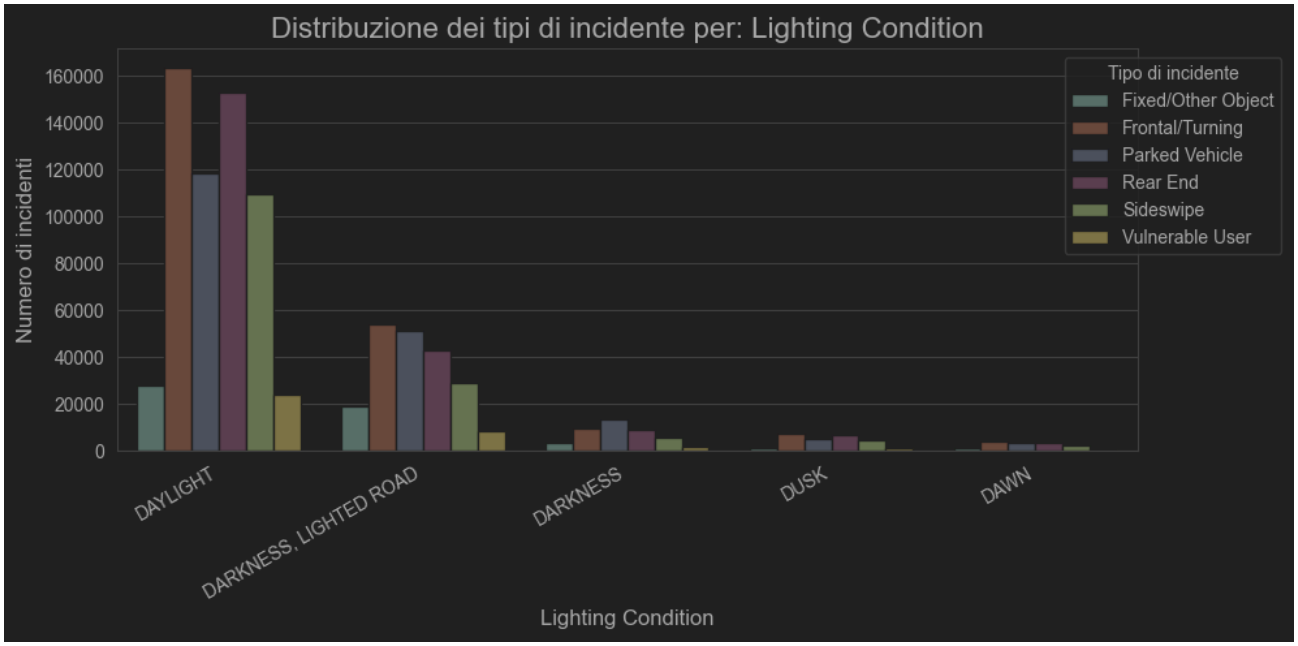
Di seguito sono riportati i tipi di incidente più frequenti attraverso un grafico a barre.

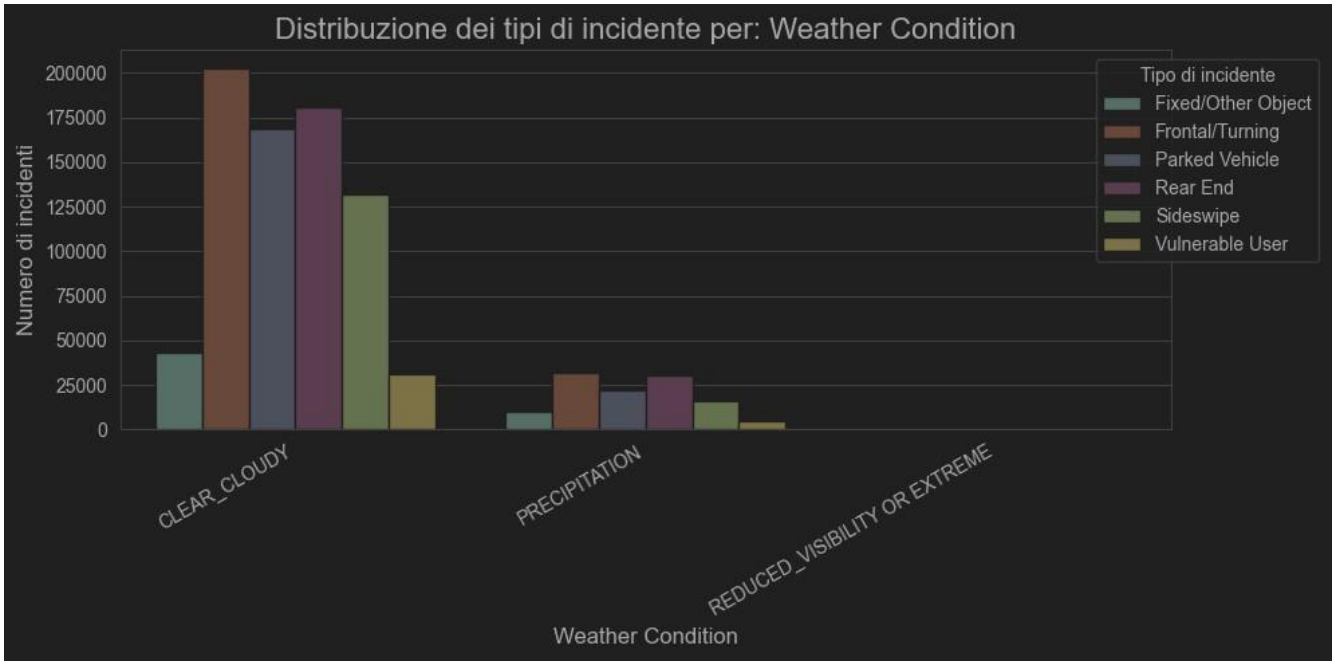


Si è quindi deciso di esplorare le possibili cause che possono aver determinato queste tipologie di incidenti.

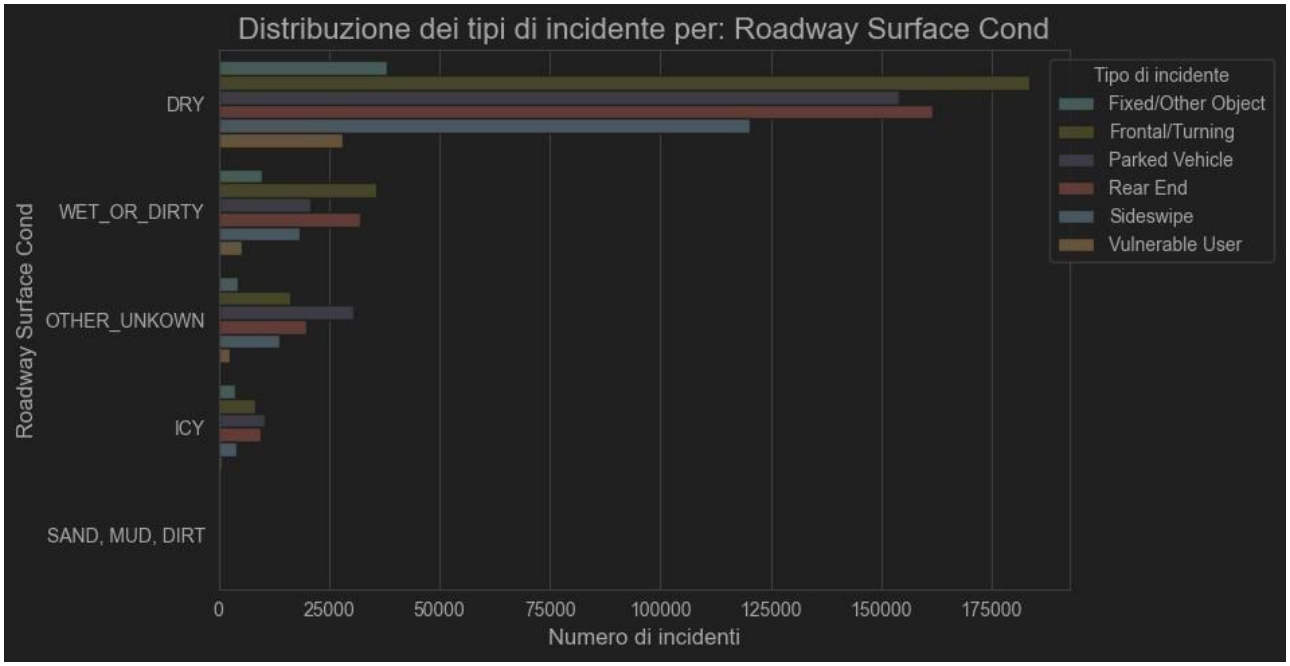
Inizialmente si sono osservati come alcune variabili esogene (variabili non legati al comportamento scorretto alla guida) potessero influenzare la presenza o meno delle varie tipologie di incidenti.

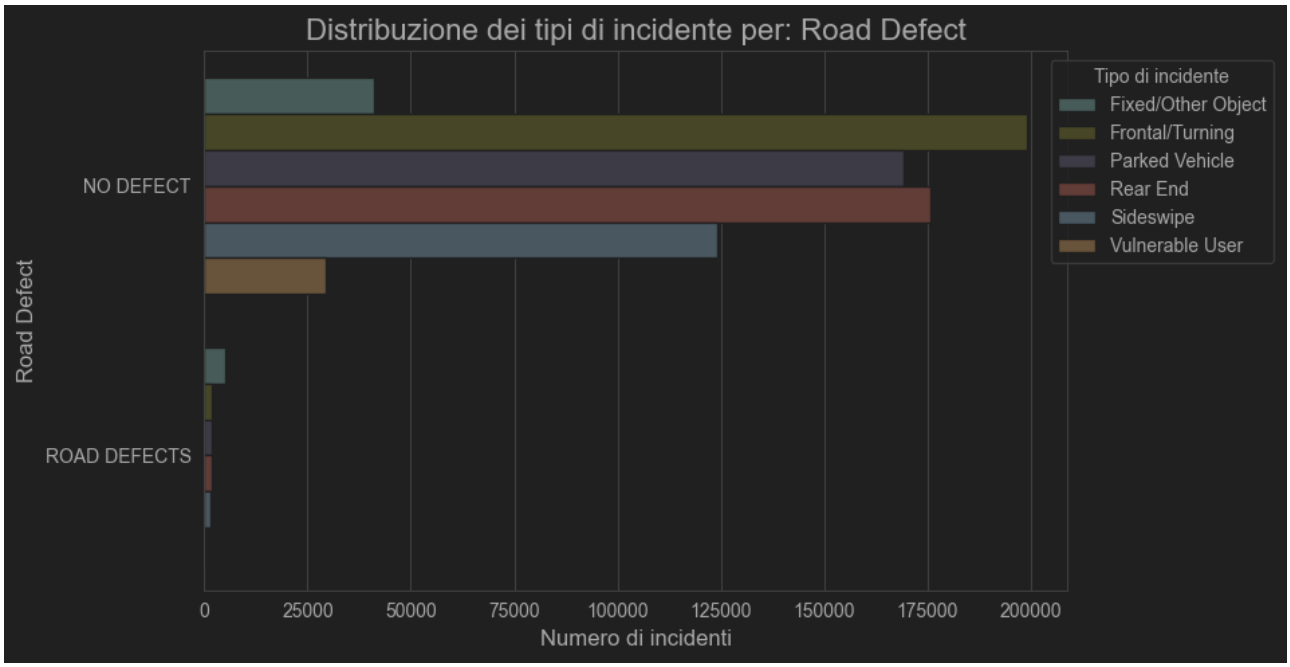
I successivi due grafici sono relativi alle condizioni atmosferiche. L'aspetto particolare è la presenza del maggior numero di incidenti in condizioni di luce e cielo pulito o nuvoloso, le quali non dovrebbero favorire gli incidenti stradali.



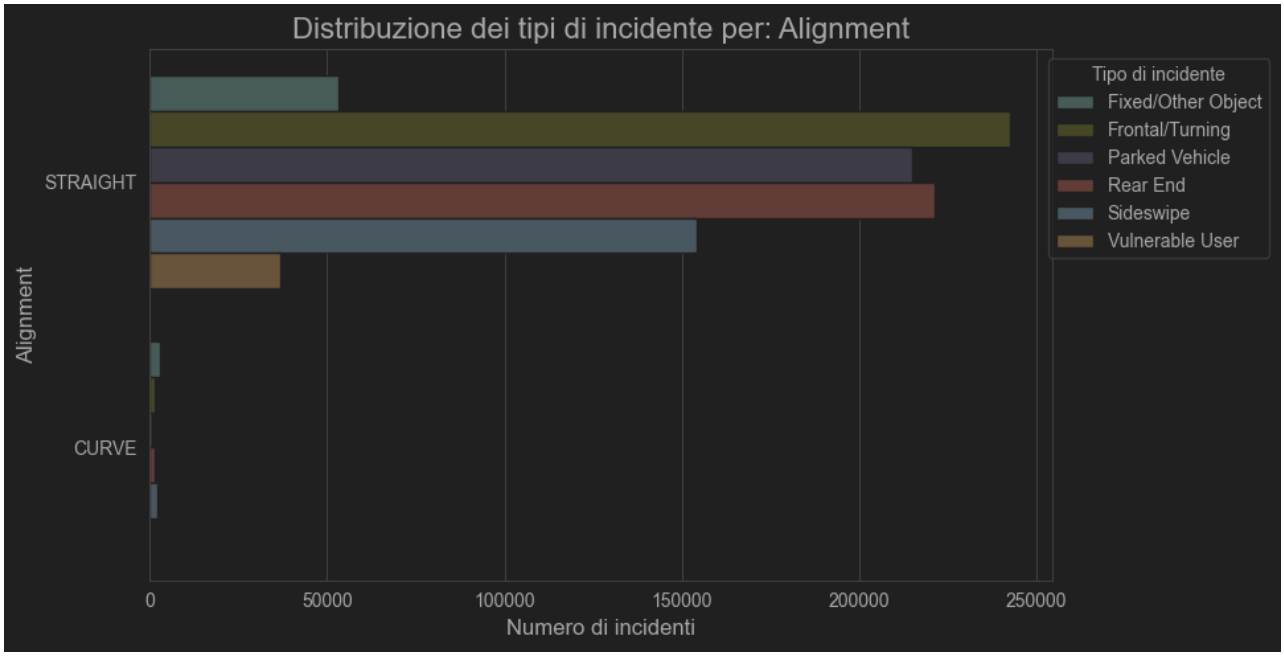


Stesso discorso vale per le condizioni della strada: sia nel grafico relativo a 'ROADWAY SURFACE CONDITION' sia in quello relativo a 'ROAD DEFECT' per la maggior parte degli incidenti stradali non ci sono condizioni sfavorevoli alla guida come una strada bagnata o una strada con presenza di detriti che possono compromettere la guida del conducente.





Infine, l'ultimo grafico è relativo alla presenza di strada dritta o di una curva. La presenza di incidenti in curva, come si osserva, è bassissima.



I risultati ottenuti con questi grafici ci vanno a suggerire che gli incidenti si verificano più per cause relative a comportamenti scorretti alla guida piuttosto che a condizioni esogene come quelle sopracitate.

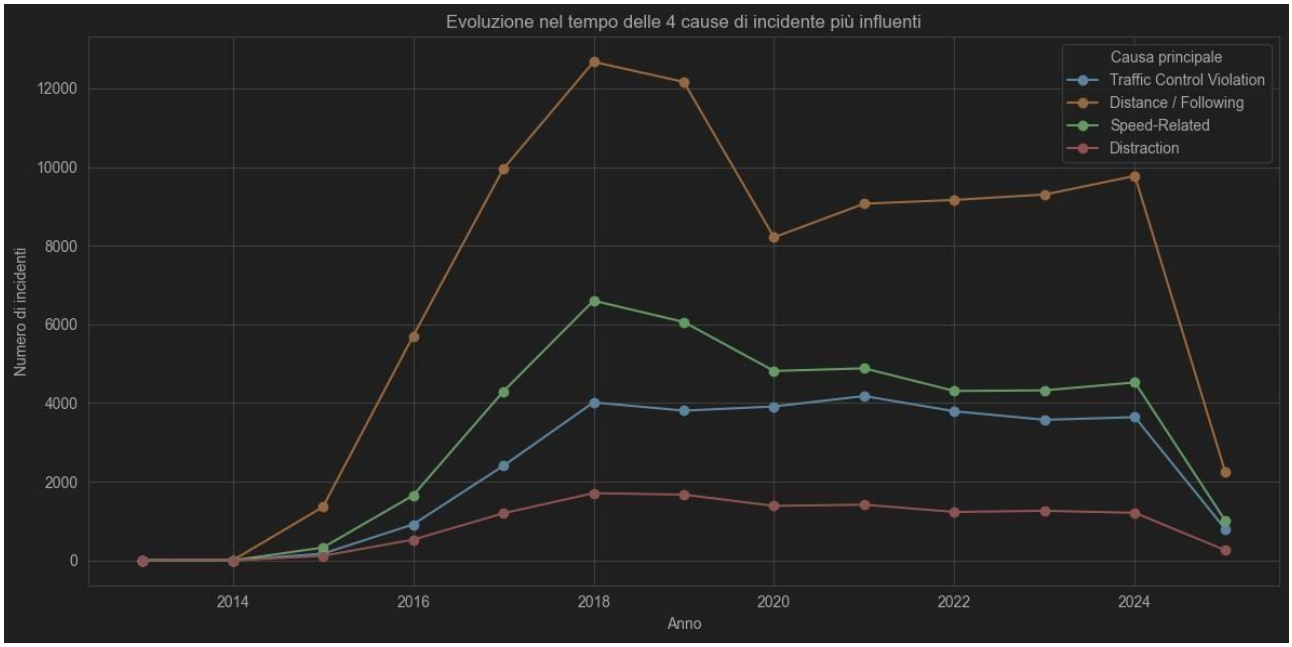
Si è quindi deciso di andare a visualizzare attraverso una heatmap che mostra la distribuzione percentuale delle cause principali degli incidenti in relazione ai tipi di incidenti normalizzata per riga.

Da questo grafico si può infatti notare che c'è prevalenza di cause endogene (comportamenti scorretti del conducente) che provocano degli incidenti stradali.

Si nota complessivamente che le cause endogene sono più determinati in valori percentuali per alcuni particolari tipi di incidente (es. per parked vehicle la causa principale è driver/skill condition con il 37.1%) invece per le cause esogene, queste sono distribuite quasi più proporzionalmente per tutte le tipologie di incidente.

I fattori ambientali e gli eventi rari, in generale, ("Environmental Factors" e "Rare Events") sono cause più influenti per gli incidenti riguardanti gli scontri con oggetti fissi (es. scontro con un albero) ricoprendo circa il 30% tra le varie cause.

Infine, un ulteriore approfondimento è stato eseguito nel grafico successivo in cui si è andato a osservare come le cause più influenti si sono evolute negli anni.



Pare che ci sia stato un picco nel 2018 e poi una lieve discesa per tutte e 4 le cause sopracitate.

ANALISI INFERENZIALE

L' analisi inferenziale riguarda un problema di classificazione che cerca di mettere in relazione la variabile di risposta 'FIRST_CRASH_TYPE' (che presenta 6 classi) con le altre features categoriche del dataset.

N.B. Il dataset prima dell'analisi è stato depurato delle variabili stringa che non ci sarebbero servite per il nostro compito di classificazione.

Il task di classificazione è stato svolto attraverso l'utilizzo di tre modelli di apprendimento supervisionato.

I primi due modelli sono stati importati dalla libreria 'pyspark.ml' e sono una Multiclass Logistic Regressione e un Random Forest Classifier.

.

Successivamente, come terzo modello, è stata implementata una tecnica ECOC (Error-Correcting-Output-Codes) utilizzabile grazie alla libreria 'sklearn'.

Tutti modelli sono stati implementati attraverso la creazione di due script contenenti entrambi due classi: lo script 'spark_models' contiene i primi due modelli e invece il modello ECOC è stato costruito nello script 'ecoc_method'

In entrambi gli script prima sono state effettuate le operazioni di preprocessing (string-indexing,vectorization e one-hot encoding), e divisione in training e test set (per tutti e tre i modelli in proporzione 70-30)

MULTICLASS LOGISTIC REGRESSION

Come primo modello si è provato a utilizzare una Multiclass Logistic Regression.

Questo è sicuramente un modello abbastanza semplice e interpretabile rispetto a Random Forest; tuttavia, è stato scelto anche per osservare quale potesse essere preliminarmente il livello di accuracy per poi operare con modelli più complessi successivamente.

La multiclass logistic regression estende la regressione logistica oltre il caso binario, permettendo di classificare esempi in più di due categorie. Questo avviene principalmente tramite l'uso della funzione softmax, che calcola la probabilità che un esempio appartenga a ciascuna delle possibili classi. L'algoritmo restituisce quindi un vettore di probabilità (una per classe) e assegna l'esempio alla classe con la probabilità più alta. In alternativa, si possono usare strategie come "one-vs-rest" (un classificatore per ogni classe contro tutte le altre) o "one-vs-one" (un classificatore per ogni coppia di classi).

Per la scelta del miglior modello, nel nostro caso, viene creata una griglia di iperparametri da selezionare mediante una 3-fold CrossValidation.

```
# Griglia di iperparametri
param_grid = ParamGridBuilder() \
    .addGrid(lr.regParam, values: [0.01, 0.1]) \
    .addGrid(lr.elasticNetParam, values: [0.0, 0.5, 1.0]) \
    .build()
```

Il primo parametro 'regParam' è il parametro di regolarizzazione (λ) che controlla l'importanza della penalizzazione dei pesi sul modello, in modo da determinare l'importanza di ciascuna feature nella previsione. Il secondo parametro 'elastcNetParam' permette di decidere il mix tra regolarizzazione L1(lasso)e L2 (ridge) combinando i vantaggi di entrambi.

Come misura di valutazione è stata scelta l'accuracy.

I risultati ottenuti, come ci si poteva aspettare sono disastrosi, come riportato nell' output seguente

```
accuracy = spark_manager.train_logistic_regression_cv(train_df, test_df)

print(f"Accuracy sul test set: {accuracy:.4f}")
Executed at 2025.05.15 14:24:56 in 3m 28s 26ms

 Accuracy sul test set: 0.4986
```

Questi risultati sono dovuti probabilmente alla presenza molte classi sbilanciate e a relazione non lineari importanti tra le features; Quindi, si è optato per la scelta di un modello più robusto come il Random Forest Classifier.

RANDOM FOREST CLASSIFIER

Il random forest classifier è un algoritmo di apprendimento supervisionato basato su un insieme di alberi decisionali. Durante l'addestramento, costruisce molteplici alberi su sottoinsiemi casuali dei dati e delle caratteristiche. Ogni albero effettua una previsione e la classe finale viene scelta tramite una votazione di maggioranza tra tutti gli alberi. Questo approccio riduce l'overfitting rispetto a un singolo albero e migliora la robustezza e l'accuratezza della classificazione, funzionando bene sia per problemi binari che multiclasse.

Il Random Forest, inoltre, è un modello più adatto a dataset con classi sbilanciate, gestisce meglio il rumore nei dati e inoltre è più adatto in caso di non linearità nel pattern dei dati.

Anche in questo caso è stata la scelta la misura dell'accuracy.

Per via di limiti computazionali è stata evitata l'impostazione di una grid-search per gli iperparametri che portavano a un crash della sessione.

Come nel modello precedente è stata impostata una 3-fold cross validation per valutare le prestazioni con i seguenti parametri fissi

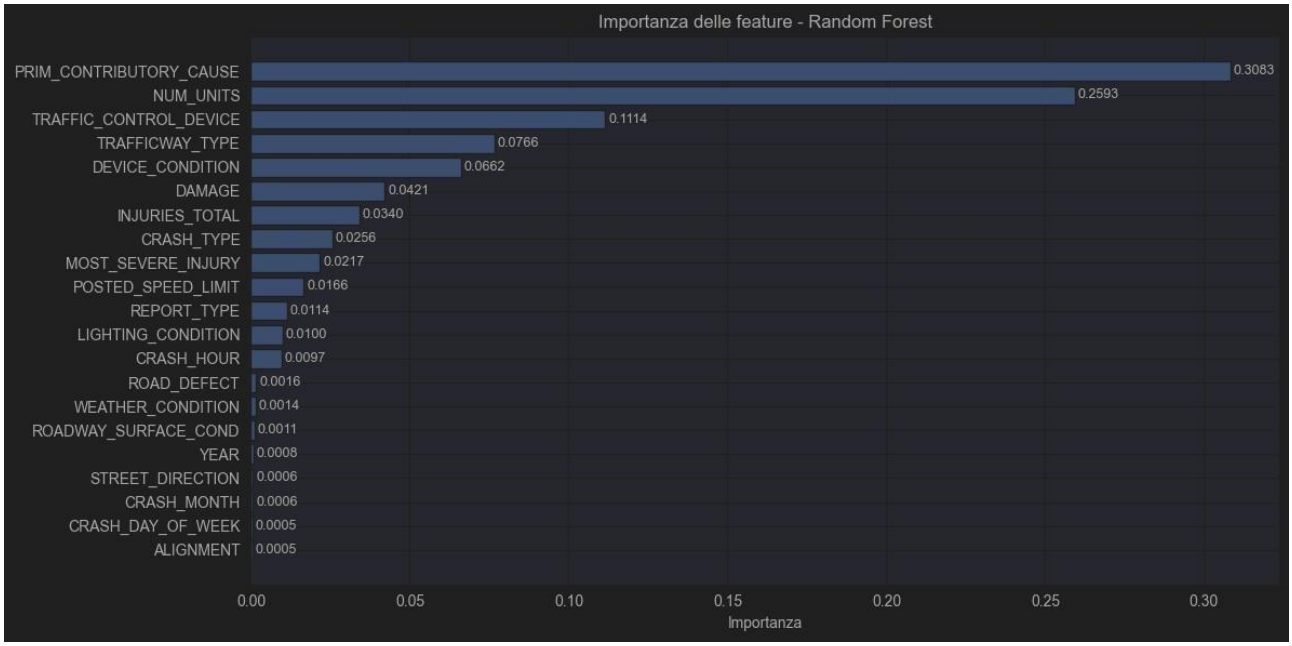
```
rf = RandomForestClassifier(
    labelCol="target",
    featuresCol="features",
    numTrees=200,
    maxDepth=10,
    minInstancesPerNode=2,
    seed=42
)
```

I risultati ottenuti, come mostrato di seguito, sono sicuramente migliori da quelli ottenuti dal modello precedente, ma comunque non sono abbastanza soddisfacenti.

```
Accuracy finale:0.5980
```

FEATURE IMPORTANCE

Inoltre, è stata effettuata una feature importance per mostrare quali sono state le variabili più influenti nella creazione della random forest.



Come intuito in precedenza per la determinazione del tipo di incidente hanno più influenza variabili comportamentali del conducente piuttosto che variabili esogene.

Infatti 'PRIM_CONTRIBUTORY_CAUSE' che presenta quasi totalmente dei labels relativi a comportamenti umani ha un valore di 0,3083 invece 'LIGHTING_CONDITION','ROAD_DEFECT', e 'WEATHER_CONDITION','ROADWAY_SURFACE_COND' hanno valori bassi (<0,01)

MODELLO ECOC

Introduzione al modello

Visti i risultati che si sono ottenuti con i precedenti due modelli, e data la distribuzione delle osservazioni nella nostra variabile di risposta che risulta essere sbilanciata si è deciso di optare per la metodologia ECOC (Error-Correcting-Output-Codes).

Il modello ECOC anziché distinguere tutte le K classi contemporaneamente, scompone il problema multiclasse in una serie di problemi di classificazione binaria più semplici.

Ogni classe K viene rappresentata attraverso un codice binario univoco(codeword) di lunghezza L.

L'insieme di questi codici forma una matrice M (codebook), di dimensioni K x L. Gli elementi del codebook sono tipicamente {-1, +1} o {0, 1}.

Nel nostro caso un tipo di matrice potrebbe essere una 6x12 (6 classi x 12 classificatori binari)

Classe	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}	C_{12}
Frontal/Turning	+1	+1	-1	-1	+1	-1	+1	-1	+1	-1	+1	-1
Rear End	+1	-1	+1	-1	-1	+1	-1	+1	-1	+1	-1	+1
Parked Vehicle	-1	+1	+1	-1	-1	-1	+1	-1	+1	+1	-1	-1
Sideswipe	-1	-1	+1	+1	+1	+1	-1	+1	-1	-1	+1	+1
Fixed/Other Object	+1	+1	+1	+1	+1	+1	-1	-1	+1	+1	+1	-1
Vulnerable User	-1	+1	+1	+1	+1	-1	+1	+1	-1	+1	-1	-1

Per ciascuna delle L colonne del codebook si addestra,quindi, un classificatore binario di base (regressione logistica,svc o altri). Ogni classificatore binario impara a distinguere le classi in base al valore del bit in quella specifica posizione del loro codice.

Quando arriva una nuova istanza da classificare: questa viene data in input a tutti gli L classificartori binari.

L'output combinato di questi L classificatori forma un "codice osservato".

Dopodiché questo codice viene confrontato con tutti i codici originali del codebook.

N.B per "codice originale" si intende la sequenza predefinita di bit che viene assegnata a ciascuna classe specifica all'inizio del processo, prima ancora di iniziare l'addestramento.

Es. Se si hanno tre classi: "Gatto", "Cane", "Uccello", il codebook potrebbe essere:

- Gatto: [1, -1, -1] <-- Questo è il codice originale per "Gatto"
- Cane: [-1, 1, -1] <-- Questo è il codice originale per "Cane"
- Uccello: [-1, -1, 1] <-- Questo è il codice originale per "Uccello"

La classe il cui codice originale è più simile al codice osservato viene scelta come predizione finale. La similarità si misura tipicamente con la **distanza di Hamming**, che misura il numero di posizioni in cui i bit differiscono tra il codice osservato e il codice originale. L'obiettivo, quindi, sarà minimizzare questa distanza.

Il concetto cardine dell'ECOC è che grazie alla struttura complessiva del codebook e il meccanismo di decisione basato sulla distanza è permesso tollerare un certo numero di

errori da parte dei classificatori binari e rendere la previsione più robusta.

Questa robustezza è garantita anche grazie alla ridondanza, infatti i codici nel codebook sono progettati per essere più lunghi del minimo indispensabile per distinguere le classe.

Impostazione del modello ECOC

La nostra matrice è stata impostata con un 'code_size' di 2 (quindi verranno addestrati 2*K classificatori binari dove K è il numero delle classi), questo garantisce una certa ridondanza alla matrice.

Il nostro metodo ECOC è stato istanziato con un ensemble di regressioni logistiche

```
base_estimator = LogisticRegression(
    random_state=42,
    solver='saga',
    max_iter=2000,
    n_jobs=1,
    class_weight='balanced',
    C=1.0
)
```

In questo caso si è deciso di impostare il parametro di regolarizzazione C a 1, il che rappresenta un buon compromesso tra una forte e una debole regolaiInoltre, è stato anche impostato il parametro class_weight='balanced' che gestisce lo squilibrio tra le classi automaticamente. In questo modo il modello impara a distinguere la classe meno frequente poiché si da più importanza agli errori sulla classe minoritaria.

Infine, viene impostato il parametro solver='saga' è un ottimizzatore che si adatta a dataset potenzialmente grandi.

Il modello ha ottenuto i seguenti risultati

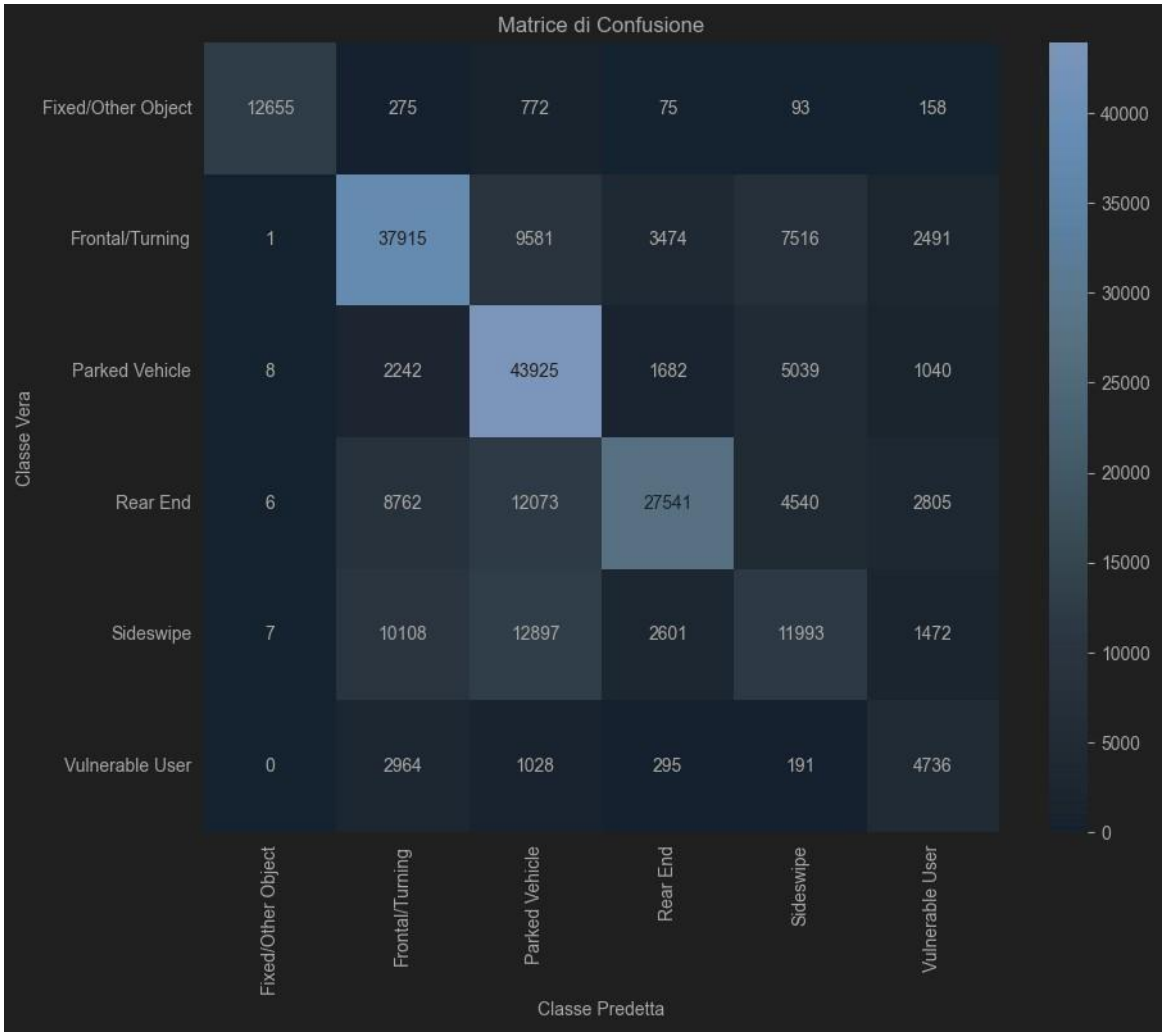
```
Inizio valutazione modello...
Accuratezza: 0.5957

Classification Report:
                    precision    recall  f1-score   support

Fixed/Other Object       1.00      0.90      0.95     14028
   Frontal/Turning       0.61      0.62      0.62     60978
    Parked Vehicle       0.55      0.81      0.65     53936
          Rear End       0.77      0.49      0.60     55727
         Sideswipe       0.41      0.31      0.35     39078
   Vulnerable User       0.37      0.51      0.43      9214

          accuracy                           0.60    232961
         macro avg       0.62      0.61      0.60    232961
      weighted avg       0.61      0.60      0.59    232961
```

Di seguito poi è riportata anche la matrice di confusione



Come osservato dai risultati ottenuti il modello ha prestazioni simili alla random forest con un accuracy e un f1 score di circa 0,60.

Complessivamente, il modello mostra prestazioni eterogenee tra le diverse categorie di incidenti. Emerge una forte capacità nel riconoscere con precisione gli impatti contro oggetti fissi o di altra natura, raggiungendo risultati quasi perfetti per questa classe. Tuttavia, si osservano difficoltà significative per altre tipologie: la classe "Sideswipe" risulta particolarmente problematica, con bassi tassi sia di corretta identificazione sia di precisione nelle sue predizioni. Anche la classificazione degli incidenti che coinvolgono utenti vulnerabili presenta criticità, con il modello che, pur identificandone circa la metà, spesso etichetta erroneamente altri incidenti come tali. Una tendenza evidente è la sovra-predizione della classe "Parked Vehicle"; sebbene molti incidenti reali con veicoli parcheggiati vengano colti, questa categoria viene frequentemente assegnata anche a incidenti di diversa natura, in particolare a scapito della corretta identificazione dei tamponamenti ("Rear End"), di cui quasi la metà non viene riconosciuta come tale. Le collisioni frontali o durante svolte ("Frontal/Turning") ottengono risultati moderati, indicando un margine di miglioramento nella distinzione da altre dinamiche simili. In sintesi, mentre alcune distinzioni sono ben apprese, il modello fatica a risolvere ambiguità tra certe classi, con una particolare tendenza a generalizzare verso "Parked Vehicle" e a sottoperformare su "Sideswipe" e "Vulnerable User".

Limiti analisi

Il modello ECOC è stato impostato scegliendo come classificatore di base una regressione logistica per via di limiti computazionali.

Infatti, nello script 'project_runner.ipynb' è presente una cella dedicata all' inizializzazione del modello e si può sostituire ad esempio con un SVC, con la possibilità di impostare dei kernel non lineari come il polinomiale e l'rbf. Un modello ECOC costruito con un ensemble di SVC avrebbe garantito sicuramente risultati migliori poiché i singoli classificatori binari in questo caso potevano catturare anche relazioni non lineari tra le varie classi.

CONCLUSIONI

Con il seguente elaborato è stato svolto un lavoro di analisi, sulle possibili cause che possono incidere sull'accadimento di diverse tipologie di incidenti stradali.

I risultati, come riportati di seguito, non sono stati abbastanza soddisfacenti a causa delle limitate risorse hardware che non hanno consentito un lavoro più approfondito.

Modello di Classificazione	Accuratezza
Multiclass Logistic Regression	0.4986
Random Forest (Multiclasse)	0.5980
ECOC Model (con Logistic Regression)	0.5964
ECOC Model (con Random Forest/SVC)	?

Tuttavia, si presuppone che con un'implementazione del modello di base più avanzato come SVC o random forest, nel modello ECOC, si possano ottenere risultati sicuramente migliori con un'attenta regolarizzazione degli iperparametri.

Sicuramente possiamo osservare un netto miglioramento tra Multiclass Logistic Regression e ECOC Model con Logistic Regression (+0,10), questo sta a significare che dividere un problema multiclasse in un sottoproblema binario ha migliorato le performance del modello.

BIBLIOGRAFIA

[1]Traffic Crashes - Crashes | City of Chicago | Data Portal