

UNIVERSITÀ POLITECNICA DELLE MARCHE

FACOLTÀ DI INGEGNERIA



*Corso di Laurea Magistrale in
Ingegneria Informatica e dell'Automazione*

Corso di DATA SCIENCE

Professore:

DOTT. DOMENICO URSINO

Di:

CRISTIAN COLAVITO
COSTANTINO TIGANO

ANNO ACCADEMICO 2022-2023

Sommario

Nei capitoli che seguono andremo ad analizzare, utilizzando i software e framework visti durante il corso di Data Science, diversi dataset trovati nel web facendo poi considerazioni sui risultati di queste analisi. I dataset che sono stati utilizzati contengono informazioni contestualmente diverse così da rendere questa relazione il meno monotona possibile. Nonostante questo inizialmente ci siamo concentrati sui dati inerenti la vendita di veicoli in Russia. Successivamente passeremo all'analisi e classificazione di dati relativi ai combattimenti di *Pokemon*. Proseguendo faremo Clustering, analisi sulle Serie temporali e analisi con i grafi su un dataset gentilmente concesso da un'azienda che gestisce il monitoraggio dei rifiuti in alcuni comuni italiani, avremo a disposizione l'intero dataset di un comune dell'Isola D'elba. Concluderemo questa serie di progetti con lo sviluppo di un chatbot integrato in Telegram per la gestione delle prenotazioni di un salone di parrucchieri.

Indice

1 Qlik Analisi Sito Vendita auto in Russia	5
1.1 Descrizione Dataset	5
1.2 Descrizione delle Dashboard e risultati	5
1.2.1 Foglio 1: Analisi Sugli Annunci	5
1.2.2 Foglio 2: Analisi sul Prezzo	6
1.2.3 Foglio 3: Analisi sul Carburante	8
1.2.4 Foglio 4: Analisi sulle Città	9
2 Tableau	11
2.1 Dataset	11
2.2 Trasformazione dei dati	11
2.3 Vista generale	11
2.4 Analisi sul brand	13
2.5 Analisi sul tipo di alimentazione	14
2.6 Andamento temporale della media prezzo con trend	15
3 PowerBI	17
3.1 ETL	17
3.2 Descrizione dei report	18
3.2.1 Pagina generale	19
3.2.2 Analisi sul Brand	19
3.2.3 Analisi sull'alimentazione	21
3.2.4 Relazione tra kilometraggio e media del prezzo auto	22
3.3 Annunci per regione	22
3.4 Media prezzo auto in base all'anno di immatricolazione	22
4 Python	25
4.1 Strumenti utilizzati	25
4.1.1 Python3	25
4.1.2 JupyterNotebook	26
5 Classificazione Battaglie Pokemon	29
5.1 Dataset utilizzato	29
5.2 Analisi Descrittiva	30
5.3 Preprocessing	32
5.4 Training	33
5.5 GridSearch CrossValidation	35
5.6 Pipeline Finale	37
6 Clustering Analisi Cittadina Isola D'elba	39
6.1 Dataset utilizzato	39
6.2 Analisi Descrittiva	40
6.2.1 Obiettivo del clustering	40
6.3 Preprocessing	40

6.4	Clustering su PCA con K-means sul dataset completo	41
6.5	Risultati	45
6.5.1	Mesi Estivi	45
6.5.2	Mesi non estivi	46
6.6	Conclusioni finali	47
7	TimeSeries	49
7.1	Pre-Processing	49
7.2	Stima parametri p,d,q	49
7.2.1	Parametro d	49
7.2.2	Parametro p	50
7.2.3	Parametro q	50
7.3	Addestramento e risultati Arima	50
7.4	Sarima	51
8	NetworkX	53
8.1	Dataset utilizzato	53
8.1.1	Obiettivo del grafo	53
8.2	Preprocessing	53
8.3	Creazione del grafo	56
8.4	Studio sul singolo nodo	58
9	Rasa Chat-bot Prenotazione salone di Parrucchieri	59
9.1	Rasa	59
9.1.1	Principi di funzionamento	59
9.1.2	Interrgrazione con altre piattaforme	60
9.2	Bot per la gestione delle prenotazioni di un salone di parrucchieri	61
9.3	Descrizione delle specifiche	61
9.4	Implementazione	61
9.4.1	Prenotazione appuntamenti	61
9.4.2	Esempio converasione per prenotare un appuntamento	64
9.4.3	Altre operazioni	65
9.5	Considerazioni	66
Bibliografia		67

Capitolo 1

Qlik Analisi Sito Vendita auto in Russia

1.1 Descrizione Dataset

Il dataset da noi utilizzato si compone di due file CSV relativi a regioni russe differenti:

- region25_en.csv, rappresenta i dati della regione Primorsky,
- region41_en.csv, rappresenta i dati della regione del Kamchatka.

Poichè l'insieme degli attributi risulta identico per entrambi, la descrizione di essi è la medesima e viene riportata nella seguente tabella:

Brand	Brand di appartenenza dell'auto
Name	Nome dell'auto
Fuel_type	Tipo di alimentazione dell'auto
Year	Anno immatricolazione auto
Km	Kilometraggio auto
Price	Prezzo auto dell'annuncio
Date	Data relativa alla pubblicazione annuncio
Location	Regione di riferimento

Tabella 1.1: Colonne della tabella del dataset

Non avendo relazioni tra tabelle, sono stati uniti i dataset in un'unica tabella e sono state fatte varie analisi

1.2 Descrizione delle Dashboard e risultati

In questa sezione andiamo ad analizzare i diversi fogli sviluppati oltre che le analisi che ne derivano.

1.2.1 Foglio 1: Analisi Sugli Annunci

In questo foglio andiamo ad analizzare i vari annunci, possiamo trovare vari **filtri**:

- il mese di pubblicazione degli annunci;
- la settimana di pubblicazione degli annunci;
- il brand del veicolo in vendita;
- il nome del veicolo in vendita.

Inoltre troviamo una **KPI** relativa agli annunci selezionati rispetto ai totali nel dataset e un **Grafico a barre** che racconta la distribuzione degli annunci nel tempo.

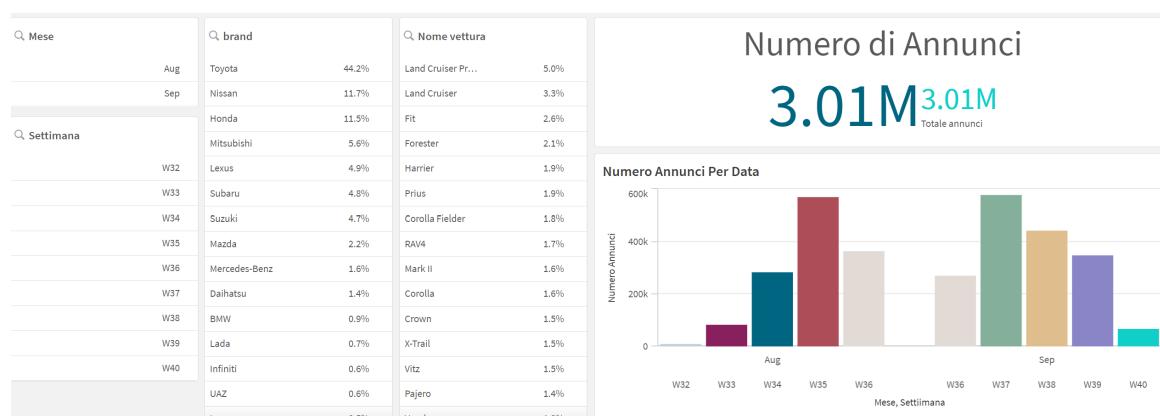


Figura 1.1: Panoramica Foglio 1

Se volessimo andare a vedere l'andamento della pubblicazione degli annunci per il mese di Agosto, notiamo un picco nella terza settimana del mese preso in esame, sul filtro relativo al brand possiamo vedere la percentuale di annunci rispetto ad un determinato brand ordinato per frequenza.

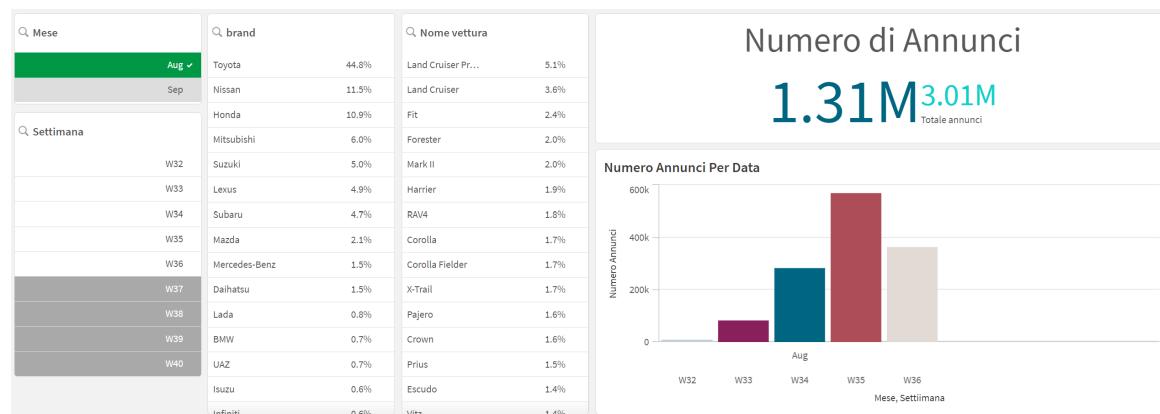


Figura 1.2: Analisi Relativa al mese di Agosto

Se invece volessimo andare a vedere l'andamento della pubblicazione degli annunci dei Top 5 Brand per numero di annunci, notiamo un picco nella terza settimana di Agosto e nella seconda di Settembre. Inoltre notiamo dal **KPI** che i top 5 Brand rappresentano la quasi totalità del dataset.

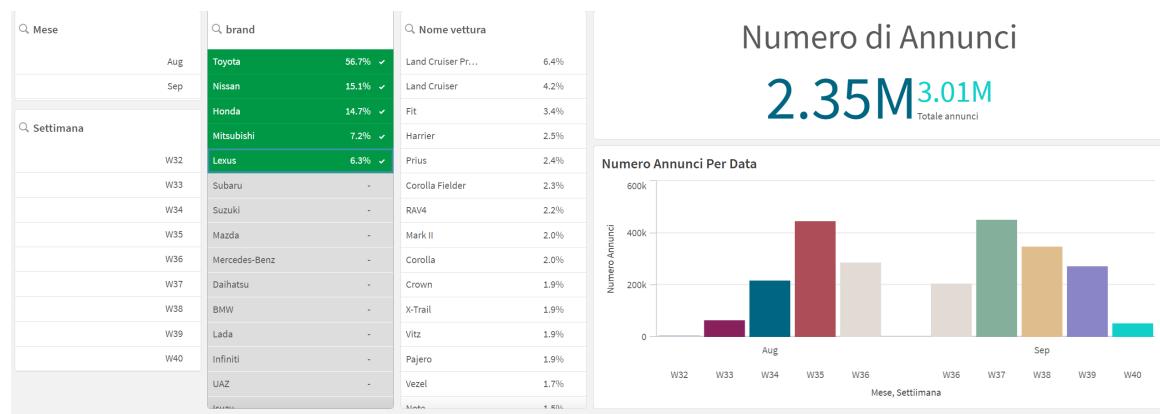


Figura 1.3: Analisi Relativa ai top 5 Brand

1.2.2 Foglio 2: Analisi sul Prezzo

In questo foglio andiamo ad analizzare l'andamento sul prezzo dei vari annunci. in particolare possiamo trovare:

- **filtro** sul Brand e Nome del veicolo;

- **KPI** relativa alla media del prezzo, sia in dollari che in Rubli, in particolare è stato utilizzato un fattore di conversione da Rublo a Dollaro di 0.0135 che era il prezzo relativo alle date presenti nei primi giorni di settembre;
- un **Grafico lineare** che rappresenta l'andamento del prezzo di un veicolo in base all'anno di immatricolazione del mezzo;
- Due **Bottoni** che ci consentono di cambiare la visualizzazione, dal grafico in Rubli a quello in Dollari.

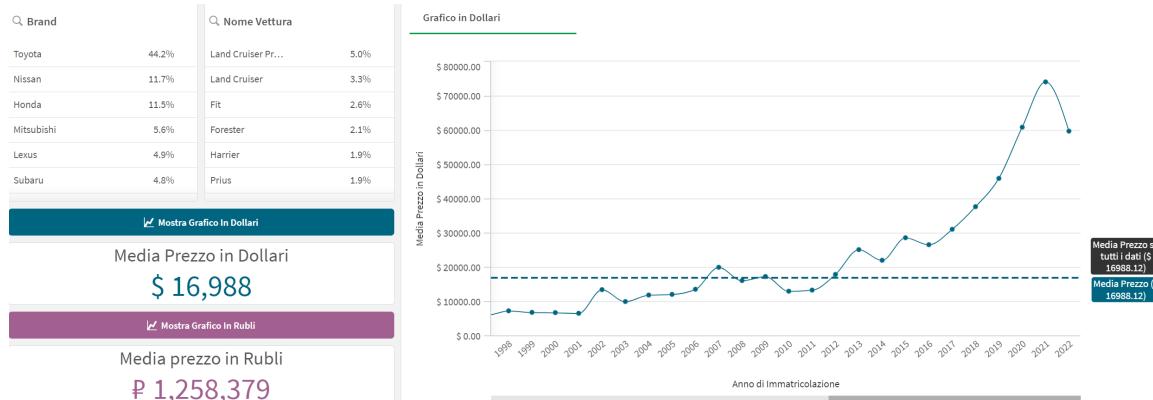


Figura 1.4: Panoramica Foglio 2

Se volessimo analizzare il prezzo degli annunci della Toyota dell'automobile Land Cruiser Prado, possiamo notare che il prezzo medio degli annunci è superiore rispetto alla media nel dataset e che la media prezzo delle auto immatricolate nel 2019 è superiore.

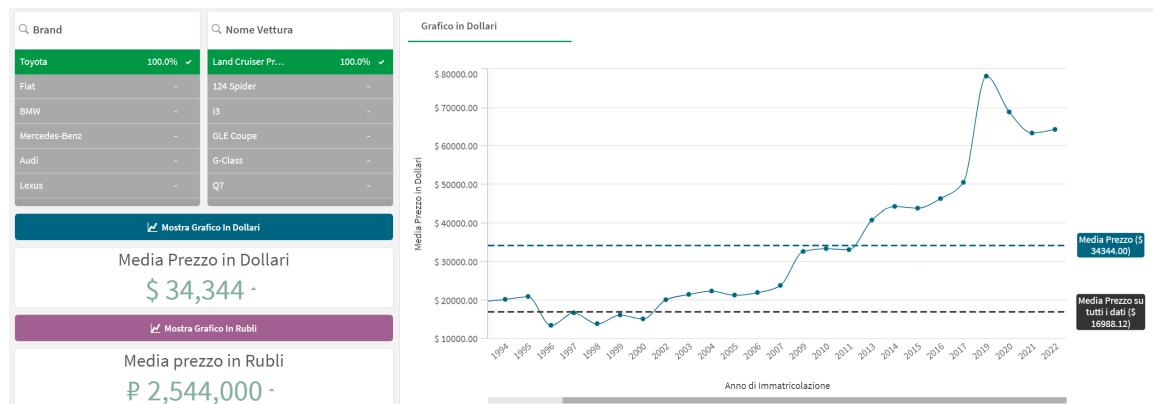


Figura 1.5: Risultato Analisi Land Cruiser Prado in Dollari

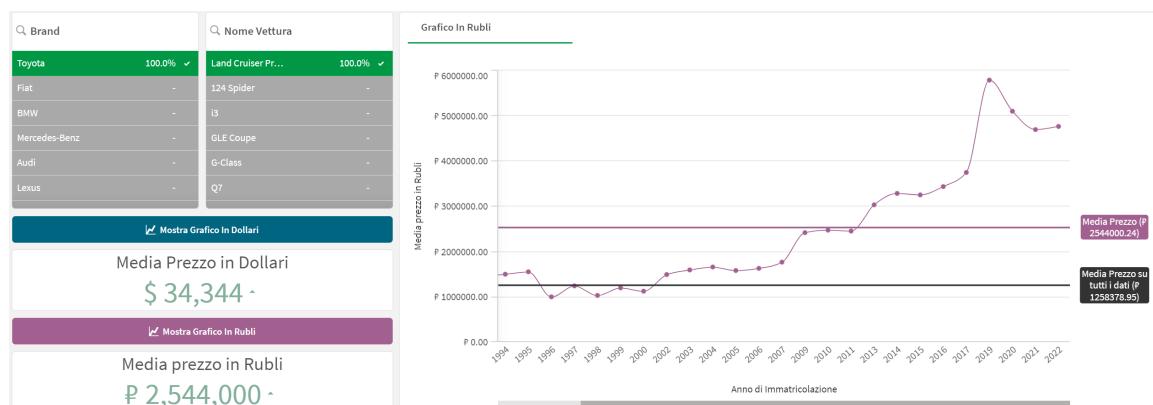


Figura 1.6: Risultato Analisi Land Cruiser Prado in Rubli

1.2.3 Foglio 3: Analisi sul Carburante

In questo foglio andiamo ad analizzare l'andamento dei prezzi in base al tipo di carburante utilizzato dalla vettura. Questo foglio è composto da:

- **Grafico a torta** per rappresentare la frequenza di un carburante rispetto a tutti gli annunci;
- **Grafico a barre** per visualizzare il prezzo medio dell'annuncio in base al tipo di carburante;
- **Tabella** per analizzare il numero di annunci in base al brand, al veicolo e al tipo di carburante.

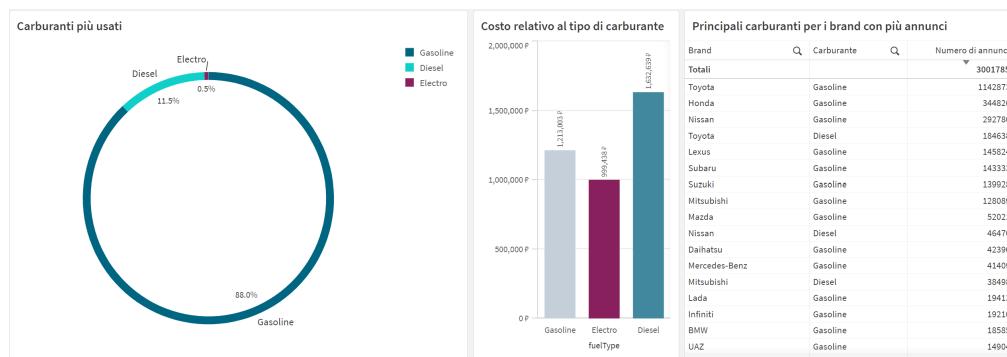


Figura 1.7: Panoramica Foglio 3

Andando ad analizzare il tipo di carburante dei top 5 brand, si può notare che la maggior parte di annunci è a benzina e che in media le automobili a Diesel hanno un prezzo superiore.

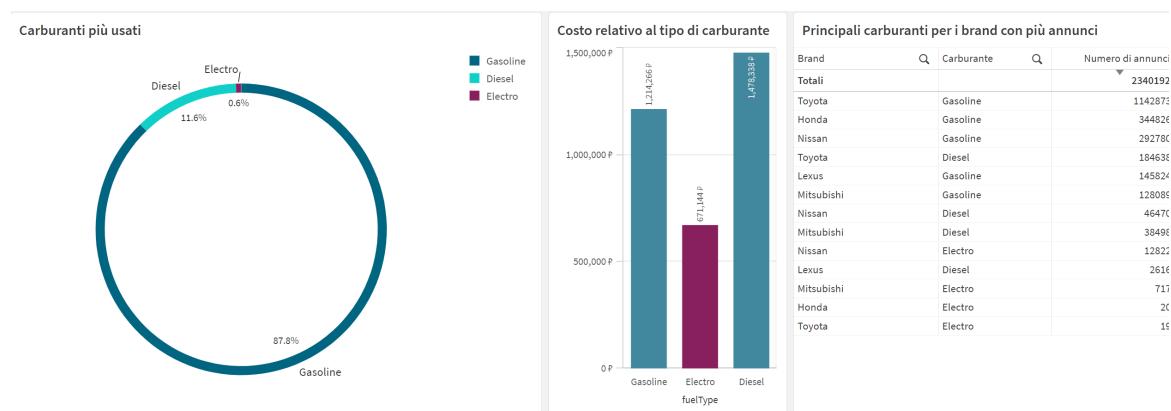


Figura 1.8: Analisi sui Top 5 Brand

Se invece volessimo andare a visualizzare gli annunci con i brand che sviluppano auto elettriche, noteremo che Nissan ha un numero elevato di annunci e vende auto elettriche a prezzi molto inferiori alla media del dataset.

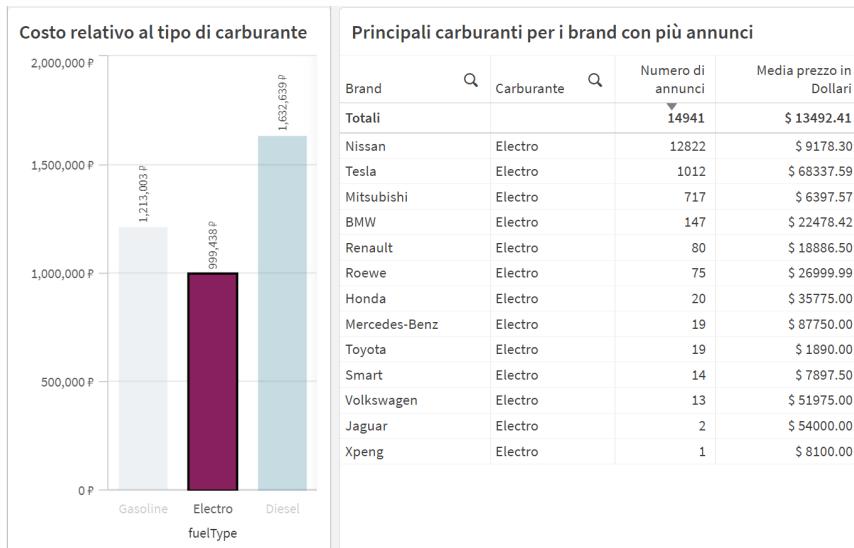


Figura 1.9: Approfondimento Auto Elettriche

1.2.4 Foglio 4: Analisi sulle Città

In questo foglio andiamo ad analizzare il numero di annunci nella varie città Russe Questo foglio è composto da:

- **Filtro** per città e Brand
 - **Mappa ad albero** per visualizzare il numero di annunci per città;



Figura 1.10: Panoramica Foglio 4

Selezionando le prime 5 città riusciamo ad analizzare il numero degli annunci dei 5 veicoli più frequenti di ogni brand.

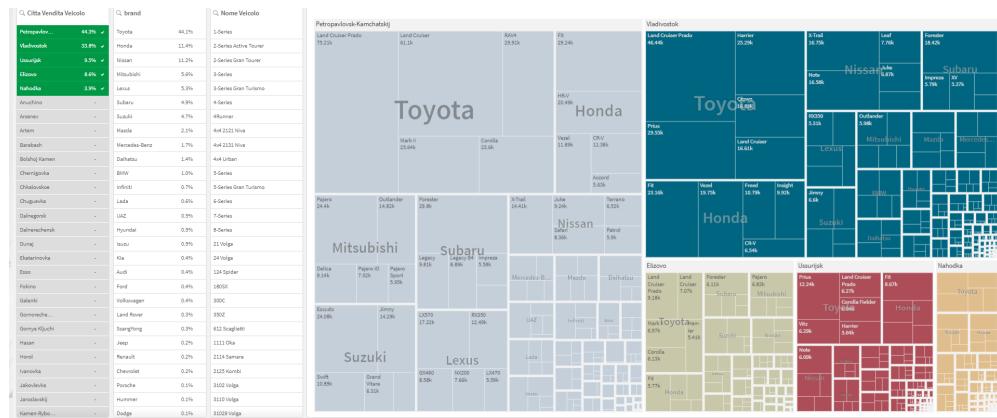


Figura 1.11: Analisi sulle top 5 città

Capitolo 2

Tableau

2.1 Dataset

Il dataset utilizzato per Tableau è il medesimo utilizzato per Qlick, esso riguarda la gestione di un sito di annunci auto su alcune località russe. Per la struttura del dataset si fa riferimento alla tabella 1.1.

2.2 Trasformazione dei dati

Considerando che il dataset si compone di due file CSV di due località differenti definiti sugli stessi attributi, si è scelto di unire i due file.

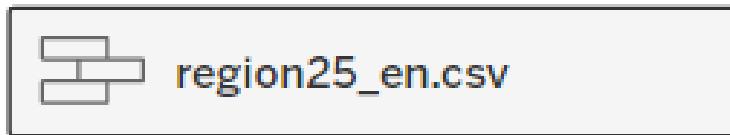


Figura 2.1: Unione file

In questo modo è stato possibile accorpare il file region41_en.csv con il file region_25en.csv in modo da poter utilizzare un unico file contenente tutti i record dei componenti. Inoltre sono state effettuate le seguenti modifiche sui dati:

- E' stato calcolato un nuovo campo da year poichè presentava una cifra in più;
- E' stato calcolato un nuovo campo per esprimere il prezzo in dollari;
- Il campo price è stato frmattato in valuta personalizzata;
- Il campo 'Date' è stato convertito nel tipo data.

2.3 Vista generale

Nella seguente dashboard si mostra una vista generale sul dataset. Sono presente quattro KPI che indicano:

- Il numero totale di annunci gestiti.

- La media annunci giornaliera.
- Il prezzo totale di tutte le auto gestite dal sito.
- Il prezzo medio di una singola auto gestito dal sito.

Inoltre viene presentato in istogramma che mostra la distribuzione degli annunci nei due mesi presenti nel dataset, confrontata con la media annunci giornaliera.

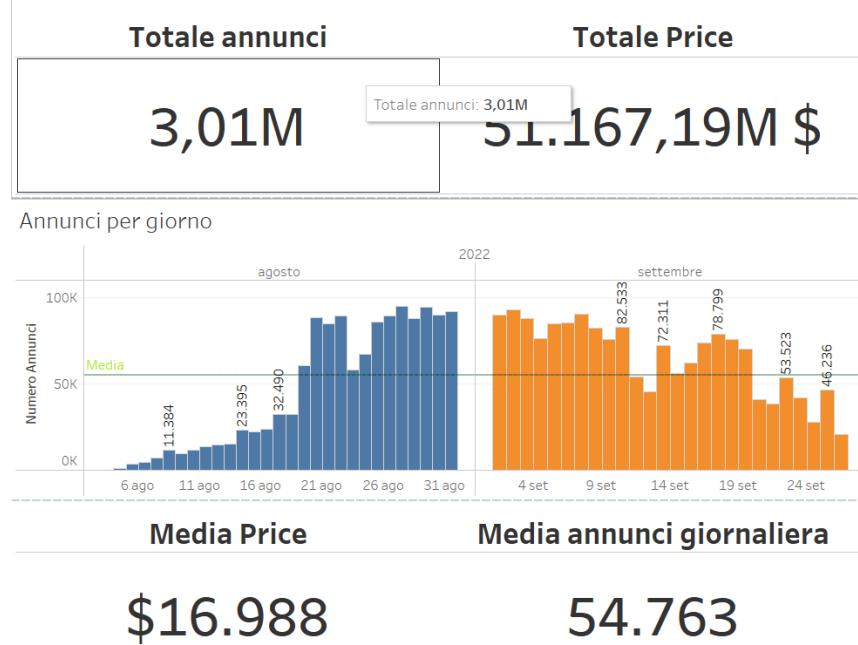


Figura 2.2: Principali KPI e numero annunci nel tempo

2.4 Analisi sul brand

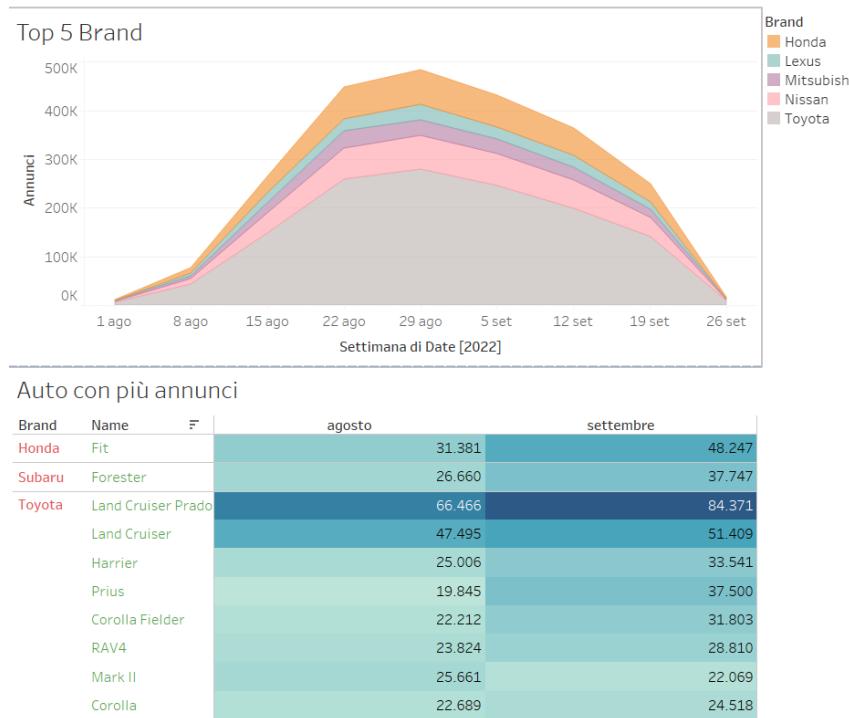


Figura 2.3: Analisi sul Brand

Tale dashboard si compone di due grafici riguardanti il brand. Il primo mostra i cinque Brand con più annunci nel sito, l'area ricoperta dalla sezione è proporzionale al numero di annunci presenti. Ciò sta ad indicare che il Brand *Toyota* è il brand con più annunci, la cui area è segnata in grigio. La visualizzazione sottostante invece mostra le dieci macchine con più annunci, segna per ognuno di esse il corrispettivo brand e visualizza nella griglia il numero di annunci di ciascuna macchina nel mese di agosto e di settembre. L'intensità del colore denota un picco facilmente leggibile. Infatti è facilmente deducibile che la *Land Crusier Pardo* è l'auto con più annunci sia ad Agosto che a Settembre.

2.5 Analisi sul tipo di alimentazione

Nelle seguenti visualizzazioni verrà mostrato un diagramma a torta per vedere in generale come le auto gestite dal sito e successivamente un grafo più analitico sull'alimentazione.

Grafico Fuel Type

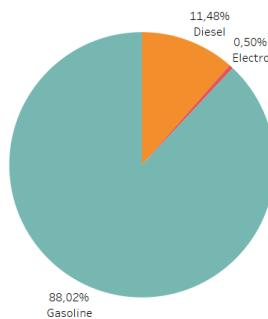


Figura 2.4: Descrizione dei tipi di alimentazione

Analisi Fuel type

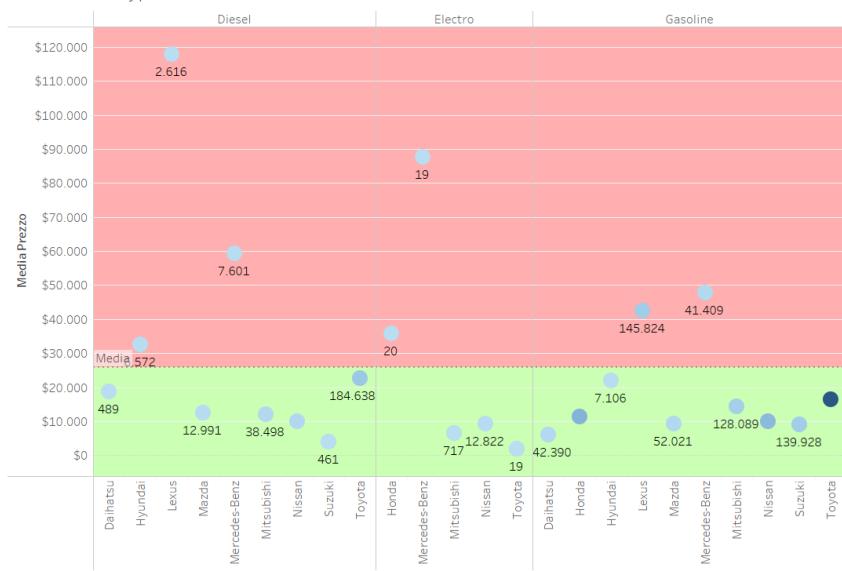


Figura 2.5: Descrizione dei tipi di alimentazione

Nel seguente grafico si prendono in esame i dieci brand, con più annunci con la corrispettiva media prezzo, divisi in base al tipo di alimentazione. Le due regioni, una in rosso e una in verde sono delimitate dalla media prezzo delle auto. Attraverso tale grafico è possibile vedere se uno specifico brand è competitivo (in base al prezzo) su ogni tipo di alimentazione. Ad esempio la Honda produce auto al di sotto della media prezzo per il tipo *Gasoline* ma superiore per le auto di tipo *Electro*.

2.6 Andamento temporale della media prezzo con trand

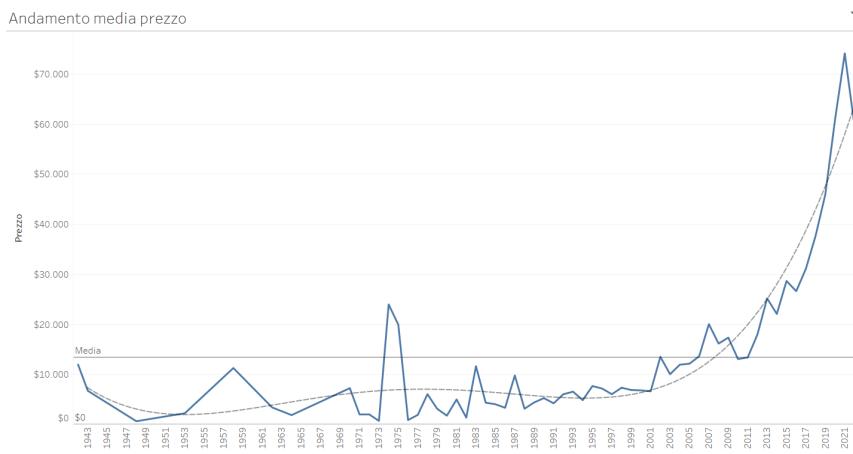


Figura 2.6: Andamento media prezzo annuale

In quest'ultimo grafico viene mostrato l'andamento della media prezzo in base agli anni delle auto presenti negli annunci. Da notare è la curva del Trand. Infatti essa sottolinea una crescita negli ultimi anni dovuta dal prezzo di auto nuove, una certa stozionarietà del prezzo negli anni centrali e un aumento del prezzo delle auto più datata, il cui prezzo probabilmente aumenta grazie al fatto che in questa fascia le auto sono considerate d'epoca. Il tipo di modello utilizzato per rappresentare la tendenza è polinomiale di quarto grado. La linea ha nel complesso un coefficiente di determinazione dello 0,89 indicando una buona descrizione del modello polinomiale e un P Value inferiore a 0,001 indicando una bassa percentuale che i risultati siano dovuti al caso.

Capitolo 3

PowerBI

Il dataset utilizzato per PowerBI è il medesimo utilizzato per Qlick e Tableau, esso riguarda la gestione di un sito di annunci auto su alcune località russe. Per la struttura del dataset si fa riferimento alla tabella 1.1.

3.1 ETL

La tabella sopra riportata è il risultato della fase di ETL effettuata utilizzando Power Query.

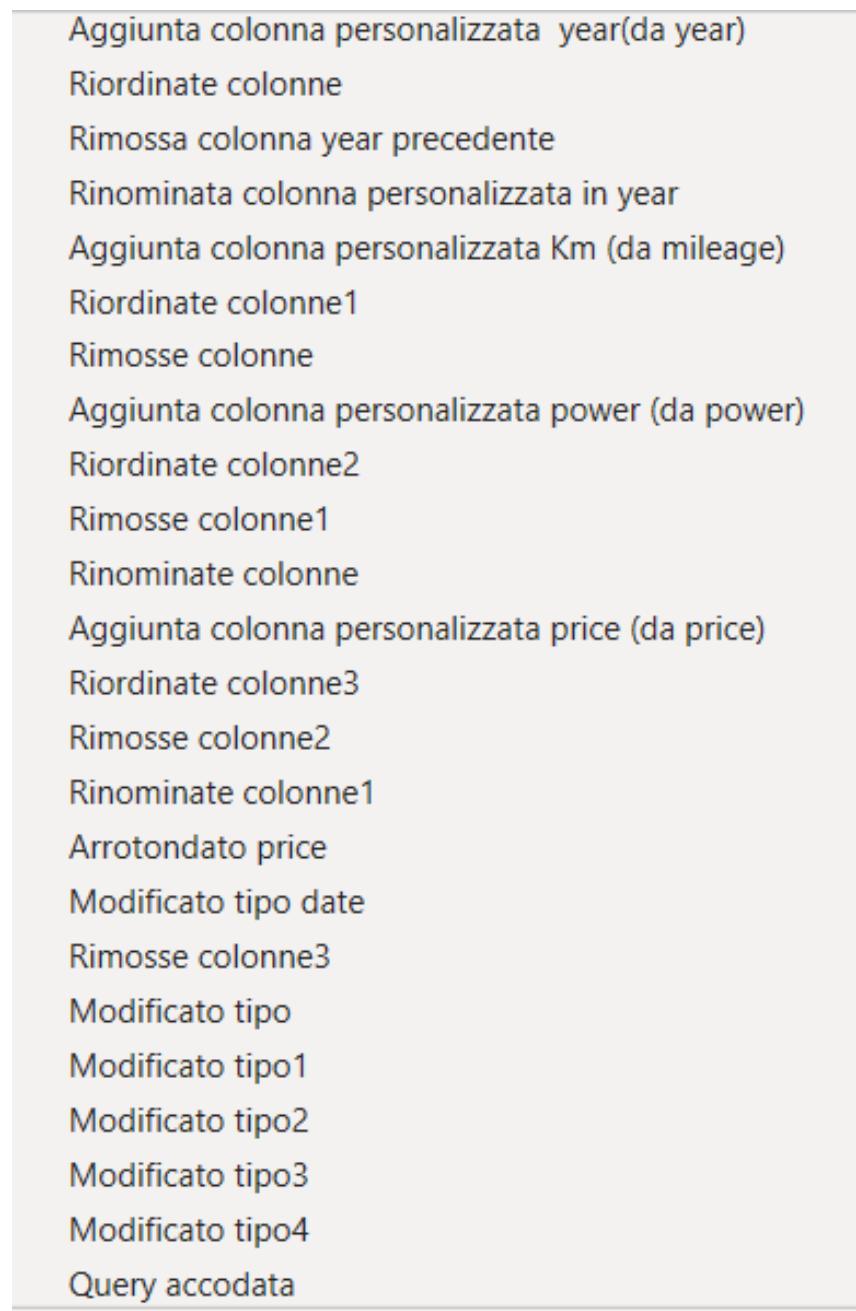


Figura 3.1: Schema passaggi ETL

Come mostrato in figura sono stati modificati diversi campi, tra cui 'year', 'mileage', 'power'. Tali campi presentavano una cifra in più, dunque abbiamo sostituito tali valori con gli stessi divisi per dieci. Inoltre, è stata modificata la valuta in dollari (poiché la precedente era in rubli). Infine sono stati rimossi alcuni campi non utili per l'analisi. Considerato che i due datasets presentavano gli stessi attributi, tali operazioni sono state ripetute per il secondo, infine le rispettive query di power query sono state unite sepicemente accondando l'una all'altra.

3.2 Descrizione dei report

In tale sezione vengono mostrati i risultati ottenuti dall'analisi su PowerBI. I report riportano essenzialmente gli stessi risultati ottenuti su QlicK e Tablau, tuttavia sono state utilizzate delle visualizzazioni un pò differenti.

3.2.1 Pagina generale

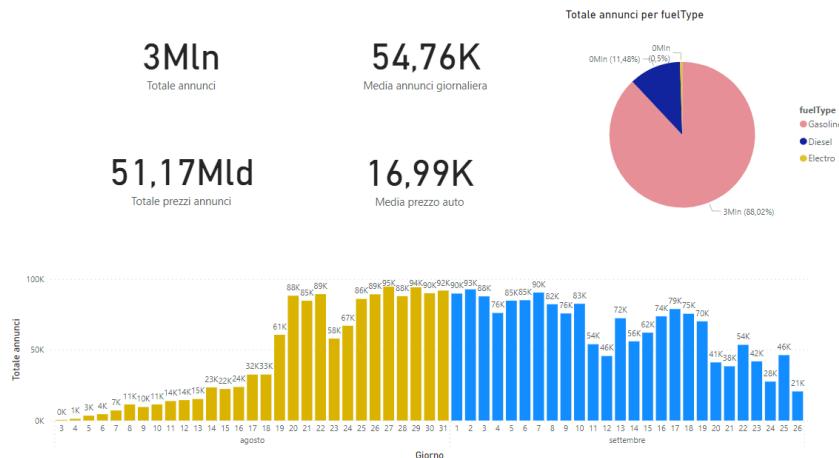


Figura 3.2: Report per la descrizione dei Datasets

In tale pagina sono stati riportati alcuni KPI per la descrizione del Dataset. In particolare, notiamo:

- Il totale degli annunci presenti sul sito.
- La media annunci giornaliera.
- Il totale del prezzo delle auto presenti.
- Il prezzo medio delle auto del Dataset.

L'istogramma in basso invece dona il numero di annunci distribuite per le date disponibili sul Dataset. Esse riguardano un arco temporale che inizia dal tre agosto e termina il 26 Settembre. Infine è stato inserito un diagramma a torta riguardo all'alimentazione più usate nelle due regioni russe. Come mostrato, le auto sono per lo più alimentate in *Gasoline* (la comune benzina), mentre le auto alimentate in elettrico risultano essere meno dell'1%.

3.2.2 Analisi sul Brand

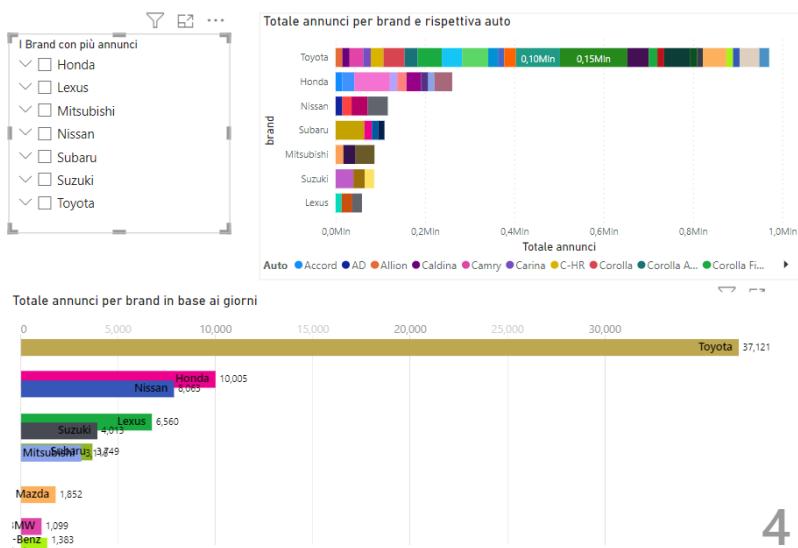


Figura 3.3: Report sul Brand PowerBI

In tale report è stata riportata un'analisi sui brand delle auto con più annunci, ovvero i brand più frequenti. In basso, nella figura, si nota un istogramma animato in cui viene visualizzato il numero di annunci dei principali brand animati in base agli annunci giornalieri. Ciò ci mostra la tendenza dei principali brand in frequenza nell'arco temporale. Nel grafico più in alto invece sono stati riportati i brand con più annunci, in cui però è possibile vedere in dettaglio il singolo brand e il contributo che danno le specifiche auto sul brand attraverso una scheda filtro a sinistra dell'istogramma. Di seguito un esempio che mostra tale dettaglio.

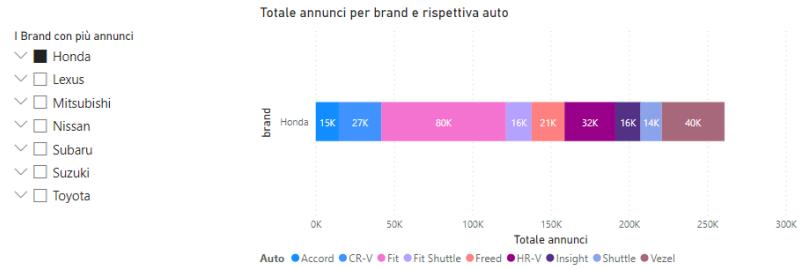


Figura 3.4: Dettaglio sul Brand Honda

Infine, tramite il menù a tendina è possibile vedere il dettaglio della singola auto.

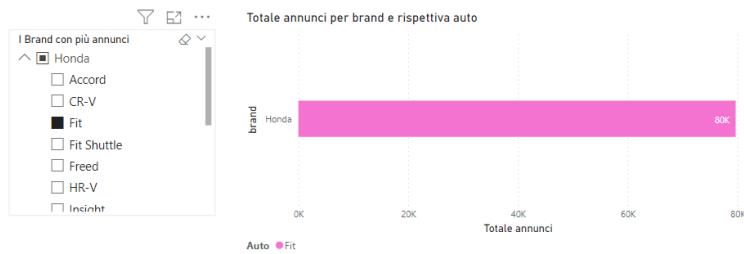


Figura 3.5: Dettaglio sull'auto Fit della Honda

3.2.3 Analisi sull'alimentazione

In questa sezione sono stati inseriti i report sul tipo di alimentazione delle auto. Come già visto nella pagina generale, il tipo di alimentazione più usato è *Gasoline*.

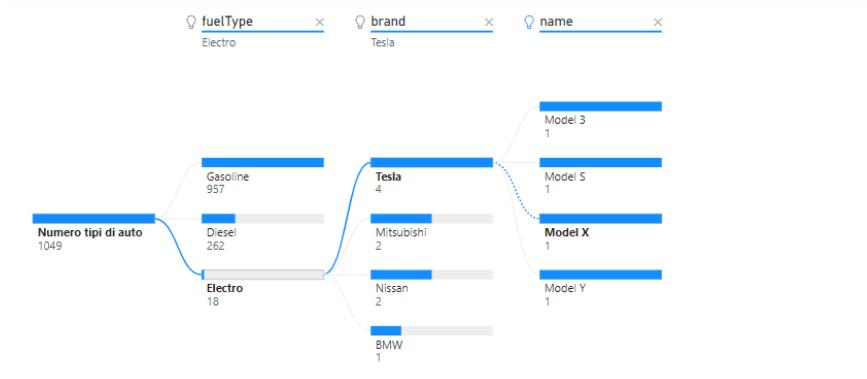


Figura 3.6: Albero di scomposizione su Fuel Type

In tale albero di scomposizione, viene riportato il numero di auto presenti nel dataset selezionate distintamente e scomposte per tipo di alimentazione. Inoltre per ogni tipo di alimentazione è possibile visionare i brand e successivamente le singole auto (con le rispettive quantità) che utilizzano l'alimentazione selezionata.

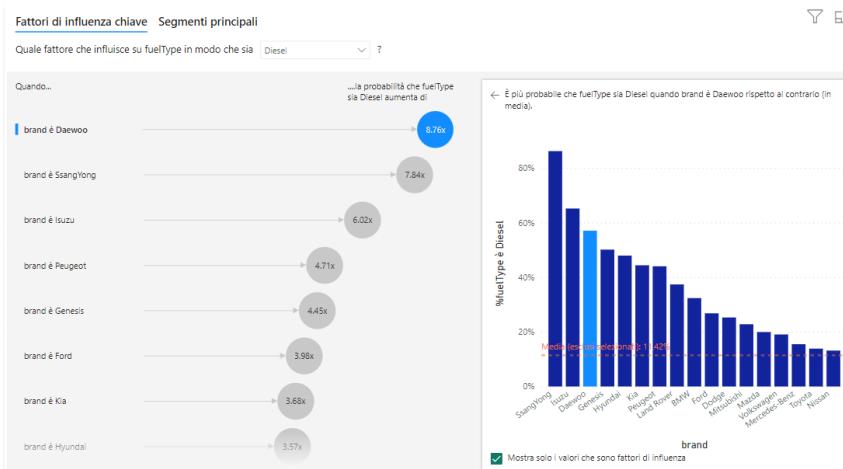


Figura 3.7: Albero di scomposizione su Fuel Type

In quest'ultimo report vengono visualizzati quali brand hanno una maggior influenza nell'uso del Diesel (è possibile effettuare l'analisi anche su altri tipi di alimentazione modificando il valore da analizzare). In questo caso si evince che seppur il brand *Daewoo* abbia minori annunci con auto a Diesel

rispetto ad altri brand, esso ha un'influenza maggiore, ovvero la probabilità che l'auto sia alimentata con il Diesel per questo brand è maggiore rispetto agli altri.

3.2.4 Relazione tra kilometraggio e media del prezzo auto

In questa pagina si è proposto di effettuare una visualizzazione che metta in relazione la media del prezzo delle auto con la media del kilometraggio di esse.

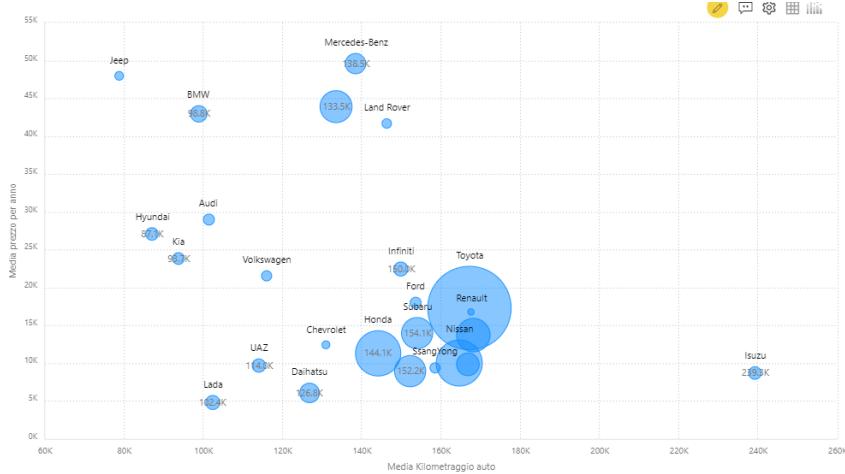


Figura 3.8: Relazione tra la media prezzo e la media del kilometraggio

Attraverso la legenda è possibile individuare i brand con il rispettivo rapporto prezzo/kilometraggio. Si individuano così più in alto della visualizzazione brand come *Mercedes*, *BMW*, *Lexus* il cui prezzo a parità di kilometraggio risulta essere maggiore rispetto ad altri brand. La dimensione dei cerchi dipende dal numero di annunci presenti nei datasets.

3.3 Annunci per regione

Nella seguente visualizzazione individuiamo le regioni interessate dal sito di annunci delle auto. In dettaglio per ogni regione individuata visualizziamo anche i brand con più annunci in tali regioni.

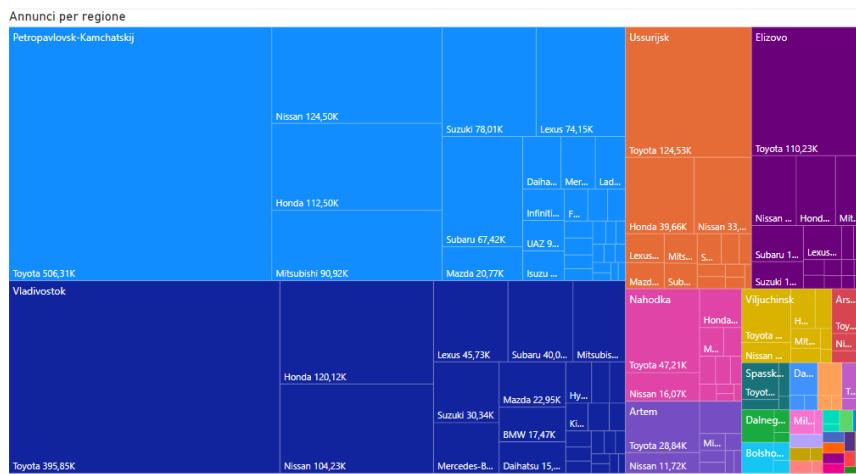


Figura 3.9: Relazione tra la media prezzo e la media del kilometraggio

3.4 Media prezzo auto in base all'anno di immatricolazione

Nella seguente visualizzazione si è riportato il monitoraggio della media del prezzo di un brand in base all'anno di immatricolazione delle auto.

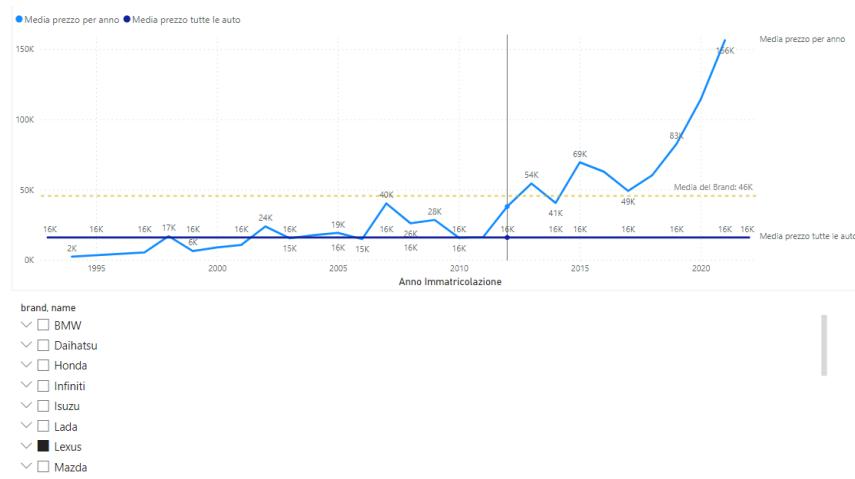


Figura 3.10: Media del prezzo in base all'anno di immatricolazione

Nella figura si nota in azzurro l'andamento della media del prezzo in al trascorrere degli anni, in blu invece la media del prezzo tra tutte le auto del dataset e infine la linea tratteggiata indica la media del brand selezionato dallo slicer.

Capitolo 4

Python

In questo capitolo verranno descritte le analisi svolte con Python [1]. In particolare andremo ad analizzare tutti gli strumenti necessari per le prossime analisi di:

- classificazione
- clustering
- time-series
- networkX

4.1 Strumenti utilizzati

Iniziamo dando una breve descrizione degli strumenti utilizzati:

4.1.1 Python3

Python 3 (fig. 4.1) è un linguaggio che offre grandi funzionalità per gestire matematica, statistica e funzioni scientifiche. Non solo, Python fornisce ottime librerie per gestire l'applicazione della Data Analysis e per questo risulta ampiamente utilizzato dai Data Scientists. Tutte le librerie Python sono accessibili gratuitamente e godono di un ottimo supporto da parte della community. Nel progetto che verrà descritto in seguito, l'analisi è stata effettuata utilizzando il linguaggio Python ed il suo ecosistema, in particolare le librerie:

- **NumPy** è una libreria fondamentale per il calcolo scientifico in Python. Fornisce supporto per l'elaborazione efficiente di array multidimensionali, insieme a un'ampia gamma di funzioni matematiche per operazioni numeriche avanzate. Sito Web: numpy.org
- **Pandas** è una libreria che fornisce strutture dati e strumenti per la manipolazione e l'analisi dei dati. Introduce due strutture dati principali: la Serie, che rappresenta un array unidimensionale con etichette, e il DataFrame, che rappresenta una struttura dati tabellare con righe e colonne. Sito Web: pandas.pydata.org
- **Scikit-learn** è una libreria di machine learning che fornisce una vasta gamma di algoritmi e strumenti per l'apprendimento automatico. Include algoritmi per la classificazione, la regressione e il clustering. Sito Web: scikit-learn.org
- **Seaborn e Matplotlib** Entrambe le librerie sono molto utili per rappresentare graficamente i dati in modo chiaro ed efficace. Sito Web: seaborn.pydata.org matplotlib.org
- **Statsmodels** è una libreria che offre funzionalità per l'analisi statistica, fornisce una vasta gamma di modelli statistici, come modelli lineari, modelli di regressione, modelli di serie temporali. Sito Web: statsmodels.org

- geopy, è una libreria che consente di determinare latitudine e longitudine di una location utilizzando un indirizzo. Sito Web: geopy.readthedocs.io.
- folium, è una libreria utilizzata per la visualizzazione interattiva delle mappe. Sito Web: gihub.folium.



Figura 4.1: Logo di Python

4.1.2 JupyterNotebook

Jupyter notebook In particolare però non è stato utilizzato Python nella maniera canonica, bensì sono stati sviluppati dei notebook [2] più o meno interattivi. Notebook Jupyter 4.2 è un'applicazione basata sul modello client-server dell'organizzazione no profit Progetto Jupyter fondata nel 2015. Permette la creazione e la condivisione di documenti web nel formato JSON, che seguono uno schema e una lista ordinata di celle input/output. Queste celle offrono tra l'altro spazio per codici, testi in markdown, formule matematiche ed equazioni o contenuti multimediali. L'elaborazione funziona su un'applicazione client basata sul web che si avvia con un browser standard. Basta che sul sistema sia installato e venga eseguito anche il server del Notebook Jupyter. I documenti Jupyter creati si possono esportare come documenti HTML, PDF, Markdown o Python o in alternativa si possono condividere con altri utenti tramite e-mail, Dropbox, GitHub o il proprio Notebook Jupyter. I due componenti centrali di Notebook Jupyter sono un set di diversi kernel (interpreti) e la dashboard. I kernel sono piccoli programmi che elaborano richieste ("request") specifiche nel linguaggio e reagiscono con relative risposte. Un kernel standard è IPython, un interprete della riga di comando che permette di lavorare con Python. Oltre 50 kernel forniscono supporto per altri linguaggi come C++, R, Julia, Ruby, JavaScript, CoffeeScript, PHP o Java. La dashboard serve da una parte come interfaccia di gestione per i singoli kernel e dall'altra come centrale per la creazione di nuovi documenti Notebook o per aprire progetti già esistenti. Notebook Jupyter è disponibile gratuitamente per tutti gli utenti grazie a una licenza BSD modificata.



Figura 4.2: Logo di Jupyter

Andando ancora più in dettaglio in realtà per lo sviluppo dei notebook è stato utilizzato Google Colab 4.3. Google Colab è uno strumento gratuito presente nella suite Google che consente di scrivere codice Python direttamente dal proprio browser. Una piattaforma online che offre un servizio di cloud hosting per notebook Jupyter dove creare documenti che contengono righe di codice, grafici, testi, link e molto altro. Si può creare un documento Google Colab direttamente da Google Drive e, proprio come un qualunque documento in G Suite, può essere condiviso con altri utenti che hanno la possibilità di modificarlo e lasciare commenti direttamente nel notebook. Il notebook Jupiter verrà poi eseguito su macchine virtuali di server Google. Ciò consente di svincolarsi dalla parte hardware e di concentrarci solamente sul codice Python e sui contenuti che si vuole integrare nel notebook.



Figura 4.3: Logo di Google Colab

Capitolo 5

Classificazione Battaglie Pokemon

In questo capitolo troverete uno studio sulla classificazione, in particolare questo classificatore dovrà predire chi fra 2 Pokemon vincerà una lotta. La libreria di riferimento per questo studio è Scikit-learn



Figura 5.1: Pokemon

5.1 Dataset utilizzato

Il dataset utilizzato per questa analisi è composto originariamente da un file CSV che contiene per ogni riga il risultato e le statistiche di una battaglia tra due Pokemon. Andiamo a vedere la semantica dei vari campi:

Index	Indice univoco della battaglia
Name	Nome del Pokemon
Type 1	Tipo principale di Pokemon
Type 2	Tipo secondario di Pokemon
HP	Punti Vita
Attack	Attacco Fisico, ovvero quanto danno causa con attacchi da mosse semplici
Defense	Difesa Fisica, ovvero quanta difesa ha contro attacchi da mosse semplici
Sp. Atk	Attacco Speciale, quanto danno causa con attacchi da mosse speciali
Sp. Def	Difesa Speciale, quanta difesa ha contro attacchi da mosse speciali
Speed	Velocità, chi ha il valore più alto attacca per primo
Generation	Generazione Pokemon, indica a quale generazione il Pokemon appartiene, attualmente ne esistono nove
Legendary	Indica se il Pokemon è considerato un pokemon leggendario, ovvero se ne esiste solo 1 all'interno di tutto il gioco (console)
Name_other	Nome del Pokemon 2
Type 1_other	Tipo principale del Pokemon 2
Type 2_other	Tipo secondario del Pokemon 2
HP_other	Punti vita Pokemon 2
Attack_other	Attacco Fisico Pokemon 2
Defense_other	Difesa Fisica Pokemon 2
Sp. Atk_other	Attacco Speciale Pokemon 2
Sp. Def_other	Difesa Speciale Pokemon 2
Speed_other	Velocità Pokemon 2
Generation_other	Generazione del Pokemon 2
Legendary_other	Leggendarità del Pokemon 2
Wins	Colonna Target, per vedere se il Pokemon 1 ha vinto la battaglia

Tabella 5.1: Dataset Battaglie Pokemon

5.2 Analisi Descrittiva

Inizialmente andiamo a visualizzare il dataset, per avere una visione più chiara di come sono i dati

	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	...	Legendary_other	Wins
39087	Karrablast	Bug	NaN	50.0	72.0	39.0	39.0	42.0	55.0		False	True
30893	NaN	Rock	Water	70.0	NaN	125.0	113.0	78.0	NaN		False	True
45278	Mega Manectric	Electric	NaN	70.0	77.0	81.0	NaN	91.0	136.0		False	True
16398	Bouffalant	Psychic	NaN	95.0	121.0	NaN	39.0	85.0	NaN		False	False
13653	Swablu	Normal	Flying	45.0	36.0	58.0	37.0	76.0	56.0		False	False

rows × 23 columns

Figura 5.2: Sample del dataset

Dopo di che andiamo a visualizzare tramite un istogramma la frequenza del tipo principale dei Pokemon.

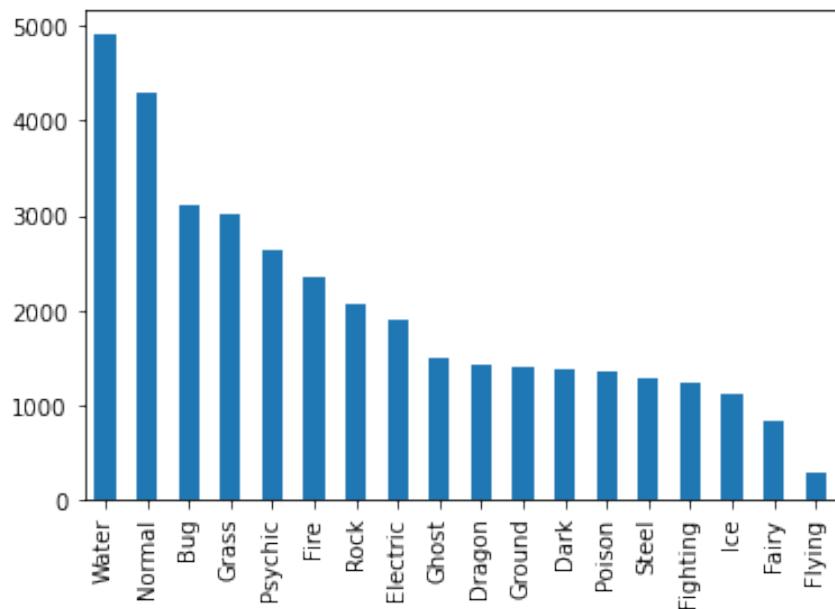


Figura 5.3: Istogramma Tipo 1 Pokemon

Infine possiamo visualizzare il numero di colonne target (colonna Wins) che hanno un valore Vero e Falso, calcoliamo la percentuale con le seguenti formule:

$$\text{PercentualeVittorie} = \frac{\text{Vittorie}}{(\text{Vittorie} + \text{Sconfitte})} * 100$$

$$\text{PercentualeSconfitte} = \frac{\text{Sconfitte}}{(\text{sconfitte} + \text{Vittorie})} * 100$$

Di seguito i risultati:

```

Number of true target columns:
18848
Number of false target columns:
21152
percentage of true values: 47.12 %
percentage of false values: 52.88 %

```

Figura 5.4: Bilanciamento del target

Possiamo notare che il dataset risulta abbastanza bilanciato.

5.3 Preprocessing

Per maggior comprensione del dataset andiamo a visualizzare i campi vuoti interni al dataset, questi infatti non possono esser dati in pasto al classificatore, Andiamo a definire qui le features:

- **Non necessarie**, ci sono alcune feature che non vengono ritenute necessarie alla classificazione, per un maggior dettaglio, si possono consultare nella tab. 5.2;
- **Continue**, ovvero formate da valori numerici, qui per i valori numerici nulli, andiamo ad assegnargli un valore medio per ogni feature. Successivamente andiamo a normalizzare questi valori, in modo da avere valori compresi tra 0 ed 1, sarà quindi utilizzato uno Scaler, per la precisione un **MinMaxScaler** che ha il compito di prendere il valore massimo e il minimo all' interno del dataset;
- **Categorie**, Qui andiamo ad utilizzare il **OneHotEncode**, ovvero andiamo ad aggiungere colonne fittizie al dataset, queste colonne avranno il nome di ogni possibile valore che assume quella colonna e come valore 1 o 0, dove 1 significa che il Pokemon appartiene a quella categoria e 0 che non ne fa parte.

Colonna	Motivo dell' esclusione
Name	Il nome del Pokemon non fornisce alcun'informazione all'analisi, aggiunge solo rumore
Generation	Questa feature non determina un Pokemon più forte
Legendary	La leggendarietà di un Pokemon non determina la probabilità di un Pokemon di vincere una battaglia
Name__other	Il nome del secondo Pokemon non fornisce alcun'informazione all'analisi, aggiunge solo rumore
Generation__other	Questa feature non determina un Pokemon più forte
Legendary__other	La leggendarietà di un Pokemon non determina la probabilità di un Pokemon di vincere una battaglia

Tabella 5.2: Feature non Necessarie

Al termine del Preprocessing il dataframe risultante sarà il seguente:

	HP	Attack	Defense	Sp. Atk	Sp. Def	...	Type	Type	Type
	2_other_Ghost	2_other_Grass	2_other_Ground						
39087	0.192913	0.372549	0.186992	0.184080	0.147826	...	0	0	0
30893	0.271654	0.408108	0.536585	0.552239	0.304348	...	0	0	0
45278	0.271654	0.397059	0.357724	0.355349	0.360870	...	0	0	0
16398	0.370079	0.612745	0.331967	0.184080	0.334783	...	0	0	0
13653	0.173228	0.196078	0.264228	0.174129	0.295652	...	0	0	0

5 rows × 84 columns

Figura 5.5: Risultato Post PreProcessing

5.4 Training

In questa sezione andiamo ad addestrare vari modelli, per determinare quello che lavora meglio su questo dataframe. In particolare prendiamo il dataframe del preprocessing senza la colonna di target e lo dividiamo in 80% training e 20% test. In questo progetto i modelli di classificazione utilizzati sono i seguenti:

- Logistic Regression
- Decision Tree
- Random Forest

Il risultato del classificatore ci restituisce i seguenti livelli di accuracy:

```
Accuracy: 0.82    ---> LogisticRegression
Accuracy: 0.79    ---> DecisionTreeClassifier
Accuracy: 0.83    ---> RandomForestClassifier
```

Figura 5.6: Accuracy del modello

Il seguente plot rappresenta l'accuratezza media e la deviazione standard dei vari modelli, in particolare si può notare che la deviazione standard ha un'oscillazione contenuta in tutti i modelli

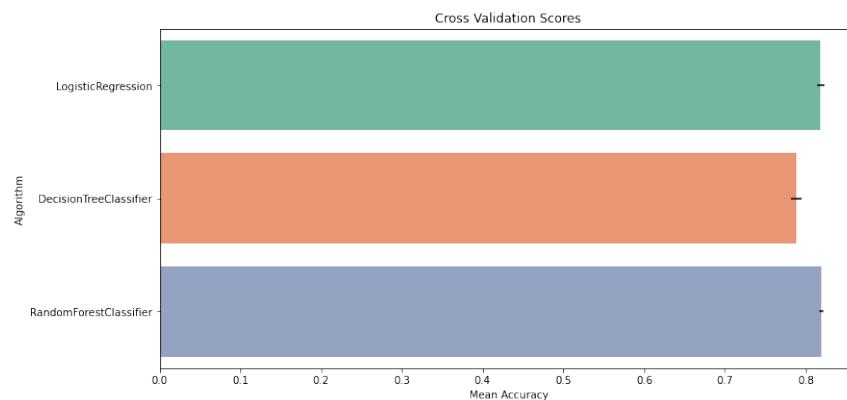


Figura 5.7: Risultati Cross Validation accuratezza e dev.standard

Di seguito troviamo le 3 matrici di confusione, Si noti che il Random forest è il modello migliore

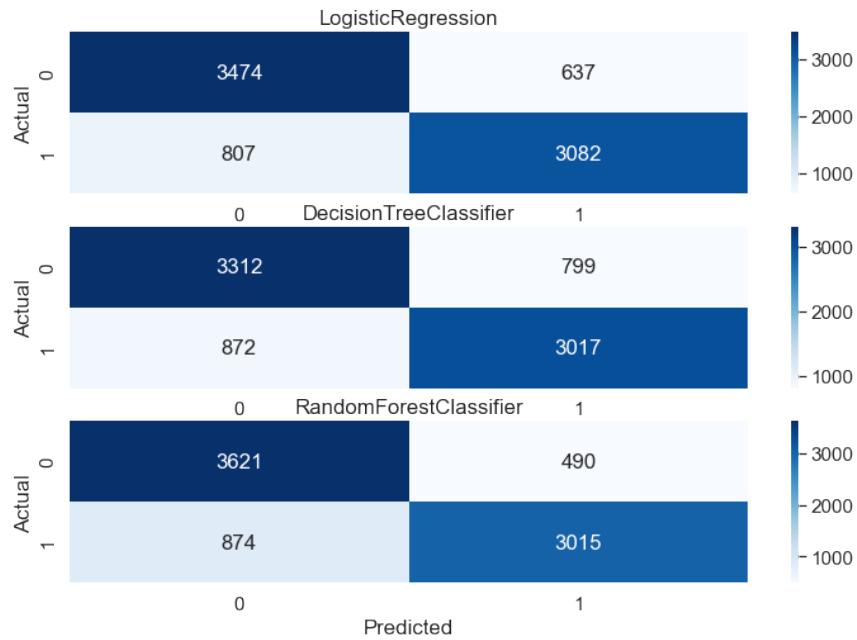


Figura 5.8: Confusion Matrix

Infine stampiamo il report per ogni modello, le metriche più importanti sono:

- $Accuracy = \frac{(veropositivo+veronegativo)}{(veropositivo+veronegativo+falsopositivo+falsonegativo)}$

- $Precision = \frac{veropositivo}{(veropositivo+falsopositivo)}$

- $Recall = \frac{veropositivo}{(veropositivo+falsonegativo)}$

- $F1 - score = \frac{2*(Precision*Recall)}{(Precision+Recall)}$

LogisticRegression Classification Report:				
	precision	recall	f1-score	support
False	0.81	0.85	0.83	4111
True	0.83	0.79	0.81	3889
accuracy			0.82	8000
macro avg	0.82	0.82	0.82	8000
weighted avg	0.82	0.82	0.82	8000
DecisionTreeClassifier Classification Report:				
	precision	recall	f1-score	support
False	0.79	0.81	0.80	4111
True	0.79	0.78	0.78	3889
accuracy			0.79	8000
macro avg	0.79	0.79	0.79	8000
weighted avg	0.79	0.79	0.79	8000
RandomForestClassifier Classification Report:				
	precision	recall	f1-score	support
False	0.81	0.88	0.84	4111
True	0.86	0.78	0.82	3889
...				
accuracy			0.83	8000
macro avg	0.83	0.83	0.83	8000
weighted avg	0.83	0.83	0.83	8000

Figura 5.9: Report Modelli

5.5 GridSearch CrossValidation

In questa sezione andiamo a determinare il modello migliore con la GridSearch. La Grid Search (ricerca a griglia) è una tecnica utilizzata nell'apprendimento automatico per determinare i migliori iperparametri di un modello. Gli iperparametri sono parametri del modello che non sono appresi durante il processo di addestramento, ma che devono essere impostati manualmente prima dell'addestramento. Di seguito sono riportati i vari parametri utilizzati per ogni modello:

```

LR_param={

    "penalty": ["l1", "l2", None],
    "max_iter": [30,50,100],
    "n_jobs": [1,3,7],
    "C": [0.3,0.7,1.0],
    "fit_intercept": [True]

}

DT_param = {

    "max_depth": [2,3,8,10],
    "max_features": [0.3,0.7,1],
    "min_samples_split": [2,3,10],
    "min_samples_leaf": [1,3,10],
    "criterion": ["gini"]

}

RF_param = {

    "max_depth": [None],
    "max_features": [0.3,0.7,1],
    "min_samples_split": [2,3,10],
    "min_samples_leaf": [1,3,10],
    "bootstrap": [False],
    "n_estimators": [100,300],
    "criterion": ["gini"]

}

```

Figura 5.10: Parametri GridSearch

Dopo aver addestrato il modello otteniamo gli score e li confrontiamo con quelli ottenuti senza GridSearch. Si noti che per il decision Tree e il Random Forest abbiamo avuto dei miglioramenti

```

score without GridSearchCV:  0.818 0.788 0.82
score with GridSearchCV:   0.818 0.832 0.854

```

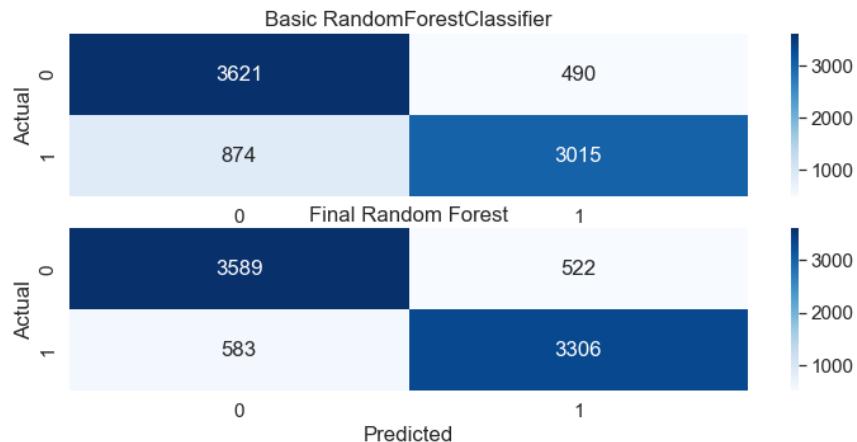
Figura 5.11: Confronto Con e Senza GridSearch

In conclusione otteniamo il classificatore migliore con i relativi iperparametri

```
The Best classifier is: RandomForestClassifier(bootstrap=False, max_features=0.3, min_samples_split=3, n_estimators=300)
And the score is: 0.854
```

Figura 5.12: Miglior Classificatore e migliori parametri

Di seguito facciamo un confronto ulteriore sulle matrici di confusione fra il random Forest Base con il Random Forest migliore e otteniamo i risultati:



5.6 Pipeline Finale

In fine abbiamo creato la pipeline in base al miglior modello con gli *hyperparametri* migliori, il seguente codice permette la corretta creazione della pipeline dove tutto lo studio precedente può esser compreso in queste poche righe:

```
from sklearn.pipeline import Pipeline, make_pipeline
from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.dummy import DummyClassifier
pipeline = DummyClassifier(strategy="uniform")

num = ["HP", "Attack", "Defense", "Sp. Atk", "Sp. Def", "Speed", "HP__other", "Attack__other", "Defense__other"]
cat = ['Type 1', 'Type 2', 'Type 1__other', 'Type 2__other']

pipeline = Pipeline(
    [
        (
            "preprocessing", ColumnTransformer(
                [
                    (
                        "cat",
                        make_pipeline(
                            SimpleImputer(strategy="most_frequent"),
                            OneHotEncoder(sparse=False, drop="first"),
                        ),
                        cat
                    ),
                    (
                        "num",
                        SimpleImputer(strategy="mean"),
                        num,
                    )
                ],
                remainder="passthrough"
            )
        ),
        ("classifier", RandomForestClassifier(bootstrap=False, max_features=0.3, min_samples_split=3, n_estimators=300))
    ]
)
```

```
        ],
        remainder="drop",
    )
),
("predictor", RandomForestClassifier(bootstrap=False, max_features=0.3, min_samples_split=3, n_
]
)

# Train the pipeline
pipeline.fit(X, y)
```

Capitolo 6

Clustering Analisi Cittadina Isola D'elba

In questo capitolo troverete uno studio sulla Clusterizzazione, La libreria di riferimento per questo studio è Scikit-learn.

6.1 Dataset utilizzato

Il dataset utilizzato per questa analisi è composto da dati reali relativi ad una ditta di gestione dei rifiuti di un paese dell'isola d'elba con attività turistica alta nel periodo estivo. Questo dataset sarà utilizzato anche per lo studio sulle TimeSeries (Capitolo 7) e sullo studio di NetworkX (Capitolo 8). Le tabelle a cui abbiamo avuto accesso sono le seguenti:

- deposits.csv (tab.6.1), ad ogni riga corrisponde un rifiuto depositato da un consumatore di un tipo di rifiuto ad una determinata data
- ecoisole.csv (tab.6.2), contiene tutte le informazioni di latitudine e longitudine di ogni ecoisola presente nella cittadina
- consumers.csv (tab.6.3), per questione di privacy i cognomi delle persone reali sono stati modificati con cognomi di: Rossi,Bianchi,Neri e Grigi

Nome Colonna	Descrizione
id	Indice univoco del deposito
consumer_id	indice relativo a chi deposita il rifiuto
waste_category_id	indice relativo al tipo di rifiuto depositato
ecoisola_id	indice relativo a dove viene depositato il rifiuto
created_at	datetime relativo all'ora di deposito

Tabella 6.1: Tabella deposits.csv colonne principali

Nome Colonna	Descrizione
id	Indice univoco dell'ecoisola
capacity	capacità massima di rifiuti dell'ecoisola
lat	latitudine
long	longitudine
address	indirizzo dell'ecoisola

Tabella 6.2: Tabella ecoisole.csv colonne principali

Nome Colonna	Descrizione
id	Indice univoco del consumer
name	Nome
surname	Cognome fittizio
address	indirizzo di residenza

Tabella 6.3: Tabella consumers.csv colonne principali

6.2 Analisi Descrittiva

Inizialmente visualizziamo il dataset relativo ai depositi, per avere una visione più chiara di come sono i dati e determinare che tipi di studi possono esser fatti, le colonne interessanti su cui basare uno studio sono le seguenti:

index	ecoisola_id	Categoria rifiuti	Data
0	9	CARTA	2021-12-30T10:55:33.0...
1	7	SECCO RESIDUO	2022-01-04T17:44:00.0...
2	14	ORGANICO	2021-12-30T11:00:13.0...
3	1	PLASTICA LATTINE	2021-12-30T11:00:48.0...
4	24	CARTA	2021-12-30T11:01:57.0...
5	15	ORGANICO	2021-12-30T11:04:31.0...
6	14	VETRO	2021-12-30T11:07:53.0...

Figura 6.1: dataset deposits

6.2.1 Obiettivo del clustering

L'**Obiettivo** dello studio che è stato individuato è di trovare un modo per clusterizzare le ecoisole in base al tipo di rifiuto che viene prevalentemente gettato e al sovraccarico delle ecoisole. In questo modo il cliente di questo studio potrà chiedere al comune in questione dei fondi extra per la costruzione di nuove ecoisole nelle zone a rischio. L'anno preso in esame per questo studio è il 2022. Inoltre siccome il dataset risulta variare molto nel periodo estivo rispetto alle altre stagioni andremo a fare 3 studi paralleli:

1. relativo ai mesi non estivi
2. relativo ai mesi estivi
3. relativo a tutto il dataset (sia mesi estivi che non estivi)

6.3 Preprocessing

Dobbiamo crearc ci 3 dataframe (uno per ogni studio) con le feature più significative per ogni ecoisola, quindi andiamo a contare dalla tabella conferimenti quante volte un tipo di rifiuto è stato gettato in un determinata ecoisola. Ogni tipo di rifiuto in realtà ha un valore diverso di riempimento quindi ad ogni deposito abbiamo aggiunto una stima di riempimento del cassonetto, questa stima ci è stata fornita da chi ci ha fornito il dataset ed è la seguente:

- Carta 2,5%

- Organico 2%
- Plastica Lattine 3%
- Secco Residuo 2,5%
- Vetro 2,7%

tutti i cassonetti di ogni ecoisola possono contenere un massimo di 1100kg dopodichè risultano saturi ed è impossibile conferire.

Otterremo il seguente risultato:

ecoisola_id	CARTA	ORGANICO	PLASTICA LATTINE	SECCO RESIDUO	VETRO
1	6237.5	8340	10374	7570	3248.1
2	11445	10844	20664	17937.5	11626.2
3	8147.5	12244	15645	10892.5	5127.3
4	7225	7510	13401	9322.5	4228.2
5	5887.5	9402	10071	7180	3126.6
6	7930	13796	17184	13722.5	6312.6
7	6695	9158	12000	9455	3272.4
8	9300	11020	18294	14005	5931.9
9	13005	21174	23367	25207.5	6615
10	17242.5	24168	30795	28122.5	10818.9
11	11457.5	17038	20268	16405	6280.2
12	7470	11496	13887	8745	4417.2

Figura 6.2: conteggio rifiuti per ecoisola di tutti i mesi del 2022

nell'immagine superiore possiamo notare che l'ecoisola 1 ha un valore relativo al cassonetto della Carta di 6237.5, questo significa che in tutto l'anno quell'ecoisola dovrà esser svuotata almeno 62 volte.

6.4 Clustering su PCA con K-means sul dataset completo

La PCA è stata utilizzata per semplificare l'analisi dei dati riducendo il numero di features da considerare, senza perdere troppa informazione per poi plottare i risultati.

Il k-means invece è un algoritmo di clustering non supervisionato utilizzato per raggruppare un insieme di dati in k cluster distinti. L'algoritmo k-means assegna i punti dati a uno dei k cluster in base alla loro distanza rispetto ai centroidi, che rappresentano il centro di ciascun cluster.

Prima di utilizzare la PCA è stato analizzato accuratamente il dataframe per vedere se esistono delle correlazioni tra le varie features

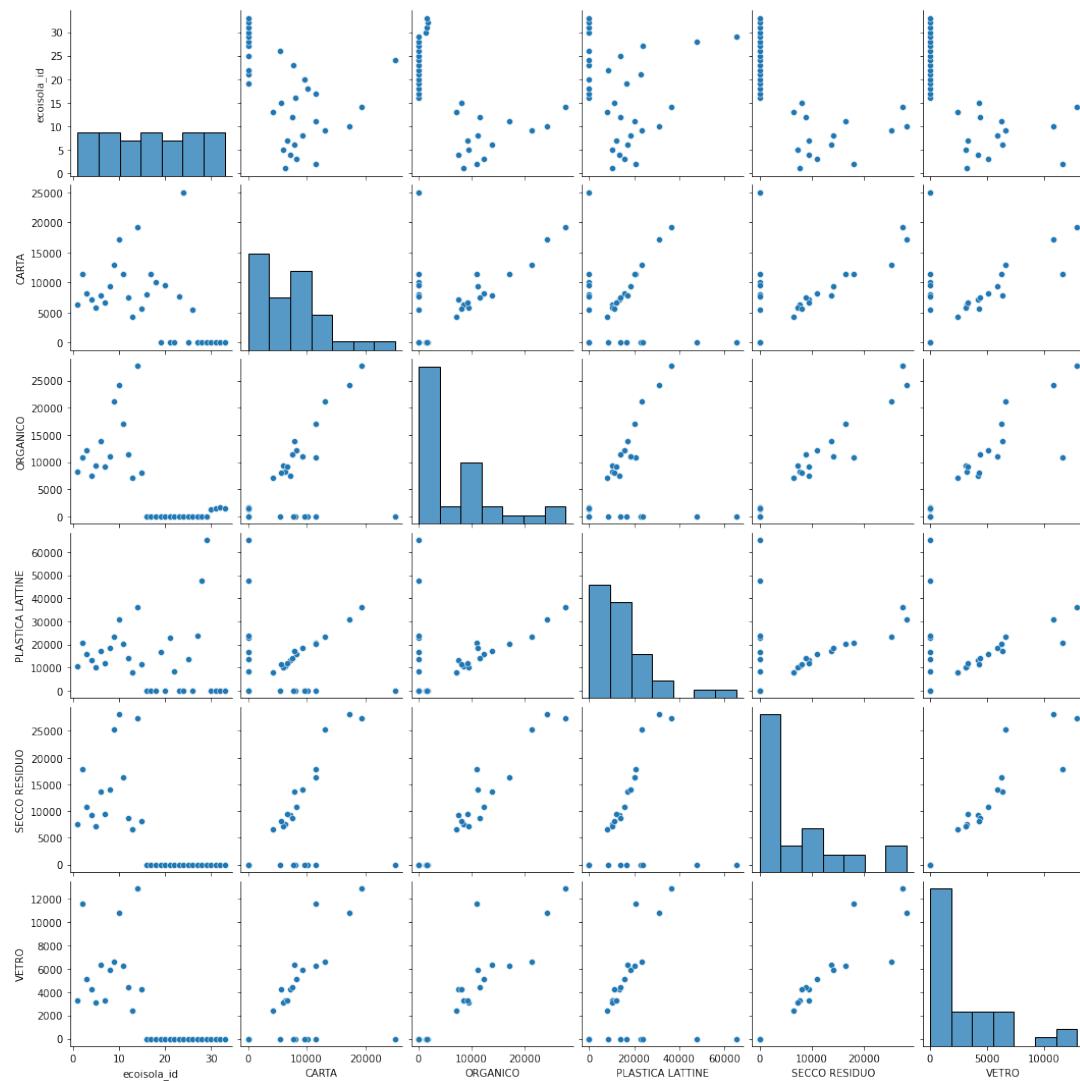


Figura 6.3: Risultato correlazione dataframe completo



Figura 6.4: Risultato Heatmap dataframe completo, più si avvicina ad 1 e più due features sono correlate

Con il seguente grafico andiamo vedere se selezionando 2 componenti della PCA la varianza è ancora contenuta e quindi non ho troppa perdita di informazione.

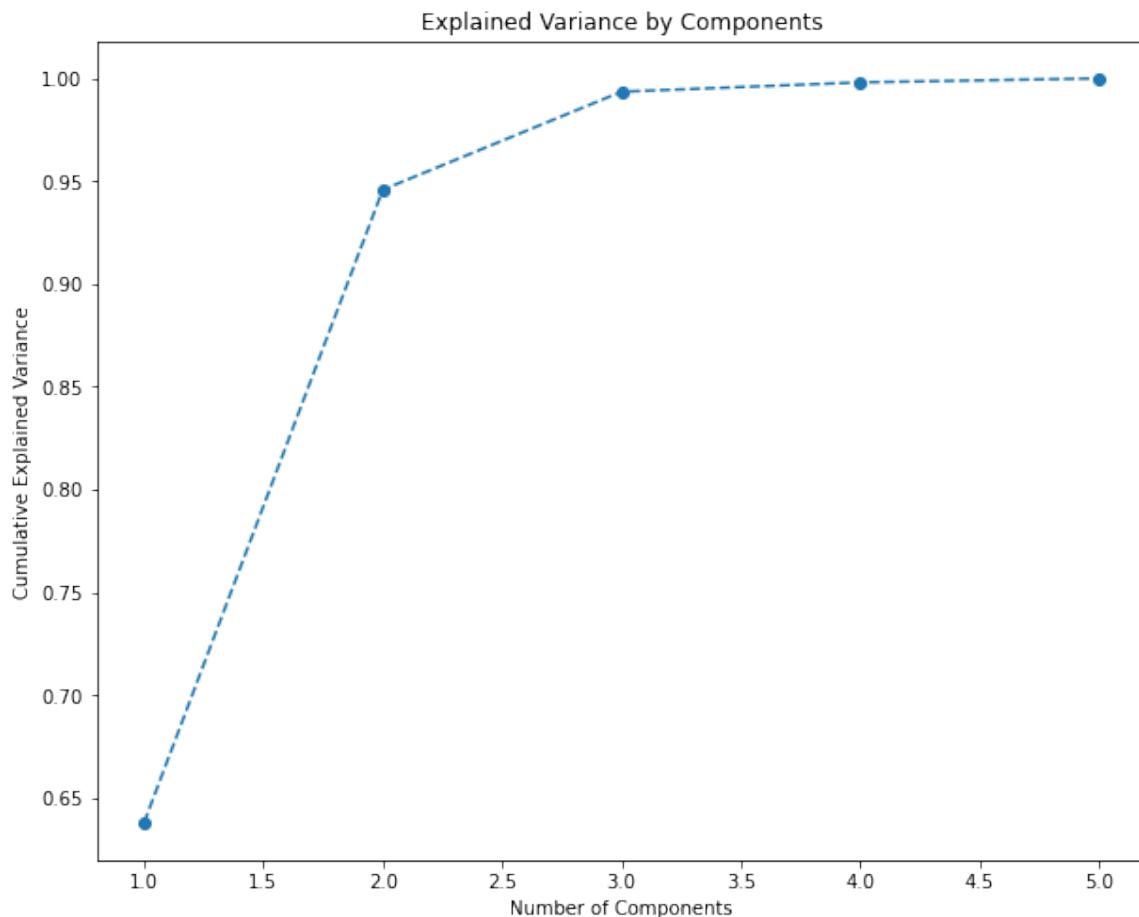


Figura 6.5: Variazione di Varianza per numero di componenti

Successivamente facciamo l'Elbow Method e lo plottiamo, questo metodo ci consente di andare a determinare il numero giusto di cluster da assegnare al K-means.

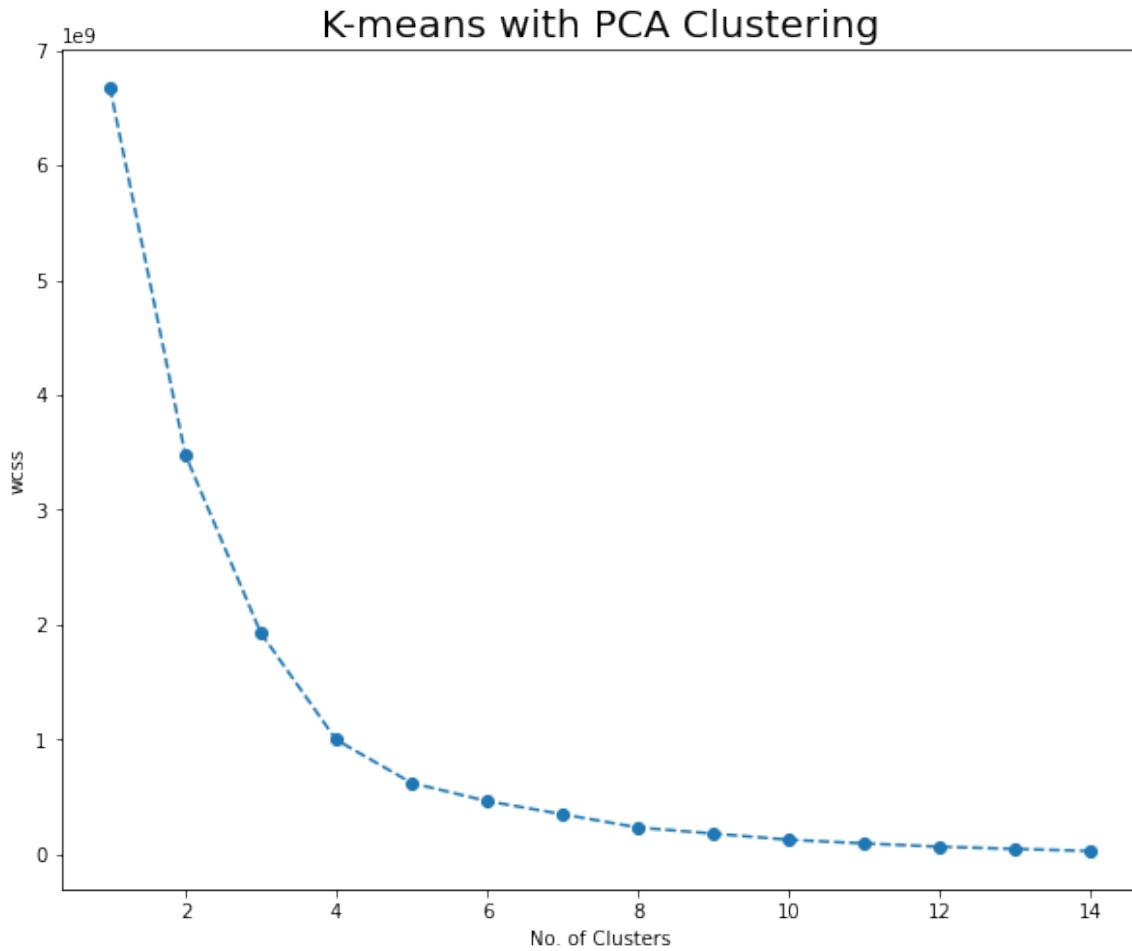


Figura 6.6: Elbow method, al K-means viene assegnato un valore di 5 cluster

Andando a plottare i risultati andiamo ad interpretare i valori di component 1, component 2 e al tipo di rifiuto, assegnamo un'etichetta ai singoli gruppi delle ecoisole.

index	CARTA	ORGANICO	PLASTICA LATTINE	SECCO RESIDUO	VETRO	Component 1	Component 2	Segment K-means PCA	Segment
28	0	0	65190	0	0	38394.31079943...	35512.7853284177	3	Plastica Alto
27	0	0	47769	0	0	23522.21885495...	26772.9338161726	3	Plastica Alto
13	19262.5	27656	36276	27307.5	12900.6	36378.71798675...	-18051.31833106...	0	Misto Alto
9	17242.5	24168	30795	28122.5	10818.9	30378.09703233...	-18240.99839520...	0	Misto Alto
26	0	0	23691	0	0	2967.1284560489	14693.3663566791	4	Plastica Basso
8	13005	21174	23367	25207.5	6615	20973.61926719...	-16264.87673861...	0	Misto Alto
20	0	0	22929	0	0	2316.6184932627	14311.0825123343	4	Plastica Basso
1	11445	10844	20664	17937.5	11626.2	13298.266646555	-9088.0841327964	1	Misto Basso
10	11457.5	17038	20268	16405	6280.2	13561.74975622...	-10322.429029327	1	Misto Basso
7	9300	11020	18294	14005	5931.9	8810.9940965724	-6099.312398662	1	Misto Basso
5	7930	13796	17184	13722.5	6312.6	8589.0714740286	-7403.8070931072	1	Misto Basso
18	0	0	16563	0	0	-3117.95686513...	11117.3568520994	4	Plastica Basso
2	8147.5	12244	15645	10892.5	5127.3	5568.3631630662	-5703.9467468072	1	Misto Basso

Figura 6.7: Risultato top 10 ecoisole a rischio maggiore di riempimento

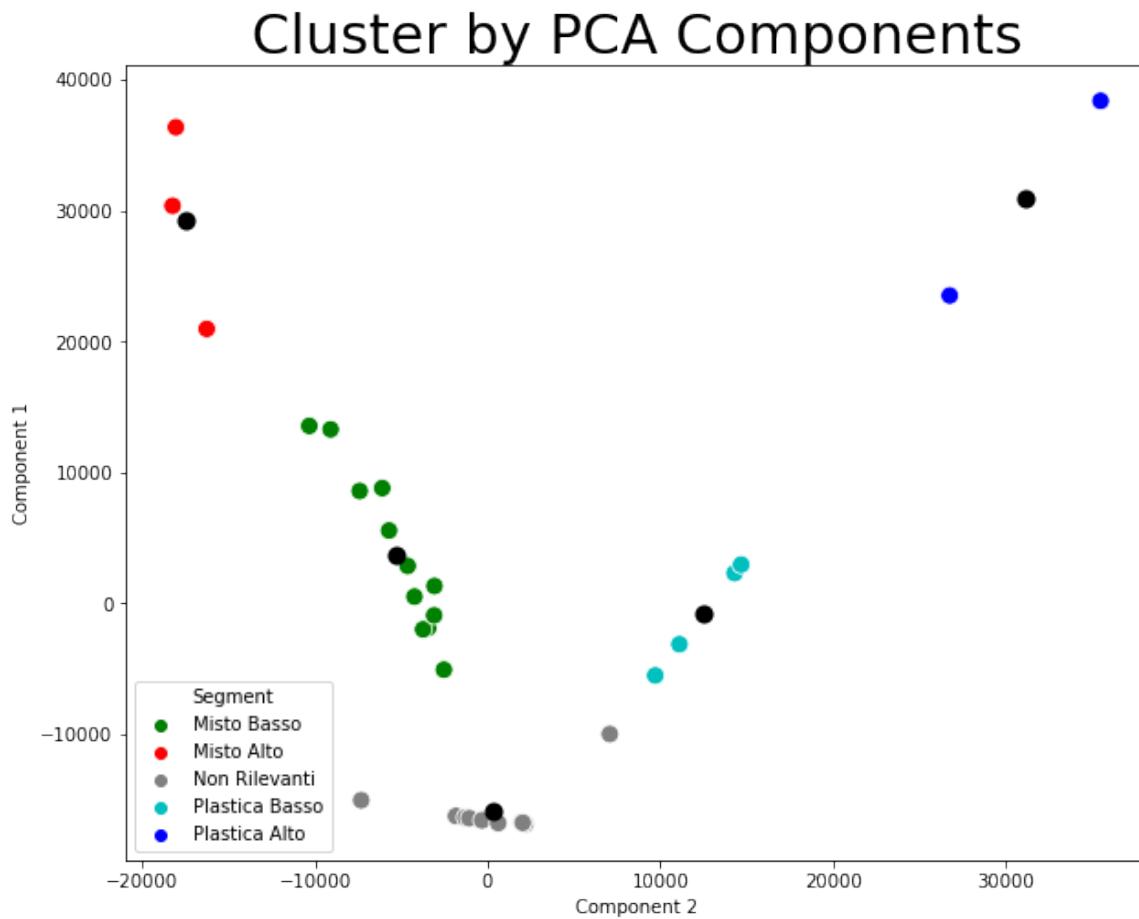


Figura 6.8: Plot finale

I gruppi a maggior rischio sono il gruppo 0(Misto alto) e il gruppo 3(plastica alto). Le ecoisole con maggior rischio in tutto l'anno sono la 28,27,13,9.

6.5 Risultati

I risultati relativi al dataset completo sono stati mostrati nel capitolo precedente, andiamo adesso a vedere i risultati riguardo i mesi estivi e non estivi.

6.5.1 Mesi Estivi

index	CARTA	ORGANICO	PLASTICA LATTINE	VETRO	Component 1	Component 2	Segment K-means PCA	Segment
28	0	0	22854	0	14339.4146346312	10186.8763831105	1	Solo Plastica Alto
13	7735	11038	14985	5448.6	13073.3164475674	-6043.4725869769	2	Misto Alto
9	6982.5	9238	12975	4749.3	10393.486669793	-5045.4516766914	2	Misto Alto
1	6985	6512	13299	7538.4	10340.6247810177	-4223.049321941	2	Misto Alto
27	0	0	14667	0	6911.8737355379	7029.1142976631	1	Solo Plastica Alto
8	4567.5	7496	9150	2643.3	5589.1726293515	-3283.2661090295	5	Misto Medio
10	3772.5	5634	7176	2365.2	3004.4581015224	-2348.0937137742	5	Misto Medio
7	3722.5	4696	7314	2516.4	2833.5833264295	-1754.4438900214	5	Misto Medio
2	3397.5	5352	6963	2435.4	2683.915803382	-2052.7758509537	5	Misto Medio
3	3525	3704	6459	1995.3	1583.6093896103	-1196.5093769408	5	Misto Medio

Figura 6.9: Risultato top 10 ecoisole a rischio maggiore di riempimento

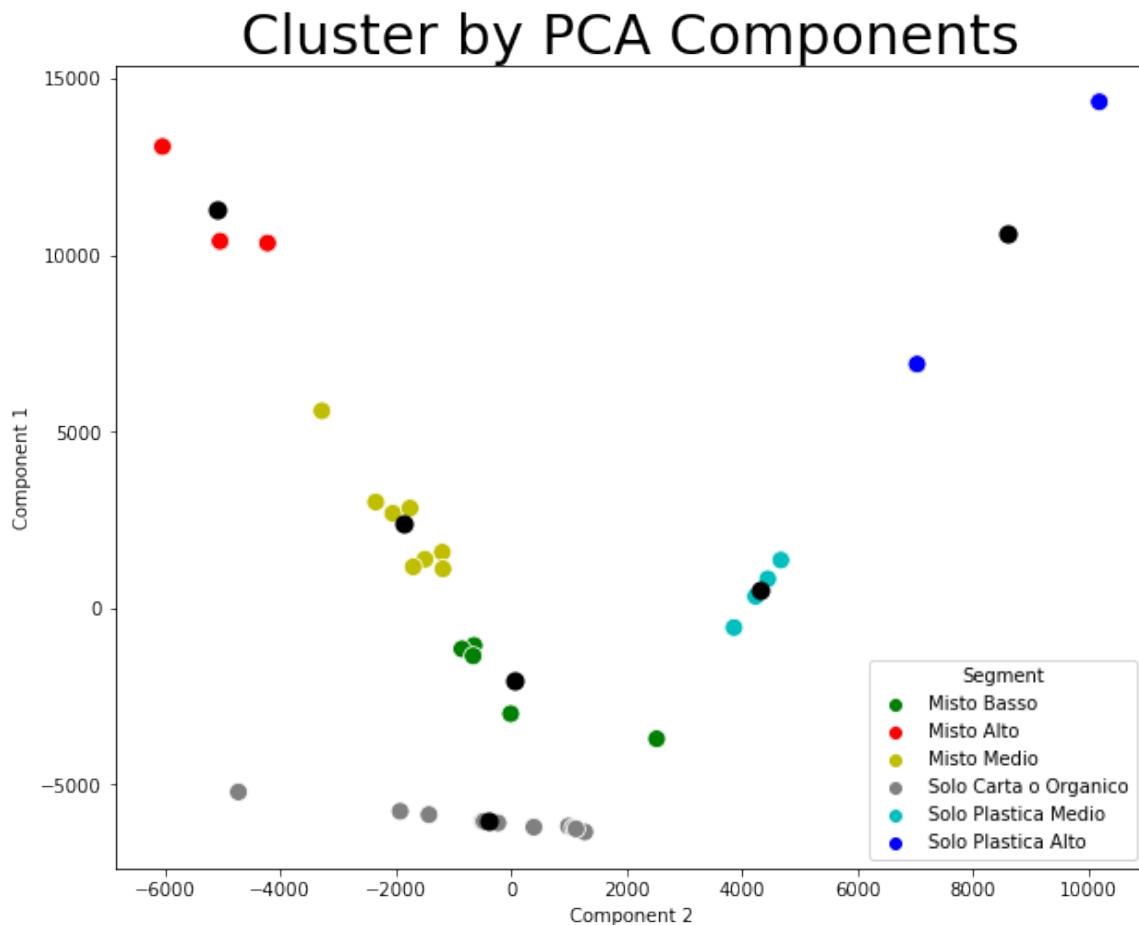


Figura 6.10: Plot finale

I gruppi a maggior rischio sono il gruppo 1(Plastica Alto) e il gruppo 2(Misto Alto). Le ecoisole con maggior rischio per i mesi estivi sono la 28,13,9,1,27.

6.5.2 Mesi non estivi

index	CARTA	ORGANICO	PLASTICA LATTINE	SECCO RESIDUO	VETRO	Component 1	Component 2	Segment K-means PCA	Segment
Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
28	0	0	42336	0	0	28532.2176227346	19365.6215214369	2	Solo Plastica Alto
27	0	0	33102	0	0	20098.3872711941	15779.6763625142	2	Solo Plastica Alto
13	11527.5	16618	21291	16512.5	7452	19747.980339694	-13800.9249678435	4	Mista Alto
9	10260	14930	17820	16677.5	6069.6	15979.6045206476	-13513.4807599169	4	Mista Alto
8	8437.5	13678	14217	15557.5	3971.7	11729.1945255005	-12387.6767069053	4	Mista Alto
10	7685	11404	13092	10955	3915	8735.0085497251	-8568.1084254953	4	Mista Alto
5	5297.5	9282	11250	9035	3812.4	5826.8163709735	-6031.1751500935	1	Mista Basso
7	5577.5	6324	10980	8580	3415.5	4646.7061348934	-4264.1647452675	1	Mista Basso
26	0	0	15141	0	0	3693.7926270507	8804.6751713282	3	Solo Plastica Basso
20	0	0	14967	0	0	3534.870547652	8737.1037550782	3	Solo Plastica Basso

Figura 6.11: Risultato top 10 ecoisole a rischio maggiore di riempimento

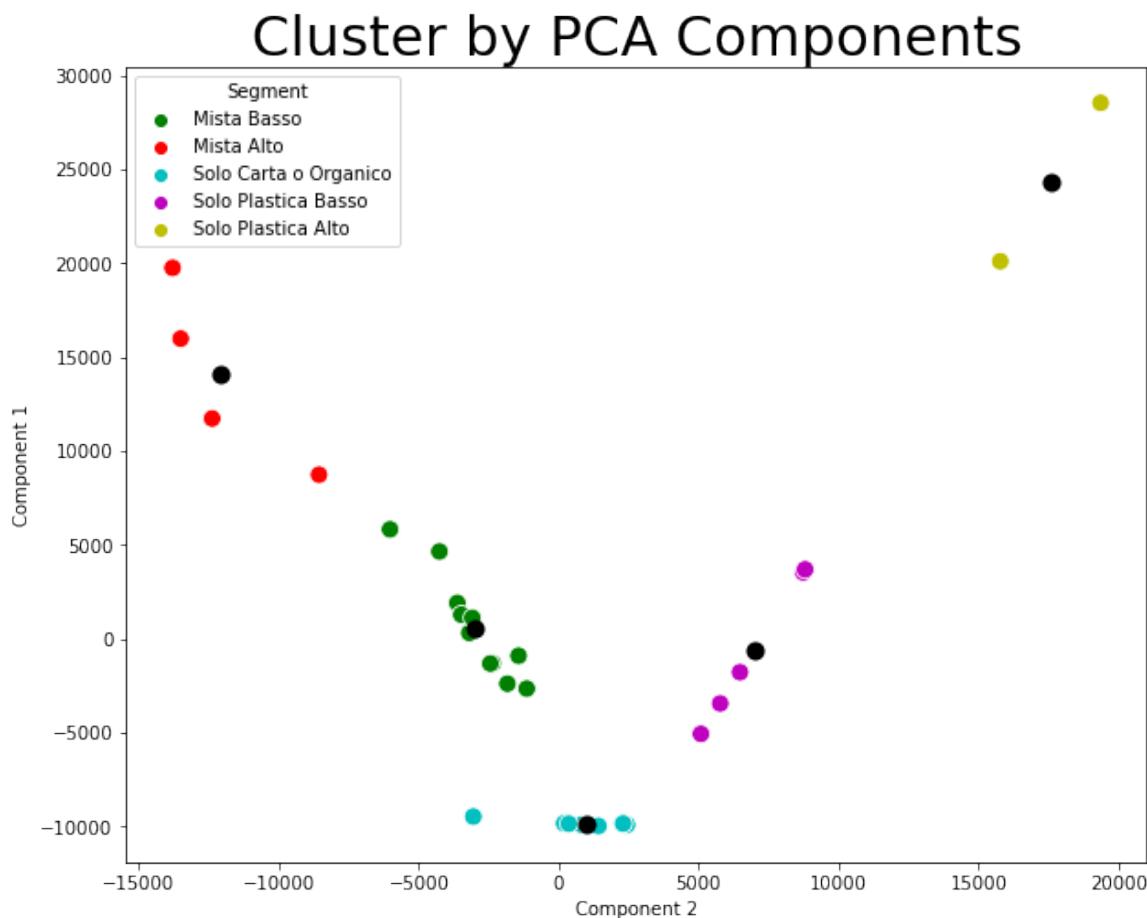


Figura 6.12: Plot finale

I gruppi a maggior rischio sono il gruppo 2(Plastica Alto) e il gruppo 4(Mista Alto). Le ecoisole con maggior rischio per i mesi non estivi sono la 28,27,13,9.

6.6 Conclusioni finali

Dai risultati di questo studio con il clustering è stato possibile determinare quali sono le ecoisole più a rischio, queste sono:

- l'ecoisola 28, questa ecoisola accetta solo plastica, nell'anno 2022 dovrebbe esser stata svuotata almeno per un totale di **651** volte;
- l'ecoisola 27, questa ecoisola accetta solo plastica, nell'anno 2022 dovrebbe esser stata svuotata almeno per un totale di **477** volte;
- l'ecoisola 13, questa ecoisola accetta tutti i tipi di rifiuto, nell'anno 2022 dovrebbe esser stata svuotata almeno per un totale di **1,232** volte;
- l'ecoisola 9, questa ecoisola accetta tutti i tipi di rifiuto, nell'anno 2022 dovrebbe esser stata svuotata almeno per un totale di **1,109** volte.

I valori considerati per lo svuotamento sono stati presi dal risultato della tab.6.7. L'ecoisola 13 e 9 hanno dei risultati molto alti perchè gestiscono 5 depositi, mentre la 28 e la 27 solo 1 deposito. La formula per calcolare lo svuotamento è data da:

$$\sum_{i=1}^n \frac{\text{Quantità del tipo di rifiuto}_i}{100}$$

Dove: - n è il numero totale di tipi di rifiuti considerati(Carta,Organico,Plastica,Secco e vetro).

- Quantità del tipo di rifiuto $_i$ rappresenta la quantità del tipo di rifiuto per l'elemento i -esimo.

Capitolo 7

TimeSeries

In questo capitolo troverete uno studio sulle serie temporali, La libreria di riferimento per questo studio è Statsmodels. Il dataset utilizzato è lo stesso utilizzato nel capitolo precedente, maggiori dettagli sul dataset li potrete trovare nel Capitolo 6.

7.1 Pre-Processing

L'oggetto di studio delle serie temporali è sostanzialmente la previsione di una determinata grandezza in base alla dimensione tempo. Prendendo dunque il file deposits.csv si è scelto di vedere l'andamento del numero di conferimenti su una singola ecoisola in base alle ore giornaliere. L'obiettivo di tale analisi è dunque di riuscire a determinare quali siano le fasce orarie ottimali per lo svuotamento dell'ecoisola. Si è dunque caricato il dataset in un dataframe, successivamente, dopo aver effettuato la drop di alcune colonne superflue al fine dell'analisi, si è creata una nuova colonna calcolata aggregando il numero di conferimenti ogni quattro ore per ogni giorno del mese di giugno. Nella seguente immagine si mostra l'andamento della serie.

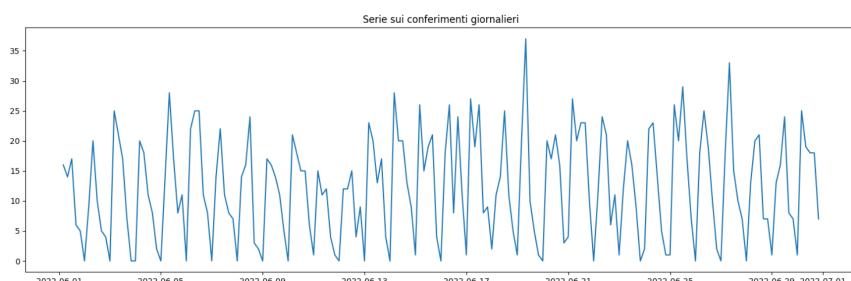


Figura 7.1: Serie Conferimenti

7.2 Stima parametri p,d,q

Per costruire il modello ARIMA si è proceduto nella stima dei tre parametri: p,d,q.

7.2.1 Parametro d

L'algoritmo ARIMA necessita dell'ordine di differenziazione per cui la serie temporale in esame deve essere stazionaria. Per trovare questo ordine abbiamo utilizzato ADF Test. Considerando il valore del P-Value, minore di 0.05 come risultato dell'ADF, è stato quantificata la stazionarità della serie, distinguibile anche dal grafico della serie. Dunque la serie non è stata differenziata, indicando con 0 il valore del parametro d.

7.2.2 Parametro p

Possiamo valutare p per ogni serie temporale analizzando i plot di auto correlazione parziale della serie e delle sue derivate. L'autocorrelazione parziale è una funzione della correlazione seriale tra le osservazioni di una serie storica, dopo aver eliminato gli effetti delle relazioni lineari intermedie. Per esempio, la funzione di autocorrelazione parziale al lag 2 misura la correlazione seriale tra Z_t e Z_{t-1} dopo aver eliminato la correlazione seriale di lag 1.

7.2.3 Parametro q

Come per p si analizza però il grafico di autocorrelazione che rappresenta di quanti termini MA si necessita per rimuovere ogni autocorrelazione nella serie stazionaria. In conclusione è stato scelto il modello con $p=3$, $d=0$ e $q=1$

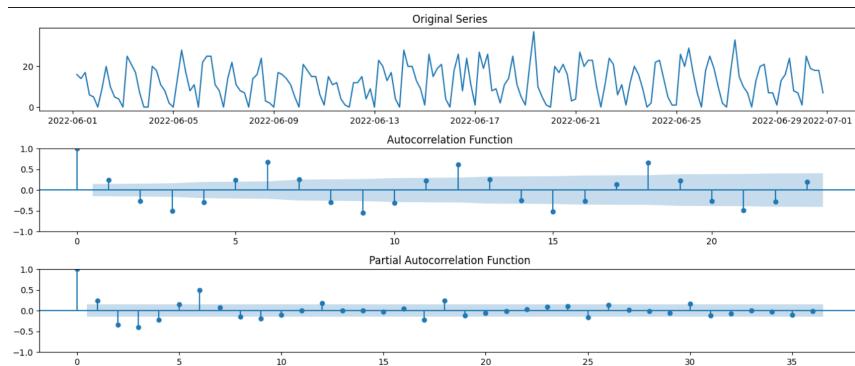


Figura 7.2

7.3 Addestramento e risultati Arima

I modelli che andremo a descrivere sono stati ottenuti confrontando il modello con i parametri descritti nella sezione precedente.

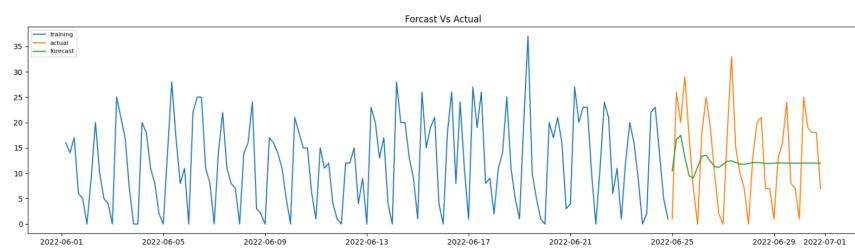


Figura 7.3: Confronto serie reale con modello Arima

Il modello ottenuto è stato addestrato considerando l'80% dei dati a disposizione, il venti percento è messo a confronto con le previsioni del modello. Si nota che inizialmente il trend viene individuato, tuttavia all'aumentare delle previsioni, il modello sembra degeneri in una retta. Risultato che può dipendere dalla non considerazione della stagionalità indotta dalla partizione in fasce orarie. Questo concetto verrà approfondito con il modello Sarimax.

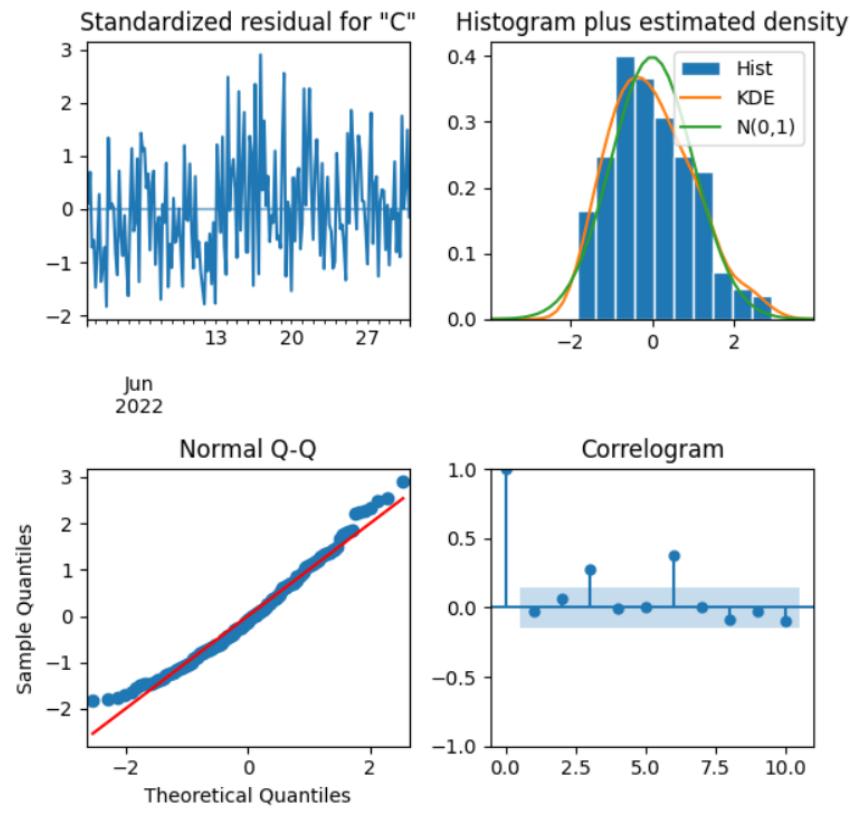


Figura 7.4: Diagnostica sul modello

Dal primo grafico possiamo vedere come il residuo sembri fluttuare intorno ad una media nulla e con una varianza più o meno uniforme per cui possiamo affermare che non c'è correlazione tra gli errori e quindi nessuna dinamica non è stata integrata nel modello. Questo viene confermato anche dal grafico in alto a destra che mostra l'istogramma più la stima della distribuzione del residuo paragonato ad una normale standard. Il grafico Normal Q-Q mostra invece come la distribuzione stimata abbia la stessa forma di quella normale meno gli estremi.

7.4 Sarima

Come anticipato, considerata l'aggregazione in fasce di orari, si è cercato di usare un modello in grado di considerare la stagionalità. Infatti è stato costruito un modello Sarima con stessi parametri p,d,q di Arima aggiungendo stagionalità e un trend lineare.

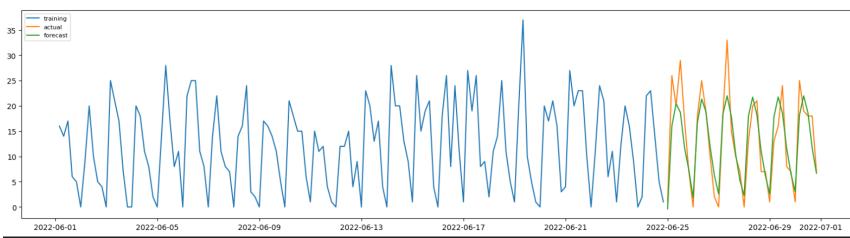


Figura 7.5: Cross validation Sarima

Anche in questo caso sono stati utilizzati l'80% dei dati come training per il modello e il restante 20% come test, sovrapponendo il modello dai valori reali. Si nota che aggiungendo la parte relativa alla stagionalità si riesce ad individuare meglio il trend della serie, avendo così previsioni più accurate.

ME (Mean Error): Rappresenta la media degli errori (differenze) tra le previsioni e i valori reali. Un valore negativo indica che le previsioni tendono a sottostimare i dati reali in media. MAE (Mean Absolute Error): È la media degli errori assoluti tra le previsioni e i valori reali. Misura l'errore medio

me	-0.5107751763481478
mae	3.2558754138670554
rmse	4.330569720939624
corr	0.8845834895990865

Tabella 7.1: Valori accuratezza modello

senza considerare la direzione degli errori. RMSE (Root Mean Square Error): È una misura della deviazione standard degli errori tra le previsioni e i valori reali. Un valore elevato indica una dispersione elevata degli errori. Corr (Correlation): Rappresenta la correlazione tra le previsioni e i valori reali. Un valore positivo e vicino a 1 indica una buona correlazione tra i dati.

Reteniamo il MAE e RMSE dei valori non troppo elevati, in realtà un errore di circa 5 conferimenti può essere accettabile nel caso di studio, riteniamo inoltre, il valore della correlazione tra previsioni e dati reali un buon valore. Concludendo, questo studio, può fungere da supporto per definire degli orari favorevoli allo svuotamento delle ecoisole, stimando in aggiunta, un fattore di riempimento per ciascun conferimento.

Capitolo 8

NetworkX

In questo capitolo troverete uno studio sui grafi, La libreria di riferimento per questo studio è Networkx.

8.1 Dataset utilizzato

Il dataset utilizzato è lo stesso utilizzato nei capitoli precedenti, maggiori dettagli sul dataset li potrete trovare nel Capitolo 6.

8.1.1 Obiettivo del grafo

L'**Obiettivo** dello studio è di creare una rete di consumatori che sia dipendente dall'ecoisola di conferimento. Successivamente grazie all'utilizzo di librerie come geopy e folium andremo a generare delle mappe con la distanza che un consumatore deve compiere per arrivare ad una ecoisola. In questo modo il cliente di questo studio potrà trovare i posti migliori per piazzare ulteriori ecoisole, in prossimità dei consumatori che generano più rifiuti, come attività commerciali e hotel. L'anno preso in esame per questo studio è il 2022 durante i mesi non estivi. Nella generazione del grafo terremo conto solo delle ecoisole che gestiscono tutti i tipi di rifiuti, i consumer considerati sono quelli che hanno effettuato nell'anno un minimo di 30 conferimenti.

8.2 Preprocessing

Dobbiamo Crearci un dataframe che metta in relazione le ecoisole, i depositi e i consumers. Prendiamo quindi da deposits l'id del consumer e l'id dell' ecoisola:

index	id	consumer_id	ecoisola_id	Data
1	87122	7032	7	2022-01-...
12	87133	4721	7	2022-01-...
14	87135	4721	7	2022-01-...
18	87139	6107	7	2022-01-...
21	87142	6107	7	2022-01-...
84	87205	5597	7	2022-01-...
85	87206	5597	7	2022-01-...
86	87207	5597	7	2022-01-...
87	87208	5597	7	2022-01-...
88	87209	6945	7	2022-01-...
91	87212	5597	7	2022-01-...

Figura 8.1: dataset prima fase

In base al id del consumer uniamo le tabelle e contiamo anche quante volte hanno conferito in ogni ecoisola.

index	ecoisola_id	consumer_id	weight
290	2	4719	34
1532	6	4724	70
1533	6	4725	31
293	2	4728	38
4330	11	4731	108
6280	15	4732	35
6281	15	4739	52
2915	9	4746	31
3465	10	4746	41

Figura 8.2: dataset seconda fase

Successivamente per ogni ecoisola assegnamo tutti gli id dei consumers.

ecoisola_id	consumer_ids
1	5082,5338,5484,5536,5687,5969,6290,6311,6367,6372,6395,6570,7069,7510...
2	4719,4728,4777,5123,5354,5607,5909,6797,6898,6992,7240,7266,7515,8073...
3	4812,5031,5093,5315,5898,6134,6198,6241,6262,6329,6420,6450,6507,6627...
4	4770,4967,5168,5439,5475,5914,6144,6333,6641,6830,6902,7018,7140,7243...
5	4806,4900,5069,5235,5368,5376,5449,5470,5657,5658,5784,5785,5896,5966...
6	4724,4725,4799,4843,4928,4984,5118,5224,5253,5302,5306,5590,5653,5779...
7	4788,4816,4933,5151,5450,5516,5597,5657,5658,5660,5782,5784,5834,5849...
8	4762,4832,4948,4984,5011,5044,5061,5203,5301,5320,5369,5377,5488,5594...
9	4746,4751,4777,4781,4793,4794,4799,4800,4803,5112,5233,5395,5452,5477...
10	4746,4749,4775,4789,4821,4858,4865,5083,5137,5173,5193,5268,5270,5284...
11	4731,4762,4819,4822,4847,4925,5074,5130,5150,5205,5232,5268,5367,5378...
12	4926,5095,5304,5567,5668,5796,5841,5950,6030,6064,6118,6120,6147,6496...
13	4862,5078,5132,5299,5332,5422,5442,5489,5572,5629,5648,5677,5922,5998...
14	4754,4755,4762,4770,4794,4799,4838,4894,4902,4908,4921,4934,4941,4949...
15	4732,4739,5011,5250,5275,5294,5494,5507,5612,5685,6256,6503,6596,6602...

Figura 8.3: dataset terza fase

Infine per ogni consumer presente nella lista andiamo ad assegnare il consumer successivo in modo da crearci per ogni riga dei legami (gli edge del grafo).

index	ecoisola_id	consumer_id_1	consumer_id_2	weight_1	weight_2
0	1	5082	5338	37	85
1	1	5082	5484	37	76
2	1	5082	5536	37	127
3	1	5082	5687	37	62
4	1	5082	5969	37	42
5	1	5082	6290	37	78
6	1	5082	6311	37	81
7	1	5082	6367	37	47
8	1	5082	6372	37	72
9	1	5082	6395	37	94
10	1	5082	6570	37	100
11	1	5082	7069	37	54
12	1	5082	7510	37	90
13	1	5082	7750	37	38
14	1	5082	7812	37	62
15	1	5082	7817	37	50
16	1	5082	7963	37	116

Figura 8.4: dataset quarta fase

8.3 Creazione del grafo

Creiamo il grafo con networkX e otteniamo le seguenti metriche:

```
NODES: 793  
EDGES: 31278  
DENSITY: 0.1  
CLUSTERING: 0.9560571164910647  
IS_CONNECTED: True
```

Figura 8.5: Metriche del grafo

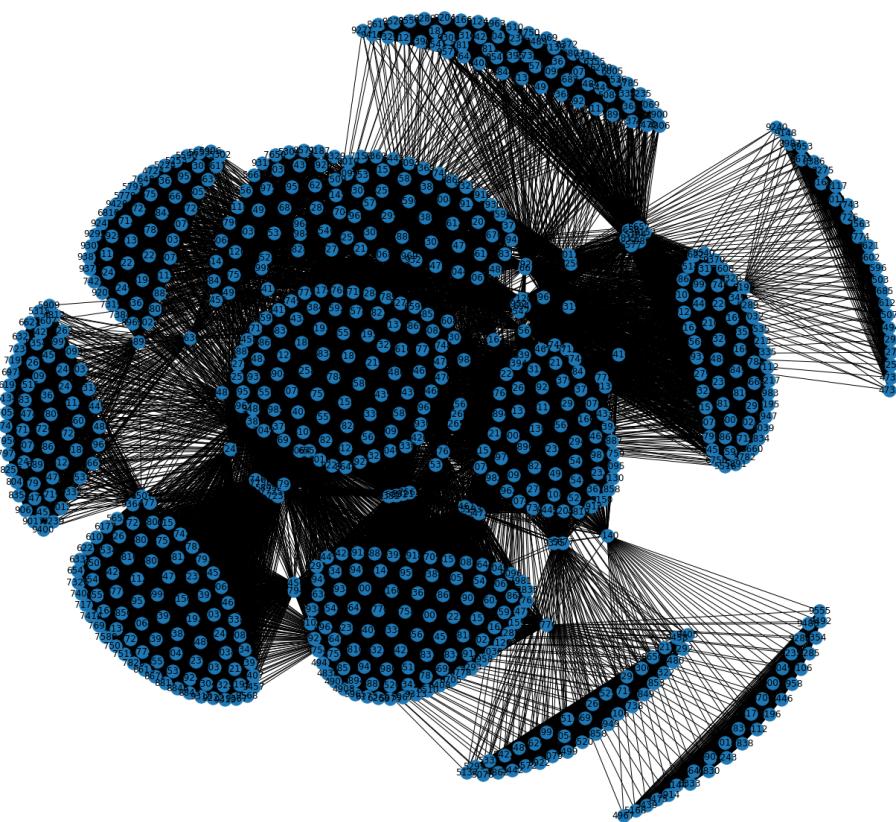


Figura 8.6: Grafo Risultante

Il risultato del grafo risulta poco interpretabile, quindi è stato fatto uno studio sulla degree centrality.

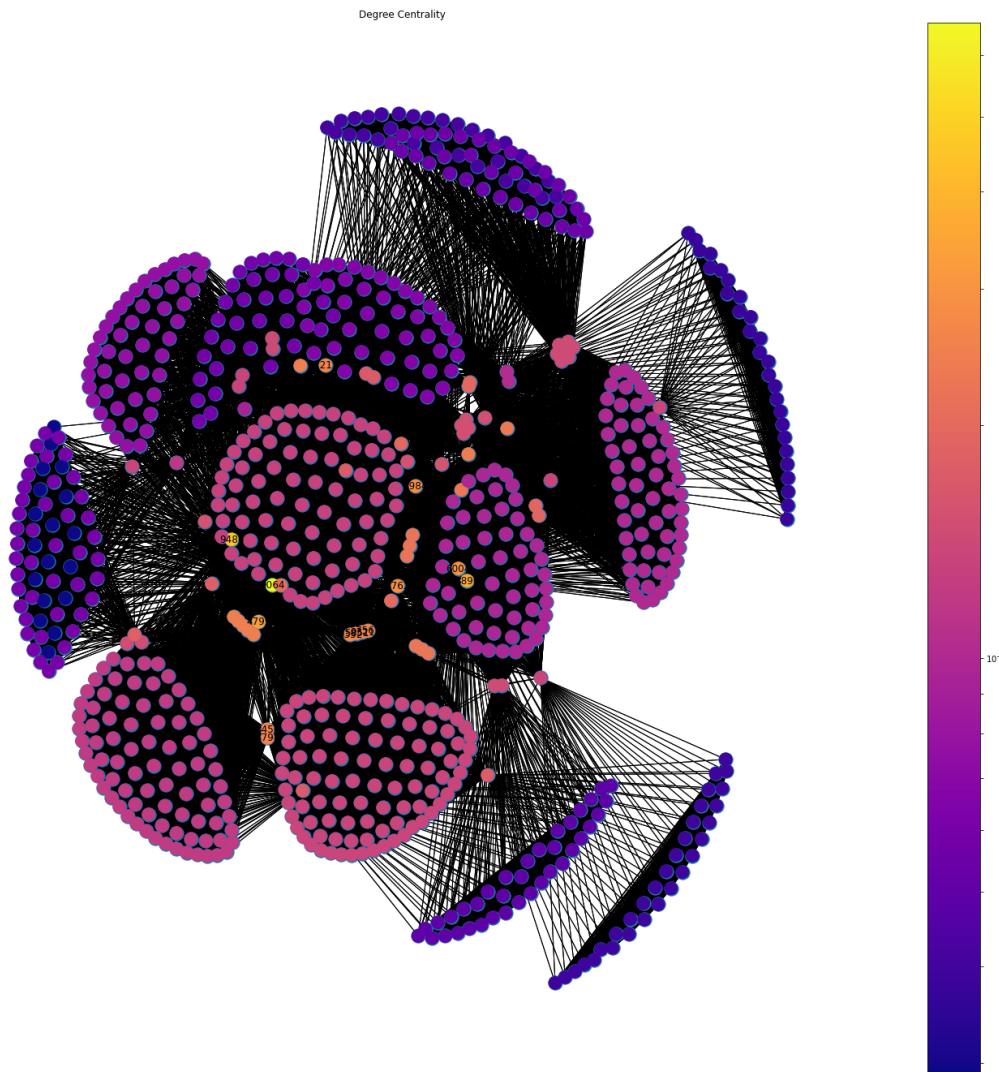


Figura 8.7: Grafo Degree Centrality

La degree centrality è una misura che valuta il numero di archi incidenti su un nodo in una rete, fornendo un'indicazione della sua importanza o centralità all'interno della rete. I nodi con un etichetta sono i top 10 consumers per centralità più alta.

consumer_id	Degree Centrality
9064	0.6616161616
9486	0.446969697
8890	0.3623737374
4799	0.3295454545
6004	0.3005050505
4762	0.2904040404
5984	0.2840909091
5851	0.2626262626
7561	0.2626262626
9321	0.2626262626

Figura 8.8: Risultato top 10 consumers

8.4 Studio sul singolo nodo

Dalla Tab.8.8 possiamo andare a selezionare un determinato nodo per studiarlo più approfonditamente, in questa parte faremo uno studio sul nodo **8890**, dalla tabella consumers prendiamo l'indirizzo, in questo caso è Piazza De Santis. Utilizzando la libreria geopy. Questa libreria nel caso in cui non dovesse trovare una via, verrà automaticamente impostata nel centro città. Successivamente verrà impostato un minmaxScaler per vedere in quali ecoisole il consumer conferisce maggiormente e per ogni ecoisola prendiamo la latitudine e longitudine.

index	id	conteggio	conteggio_scalato	lat	lng
0	11	209	1	42.7679626	10.3957095
1	7	60	0.2836538462	42.7688738	10.398032
2	10	51	0.2403846154	42.765367	10.395023
3	13	30	0.1394230769	42.7702749	10.3975797
4	3	1	0	42.7676371724	10.4005926525
5	12	1	0	42.7684363397	10.3982066084

Figura 8.9: Risultato per ogni ecoisola utilizzata

con la libreria Folium abbiamo realizzato una mappa interattiva dove il pin rappresenta l'indirizzo del consumer e i cerchietti le ecoisole dove il consumer conferisce maggiormente su una scala di colore, cliccando sui cerchietti saranno presenti ulteriori informazioni come in nome dell'ecoisola, il numero di conferimenti e la distanza.

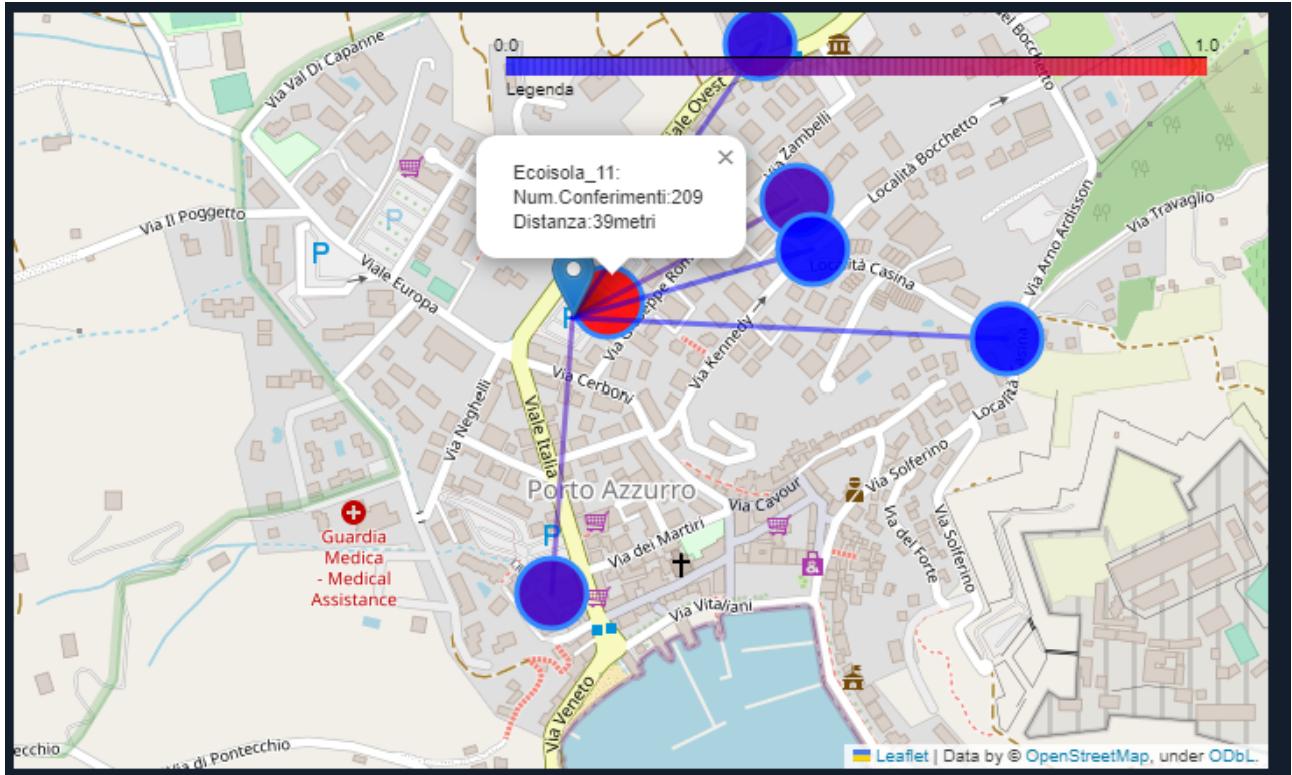


Figura 8.10: Risultato per ogni ecoisola utilizzata

Quest'analisi dettagliata può esser eseguita per ogni consumer presente nel dataset

Capitolo 9

Rasa Chat-bot Prenotazione salone di Parrucchieri

Un chatbot è una particolare tipologia di software in grado di simulare una conversazione con un essere umano. Questi tipi di software utilizzano degli algoritmi per la comprensione del linguaggio naturale (NLU) ed elaborano una risposta in funzione di ciò che un utente scrive come input; consentendo inoltre di elaborare l'informazione anche delle conversazioni passate. I principali vantaggi che spingono le aziende ad avere un chatbot sono:

- Riduzione dei costi
- Servizio multilingue
- Servizio attivo 24 ore su 24 7 giorni su 7.

9.1 Rasa

Rasa è un framework open source per la creazione di chatbot basati sul machine learning supervisionato che gode di una community molto ampia per cui gli sviluppatori sono sempre pronti a rispondere ad eventuali domande e chiarimenti.



Figura 9.1: Logo Rasa

9.1.1 Principi di funzionamento

Rasa è composto da due parti: Rasa Core e Rasa NLU. Rasa NLU è la componente che si occupa di comprensione del messaggio fornito dall'utente e della sua classificazione. Rasa NLU riesce ad analizzare e comprendere la grammatica della frase al fine di classificarla per poter costruire la risposta opportuna (generata da Rasa Core). Rasa Core è invece elabora le risposte da fornire all'utente in base agli input forniti dall'utente nel presente e nel passato. Di seguito una descrizione delle componenti principali di Rasa NLU e Core:

- **RasaCore.Domain** il dominio di applicazione del bot. Viene descritto all'interno del file domain.yml. Esso contiene; Elenco *intent*, elenco *slot*, elenco *forms*, elenco *responses*.
- **RasaCore.Stories** rappresentano gli esempi di conversazione possibili. Attraverso le stories vengono definiti i diversi pattern conversazionali intraprendibili dal bot alternando intent ed actions.
- **RasaCore.Rules** Sono anch'essi esempi di conversazione, tuttavia le rules obbligano il bot di rispondere seguendo il percorso indicato dalla rule. Vengono molto utilizzati per gestire le forms.
- **RasaCore.Actions** Azioni eseguibili dal bot implementate in python. Grazie alle actions è possibile effettuare delle chiamate al database o, in generale, ritornare risposte complesse generate dinamicamente. Per definire nuove actions basta instanziare un oggetto della classe Action del package rasa_sdk.
- **RasaCore.Form** sono dei pattern che permettono al bot di raccogliere frammenti di informazione riempiendo gli slots. Definito un form è fondamentale definire i messaggi automatici che guideranno l'utente al riempimento del form. Questi messaggi possono essere dei semplici responses o anche delle custom actions. Rasa cercherà di lanciare delle risposte nominate nella maniera che segue: utter_ask_<nome_slot> in caso di response o action_ask_<nome_slot> in caso di custom action. Inoltre è possibile (ed è una best practice) definire una classe di validazione del form estendendo la classe FormValidationAction del package rasa_sdk con nome validate_<nome_form> per effettuare la validazione sul corretto riempimento dello slot attuale. Per effettuare la validazione di uno slot in particolare è sufficiente aggiungere un metodo alla classe chiamato validate_<nome_slot>.
- **RasaCore.Slots** sono "memorie" preposte a ricordare alcuni valori dell'input dell'utente. Vengono utilizzati per salvare dei valori che possono essere utilizzati dal bot per, ad esempio, riempire una form.
- **RasaNLU.Intent** costituiscono le finalità richieste dall'utente. Per definire un intent, e conseguentemente addestrare il bot a riconoscerle, è necessario specificare diversi esempi di frasi di quel tipo.
- **RasaNLU.Entities** sono oggetti utili a comprendere meglio l'intent della frase e possono essere associate al riempimento delle slots. Ad esempio nella frase "vorrei prenotare per un *Taglio uomo*" la parola taglio uomo potrebbe essere interpretata come l'entità dell'intent prenotazione taglio per estrarre come slot il taglio che l'utente vuole prenotare.
- **RasaNLU.Pipeline** è una sequenza di step per la realizzazione del classificatore di intent le cui specifiche vengono descritte nel file config.yml.

9.1.2 Interrgrazione con altre piattaforme

Rasa permette agevolmente di integrare il bot creato con Rasa all'interno delle piattaforme di messaggistica più comuni come Telegram, Messenger o Slack tramite codici di verifica forniti dalle piattaforme stesse. Nel bot realizzato tramite Ngrok è stato possibile intergrare il bot per Telegram.

9.2 Bot per la gestione delle prenotazioni di un salone di parrucchieri

La prenotazione e la gestione degli appuntamenti in un salone è un processo che (nel caso in cui non sia automatizzato) comporta deviazioni in termini di tempo all'atto lavorativo. Generare un'applicazione in grado di gestire questa tematica risulta sicuramente un'ottimizzazione. Tuttavia si perderebbe il lato confidenziale tra utente e produttore del servizio. Dunque, in questo caso, la realizzazione di un bot risulta particolarmente adatta, in quanto al cliente viene data l'impressione di parlare con un operatore. Da questo parte l'idea di realizzare un bot per gestire le prenotazioni di un salone.

9.3 Descrizione delle specifiche

Al fine di poter interagire con gli utenti, il chatbot realizzato dovrà rispondere a molti pattern conversazionali. Anche se le possibilità di conversazione sono molte, sappiamo che il nostro bot dovrà "lavorare" in un ambiente specifico (salone di parrucchieri) per cui le domande che potranno essergli sottoposte sono più o meno prevedibili. Il bot dovrà essere in grado di gestire, tramite conversazione con i clienti, le seguenti operazioni:

- Prenotazione di appuntamenti
- Modifica appuntamenti
- Eliminazione appuntamenti
- Risposte riguardo ad informazioni generali riguardo il salone
- Curiosità e consigli su prodotti per capelli

9.4 Implementazione

Di seguito vengono riportate le operazioni per cui il chatbot è stato realizzato, facendo riferimenti ai file messi a disposizione da Rasa.

9.4.1 Prenotazione appuntamenti

Per implementare tale operazione sono state utilizzate le seguenti funzionalità di Rasa:

Rasa NLU:

- **Intent _ Prenotazione:** Definisce le frasi che verranno classificate per interpretare l'intento, da parte dell'utente, di voler procedere con la registrazione di un nuovo appuntamento.
- **Intent _ Inserimento _ Taglio:** Definisce l'intento per cui, tramite l'entità taglio, verrà estratto il valore dello slot taglio durante l'esecuzione della form.
- **Intent _ Inserimento _ data:** Definisci l'intento di inserire una data da parte dell'utente, come vedremo, nell'esecuzione della form sarà importante per l'estrazione della data in cui si vuol prenotare
- **Intent _ Inserimento _ recapito:** Definisce l'intento di inserire un numero telefonico, a tale intent corrisponderà il valore dello slot recapito da salvare per archiviare la prenotazione
- **Intent _ Inserimento _ orario :** Analogico all'intent inserimento recapito, sarà utilizzato per salvare l'orario della prenotazione.
- **Intent _ Conferma :** Definisce le frasi per cui verrà riconosciuta la volontà da parte dell'utente di confermare la prenotazione. Analogico per intent _ nega.

Rasa Domain: Nel file *domain* sono presenti tutte le risposte che il bot dovrà eseguire in base alla classificazione degli intent. In particolare saranno presenti anche le *utter_ask_<slotname>* che definiscono le domande da parte del bot per procedere con la *Collezione_dati_Form*. Nel file *domain* sono definite anche le **Forms** e le **Slots**. La form dedicata alla prenotazione di un appuntamento è la seguente:

```
forms:
  colleziona_dati_prenotazione_form:
    required_slots:
      - nome
      - cognome
      - recapito
      - taglio
      - data
      - orario
      - conferma
```

Figura 9.2: Form di prenotazione

Essa, ha al di sotto dell'intestazione *required_slots* l'elenco delle slot in riferimento alla prenotazione. Di seguito l'immagine della definizione di alcune slot significative:

```
nome:
  type: text
  mappings:
    - type: from_text
      conditions:
        - active_loop: colleziona_dati_prenotazione_form
          requested_slot: nome
```

Figura 9.3: Slot Nome

Tale definizione di slot viene riempita in base all'input dell'utente durante l'esecuzione della form *collezione dati prenotazione form* dedicata alle prenotazioni.

```
taglio:
  type: text
  mappings:
    - type: from_entity
      entity: taglio
      intent: inserimento_taglio
```

Figura 9.4: Slot Taglio

Lo slot taglio, anch'esso di tipo text, si differenzia dal primo poichè il valore viene estratto dal valore dell'entità di riferimento, in questo caso è l'entità taglio.

```
conferma:
  type: bool
  mappings:
    - type: from_intent
      value: true
      intent: conferma
      conditions:
        - active_loop: colleziona_dati_prenotazione_form
          requested_slot: conferma
    - type: from_intent
      value: false
      intent: nega
      conditions:
        - active_loop: colleziona_dati_prenotazione_form
          requested_slot: conferma
```

Figura 9.5: Slot Conferma

Lo slot conferma invece è di tipo booleano e avrà il valore *true* se verrà classificato in rasa nlu l'intent conferma, a *false* per l'intent nega.

Rasa Action Rasa mette a disposizione delle funzionalità che permettono di creare delle risposte customizzate da parte del bot attraverso l'interazione tra rasa_core e dei file python importando opportuni package. Per quanto riguarda le forms, estendendo la classe *FormValidationClass* è possibile definire delle funzioni per gestire la validazione dei valori delle slots.

```
class ValidatePrenotazioneForm(FormValidationAction):
    def name(self) -> Text:
        return "Valida_collezione_dati_prenotazione_form"

    def validate_recapito(self, slot_value: Any, dispatcher: CollectingDispatcher, tracker: Tracker, domain: DomainDict,) try:
        slot_value=int(slot_value)
        slot_value=str(slot_value)
        numero= phonenumbers.parse(slot_value, "IT")
        if phonenumbers.is_valid_number(numero):
            return {"recapito":slot_value}
        else:
            dispatcher.utter_message(text="Il numero inserito non è corretto")
            return {"recapito": None}
    except:
        dispatcher.utter_message(text="Il numero di telefono sembra sbagliato")
        return { "recapito": None}
```

Figura 9.6: Validate slot recapito

Esempio di una funzione per validare il valore di uno slot è quello mostrato in figura, il quale verifica che ciò che è stato inserito da parte dell'utente risulta un numero telefonico corretto, in caso contrario non accetta il valore e richiede di inserire il numero. Infine per gestire tutte le prenotazioni del salone è stato creato un file json per archiviare gli appuntamenti, dunque l'ultima funzione che viene effettuata, una volta terminata la form, è una funzione che scrive sul file i dati estratti durante la conversazione.

```

class Prenotazione(Action):
    def name(self) -> Text:
        return "scrivi_prenotazione"

    @async def run(self, dispatcher: CollectingDispatcher, tracker: Tracker, domain: DomainDict) -> List[Dict[Text, Any]]:
        if tracker.get_slot("conferma"):
            nome = tracker.get_slot("nome")
            cognome = tracker.get_slot("cognome")
            recapito = tracker.get_slot("recapito")
            taglio = tracker.get_slot("taglio")
            data = tracker.get_slot("data")
            orario = tracker.get_slot("orario")
            with open("Prenotazioni.json", "w") as f:
                dict_pren = json.load(f)
                dict_pren["giorno"][data][taglio][orario]["nome"] = nome
                dict_pren["giorno"][data][taglio][orario]["cognome"] = cognome
                dict_pren["giorno"][data][taglio][orario]["recapito"] = recapito
                dict_pren["giorno"][data][taglio][orario]["taglio"] = taglio
            with open("Prenotazioni.json", "w") as f:
                json.dump(dict_pren, f, indent=4)

            dispatcher.utter_message(text="Grazie per aver inserito i tuoi dati. L'operazione si è conclusa con successo.")
            return [AllSlotsReset()]

        else:
            dispatcher.utter_message(text="L'operazione non ha avuto successo")

```

Figura 9.7: Funzione di aggiornamento del file Prenotazioni.json

RasaRule Per attivare la form che permette di inserire una nuova prenotazione è stata utilizzata una *rule* che obbliga il bot, una volta classificato l'intento _prenotazione di rispondere con l'esecuzione della form.

```

- rule: Esegui_appuntamento
  steps:
    - intent: prenotazione
    - action: colleziona_dati_prenotazione_form
    - active_loop: colleziona_dati_prenotazione_form

```

Figura 9.8: Rule per attivare la Form per la prenotazione

9.4.2 Esempio conversazione per prenotare un appuntamento





Con il corrispettivo inserimento nel file json:

```
"2023-08-12": {
    "turno": {
        "8-9": {},
        "9-10": {
            "nome": "costantino",
            "cognome": "tigano",
            "telefono": "3496874563",
            "taglio": "Taglio uomo"
        },
        "10-11": {},
        "11-12": {},
        "12-13": {},
        "14-15": {},
        "15-16": {},
        "17-18": {},
        "18-19": {}
    }
}
```

Figura 9.9: Aggiornamento file json

9.4.3 Altre operazioni

Per quanto riguarda l'operazione di modifica e eliminazione degli appuntamenti, esse sono state implementate in maniera analoga alla prenotazione. Infine le operazioni riguardo alle curiosità, consigli per

prodotti e info generali sul salone, esse sono state gestite tramite le stories. Riportiamo un esempio:

```
- story: curiosità_path
  steps:
    - intent: saluto
    - action: utter_saluto
    - intent: curiosità
    - action: utter_curiosità
```

Figura 9.10: Story per l'intento curiosità

9.5 Considerazioni

Il bot sviluppato è sicuramente semplicistico ed ottimizzabile. Si potrebbe senz'altro migliorare la parte di prenotazione inserendo ad esempio un calcolo stimato della durata dell'operazione. Ciò nella realizzazione del bot è resa standard ad un'ora. In ogni caso sviluppare questo bot è stato molto utile per capire il funzionamento di un framework complesso e molto potente come RASA, oltre che le basi della NLP.

Bibliografia

- [1] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009.
- [2] Thomas Kluyver, Benjamin Ragan-Kelley, Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, et al. *Jupyter Notebooks - a publishing format for reproducible computational workflows*, pages 87–90. IOS Press, 2016.

