Progetto ingegneria della conoscenza AA 2022-2023


# Train delay project

Vito Proscia mat. 735975

# Table of content



# Contesto

Il **settore ferroviario** riveste un ruolo cruciale nel trasporto pubblico andando a connettere diverse città e regioni permettendo ai cittadini di spostarsi per diversi motivi: lavorativi, di studio, di piacere, etc...

Tuttavia, sia **l'organizzazione** che i **ritardi** nei servizi ferroviari possono causare disagi significativi per i passeggeri.

Il progetto che si sta presentando è incentrato su questo tema per migliorare l'esperienza di viaggio, incoraggiando le persone a spostarsi con i mezzi pubblici sia per motivi economici che ecologici andando a ridurre in maniera significativa l'inquinamento nelle città.

# Obiettivo

L'obiettivo del progetto è la realizzazione di un **sistema di ricerca treni** capace di trovare automaticamente il miglior itinerario di viaggio sulla base della stazioni di partenza e di arrivo.

Il sistema progettato, grazie all'impiego di un modello di intelligenza artificiale (AI) offrirà per ogni treno una **predizione del probabile ritardo**, andando a ridurre significativamente il disagio da parte di noi viaggiatori.

# Acquisizione dati

Sia per la costruzione delle base di conoscenza che per la ricerca e per l'apprendimento automatico si sono recuperati i dati relativi alla schedule dei treni e alla informazioni relative alle stazioni.

I dati utilizzati per la realizzazione del sistema in tutte le sue parti sono stati recuperati sia per mezzo dell'interrogazione tramite API al sito: viaggiatreno, in particolare le info. relative ai treni.

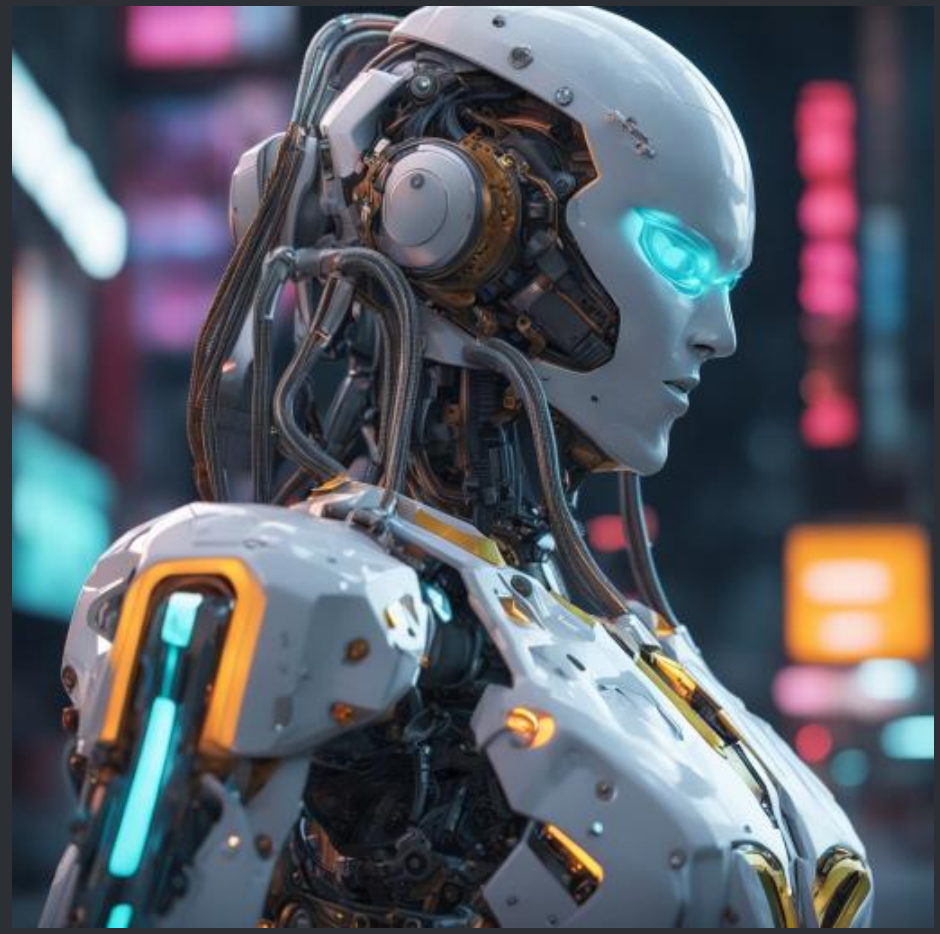
Mentre i dati relativi alle stazioni ferroviarie sono stati reperiti dal repository trenitalia.

# Rappresentazione della conoscenza

Per la realizzazione del motore di ricerca è stato necessario studiare come meglio rappresentare formalmente i dati, questo è uno step fondamentale sul quale si poggia tutto il sistema.

Per gestire la conoscenza acquisita si è progettata una **knowledge base** (base di conoscenza), una raccolta strutturata di informazioni per facilitare la gestione di dati da parte del sistema.

Si è utilizzato il Prolog, linguaggio logico, per realizzare la KB, definendo **fatti e regole** riguardanti la schedule dei treni e le stazioni.



# Rappresentazione della conoscenza

In funzione all'obiettivo si è pensato di descrivere ogni treno da:

- *train_id*, identificatore univoco del treno;
- *train_type*, tipo di treno (Regionale/Nazionale);
- *origin_id*, stazione di partenza;
- *arrival_id*, stazione di arrivo;
- *departure_time*, orario di partenza (HH:MM);
- *arrival_time*, orario di arrivo (HH:MM);
- *stops*, lista delle fermate.

```
train(320, nazionale, s01700, s01301, '15:10', '15:58', [s01700, s01307, s01301]).
train(321, nazionale, s01301, s01700, '18:02', '18:50', [s01301, s01307, s01700]).
train(322, nazionale, s01700, s01301, '17:10', '17:58', [s01700, s01307, s01301]).
train(323, nazionale, s01301, s01700, '20:02', '20:50', [s01301, s01307, s01700]).
train(324, nazionale, s01700, s01301, '19:10', '19:58', [s01700, s01307, s01301]).
train(325, nazionale, s01301, s01700, '22:02', '22:50', [s01301, s01307, s01700]).
```

# Rappresentazione della conoscenza

Mentre ogni stazione si è pensato di rappresentarla da:

- *station_id*, identificatore univoco della stazione;
- *station_name*, nome delle stazione;
- *region_name*, regione.

```
1  station(s06950, "BINARIO S.MARCO VECCHIO", "Toscana").
2  station(s11113, "BISCEGLIE", "Puglia").
3  station(s00870, "BISTAGNO", "Piemonte").
4  station(s01202, "BISUSCHIO VIGGIU", "Lombardia").
5  station(s11501, "BITETTO PALO DEL COLLE", "Puglia").
6  station(s03313, "BIVIO D`AURISINA", "Friuli Venezia Giulia").
```

# Rappresentazione della conoscenza

Una volta costruita la knowledge base, per andare ad interagire con quest'ultima si sono progettate delle **query predefinite** che, in base alle esigenze dell'utente, sono utili ad **estrapolare i dati di interesse**.

Ecco le principali:

```
% Rule for finding all trains departing from a station
trains_departure_from_station_name(StationName, Trains) :-
    findall(TrainID, (station(DepartureStationID, StationName, _),
train(TrainID, _, DepartureStationID, _, _, _, _)), Trains).

```

Regola 1: Restituisce la lista di tutti i treni (`Trains`) che partono da una determinata stazione (`StationName`)

```
% Rule for finding all trains leaving a station after a specific t
ime (HH:MM)
trains_departure_from_station_name_at_time(StationName, Departure,
Trains) :-
    findall(TrainID, (station(DepartureStationID, StationName, _),
train(TrainID, _, DepartureStationID, _, DepartureTime, _, _), equ
al_major_time(DepartureTime, Departure)), Trains).

```

Regola 2: Restituisce la lista di treni (`Trains`) che partono da una stazione specifica (`StationName`) ad un determinato orario di partenza (`Departure`)

# Rappresentazione della conoscenza

```
% Rule for finding all trains between two stations
trains_departure_between_stations_name(DepartureStationName, ArrivalStationName, Trains) :-
    findall(TrainID, (station(DepartureStationID, DepartureStationName, _),
        station(ArrivalStationID, ArrivalStationName, _),
        train(TrainID, _, DepartureStationID, ArrivalStationID, _, _, _)),
        Trains).
```

Regola 3: Restituisce la lista di treni (`Trains`) che collegano due stazioni specifiche

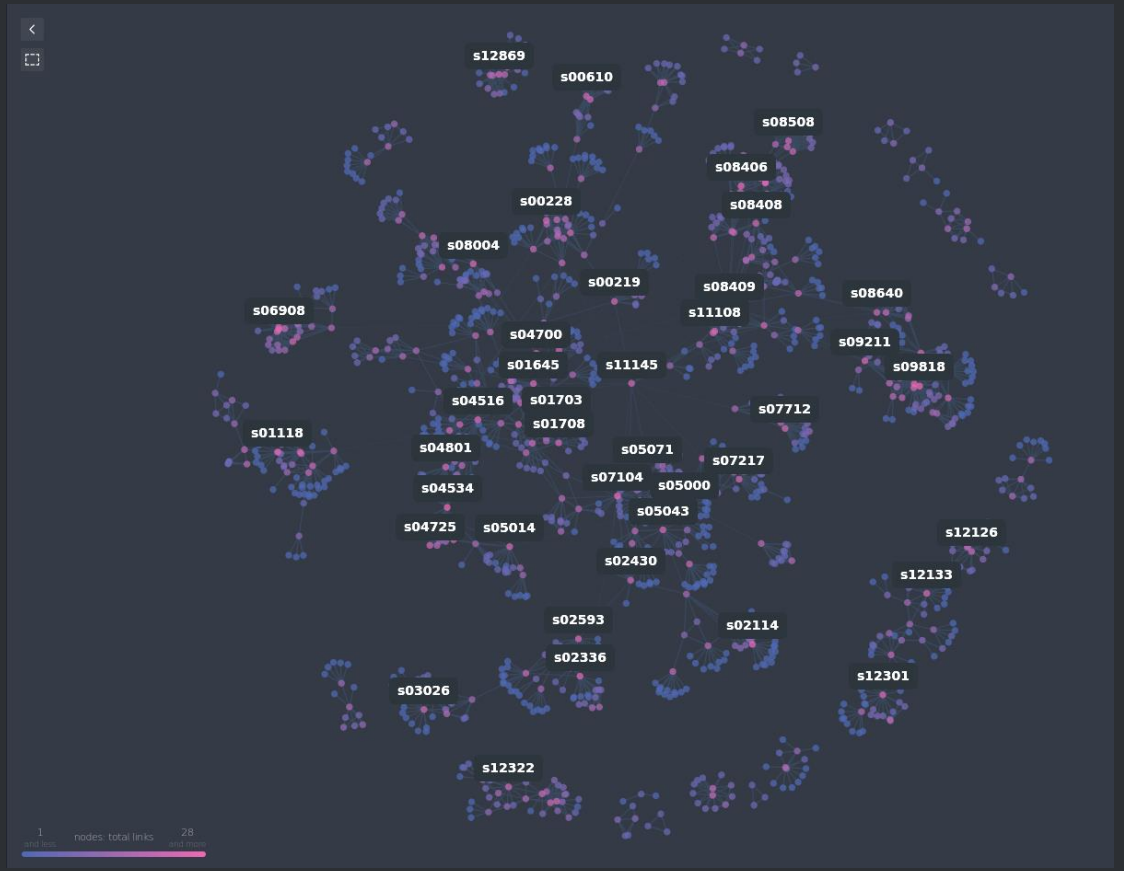
```
% Rule for finding all trains between two stations after a specific time (HH:MM)
trains_departure_between_stations_name_at_time(DepartureStationName, ArrivalStationName, Time, Trains) :-
    findall(TrainID, (station(DepartureStationID, DepartureStationName, _),
        station(ArrivalStationID, ArrivalStationName, _),
        train(TrainID, _, DepartureStationID, ArrivalStationID, DepartureTime, _, _),
        equal_major_time(DepartureTime, Time)),
        Trains).
```

Regola 4: Restituisce la lista di treni (`Trains`) che collegano due stazioni specifiche, dopo un certo orario

# Ricerca

Per la funzionalità di **ricerca del migliore itinerario** di viaggio è stato utile progettare e costruire un **grafo** che avesse come nodi *station_id* e la presenza di un arco tra due nodi indicasse un collegamento ferroviario tra le due stazioni.

Il sistema adopera **l'algoritmo di Dijkstra** che trova il percorso migliore tra due nodi in un grafo pesato, in questo caso specifico "percorso migliore" si traduce in **minor numero di stazioni** da percorrere, quindi il peso per ogni arco è stato impostato ad uno.



# Machine learning

Una delle funzionalità principali del sistema è fornire **previsioni sul probabile ritardo** che un treno potrà effettuare.

Questo è stato possibile addestrando un modello di **Machine Learning** sui dati precedentemente raccolti e raffinati.

In particolare si sono sperimentati due task:

- **Regressione**, per predire il valore del probabile ritardo;
- **Classificazione binaria**, per predire se il treno farà o meno ritardo.

Il task che ha avuto maggiore successo è stato quello delle classificazione, per questo è stato integrato nel sistema di ricerca.

# Data pre-processing

Per poter utilizzare i **dati** ottenuti come learning per i modelli si è dovuto necessariamente effettuare delle operazioni su di essi per **migliorarne la qualità e l'usabilità**.

In prima istanza si sono andate ad eliminare tutte le osservazioni ritenute problematiche, come valori mancanti e incoerenti, successivamente si sono andati a sostituire alcuni caratteri delle feature nominali per meglio gestire il dataset poi, in base ai due task che si andranno ad effettuare, si sono trasformati i valori di alcuni attributi, si è effettuata una binarizzazione dei valori di *train_type* e di *delay* per la classificazione.

In seconda istanza si è andati a fare **l'ingegnerizzazione delle features** separando quelle di input da quella target (*delay*), per poi sceglie tra quelle di input quelle che meglio si adattavano ai modelli selezionati.

# Regressione e classificazione

Sia per la regressione che per la classificazioni si è utilizzata la **Random Forest**, modello di apprendimento supervisionato facente parte delle categoria ensemble learning.
La random forest combina vari alberi decisionali, ognuno addestrato su un subset casuale dei dati e su un subset casuale delle caratteristiche.

Per ogni nodo di un albero viene scelto il miglior attributo di split secondo un determinato criterio, nel nostro caso per:

- **Regressione** si è usato il *"mean squared error";*
- **Classificazione** si è usato *"gini"* che si basa sulla probabilità.

Alla fine ogni albero effettua una previsione, la random forest le combina per ottenere un risultato finale più accurato e stabile.

# Risultati

| Metrica | Valore |
|---|---|
| MAE | 1.27 |
| MSE | 4.86 |
| R^2 score | 0.58 |

Risultati regressione

| Metrica | Valore |
|---|---|
| Precision | 0.94 |
| Recall | 0.87 |
| F1 score | 0.90 |
| AUC-ROC | 0.91 |

Risultati classificazione

# Risultati

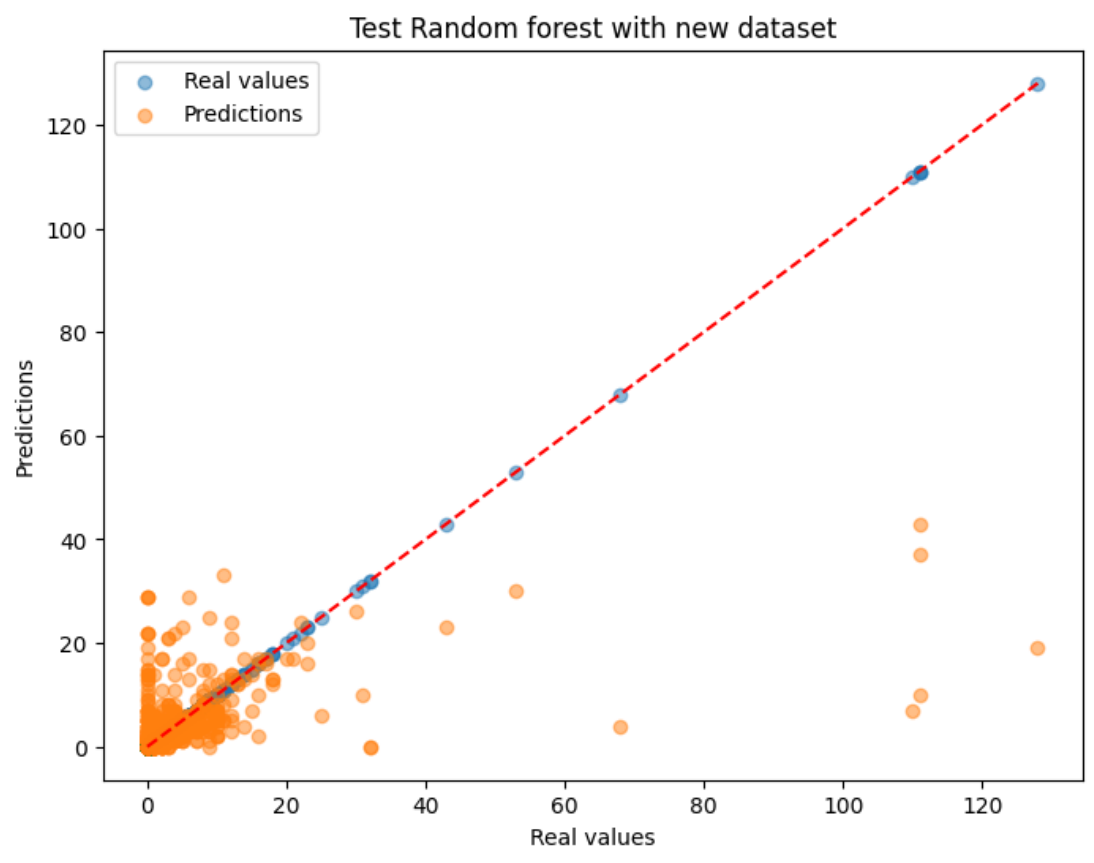



Grafico di confronto tra valori effettivi e previsioni

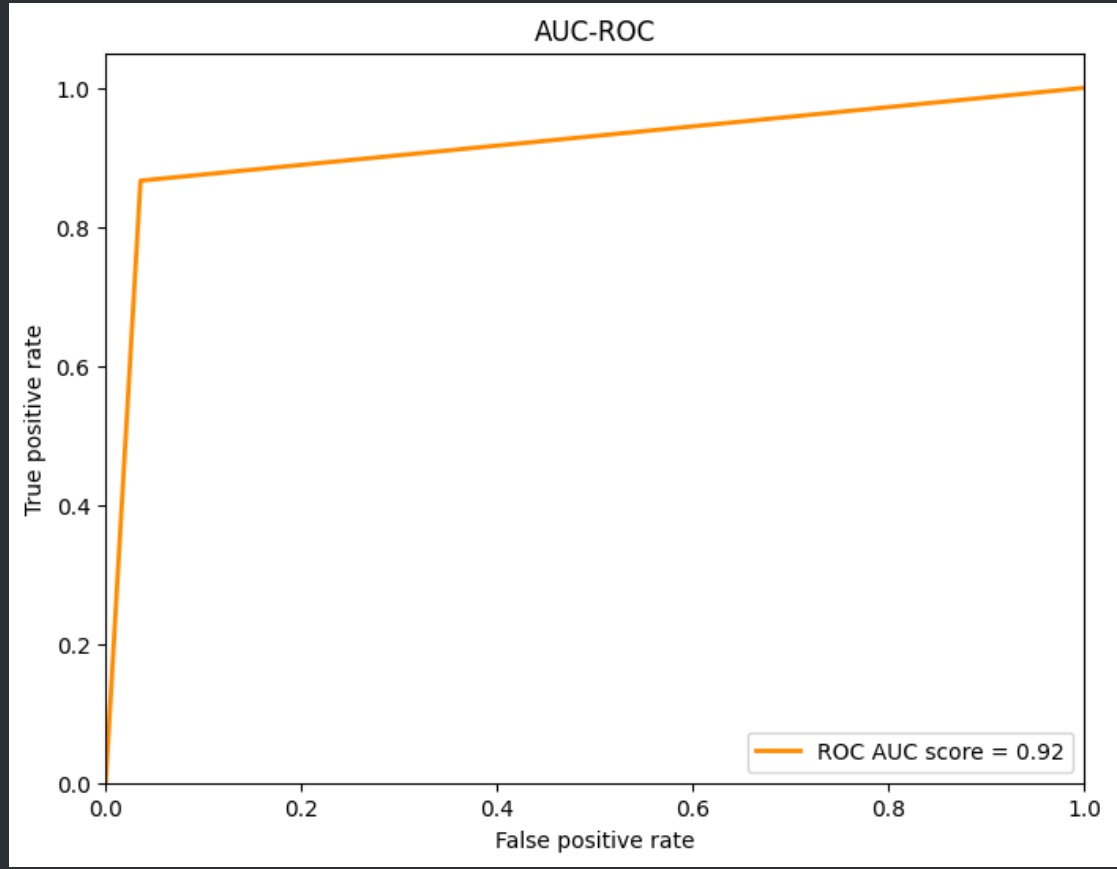



Grafico ROC (Receiver Operating Characteristic curve)

# Conclusioni

Analizzando i risultati, si è scelto di **integrare**, nel sistema, il modello prodotto dal task di **classificazione binaria**, mentre quello prodotto dalla regressione si è ritenuto troppo "immaturo", necessitando, quindi, di ulteriori ricerche e sperimentazioni.

L'obiettivo principale era sviluppare un sistema che potesse **aiutare le persone** a trovare la migliore soluzione per il viaggio in treno, andando a combinare la **ricerca automatica** di un itinerario e **l'intelligenza artificiale** per predire se il treno farà ritardo, andando così a limitare il disagio da parte dei viaggiatori.

Considerando il tutto si può dire di aver **raggiunto l'obiettivo** preposto, ottenendo un sistema efficiente e soprattutto d'aiuto per noi viaggiatori.