



UNIVERSITÀ DEGLI STUDI DI BARI ALDO MORO

DOCUMENTAZIONE DI PROGETTO ICON 21/22

V APPELLO

Link repository progetto: <https://github.com/marcosallustio/MoveAssistant>

Il progetto è stato svolto da:

- Marco Sallustio
 - Matricola: 723948

Introduzione

Il progetto consiste nella creazione di un'applicazione chiamata "Move Assistant" che possiede delle funzionalità utili per chi vuole viaggiare con mezzi di trasporto pubblico o anche per chi preferisce spostarsi in altro modo, all'interno della città di Bari. Infatti, con questa applicazione sarà possibile trovare il percorso più breve per arrivare a una delle stazioni di Bari, selezionando un punto di partenza ed uno di arrivo tra le stazioni presenti; oppure, sarà possibile prevedere la probabilità che un autobus sia affollato in base ad alcuni fattori inseriti dall'utente; sarà anche possibile ottenere delle informazioni aggiuntive sulle varie corse dei mezzi di trasporto pubblico, interrogando direttamente il sistema; ed infine, è presente una funzionalità che aiuterebbe in questo caso più chi si occupa della gestione dei vari autobus della città di Bari, che permette di assegnare una classe (A,B,C,D) ad ogni autobus, in base a delle caratteristiche che è possibile inserire.

Nella realizzazione del progetto, sono stati trattati i seguenti argomenti spiegati durante le lezioni:

1. Implementazione di una Bayesian Network per prevedere la probabilità che un autobus sia affollato in base ad alcuni fattori inseriti direttamente dall'utente stesso.
2. Classificazione attraverso Logistic Regression, Support Vector Machines Random Forest Classifier e Decision Tree Classifier, valutando i seguenti modelli attraverso la K-Fold Cross Validation, utilizzando diverse metriche.
3. Ricerca in un grafo in cui il sistema calcola il percorso migliore, dato un punto di partenza scelto dall'utente, per raggiungere diverse stazioni della città di Bari. La ricerca viene svolta utilizzando l'algoritmo A*.
4. Creazione e interrogazione di una piccola base di conoscenza, in cui l'utente potrà effettuare alcune query al sistema, per ottenere diverse informazioni aggiuntive (orario feriale/festivo della prima ed ultima corsa di un autobus, presenza della pedana per disabili all'interno di un determinato autobus, possibilità di comprare il biglietto direttamente su un autobus...).

Librerie e materiale utilizzato

Il progetto è stato realizzato interamente in Python utilizzando le seguenti risorse:

- Libreria **pgmpy** per la creazione della Bayesian Network
- Libreria **PyQt5** per la creazione di tutte le interfacce grafiche dell'applicazione
- Libreria **Pandas**
- Libreria **sklearn** per l'utilizzo di vari algoritmi di Machine Learning
- Libreria **pytholog** per l'utilizzo della programmazione logica in Python
- Libreria **Folium** per la visualizzazione della mappa della città di Bari utilizzata per la ricerca su grafo
- Dataset con vie e luoghi della città di Bari creato direttamente da noi inserendo le varie coordinate di ogni luogo/via, il relativo nome e la distanza – in metri – tra i due luoghi/vie (*fermate.csv*)
- Dataset che contiene varie caratteristiche per classificare un autobus (*dataset.csv*)

Manuale utente

Il programma viene avviato dal file *MoveApp.py*. Una volta avviato il file, si potrà visualizzare il seguente menu, in cui sono presenti tutte le varie funzionalità dell'applicazione:



I pulsanti presenti all'interno di questo menù sono:

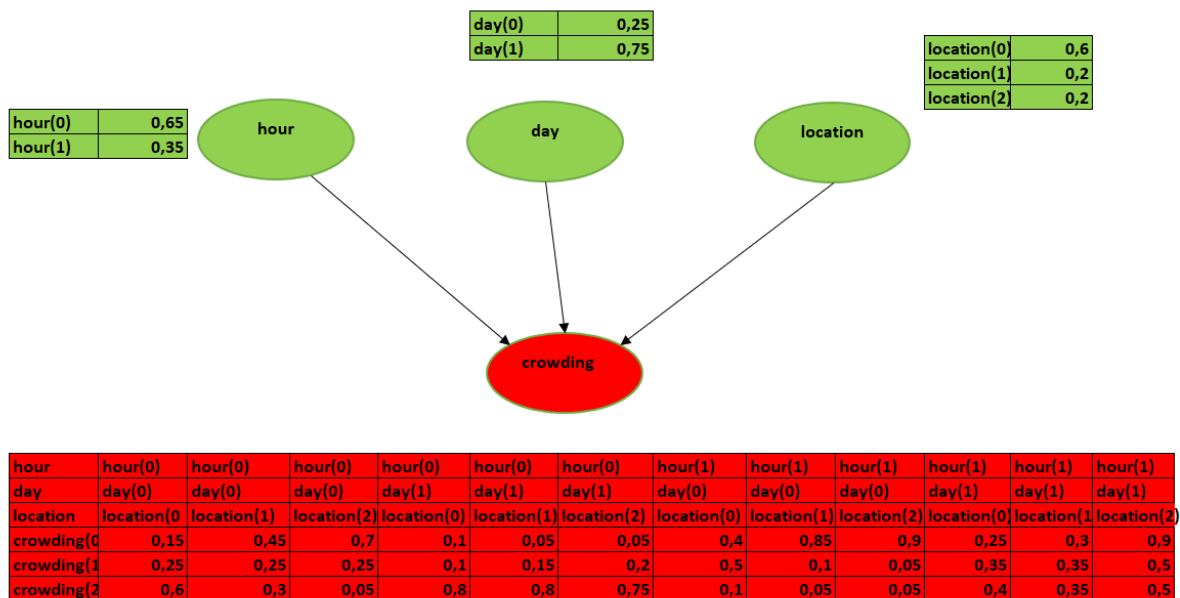
- **Trova un percorso** → Aprirà una nuova finestra che permetterà di selezionare un punto di partenza ed uno di arrivo. Una volta selezionati, sarà calcolato il percorso più breve e la relativa distanza.

- **Classificazione** → Aprirà una nuova finestra in cui sarà possibile effettuare la classificazione degli autobus e sarà possibile valutare l'efficienza di ogni classificatore utilizzato.
- **Predizione affollamento** → Aprirà una nuova finestra in cui sarà chiesto all'utente in quali condizioni desidera viaggiare e, solo dopo sarà possibile visualizzare la probabilità di affollamento dell'autobus.
- **Base di conoscenza** → Aprirà una nuova finestra in cui l'utente potrà chiedere al sistema alcune informazioni aggiuntive di vario tipo riguardanti le corse di ogni autobus.

Scelte progettuali (1-Bayesian Network)

Mi sono servito della rete Bayesiana per prevedere la probabilità che un autobus sia molto affollato, mediamente affollato o poco affollato in base a diversi fattori inseriti dall'utente.

La struttura della rete Bayesiana è la seguente:



Come si può già vedere dall'immagine, nella rete Bayesiana sono presenti quattro nodi:

- **Crowding**: indica la probabilità che un autobus sia affollato, fornendo tre livelli di affollamento: poco affollato [crowding(0)], mediamente affollato [crowding(1)] e molto affollato [crowding(2)].

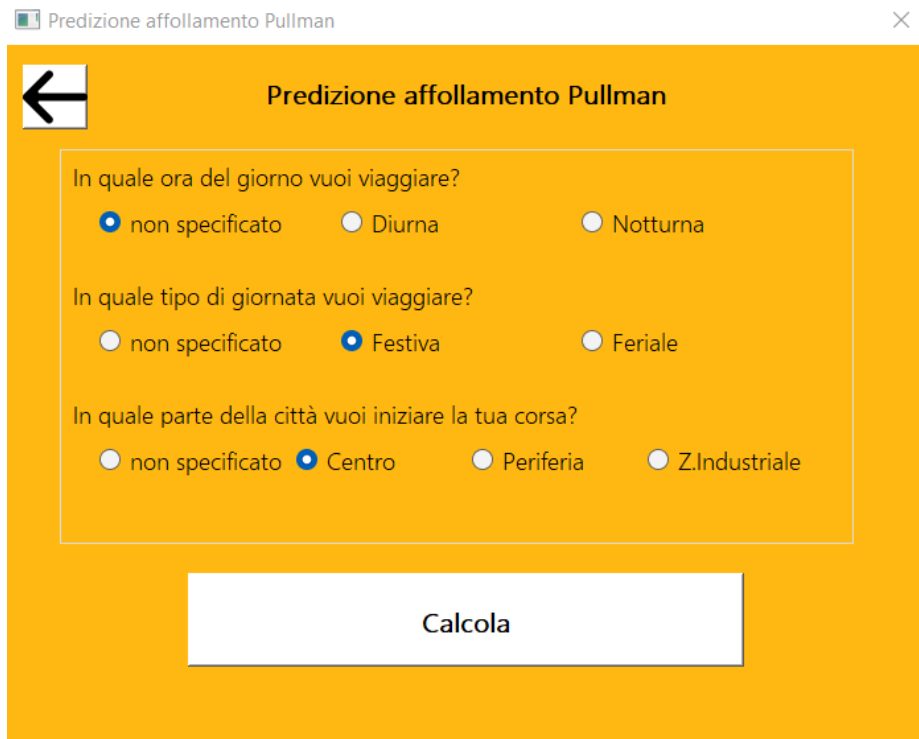
La probabilità di questo nodo è influenzata dai seguenti nodi:

- **Hour**: indica la probabilità che una persona viaggi durante un orario diurno [hour(0)] o un orario notturno [hour(1)]
- **Day**: indica la probabilità che una persona viaggi durante un giorno festivo [day(0)] o un giorno ferialo [day(1)]
- **Location**: indica la probabilità che una persona inizi la sua corsa in centro [location(0)], in periferia [location(1)] o in zona industriale [location(2)]

Tutti i vari valori di probabilità sono stati assegnati ipotizzando dei valori quanto più vicini possibili alla realtà. Ad esempio, potrebbe essere logico pensare che la probabilità che una persona viaggi durante un orario diurno è più alta rispetto alla probabilità che una persona viaggi durante la notte;

oppure che la probabilità che una persona inizi la sua corsa da una fermata in centro è più alta rispetto alla probabilità che una persona inizi la sua corsa da una fermata in zona industriale.

Una volta creata la rete bayesiana, viene chiesto all'utente attraverso un'interfaccia grafica in che condizioni desidera viaggiare, come mostrato dalla seguente immagine:



Predizione affollamento Pullman

←

Predizione affollamento Pullman

In quale ora del giorno vuoi viaggiare?

☒ non specificato ☐ Diurna ☐ Notturna

In quale tipo di giornata vuoi viaggiare?

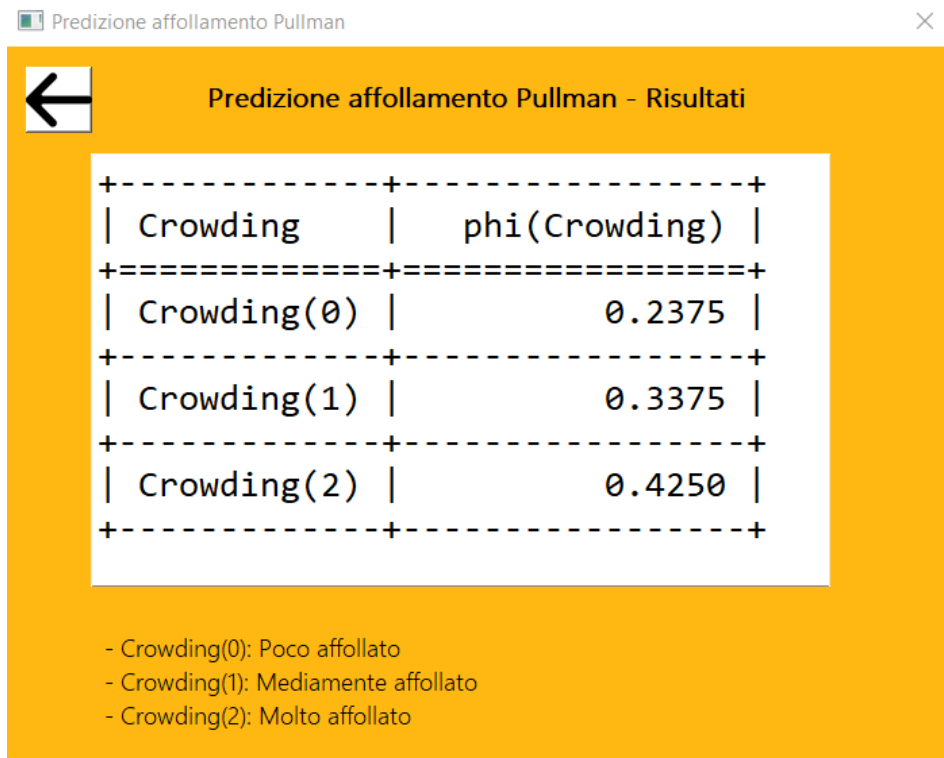
☐ non specificato ☒ Festiva ☐ Feriale

In quale parte della città vuoi iniziare la tua corsa?

☐ non specificato ☒ Centro ☐ Periferia ☐ Z.Industriale

Calcola

L'utente può anche non indicare obbligatoriamente una risposta (quindi selezionare l'opzione "non specificato"), ma l'importante è che indichi una preferenza per almeno una delle tre domande. Ad esempio, nella precedente immagine l'utente indica che vuole viaggiare in una giornata festiva e vuole iniziare la sua corsa in una fermata in centro e preferisce non specificare in quale ora del giorno vuole viaggiare. Il sistema, in base alle probabilità fornite per creare la rete bayesiana, calcolerà le seguenti probabilità:



La libreria che implementa la rete bayesiana è *pgmpy*, che permette di creare le tabelle di distribuzione di probabilità e fare inferenza sulla predizione dell'affollamento, basandosi sulla struttura della rete Bayesiana.

Scelte progettuali (2- Apprendimento supervisionato)

Ho sfruttato l'apprendimento supervisionato per classificare il grado di qualità degli autobus che effettuano il trasporto pubblico nella città di Bari, in modo che chi si occupa della parte amministrativa, possa avere un quadro di riferimento dettagliato sui pullman, per mantenere un livello alto dei mezzi in circolazione.

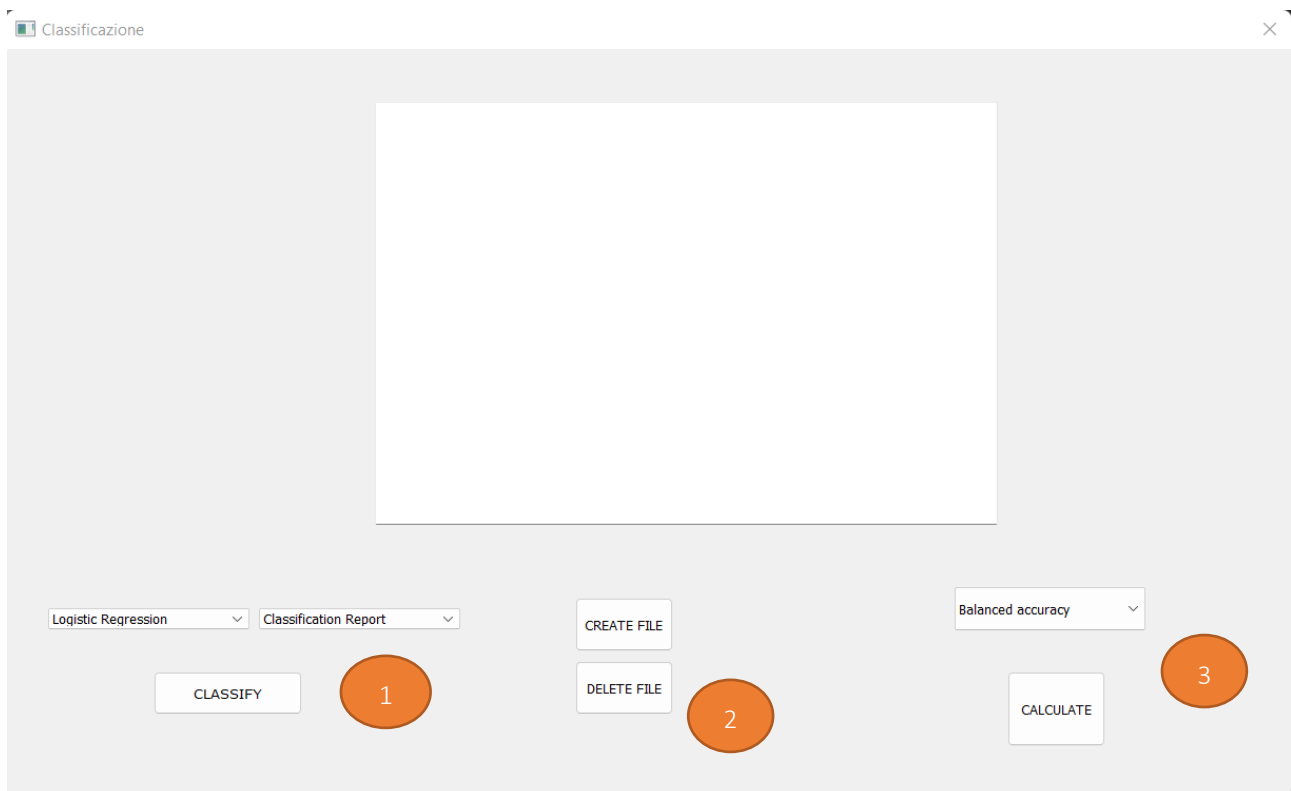
Ogni autobus può essere assegnato ad una delle quattro classi presenti:

- Classe A
- Classe B
- Classe C
- Classe D

Quindi, se un pullman ha delle buone caratteristiche, ovviamente sarà assegnato ad una classe di livello medio-alto, come la classe A o B. Al contrario, se un pullman possiede delle caratteristiche un po' obsolete o delle caratteristiche che danno un riscontro negativo tra i viaggiatori, allora sarà assegnato ad una classe di livello medio-basso, come la classe C o D.

Per implementare tutto ciò, ho utilizzato un dataset trovato online che però si occupa della classificazione delle auto. Per adattare questo dataset al nostro caso d'uso, ho modificato il tipo di attributi presenti e i relativi valori. (Vedere sezione [dataset](#))

Anche in questo caso, ho utilizzato un'interfaccia grafica che si presenta nel modo seguente:



Nel punto 1, come si può vedere è presente un menù a tendina in cui è possibile selezionare varie opzioni che riguardano la scelta dell'algoritmo di classificazione.

Nel mio caso, ho utilizzato dei modelli di apprendimento supervisionato, sfruttando la libreria *sklearn*, ovvero: **Logistic Regression, Support Vector Machines, Random Forest Classifier e Decision Tree Classifier.**

L'obiettivo è quello di fornire dei dati relativi agli autobus, ed ottenere, attraverso i classificatori, delle predizioni il più accurate possibili.

Selezionando quindi, uno dei tre classificatori e l'opzione "Classification Report" sarà possibile visualizzare tutte le varie metriche relative a quel classificatore all'interno di una tabella in modo tale da scegliere il classificatore più adatto:

REPORT LOGISTIC REGRESSION

Logistic Regression Table:

	precision	recall	f1-score	support
0	0.67	0.35	0.46	17
1	0.54	0.64	0.58	11
2	0.72	0.58	0.64	83
3	0.87	0.95	0.91	235
accuracy			0.82	346
macro avg	0.70	0.63	0.65	346
weighted avg	0.81	0.82	0.81	346

The screenshot shows the application interface with the 'Logistic Regression' dropdown selected and the 'Classification Report' dropdown also selected. The 'Logistic Regression Table' is displayed in the center, showing precision, recall, f1-score, and support for each class (0, 1, 2, 3) and overall metrics (accuracy, macro avg, weighted avg). The table data is as follows:

REPORT SUPPORT VECTOR MACHINES(SVM)

Classificazione

Support Vector Machines Table:

	precision	recall	f1-score	support
0	1.00	0.82	0.90	17
1	0.69	1.00	0.81	11
2	0.97	0.87	0.92	83
3	0.97	1.00	0.99	235
accuracy			0.96	346
macro avg	0.91	0.92	0.91	346
weighted avg	0.96	0.96	0.96	346

Support Vector Machines

Classification Report

CREATE FILE

Balanced accuracy

CLASSIFY

DELETE FILE

CALCULATE

REPORT RANDOM FOREST CLASSIFIER

Classificazione

Random Forest Classifier Table:

	precision	recall	f1-score	support
0	0.88	0.82	0.85	17
1	0.56	0.91	0.69	11
2	0.99	0.90	0.94	83
3	1.00	1.00	1.00	235
accuracy			0.97	346
macro avg	0.85	0.91	0.87	346
weighted avg	0.97	0.97	0.97	346

Random Forest Classifier

Classification Report

CREATE FILE

Balanced accuracy

CLASSIFY

DELETE FILE

CALCULATE

REPORT DECISION TREE CLASSIFIER



Come si può vedere, nella report risultante, sono presenti 4 classi (0,1,2,3) e per ognuna vengono forniti i relativi risultati. Queste classi corrispondono alle 4 classi (A,B,C,D) in cui i vari autobus saranno classificati.

Vengono visualizzate le classi 0,1,2,3 perché durante la fase di pre-processing, viene utilizzato l'*Ordinal Encoder* per gestire le feature ordinali nella suddivisione di train e test set.

Questo viene fatto perché i modelli di apprendimento automatico richiedono che tutte le variabili di input e output siano numeriche; quindi dato che nel nostro dataset, abbiamo dati categoriali bisogna codificarli in numeri prima di poterli adattare e valutare un modello. Successivamente, vengono suddivise le feature obiettivo applicando un *Label Encoder*, che trasforma, anche in questo caso, i valori di tipo stringa in interi.

Selezionando invece, uno dei classificatori e l'opzione "Classification from File" sarà possibile predire la classe a cui un autobus appartiene, in base a delle caratteristiche dell'autobus presenti su un file locale che sarà creato manualmente.

Per creare questo file, come si può vedere dal punto 2, abbiamo due pulsanti: "Create File" e "Delete File". Cliccando su "Create File" sarà aperta la seguente finestra in cui sarà possibile inserire le caratteristiche di un autobus:

CLASSIFICATION

Inserisci il valore del Pullman

☐ vhigh ☐ high ☐ med ☐ low

Inserisci il livello di manutenzione del Pullman

☐ vhigh ☐ high ☐ med ☐ low

Inserisci il numero di posti del Pullman

☐ 40 ☐ 48 ☐ 62 ☐ 65more

Inserisci la possibilità di comprare il biglietto sul Pullman

☐ no ☐ yesadd ☐ yes

Inserisci il numero di porte del Pullman

☐ 2 ☐ 3 ☐ more

Inserisci il livello di sicurezza dei dispositivi per disabili

☐ low ☐ med ☐ high

Invia

È obbligatorio selezionare un'opzione per ogni domanda.

Selezionando tutte queste caratteristiche, sarà creato un file locale, chiamato "*classification.txt*" su cui sarà effettuata la classificazione.

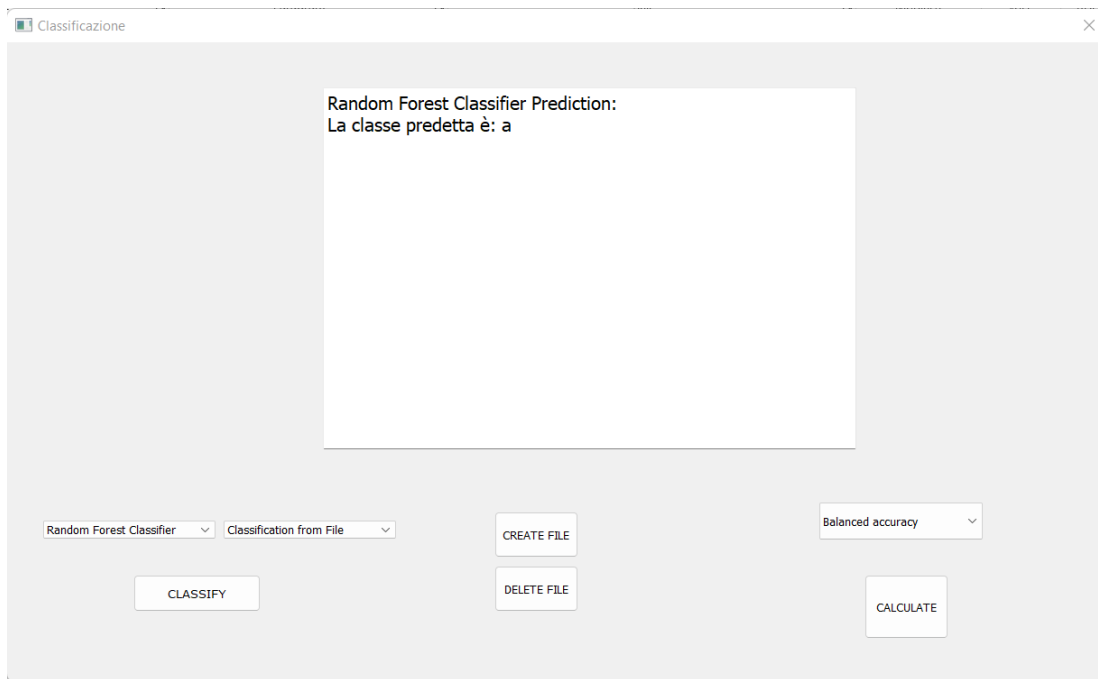
A questo punto, sarà possibile selezionare uno dei classificatori e l'opzione "Classification from file" e sarà visualizzata la classe a cui appartiene l'autobus.

Dopo di che, se si vuole effettuare una nuova classificazione con altre caratteristiche, si dovrà cancellare il file precedente attraverso il pulsante "Delete File".

Supponiamo di inserire le seguenti caratteristiche:

- Costo del pullman: basso(low)
- Costo di manutenzione: alto(high)
- Numero di posti a sedere: 48
- Possibilità di comprare il biglietto sul pullman: sì ma con un costo aggiuntivo (yesadd)
- Numero di porte del pullman: più di 3(more)
- Livello di sicurezza dei dispositivi per disabili: alto(high)

Utilizzando il Random Forest Classifier, avremo il seguente risultato:



Il punto 3, invece, riguarda la valutazione dell'efficacia dei classificatori utilizzando la metodologia della **K-Fold Cross Validation** dividendo il training set in dieci insiemi (fold, $k = 10$).

In questa metodologia, bisognerà specificare una metrica per la valutazione e sarà possibile sceglierla tra le opzioni presenti:

- Balanced Accuracy

- Accuracy

- F1

- Negative Mean Squared Error

In particolare, è possibile scegliere tra l'Accuracy e la Balanced Accuracy per evidenziare le differenze fra le due metriche; questo perché la Balanced Accuracy, viene utilizzata in caso di dataset sbilanciati. Nel nostro caso, infatti, sono presenti molti più esempi di classe D rispetto a tutti gli altri e come si può vedere i risultati tra le due metriche sono molto diversi:

ACCURACY

Logistic Regression:

0.7886207823632209

Support Vector Machines:

0.9091275709100686

Random Forest Classifier:

0.8738573733028632

BALANCED_ACCURACY

Logistic Regression:

0.6072242151787606

Support Vector Machines:

0.8096307998221874

Random Forest Classifier:

0.8006788865042452

È possibile anche utilizzare la metrica F1 e in particolare la metrica Negative Mean Squared Error che calcola la differenza tra le features predette e quelle reali, facendo il quadrato di ciò che si ottiene.

Questa metrica è utile per garantire che il nostro modello addestrato non abbia previsioni anomale con errori enormi, dato che come abbiamo detto, questa metrica attribuisce un peso maggiore a questi errori.

Scelte progettuali (3- Ricerca percorso in un grafo)

Un'altra funzionalità che è stata implementata è la ricerca del percorso più breve in un grafo. Questa funzionalità viene utilizzata per permettere a un utente di orientarsi e trovare il percorso più breve per andare da un qualsiasi punto, tra quelli presenti nel dataset, della città di Bari, a una delle stazioni principali di Bari. Nel mio caso, per comodità si è preferito scegliere le seguenti stazioni:

- Bari Centrale
- Bari Policlinico
- Bari Marconi
- Brigata Bari
- Bari Sud Est
- Bari Parco Sud
- Bari Quintino Sella

Andando più nel dettaglio, per implementare questa funzionalità ho utilizzato un file chiamato *GraphUtilities* che contiene al suo interno le classi per la creazione dei nodi e archi che compongono un grafo, e anche una classe che rappresenta il percorso che collega i diversi nodi del grafo. In questo modo viene creata la struttura del mio grafo.

In seguito, viene utilizzato il file *LocationUtilities* che contiene delle classi utili per la gestione dei vari luoghi; in particolare avrà delle funzioni che permetteranno di caricare i luoghi e le vie direttamente dal dataset.

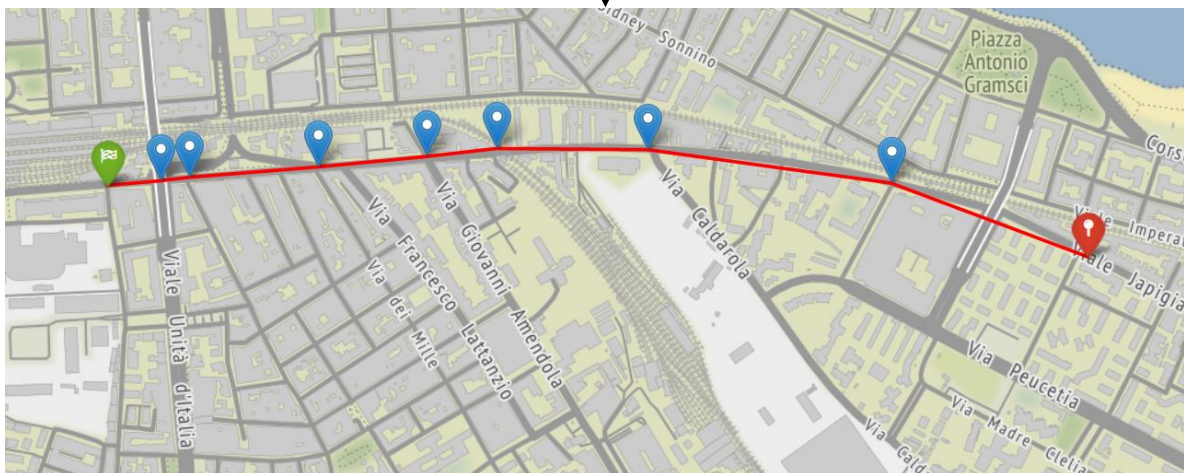
Per la ricerca del grafo, è stato implementato ***l'algoritmo A****, che ovviamente dovrà utilizzare una funzione euristica; nel mio caso ho scelto di utilizzare come funzione euristica la distanza euclidea che permette di calcolare la distanza in linea d'aria tra due punti rappresentati da coordinate. Dopo aver eseguito l'algoritmo A*, il risultato viene visualizzato graficamente su una mappa, utilizzando la libreria *Folium*.

Tutto ciò che riguarda questa funzionalità, viene eseguita in maniera interattiva, utilizzando la seguente interfaccia grafica:



Cliccando direttamente sulla mappa sarà aperta l'intera mappa con tutti i vari punti (in blu) e tutte le stazioni raggiungibili (in verde).

Ora, supponiamo che ci troviamo in Viale Japigia e vogliamo raggiungere la stazione centrale; cliccando sul pulsante "Calcola il percorso migliore" sarà visualizzata la distanza totale tra i due punti e sarà possibile visualizzare anche graficamente il percorso in modo che l'utente possa in qualche modo orientarsi:



Scelte progettuali (4-Interrogazione base di conoscenza)

Questa funzionalità presente nell'applicazione permette agli utenti di interrogare una base di conoscenza con lo scopo di ottenere le seguenti informazioni:

- Nome e ID di tutte le fermate
- Orario feriale da cui si ottiene l'ID delle fermate e il relativo orario della prima corsa della giornata e l'orario dell'ultima corsa della giornata
- Orario festivo da cui si ottiene l'ID delle fermate e il relativo orario della prima corsa della giornata e l'orario dell'ultima corsa della giornata
- Presenza o no della pedana per disabili sui vari autobus identificati da un ID
- Possibilità di comprare il biglietto direttamente sull'autobus, oppure no
- Tabella completa di tutte le fermate con i relativi nomi, ID, orari feriali/festivi

Per creare questa base di conoscenza è stata utilizzata la libreria *pytholog* che permette di utilizzare la programmazione logica in Python e in particolare di creare basi di conoscenza e interrogarle attraverso le varie asserzioni.

Come abbiamo già detto, una volta creata la base di conoscenza, è possibile interrogarla effettuando query su di essa.

Le query permettono di verificare se una clausola è conseguenza logica della base di conoscenza; nel caso in cui lo fosse il sistema risponderà “Yes”, altrimenti “No”.

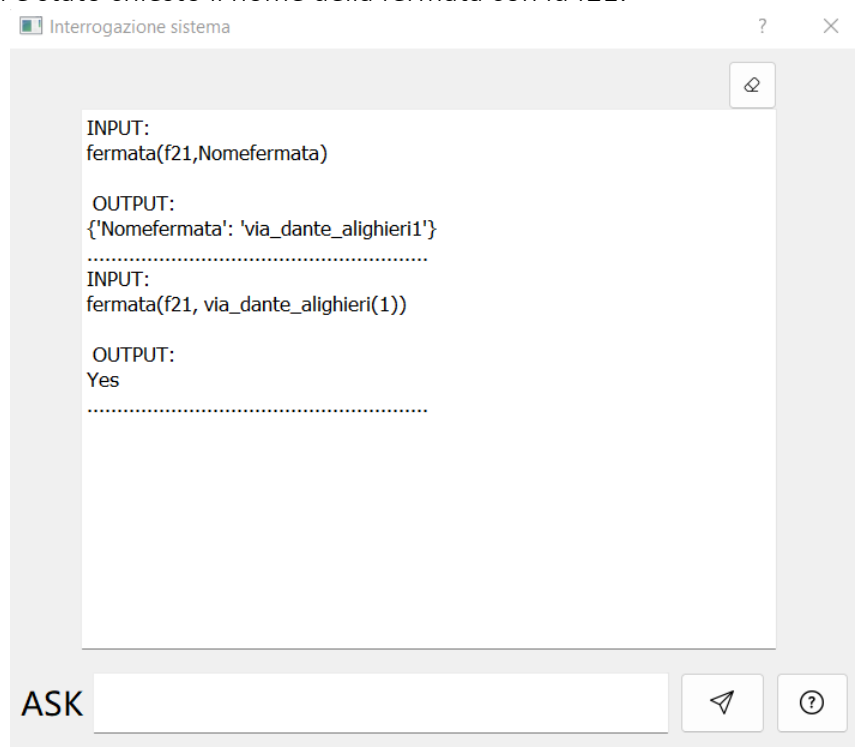
Inoltre, è possibile anche comprendere variabili, in modo che la risposta comprende tutti i valori che possono essere assunti dalla variabile: ad esempio, formulando la query

orario_feriale(Idfermata, h06_01, h20_01), la risposta comprenderà tutti gli ID delle fermata la cui prima corsa è alle 06:01 e la cui ultima corsa è alle 20:01.

Tutte le query che si possono utilizzare sono le seguenti:

SINTASSI QUERY	ESEMPIO DI UTILIZZO
<i>fermata(Idfermata, Nomefermata)</i>	<i>fermata(f21, via_dante_alighieri(1))</i>
<i>orario_feriale(Idfermata, Hprimacorsa, Hultimacorsa)</i>	<i>orario_feriale(f34, h05_18, h22_18)</i>
<i>orario_festivo(Idfermata, Hprimacorsa, Hultimacorsa)</i>	<i>orario_festivo(f47, h09_14, h00_14)</i>
<i>pedana_disabili(Idautobus, Si/no)</i>	<i>pedana_disabili(id3, n)</i>
<i>compra_biglietto_sul_bus(Idautobus, Si/no)</i>	<i>compra_biglietto_sul_bus(id2, y)</i>
<i>tabella_fermata_feriale(I, N, H1, H2):- fermata(I, N), orario_feriale(I, H1, H2)</i>	<i>tabella_fermata_feriale(f21, via_dante_alighieri(1), h09_14, h00_14)</i>
<i>tabella_fermata_festivo(I, N, H1, H2):- fermata(I, N), orario_feriale(I, H1, H2)</i>	<i>tabella_fermata_feriale(f58, viale_john_fit zgerald_kennedy, h08_06, h21_06)</i>

Anche in questo caso, viene utilizzata un’interfaccia grafica in cui è possibile scrivere direttamente la query, e la relativa risposta sarà visualizzata sulla schermata come nella seguente immagine in cui sono state fatte due interrogazioni, chiedendo prima se esistesse la fermata con id f21 in Via Dante Alighieri(1) e poi è stato chiesto il nome della fermata con id f21:



Dataset (Ricerca percorso in un grafo)

Per la ricerca del percorso in un grafo, è stato creato un file csv contenente i seguenti attributi:

- long1 → valore di longitudine del primo punto
- lat1 → valore di latitudine del primo punto
- long2 → valore di longitudine del secondo punto
- lat2 → valore di latitudine del secondo punto
- Name → nome del luogo/via
- Lenght → distanza dal punto1 al punto2

Come si può notare per ogni riga vengono salvate le coordinate di due punti; questo serve per facilitare la creazione del grafo, utilizzando queste coordinate per collegare già due punti all'interno del grafo.

Le coordinate sono state ricavate da Google Maps, così come la distanza tra due punti.

Dataset (Apprendimento supervisionato)

Il dataset utilizzato per effettuare la classificazione è stato trovato online al seguente link (<https://www.openml.org/search?type=data&sort=runs&id=40975&status=active>). Questo dataset, però, si occupa della classificazione per quanto riguarda le auto; quindi, per adattare questo dataset al nostro caso d'uso, ho modificato il tipo di attributi presenti e i relativi valori. Gli attributi presenti nel dataset sono:

- Value → Costo dell'autobus.
Può assumere i seguenti valori:
 - vhigh (molto alto)
 - high (alto)
 - med (medio)
 - low (basso)
- Maint → costo di manutenzione dell'autobus:
Può assumere i seguenti valori:
 - vhigh (molto alto)
 - high (alto)
 - med (medio)
 - low (basso)
- Seats → numero di posti dell'autobus
Può assumere i seguenti valori:
 - 40
 - 48
 - 62
 - 65more (65 e più)
- Ticket_on_bus → possibilità di comprare il biglietto sull'autobus
Può assumere i seguenti valori:
 - no
 - yesadd (si con supplemento)
 - yes

- Port_number → numero di porte dell'autobus
Può assumere i seguenti valori:
 - 2
 - 3
 - more (più delle opzioni precedenti)
- safety_devices_disabled → livello di sicurezza dei dispositivi per disabili
Può assumere i seguenti valori:
 - low
 - med
 - high
- Class → Classe a cui l'autobus appartiene
Può assumere i seguenti valori:
 - A
 - B
 - C
 - D

Implementazioni future

In futuro, si potrebbero aggiungere le seguenti feature per migliorare l'applicazione

1. Aggiunta di nuove vie/luoghi all'interno del dataset utilizzato per la ricerca del percorso
2. Aggiunta di particolari condizioni che si potrebbero verificare lungo un percorso (ad esempio situazioni di traffico o strade bloccate).
3. Aggiunta di nuove asserzioni all'interno della base di conoscenza
4. Aggiunta di nuovi nodi all'interno della Bayesian Network per rendere la predizione più realistica possibile

Conclusioni

Nonostante diverse difficoltà incontrate e nonostante il lungo tempo dedicato alla realizzazione del progetto (circa 3 mesi), posso ritenere di aver tratto il meglio da questa esperienza grazie all'apprendimento di nuove conoscenze che mi saranno sicuramente utili in futuro.

Ringrazio per l'attenzione.

Marco Sallustio