DIPARTIMENTO
DI INFORMATICA

DOCUMENTAZIONE DELL'APPLICATIVO "PiGri Airlines"

Docente: Nicola Fanizzi

Corso: Ingegneria della conoscenza

Realizzato da:

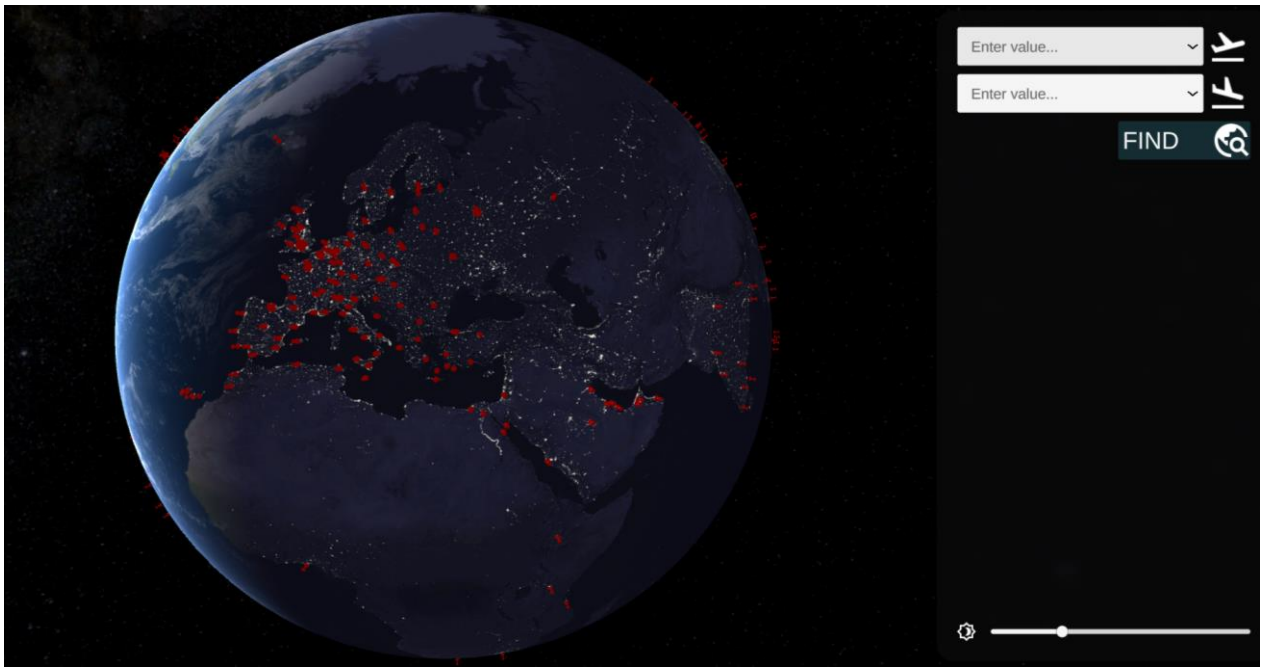
- *GIORGIO GRIMALDI* – **Matricola:** 717579 – **E-mail:** g.grimaldi23@studenti.uniba.it
- *GIULIANO PICILLI* – **Matricola:** 717580 – **E-mail:** g.picilli@studenti.uniba.it

Repository del progetto: https://github.com/Giuly123/PiGri-Airlines

INDICE

1. Introduzione

Questo documento illustra l'utilizzo della prima versione dell'applicativo PiGri Airlines, il cui nome nasce dalle iniziali dei nostri cognomi: "Picilli e Grimaldi".



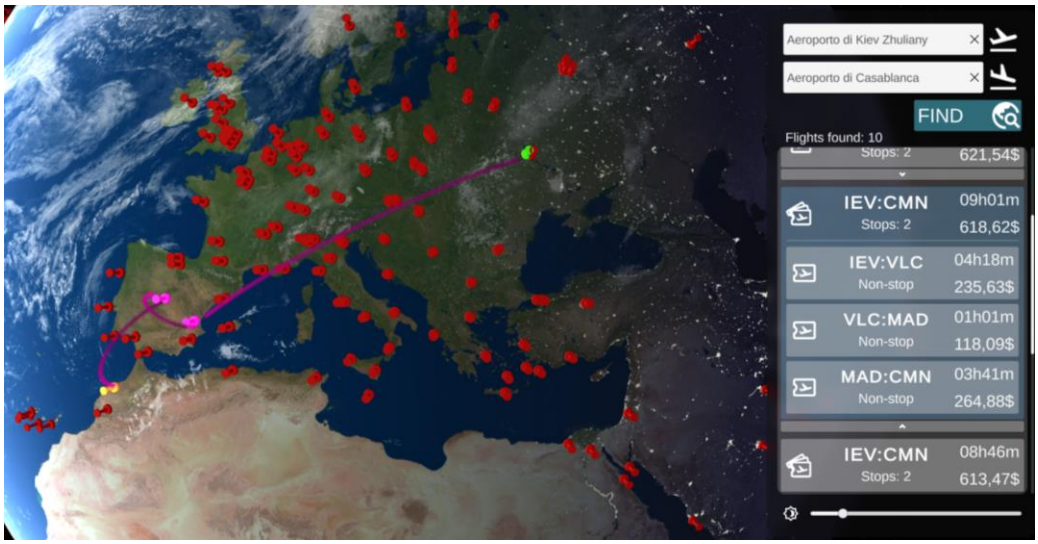
Il sistema è uno strumento per la mobilità intelligente, pensato visualizzare le tratte aeree esistenti. L'idea alla base del progetto è quella di consentire agli utenti di usufruire in modo efficiente del sistema dei voli, permettendo loro di trovare il percorso più rapido per spostarsi da un aeroporto di partenza ad uno di arrivo. Inoltre, l'applicazione farà una predizione dei prezzi e delle durate della tratta.

Il programma consente di navigare intorno ad un modello 3D della terra sul quale appariranno dei "Pin", ognuno dei quali rappresenta uno degli oltre 260 aeroporti presenti nella base di conoscenza.

I pin vengono generati e posizionati a runtime:

- Vengono prelevate le coordinate sessagesimali presenti nella base di conoscenza;
- Vengono convertite in coordinate euleriane;
- Sulla base di quest'ultime il sistema genera e posiziona il pin sul modello 3D della terra.

La UI si presenta nel seguente modo:



Il primo campo di testo consente di inserire l'aeroporto di partenza, il secondo quello di arrivo. Queste informazioni possono essere popolate anche cliccando i pin dal modello della terra. Ad esempio, cliccando il pin dell'aeroporto di Bari questo verrà automaticamente inserito nel campo vuoto.

Una volta popolati entrambi i campi sarà possibile, tramite il tasto 'Find', effettuare la ricerca. Questa verrà effettuata combinando le oltre 10000 rotte presenti nel sistema, e restituirà i primi dieci risultati ordinati per lunghezza del tragitto. Inoltre, per ogni tratta, verrà mostrato il numero degli scali e le stime di prezzo e durata del viaggio. Le suddette stime verranno calcolate sulla base di oltre 43000 esempi presenti nel training set.

L'utente, inoltre, cliccando su uno dei risultati potrà visualizzare il percorso corrispondente sul modello 3D.

2. Struttura del progetto

L'applicativo è stato sviluppato interamente in Unity, utilizzando il linguaggio C# per programmare gli script. Unity è un engine grafico che permette di realizzare applicativi 3D.

Abbiamo utilizzato le seguenti librerie:

- Accord.NET (framework): per realizzare il regressore;
- Swipl.cs (libreria): per interfacciarci con la knowledge base;

Il progetto, recuperabile tramite il repository di GitHub, è liberamente consultabile e modificabile tramite lo Unity Editor.

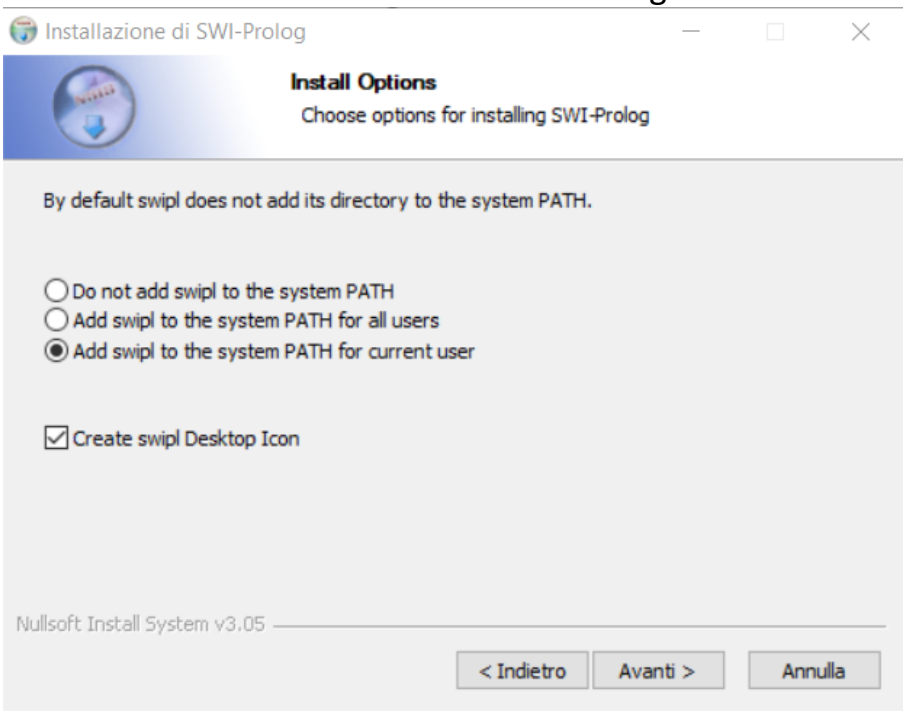
I principali file di progetto risiedono nella cartella 'Assets' e sono separati in base al tipo di risorsa nelle rispettive cartelle:

- Scripts (contente i vari scripts scritti nel linguaggio C#)
- Scenes
- Prefabs
- Materials
- Textures
- StreamingAssets (contente la base di conoscenza -> 'airports.pl', il training set per allenare il regressore -> 'travells.csv')

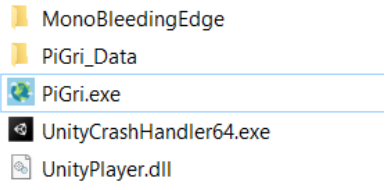
3. Requisiti funzionali e avvio

Per avviare il programma è necessario installare:

- Swi-Prolog (sito ufficiale). Una volta nel sito, andare nella sezione download, selezionare 'stable release' e scaricare la versione adatta al proprio sistema operativo. Eseguito il download, è necessario installare il programma sulla propria macchina e spuntare la casella 'Add swipl to the system PATH for current user' come mostrato anche nella figura:



Il software viene rilasciato sottoforma di archivio (PiGri-Airlines.zip) contenente l'eseguibile (PiGri.exe) e i file necessari all'avvio.



Una volta avviato si presenterà così:

Se si vuole aprire il progetto per consultarlo è necessario installare Unity Editor 2020.3.19f1.

Per mettere l'applicazione in modalità "finestra" sarà necessaria la combinazione di tasti "Alt + INVIO".

4. Dataset utilizzato

Il dataset usato per l'addestramento del sistema è stato costruito utilizzando le API fornite da Travelpayouts.com, per ottenere le informazioni reali relative alle rotte (voli diretti) presenti nella base di conoscenza. Il dataset ammonta a 43638 esempi, i quali sono strutturati come segue:

	A	B	C	D	E	F
43623	2	NAT	FOR	42	60	
43624	6	NAT	GYN	144	170	
43625	7	NAT	GYN	280	170	
43626	1	ECP	TPA	2520	89	
43627	2	ECP	TPA	1335	89	
43628	2	ECP	IAH	161	130	
43629	2	ECP	DCA	149	130	
43630	3	ECP	DCA	121	132	
43631	4	ECP	DCA	132	131	
43632	5	ECP	DCA	263	131	
43633	3	ECP	DEN	316	223	
43634	7	ECP	DEN	152	227	
43635	2	ECP	IAH	161	130	
43636	2	ECP	ATL	83	69	
43637	2	ECP	MCO	162	420	
43638	3	ECP	ORD	191	310	
43639	2	ECP	CLT	130	97	
43640	3	ECP	CLT	130	107	
43641	4	ECP	CLT	130	95	
43642	6	ECP	CLT	168	95	
43643	8	ECP	CLT	172	95	
43644	5	ECP	MCI	195	130	
43645						

- Colonna "A": numero da 1 a 12 per indicare il mese in cui verrà effettuato il volo;
- Colonna "B": codice "IATA" dell'aeroporto di partenza;
- Colonna "C": codice "IATA" dell'aeroporto di arrivo;
- Colonna "D": il prezzo del volo in euro;
- Colonna "E": la durata del volo in minuti.

5. Scelte progettuali

Nell'applicativo software si è scelto di utilizzare:

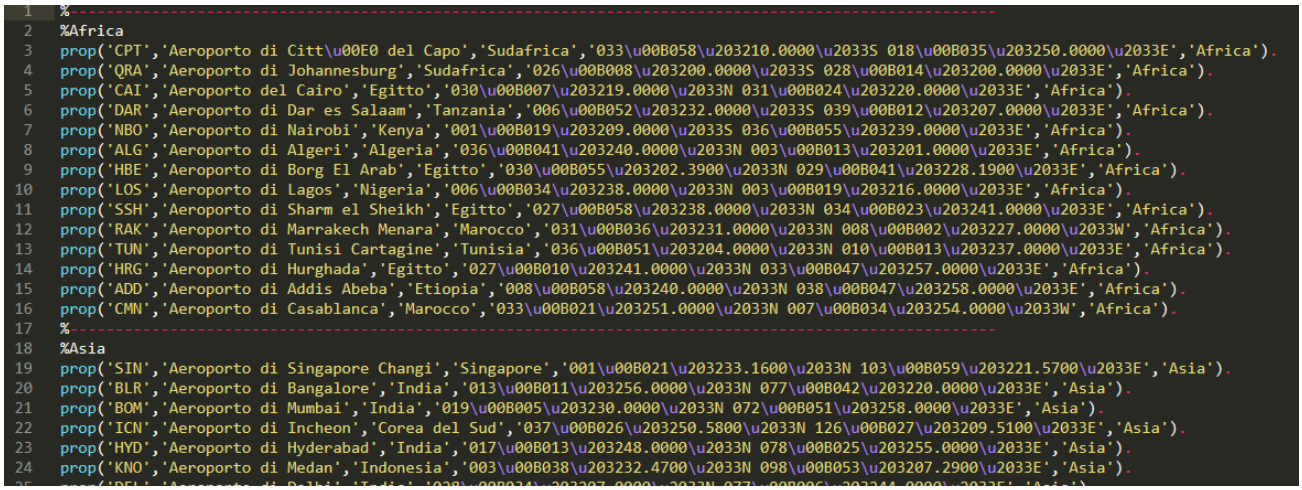
- Una base di conoscenza, scritta interamente dal gruppo sulla base di informazioni reali, che verrà interrogata tramite query.

- Una variante dell'algoritmo di ricerca informata A* che trova il percorso più rapido per ogni punto di partenza e punto di arrivo.

- Un regressore in grado di restituire la previsione del costo del biglietto e della durata del viaggio sulla base dei risultati ottenuti precedentemente dall'algoritmo di ricerca.

5.1. Modellazione Knowledge Base

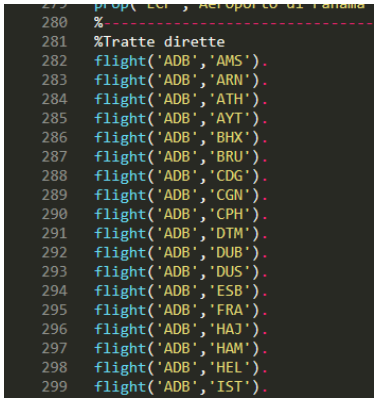
La knowledge base è composta da un insieme di regole e fatti modellati nel linguaggio Prolog, dove per ognuno dei 267 aeroporti abbiamo:



```
%-----------------------------------------------------------------------------------------------
%Africa
prop('CPT','Aeroporto di Citt\u00E0 del Capo','Sudafrica','033\u00B058\u203210.0000\u2033S 018\u00B035\u203250.0000\u2033E','Africa').
prop('QRA','Aeroporto di Johannesburg','Sudafrica','026\u00B008\u203200.0000\u2033S 028\u00B014\u203200.0000\u2033E','Africa').
prop('CAI','Aeroporto del Cairo','Egitto','030\u00B007\u203219.0000\u2033N 031\u00B024\u203220.0000\u2033E','Africa').
prop('DAR','Aeroporto di Dar es Salaam','Tanzania','006\u00B052\u203232.0000\u2033S 039\u00B012\u203207.0000\u2033E','Africa').
prop('NBO','Aeroporto di Nairobi','Kenya','001\u00B019\u203209.0000\u2033S 036\u00B055\u203239.0000\u2033E','Africa').
prop('ALG','Aeroporto di Algeri','Algeria','036\u00B041\u203240.0000\u2033N 003\u00B013\u203201.0000\u2033E','Africa').
prop('HBE','Aeroporto di Borg El Arab','Egitto','030\u00B055\u203202.3900\u2033N 029\u00B041\u203228.1900\u2033E','Africa').
prop('LOS','Aeroporto di Lagos','Nigeria','006\u00B034\u203238.0000\u2033N 003\u00B019\u203216.0000\u2033E','Africa').
prop('SSH','Aeroporto di Sharm el Sheikh','Egitto','027\u00B058\u203238.0000\u2033N 034\u00B023\u203241.0000\u2033E','Africa').
prop('RAK','Aeroporto di Marrakech Menara','Marocco','031\u00B036\u203231.0000\u2033N 008\u00B002\u203227.0000\u2033W','Africa').
prop('TUN','Aeroporto di Tunisi Cartagine','Tunisia','036\u00B051\u203204.0000\u2033N 010\u00B013\u203237.0000\u2033E','Africa').
prop('HRG','Aeroporto di Hurghada','Egitto','027\u00B010\u203241.0000\u2033N 033\u00B047\u203257.0000\u2033E','Africa').
prop('ADD','Aeroporto di Addis Abeba','Etiopia','008\u00B058\u203240.0000\u2033N 038\u00B047\u203258.0000\u2033E','Africa').
prop('CMN','Aeroporto di Casablanca','Marocco','033\u00B021\u203251.0000\u2033N 007\u00B034\u203254.0000\u2033W','Africa').
%-----------------------------------------------------------------------------------------------
%Asia
prop('SIN','Aeroporto di Singapore Changi','Singapore','001\u00B021\u203233.1600\u2033N 103\u00B059\u203221.5700\u2033E','Asia').
prop('BLR','Aeroporto di Bangalore','India','013\u00B011\u203256.0000\u2033N 077\u00B042\u203220.0000\u2033E','Asia').
prop('BOM','Aeroporto di Mumbai','India','019\u00B005\u203230.0000\u2033N 072\u00B051\u203258.0000\u2033E','Asia').
prop('ICN','Aeroporto di Incheon','Corea del Sud','037\u00B026\u203250.5800\u2033N 126\u00B027\u203209.5100\u2033E','Asia').
prop('HYD','Aeroporto di Hyderabad','India','017\u00B013\u203248.0000\u2033N 078\u00B025\u203255.0000\u2033E','Asia').
prop('KNO','Aeroporto di Medan','Indonesia','003\u00B038\u203232.4700\u2033N 098\u00B053\u203207.2900\u2033E','Asia').
```

- Il codice "IATA" dell'aeroporto;
- Il suo nome escapato;
- La nazione di appartenenza;
- Le coordinate in forma sessagesimale escapate;
- Il continente di appartenenza.

Dove per ognuna delle 10342 rotte (voli diretti) abbiamo:



```
%-----------------------------
%Tratte dirette
flight('ADB','AMS').
flight('ADB','ARN').
flight('ADB','ATH').
flight('ADB','AYT').
flight('ADB','BHX').
flight('ADB','BRU').
flight('ADB','CDG').
flight('ADB','CGN').
flight('ADB','CPH').
flight('ADB','DTM').
flight('ADB','DUB').
flight('ADB','DUS').
flight('ADB','ESB').
flight('ADB','FRA').
flight('ADB','HAJ').
flight('ADB','HAM').
flight('ADB','HEL').
flight('ADB','IST').
```

- Codice “IATA” dell’aeroporto di partenza;
- Codice “IATA” dell’aeroporto di arrivo.

5.2. Algoritmo di path finding

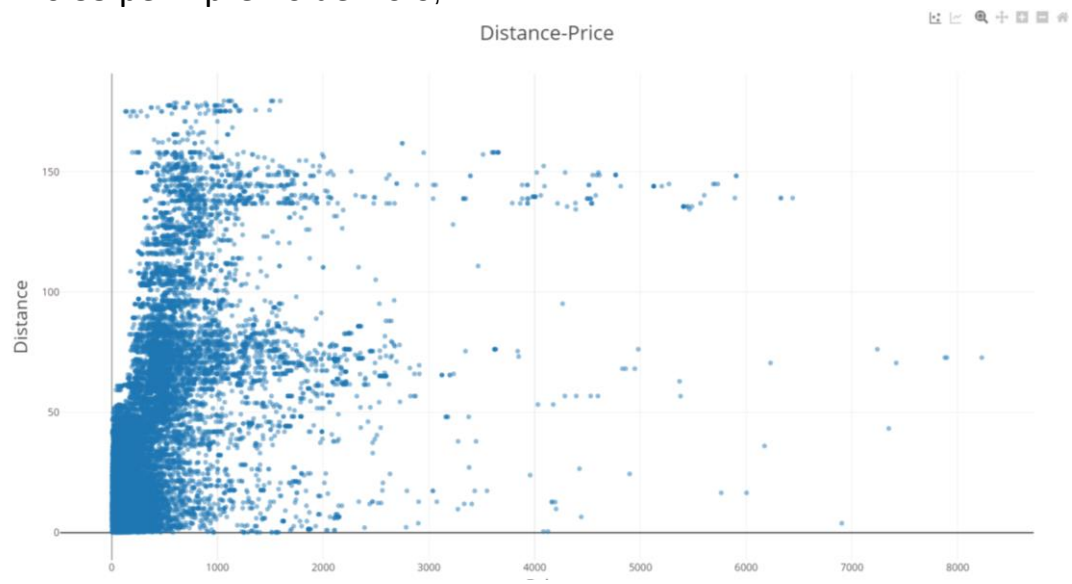
Abbiamo utilizzato una variante dell’algoritmo A* che invece di restituire un singolo risultato restituisce gli N risultati migliori trovati entro un tempo massimo di timeout. Inoltre, l’esplorazione del grafo è stata limitata imponendo un limite di costo in modo tale da ritrovare unicamente i voli con un massimo di due scali. La funzione euristica utilizzata tiene conto della distanza (normalizzata) tra le coordinate dell’aeroporto preso in esame e quelle dell’aeroporto di destinazione.

5.3. Regressore

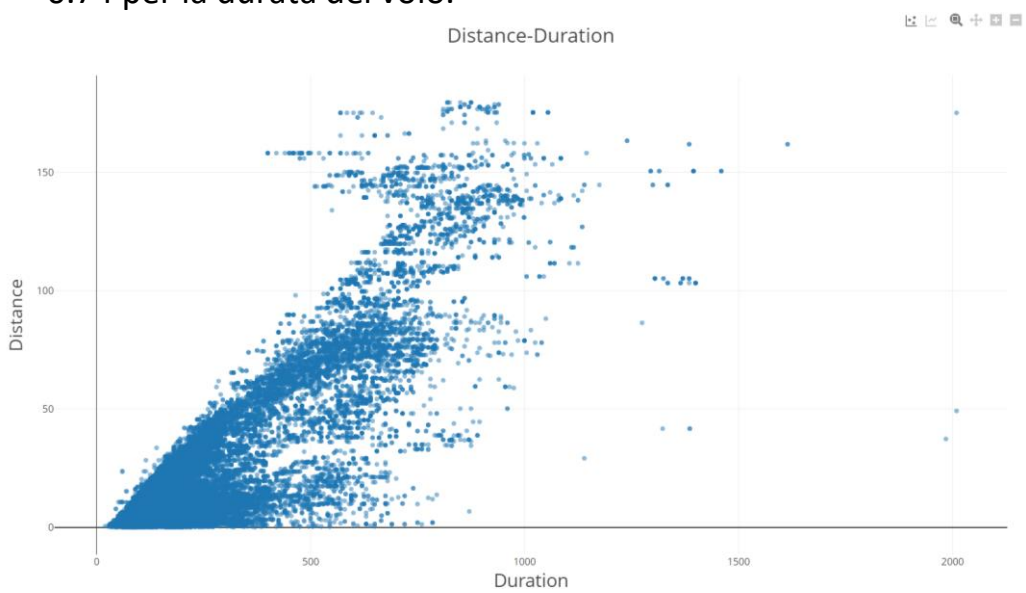
Per il regressore è stata utilizzata l’implementazione di Multivariate Linear Regression messa a disposizione dalla libreria Accord.NET. Questa permette di associare a una o più feature di input una o più feature di output.

La libreria, inoltre, fornisce una metrica di valutazione basata sul coefficiente di determinazione (R^2) la quale risulta pari a:

- 0.35 per il prezzo del volo;



- 0.74 per la durata del volo.



6. Implementazioni future

In futuro, alcune feature che potrebbero essere implementate sono:

- Aggiungere informazioni riguardanti le emissioni di CO2 emesse da ogni volo;
- Aggiungere il viaggio di ritorno data l'andata;
- Scelto un viaggio, suggerimenti su dove alloggiare, posti da visitare e ristoranti del luogo;
- Mostrare il traffico aereo in tempo reale;
- Tenere conto del mese per la predizione dei prezzi.

7. Conclusioni

Il gruppo ha visto questo progetto come un'opportunità di applicare le conoscenze teoriche apprese durante il corso. Nonostante il progetto sia stato svolto solo durante il mese di gennaio, possiamo ritenerci soddisfatti del lavoro svolto.

Ringraziamo per l’attenzione!