

Università Politecnica delle Marche
Dipartimento di Ingegneria dell'Informazione
Facoltà di Ingegneria Informatica e dell'Automazione

Corso di Data Science



Analisi dell'utilizzo delle applicazioni mobile presenti nel Play Store di Google

Professore:
Prof. Ursino Domenico
Dott. Virgili Luca
Dott. Marchetti Michele

Studenti:
Bernovschi Denis
Giacomini Alex
Incicco Emanuele

A.A 2021/2022

Indice

1 Introduzione Qlik, Tableau e Power BI	13
1.1 Tools	14
1.2 Dataset	14
1.3 Descrizione Campi	14
2 Qlik	17
2.1 Manipolazione e Pre-Processing dei Dati	17
2.2 Funzioni Utilizzate	17
2.3 Note Analisi	19
2.4 Analisi Temporale	20
2.5 Analisi produttività	22
2.6 Analisi Categorie	24
2.7 Analisi Longevità	26
2.8 Analisi Requisiti e Dimensioni	27
2.9 Analisi Costi	28
2.10 Analisi Varie	29
3 Tableau	31
3.1 Misure	31
3.2 Analisi Temporale	33
3.3 Analisi produttività	35
3.4 Analisi Categoria	38
3.5 Analisi Longevità	40
3.6 Analisi Requisiti e Dimensioni	41
3.7 Analisi Costi	42
3.8 Box & Whisker	43
3.9 Analisi Varie	45
4 Power BI	47
4.1 Misure & Pre-processing	47
4.2 Analisi Temporale	49
4.3 Analisi Produttività	51
4.4 Analisi Categoria	53
4.5 Analisi Longevità	56
4.6 Analisi Costi	57
4.7 Box & Whiskers	58
4.8 Analisi Requisiti e Dimensioni	60
4.9 Analisi Varie	61
4.10 Utilizzo di R in PowerBI	62

4.11 ARIMA	64
4.12 Analisi Correlazioni	67
5 Conclusioni Qlik, Tableau e Power BI	69
6 Python	71
6.1 Serie Temporali	71
6.1.1 Dataset	71
6.1.2 Visualizzazione dei Dati	73
6.1.3 Correlazioni	77
6.1.4 StatsModels	81
6.1.5 Arima	82
6.1.6 SARIMA	84
6.1.7 Out-of-time cross validation	85
6.2 Cluster	87
6.2.1 Dataset	87
6.2.2 K-Means Clustering	88
6.2.3 Hierarchical Clustering	89
6.2.4 Clustering Price - Surface	89
6.2.5 Cluster All	90
6.2.6 DBSCAN	92
6.3 Classificazione	95
6.3.1 Dataset	95
6.3.2 Struttura Dataset	95
6.3.3 Visualizzazione del dataset di partenza	96
6.3.4 Pre-processing - Fase 3	96
6.3.5 Correlazione	97
6.3.6 Classificatori	97
6.3.7 Addestramento dei classificatori	100
6.3.8 Classificazione con GridSearch	102
6.3.9 Model Ensemble	103
7 Social Network Analysis	107
7.1 Dataset	107
7.2 ETL	108
7.3 Rappresentazione	108
7.4 Caratteristiche della rete	109
7.5 Degree Centrality	109
7.5.1 Layouts di Degree Centrality	109
7.5.2 Eigenvector Centrality	111
7.5.3 Closeness Centrality	112
7.5.4 Current flow closeness Centrality	114
7.5.5 Betweennes Centrality	115
7.5.6 Load Centrality	116
7.5.7 Harmonic centrality	117
7.5.8 Confronto "Top Five" Nodi	118
7.5.9 Dispersion	118
7.5.10 Vote Rank	118
7.6 Clique, K-core e K-plex	119
7.6.1 Clique	119

7.6.2	K-Core	120
7.6.3	K-Shell	121
7.6.4	K Crust	122
7.7	K-Corona	123
7.7.1	K-Truss	124
7.8	Ego Networks, Triadi e Articulation Points (Cut-Vertices)	125
7.8.1	Ego Networks	125
7.8.2	Triadi	126
7.8.3	Articulation Points (Cut-Vertices)	126
7.9	Community Detection	127
7.10	Link Prediction	130
7.10.1	Jaccard Coefficient	131
7.10.2	Resource Allocation Index	132
7.10.3	Triadic closure	133
7.10.4	Adamic Adar Index	134
7.10.5	Preferential Attachment	134
7.10.6	Tempistiche	135
8	Chatbot	137
8.1	Tecnologie Utilizzate	137
8.2	Framework Rasa	138
8.2.1	NLU	139
8.2.2	Slots	140
8.2.3	Rules	140
8.2.4	Story	140
8.2.5	Action	142
8.2.6	Pipeline	142
8.2.7	Integrazione con Telegram	143
8.3	Funzionalità Aggiuntive	144
8.3.1	Test Stories	144
8.3.2	File CSV	146
8.4	Il nostro FoodOrderingBot	147
8.4.1	Help Command	147
8.4.2	Suggerimento Piatto	147
8.4.3	Intolleranza	148
8.4.4	Tipologia di dieta	148
8.4.5	Ordine	149
8.4.6	Suggerire una modifica al menu	150
8.4.7	Richiesta suggerimento bevanda	150
8.4.8	Informazione apporto calorico	150
8.4.9	Show Entire Menu	150

Elenco delle figure

1.1	Magic Quadrant Gartner	13
2.1	Logo Qlik	17
2.2	Funzione Make Date	18
2.3	Funzione Condizionale if	18
2.4	Funzione di verifica e pulizia	18
2.5	Funzione di verifica e pulizia	18
2.6	Editor Caricamento Dati	19
2.7	Over Time Analysis	20
2.8	Analisi produttività	22
2.9	Analisi delle diverse Categorie	24
2.10	Radar Plot per Categorie su base annua	24
2.11	Analisi Longevità	26
2.12	Analisi Requisiti e Dimensioni	27
2.13	Analisi Costi	28
2.14	Ulteriori Analisi	29
3.1	Logo Tableau	31
3.2	Interfaccia Grafica per la creazione di misure ad Hoc	31
3.3	Misura per il calcolo della dimensione delle Apps	32
3.4	Misura per il calcolo del rating	32
3.5	Misura per il calcolo del prezzo delle Apps	32
3.6	Overtime Analysis	33
3.7	Analisi produttività	36
3.8	Analisi produttività pt.2	37
3.9	Analisi delle diverse Categorie	38
3.10	Analisi delle diverse Categorie pt.2	39
3.11	Analisi Longevità	40
3.12	Analisi Requisiti e Dimensioni	41
3.13	Analisi Costi	42
3.14	Box & Whisker	43
3.15	Ulteriori Analisi	45
4.1	Logo PowerBI	47
4.2	Overtime Analysis	49
4.3	Analisi produttività	51
4.4	Analisi Categoria	53
4.5	Analisi Categoria (utilizzo)	54
4.6	Analisi Categoria (sviluppo)	54
4.7	Analisi Longevità	56

4.8 Analisi Costi	57
4.9 Box & Whisker	58
4.10 Analisi Requisiti e Dimensioni	60
4.11 Various Analysis	61
4.12 R Code	62
4.13 Bar Plot sviluppato in R	62
4.14 Analisi con ARIMA	64
4.15 Analisi con ARIMA pt.2	65
4.16 Dettagli Analisi con ARIMA	65
4.17 Previsione 2021-2024	66
4.18 Correlazioni	67
6.1 Logo Python	71
6.2 WordCloud Plot	72
6.3 Utilizzo di dtypes	72
6.4 Dataset post-preprocessing	72
6.5 Parametri Meteorologici di Tocumen, Santiago, David	74
6.6 Carico Elettrico della repubblica Panamense	74
6.7 Carico Elettrico su base mensile	75
6.8 Carico Elettrico su base settimanale	75
6.9 Carico Elettrico su base oraria	76
6.10 PairPlot relativo alla città di Tocumen	77
6.11 PairPlot relativo alla città di Santiago e David	77
6.12 Analisi della corellazione tramite il metodo CorrCoef	78
6.13 Scatter Plot Temperatura e Net Demands	79
6.14 Scatter Plot per Precipitazioni e Net Demands	79
6.15 Lowess Plot per Temperatura e Net Demands	79
6.16 Plot di Autocorrelazione e derivate successive	81
6.17 Plot di Partial Auto Correlation	82
6.18 Arima Models & Results	83
6.19 <i>Plotting</i> dei Residui	83
6.20 Forecasting con SARIMAX	84
6.21 <i>Plotting</i> dei Residui	85
6.22 SARIMA with out-of-time cross validation	86
6.23 Elbow Method	88
6.24 Clustering Price - Location	89
6.25 Hierarchical Clustering	90
6.26 Elbow Method	90
6.27 Clustering Price - Surface	91
6.28 Elbow Method	91
6.29 Clustering	92
6.30 Silhouette	93
6.31 Nearest Neighbors	93
6.32 Clustering DBSCAN	94
6.33 Estrazione delle istanze per il set di Train	96
6.34 Dataset Immobiliare post-preprocessing	97
6.35 Codificazione dei range per Features Price, Area, Kitchen_area	98
6.36 Database complessivo finale	98
6.37 Matrice di correlazione delle varie features	99
6.38 Barplot dei risultati	101

6.39 Roc-Curve RF-Classifier	101
6.40 CM RF-Classifier	102
6.41 GridSearch Parameters RF-Classifier	103
6.42 CM - Test Ensemble Model	104
7.1 Logo NetworkX	107
7.2 Social Network Plot	108
7.3 Degree Centrality	110
7.4 Spring Degree Centrality Layout	110
7.5 Spiral Degree Centrality Layout	110
7.6 Fruchterman Reingold Degree Centrality Layout	111
7.7 Kamada Kawai Degree Centrality Layout	111
7.8 Eigenvector Centrality Distribution Plot	112
7.9 Closeness Centrality Distribution Plot	113
7.10 Current flow closeness Centrality Distribution Plot	114
7.11 Betweenness Centrality Distribution Plot	115
7.12 Harmonic Centrality Distribution Plot	117
7.13 K-Core	120
7.14 K-Shell	121
7.15 K-Crust	122
7.16 K-Corona	123
7.17 K-Truss	124
7.18 La Ego-Network Relativa al nodo 5440	125
7.19 Le principali Comunità all'interno della Network	128
7.20 Kamada Kawai Layout con le diverse comunità	129
7.21 Comunità 27	130
7.22 Comunità 27 con archi rimossi	131
7.23 Tempi di esecuzione (s) dei diversi algoritmi	135
8.1 Framework Rasa	138
8.2 Framework Rasa "Actions"	138
8.3 Framework Rasa "Data"	139
8.4 Intent NLU	139
8.5 Lookup Table	139
8.6 Slots	140
8.7 Rules	140
8.8 Story's	141
8.9 Actions	142
8.10 RegexEntityExtractor	142
8.11 NGROK	143
8.12 Chat Telegram	144
8.13 Esempio di Test Story	145
8.14 Intent Confusion Matrix	145
8.15 Intent Prediction Confidence Distribution	146

Listings

7.1	ETL per creazione del grafo	108
7.2	Funzioni utili per la presentazioni del grafo	108
7.3	Implementazione della load centrality relativa al nodo 5440 con un cutoff pari a 3	116
7.4	Implementazione del calcolo della dispersion relativa al nodo 0 e il nodo 4924 . .	118
7.5	Implementazione dell'algoritmo di Vote Rank	118
7.6	Funzioni di NetworkX per le trovare le informazioni principali	119
7.7	Funzioni di NetworkX per le trovare le Clique	119
7.8	Funzioni di NetworkX per le K-core	120
7.9	Funzione K-Shell	121
7.10	Funzione K-Crust	122
7.11	Funzione K-Corona	123
7.12	Funzione K-Truss	124
7.13	Implementazione del Girvan-Newman algorithm	127
7.14	Algoritmo di previsione basato sul Jaccard Coefficient	131
7.15	Algoritmo di previsione basato sull'indice di allocazione delle risorse	132
7.16	Algoritmo di previsione "Triadic Closure"	133
8.1	comando ngrok per attivare il tunnel	143
8.2	configurazione integrazione con telegram	143

Capitolo 1

Introduzione Qlik, Tableau e Power BI

Il progetto ha come obiettivo cardine l'apprendimento di diversi applicativi leader di mercato per la **analytics** e **business intelligence**. Questi applicativi vengono riportati nel *Magic Quadrant* di Gartner seguente, distribuendole per forza e importanza. In questa tesina andiamo quindi a rappresentare il funzionamento e le possibili applicazioni dei principali tools: *Power Bi*, *QLik*, *Tableau*.



Figura 1.1: Magic Quadrant Gartner

1.1 Tools

QLik Sense Uno dei software analizzati in questo progetto è **Qlik Sense**, un prodotto di visualizzazione e rilevazione dati che consente di creare visualizzazioni flessibili e interattive. Qlik Sense fa parte dell' "ecosistema" Qlik, uno dei principali strumenti di Business Intelligence, che permette di effettuare analisi Descrittiva e Diagnostica.

Tableau Un'ulteriore software appreso durante questo studio è **Tableau**, il quale analogamente a *QLik* ci permette di svolgere le medesime analisi, introducendo però l'opzione aggiuntiva relativa all'analisi Predittiva di specifiche features che cambiano temporalmente.

Microsoft PowerBI L'ultimo software utilizzato è **PowerBI**, dove in aggiunta alle analisi possibili tramite i tool precedenti, qui è possibile svolgere ulteriori studi sia predittivi che prescrittivi. Inoltre, tramite questo applicativo, è possibile utilizzare diversi oggetti visivi tramite linguaggi di programmazione come Python e R.

1.2 Dataset

Il progetto è stato realizzato utilizzando il Dataset ***Google Play Store Apps***¹ il quale riporta le maggiori informazioni riguardo allo sviluppo, la pubblicazione e l'utilizzo di oltre due milioni di applicazioni distinte. Il dataset è stato scelto e scaricato da *Kaggle* e contiene oltre 23 attributi. Questi verranno descritti successivamente nella tabella Tab. 1.1. Il dataset è composto da un unico file CSV, il quale risulta essere stato aggiornato con data ultima, Giugno 2021. Si è scelto questo dataset per visualizzare come l' "evoluzione" tecnologica ha influenzato lo sviluppo delle applicazioni nell'ecosistema **Google Inc.**, con riferimento al decennio 2010-2021.

1.3 Descrizione Campi

Nella tabella Tab. 1.1 riportiamo tutte le colonne presenti nel dataset scelto, con una breve descrizione per ciascun campo al fine di comprendere appieno il significato di tali campi.

¹Link al dataset, <https://www.kaggle.com/gauthamp10/google-playstore-apps>

N°	Nome Colonna	Descrizione
1	App Name	Nome applicazione
2	App Id	Id relativo all'applicazione
3	Category	Categoria dell'applicazione
4	Rating	Valore medio recensioni
5	Rating Count	N° di recensioni
6	Installs	N° di installazioni approssimativo
7	Minimum Installs	N° minimo di installazioni
8	Maximum Installs	N° massimo di installazioni
9	Free	Applicazione gratuita (True/False)
10	Price	Prezzo
11	Currency	Valuta
12	Size	Dimensione dell'applicazione (MB o KB)
13	Minimum Android	Versione minima di Android richiesta
14	Developer Id	Id sviluppatore
15	Developer Website	Sito web sviluppatore
16	Developer Email	E-mail sviluppatore
17	Released	Data di rilascio
18	Privacy Policy	Policy per la privacy
19	Last Updated	Data ultimo aggiornamento
20	Content Rating	Classificazione contenuti
21	Ad Supported	Pubblicità all'interno dell'applicazione (True/False)
22	In app purchases	Acquisti all'interno dell'applicazione (True/False)
23	Editor Choice	Se valutata come "Scelta Dell'Editore"
24	Scraped Time	Data Import Valori

Tabella 1.1: Descrizione Campi

Capitolo 2

Qlik

In questo capitolo andiamo ad introdurre tutto ciò che è inherente allo studio del dataset presentato precedentemente, in particolare andremo a descrivere l'utilizzo del tool **Qlik**. Ecco quindi che oltre ai primi step di manipolazione e pre-processing, andremo ad osservare le varie analisi svolte ponendo attenzione su ciò che esse descrivono.



Figura 2.1: Logo Qlik

2.1 Manipolazione e Pre-Processing dei Dati

Essendo, all'interno del dataset, presenti campi con tipologie di dato differenti si è reso necessario un pre-processing dei dati. Al fine di apprendere al meglio il software Qlik, si è deciso di realizzare questo "pre-processing" al suo interno, impostando delle funzioni che estraessero le informazioni nel formato a noi necessario, senza utilizzare software terzi per realizzare questo processo in fase di importazione dei dati.

2.2 Funzioni Utilizzate

Make Date Nella figura Fig. 2.2 viene riportata la prima funzione utilizzata. Attraverso le funzioni **SubField** e **MakeDate** siamo riusciti a convertire la data dal formato iniziale "May 20,2020" al formato finale "20/05/2020". Tra le altre funzioni utilizzate c'è la **Year**, la quale ci ha permesso di estrarre l'anno da una specifica data.

Istruzione Condizionale Un'ultima funzione adottata al fine di fornire una più chiara visualizzazione è stata creata tramite il costruttivo **IF**.

```

1 =MakeDate(SubField(Released, ',', 2),
2   Num(Month(Date#(SubField(SubField(Released,',',1),',',1), 'MMM'))),
3   SubField(SubField(Released,',',1),',',2))

```

Figura 2.2: Funzione Make Date

Replace La funzione **Replace** ci ha permesso di convertire i numeri decimali, rappresentati con il punto, in numeri decimali con la virgola, permettendo così una rappresentazione uniforme del formato.

```

1 =if(Replace(Price,'.',',')>= 0 and Replace(Price,'.',',')<1, '<1$',
2 if(Replace(Price,'.',',')>= 1 and Replace(Price,'.',',')<2, '1;1,99$', 
3 ...
4 if(Replace(Price,'.',',')>= 20 and Replace(Price,'.',',')<50, 8, '20; 49,99 ))))
      ))))

```

Figura 2.3: Funzione Condizionale if

Istruzione di verifica rating Nella figura Fig. 2.4 si ha l'ultima funzione utilizzata. Questa ci ha permesso di calcolare la media dei valori di rating delle applicazioni. Da notare la condizione settata che la lunghezza del campo Rating debba essere uguale a 3 (i valori di rating spaziano da 0,00 a 5,00), in questo modo filtriemo soltanto i valori di interesse. Un'altra condizione da notare è l'esclusione del valore 0,00. Si vuole infatti calcolare la media dei valori di rating fra i soli rating non nulli. Infine, si è utilizzata la funzione di Replace per far in modo di convertire la stringa in numero decimale e rendere possibile il calcolo della media.

```

1 =Avg ({<Rating = {'=len(trim(Rating))=3'}, Rating=-{'0.0'}>} Replace(Rating, '.', ','))

```

Figura 2.4: Funzione di verifica e pulizia

Istruzione per processing sull'attributo Size Nella figura Fig.2.5 siamo andati ad analizzare e filtrare i valori, normalizzandoli al fine di esprimere le misure successive secondo il medesimo ordine di grandezza. In particolare, abbiamo estratto i valori di size diversi da 0, poi, essendoci

```

1 =AVG({<Size=-{'0'}>} NUM(
2 if(FindOneOf(Size, 'M')>1, NUM(replace(SubField(Size,'M',1),'.',',',)),
3 if(FindOneOf(Size, 'k')>1, NUM(replace(SubField(Size,'k',1),'.',',','))/1000,
4 if(FindOneOf(Size, 'G')>1, NUM(replace(SubField(Size,'G',1),'.',',','))*1000,
5 NUM(replace(Size,'.',','))/1000
6 )))

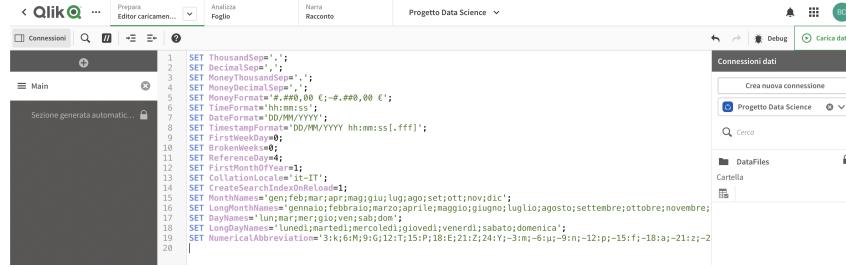
```

Figura 2.5: Funzione di verifica e pulizia

valori espressi con unità di misura differenti, abbiamo riportato tutti i valori all'unità di misura MB. Infine, una volta convertito poi il valore in numero decimale si è calcolata la media.

2.3 Note Analisi

Disclaimer Anziché utilizzare le formule sopra descritte avremmo potuto agire a livello di *pre-processing* impostando un nuovo tipo di formato di dato nella schermata **editor caricamento dati**, visibile nella Figura Fig. 2.6. Abbiamo però deciso di utilizzare le formule per approfondire la conoscenza di queste ultime e seguire quanto fatto durante le esercitazioni.



```

< Qlik > ... Prepara Editor caricamen... Analizza Foglio Molla Racconto Progetto Data Science
Conessioni | Q | W | E | ? | Connessioni dati
Main
Sezione generata automaticamente
1. SET ThousandSep(',');
2. SET DecimalSep(',');
3. SET DateThousandSep(',');
4. SET MoneyDecimalSep(',');
5. SET MoneyFormat="#,##0,00 C-#,##0,00 €";
6. SET DateFormat="DD/MM/YYYY";
7. SET DataFormat="DD/MM/YYYY hh:mm:ss,ffff";
8. SET TimestampFormat="DD/MM/YYYY hh:mm:ss,ffff";
9. SET FirstWeekday=0;
10. SET ReferenceDay=0;
11. SET ReferenceDay=4;
12. SET FirstMonthOfYear=1;
13. SET MonthNames="gen;feb;mar;apr;mag;giu;lug;ago;set;ott;nov;dic";
14. SET CreateSearchIndexOnReload=1;
15. SET MonthNames="gen;feb;mar;apr;mag;giu;lug;ago;set;ott;nov;dic";
16. SET LongMonthNames="gennaio;febbraio;märz;april;mai;giugno;luglio;agosto;settembre;ottobre;novembre";
17. SET DayNames="lunedì;martedì;mercoledì;giovedì;venerdì;domenica";
18. SET LongDayNames="lunedì;martedì;mercoledì;giovedì;venerdì;domenica";
19. SET NumericalAbbreviation="3;k;6;M;9;G;12;T;15;P;18;E;21;z;24;Y;-3:m;-6;j;-9;n;-12;p;-15:f;-18;a;-21;z;-2"
|
```

Figura 2.6: Editor Caricamento Dati

Note Numero di Installazioni Le analisi che utilizzano il numero di installazioni fanno uso della colonna del dataset *Maximum Installs*. Utilizzando tale stima le analisi saranno quindi approssimate per eccesso.

2.4 Analisi Temporale

In questa sezione, abbiamo rappresentato l'analisi temporale effettuata nel nostro dataset. Abbiamo raffigurato come si è evoluto il settore dello sviluppo delle applicazioni assieme al loro utilizzo nel corso del decennio preso in analisi. Il tutto è rappresentato nella figura Fig.2.7. Le analisi in questione sono:

- N° di installazioni nel decennio 2010 – 2021
- Dimensione media delle applicazioni nel decennio 2010 – 2021
- N° di applicazioni sviluppate nel decennio 2010 – 2021
- N° di applicazioni aggiornate nel decennio 2010 – 2021

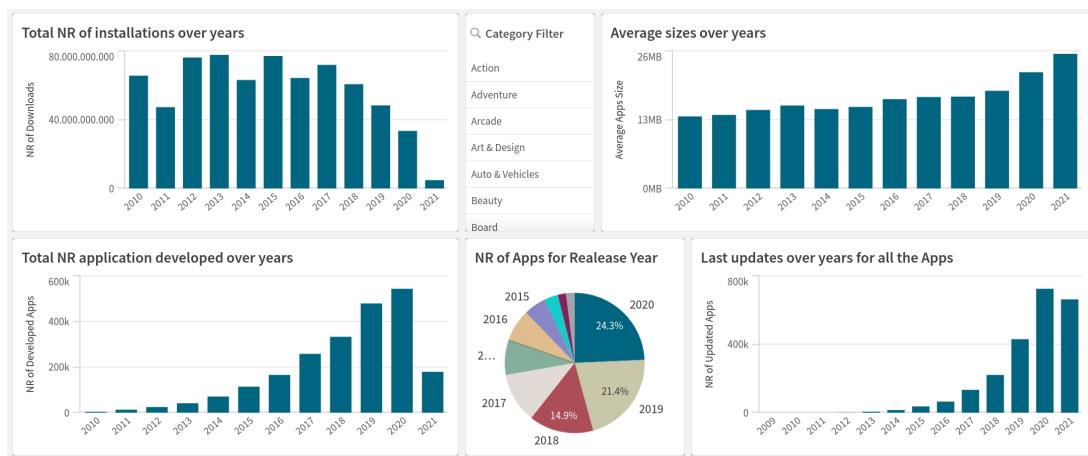


Figura 2.7: Over Time Analysis

Osservazioni

Osservando la figura precedente (Fig. 2.7) una prima considerazione può essere fatta guardando l'istogramma che riporta il numero di applicazioni sviluppate per ciascun anno di rilascio. Si ha infatti che con l'aumentare degli anni aumentano anche il numero di applicazioni che vengono rilasciate, un incremento significativo lo si ha avuto nel 2019 e 2020, questo potrebbe essere dovuto anche al Covid-19 che ha reso necessario sviluppare applicazioni per fornire i servizi che inizialmente erano offerti *vis-à-vis*. Il fatto che per il 2021 si abbia un numero relativamente basso di applicazioni sviluppate dipende dal fatto che il database utilizzato, come detto precedentemente, è stato aggiornato l'ultima volta a metà del 2021, quindi contiene dati incompleti per tale anno.

Osservando l'istogramma con il numero di installazioni per ciascun anno di rilascio delle applicazioni possiamo osservare che i valori oscillano di poco, ad eccezioni degli ultimi anni per cui, essendo le applicazioni più recenti, si ha, per ora, un numero relativamente basso di installazioni. Guardando l'analisi sul numero di applicazioni che hanno avuto un ultimo aggiornamento nel corso degli anni (istogramma in basso a destra), possiamo osservare che un numero rilevante di applicazioni sono state aggiornate l'ultima volta fra il 2019 e il 2021. Rimane una piccola parte di applicazioni che risultano aggiornate per l'ultima volta prima del 2019, confermando il fatto che ci sono sempre applicazioni la cui vita termina perché hanno avuto poco successo nel mercato. Dal grafico a torta possiamo prendere atto delle porzioni di applicazioni per ciascun anno di

rilascio sul numero totale di applicazioni analizzate.

Mentre per quanto riguarda la dimensione media delle applicazioni è osservabile come negli ultimi anni, abbiamo un trend positivo e questo è giustificabile dal fatto che diverse categorie di applicazioni, in particolare i *video games*, avendo a disposizione più *storage*, hanno innovato sempre di più, senza porre particolare attenzione all'aspetto del costo computazionale e al peso in termini di spazio di memoria richiesto.

2.5 Analisi produttività

Nell'analisi della produttività abbiamo analizzato il comportamento dei vari sviluppatori, in particolare oltre all'aspetto economico abbiamo deciso di analizzare anche la loro produttività espressa sia in termini di longevità delle loro applicazioni che in termini di numero di download delle medesime. Il tutto è osservabile nella figura Fig.2.8.

Note utili Attraverso il dataset è osservabile come ciascun singolo sviluppatore può anche aver messo sul mercato applicazioni in anni diversi e quindi non necessariamente sempre presenti nel corso del decennio.

La stima dei profitti è stata realizzata considerando il prezzo di acquisto delle app, escludendo quindi gli acquisti in-app, per ciascun sviluppatore.

La longevità degli sviluppatori è espressa come la massima durata di vita fra le varie durate di vita delle applicazioni da loro sviluppate. Tale valore è la differenza fra la data di ultimo aggiornamento dell'applicazione e la data di rilascio.

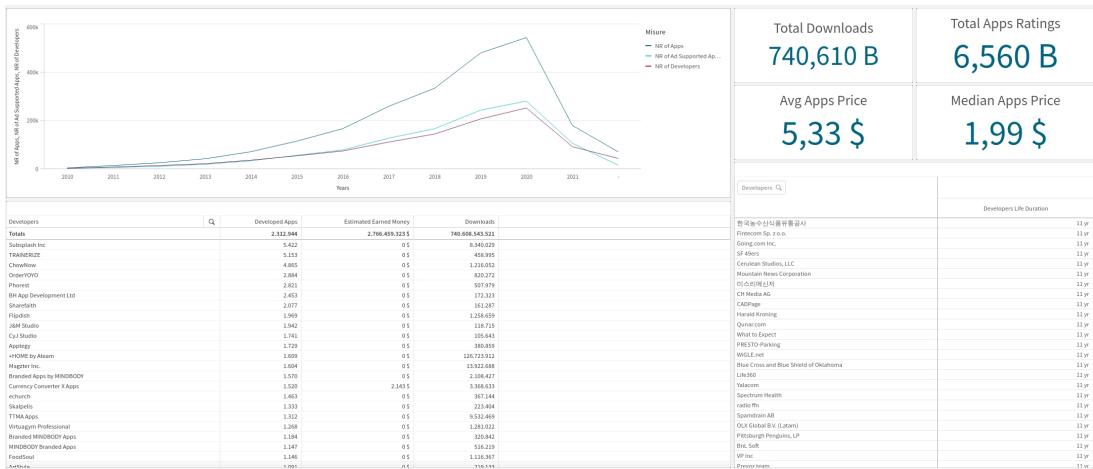


Figura 2.8: Analisi produttività

Osservazioni

Osservando la figura Fig.2.8, possiamo concludere che il numero di sviluppatori e, di conseguenza, gli indicatori relativi al numero di applicazioni sviluppate hanno avuto un notevole incremento nel corso degli anni, eccetto il dato relativo all'anno 2021, per ragioni già esplicate in precedenza. I KPI si possono riassumere nell'affermazione che gran parte delle applicazioni contenute nel dataset sono free e assai recensite.

Dalla tabella a sinistra, osservando gli sviluppatori che hanno presumibilmente realizzato maggior profitto, possiamo affermare che hanno realizzato poche applicazioni. Questo ci fa pensare quindi che è fondamentale l'originalità dell'idea derivante dalla fantasia dello sviluppatore per la commercializzazione di un app di successo piuttosto che la quantità di applicazioni mediocri realizzate.

Mentre nella tabella a destra, è possibile visionare come gli sviluppatori cercano di rendere le loro applicazioni più longeve, così da avere profitti marginali più bassi nel breve periodo, ma che nel lungo portano conoscenza ed introiti più elevati. Si possono infatti osservare valori di durata di vita delle applicazioni elevati, come 11 anni. Questa longevità è comunque generata dal successo che esse hanno ottenuto dalla loro entrata nello Store. Infine, il grafico a linee dimostra come l'andamento del numero di applicazioni sviluppate sia direttamente proporzionale al numero di

produttori attivi in un certo periodo, con coefficiente di crescita molto più elevato, possibile sintomo che, in periodi di attività, gli sviluppatori possano mettere mano anche su molteplici applicazioni contemporaneamente.

2.6 Analisi Categorie

Nello studio relativo alle categorie siamo andati ad analizzare quali sono le categorie maggiormente trainanti, sia in termini di applicazioni free che paid apps, sia considerando altri aspetti di monetizzazione come le pubblicità e gli in-App Purchases, tenendo come focus principale appunto le categorie. Il tutto è visibile nella figura Fig. 2.9. Mentre nella figura Fig. 2.10, abbiamo posto l'attenzione su come, nel corso degli anni, diverse categorie di applicazioni hanno preso il sopravvento su altre.

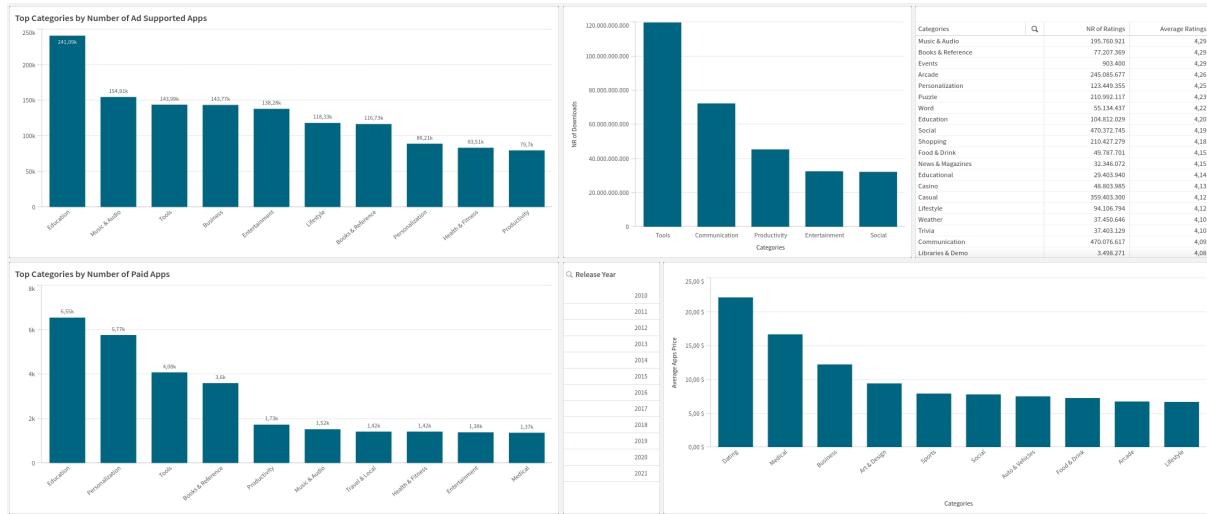


Figura 2.9: Analisi delle diverse Categorie

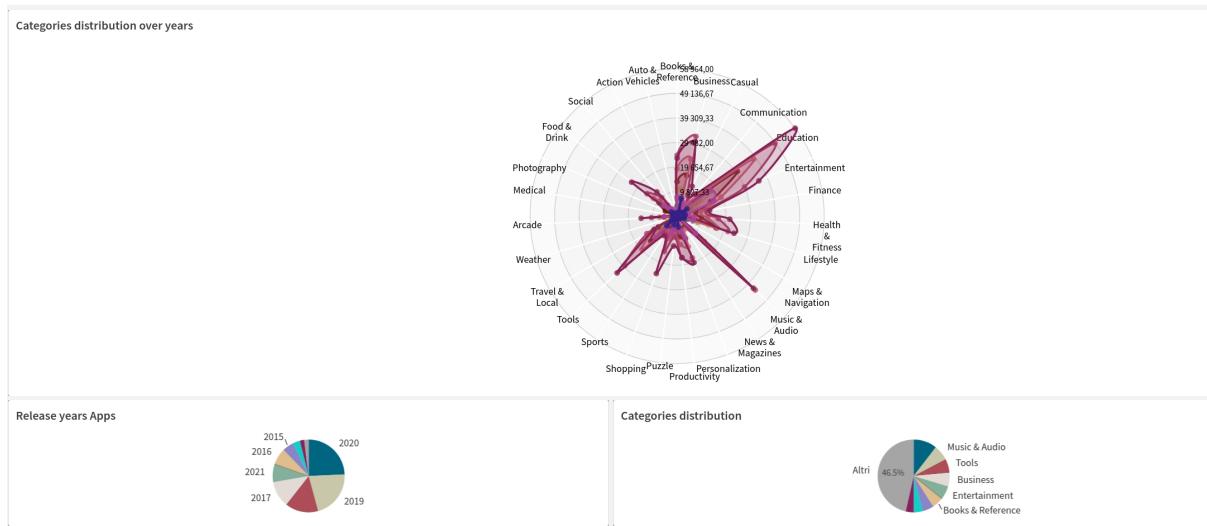


Figura 2.10: Radar Plot per Categorie su base annua

Osservazioni

Dalla figura Fig.2.10 si nota come i settori maggiormente trainanti nel corso degli anni sono stati: *Travel*, *Photography* (grazie al continuo miglioramento del comparto fotografico dei diversi smartphone presenti nel mercato), *Educational*, il quale anche nel corso del 2020/2021 ha avuto

un notevole incremento per motivi ormai noti come la didattica digitale, etc...

Infine, dalla figura Fig. 2.9 è possibile osservare che le categorie principali per numero di applicazioni contenenti pubblicità sono *Education*, *Music & Audio*, *Tools*. Mentre le categorie principali per numero di applicazioni a pagamento sono *Education*, *Personalization*, *Tools*.

Per quanto riguarda invece il prezzo medio più alto, le categorie citate precedentemente vengono surclassate da categorie come *Dating*, *Medical*, *Business*, *Art & Design*, *Sports*. Infine, per quanto concerne il numero di *downloads*, abbiamo una notevole disparità tra la prima categoria, ovvero i *Tools* e le successive categorie.

Per concludere, la tabella ci permette di osservare come le categorie con valor medio di recensioni più alto sono *Music & Audio*, *Book & Reference*, *Events*, mentre quelle più recensite sono *Action*, *Tools*, *Social*, *Communication*.

2.7 Analisi Longevità

Nella seguente analisi, figura Fig. 2.11, abbiamo studiato come le diverse applicazioni, presenti all'interno del *dataset*, vengono manutenute dai vari sviluppatori.

Note utili Longevità \vdash Differenza (espressa in anni) tra la data di ultimo aggiornamento e la data di rilascio dell'applicazione. All'interno del *bar chart* sono riportate le 10 categorie principali che posseggono le applicazioni con la media di longevità più elevata, a fianco di ciascuna categoria abbiamo riportato anche il valore di *rating* medio.

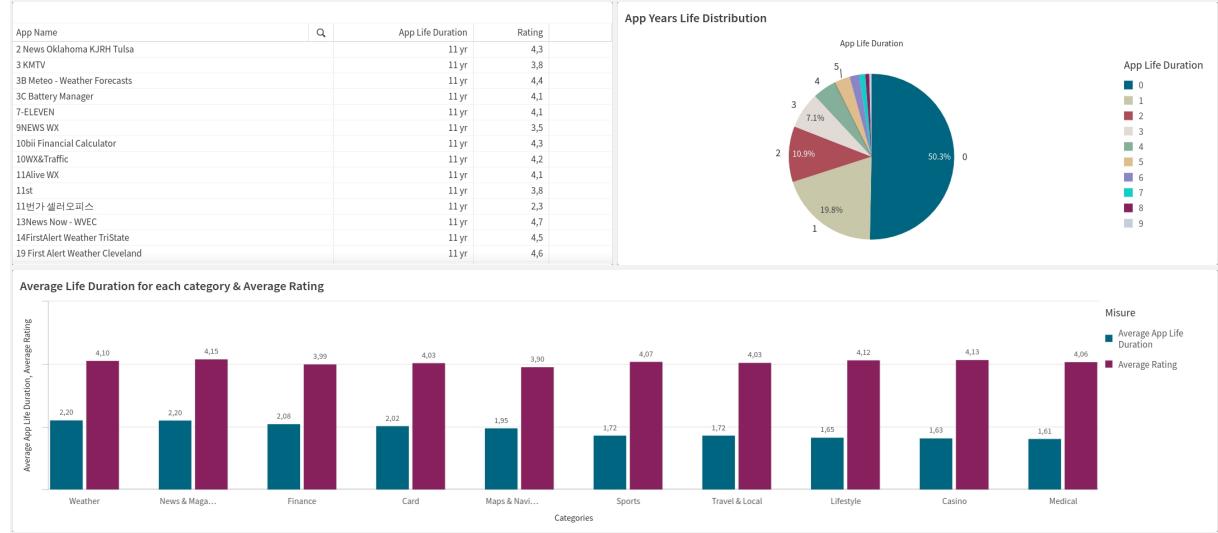


Figura 2.11: Analisi Longevità

Osservazioni

Osservando la figura Fig. 2.11 possiamo individuare due principali informazioni estratte dal *dataset*. La prima può essere osservata dal *pie chart*: escludendo, dalla fetta con la durata degli anni pari a 0, ovvero il numero di applicazioni rilasciate nel 2021, il numero residuo che spicca maggiormente rappresenta le applicazioni che sono state "abbandonate" dai rispettivi *developers* entro un anno dal loro rilascio sul *Play Store* di *Google*, quindi non più aggiornate oltre lo stesso anno del loro rilascio. Come si può osservare tale numero è molto elevato, questo conferma il fatto che soltanto una piccola porzione di applicazioni rilasciate riescono a sopravvivere, molte invece muoiono entro un anno dal loro rilascio.

L'altra informazione ricavata è che le categorie di applicazioni in media più longeve sono quelle appartenenti alle categorie *Weather*, *News & Magazines*, indice che le applicazioni provengono da potenze del mercato ormai consolidate anche in televisione, e che si sono espansse fino al campo degli applicativi.

Nella tabella in alto si possono osservare i valori di durata di vita e il rating medio delle singole applicazioni presenti nel dataset.

2.8 Analisi Requisiti e Dimensioni

Nella figura Fig. 2.12 abbiamo riportato le *Minimum Android Version Analysis*.

In particolare abbiamo posto attenzione su uno degli aspetti cardine dello sviluppo di applicazioni, il sistema minimo di *Android* supportato. Abbiamo quindi svolto tali analisi tenendo in considerazione la versione minima di *Android* necessaria per installare, e quindi utilizzare, le applicazioni.

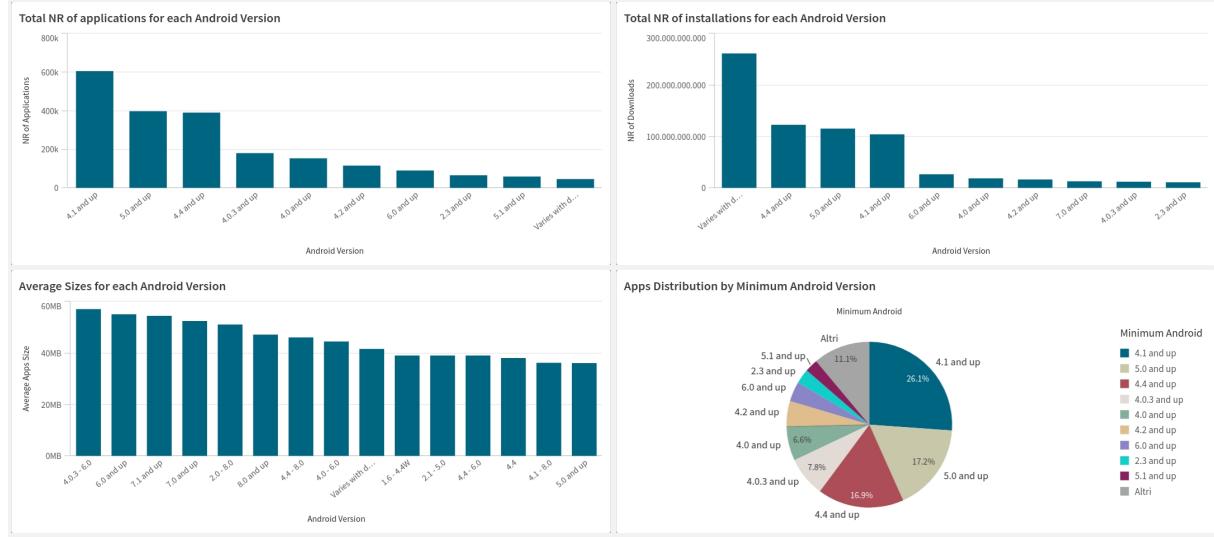


Figura 2.12: Analisi Requisiti e Dimensioni

Osservazioni

Come possiamo osservare dalla figura Fig. 2.12 la maggior parte delle applicazioni richiedono come versione minima *Android 4.1*. Il motivo di questa scelta potrebbe essere quello di un trade-off fra il cercare di coprire un numero quanto più elevato possibile di dispositivi senza rinunciare però alle feature messe a disposizione dai nuovi sistemi operativi *Android*.

Nella parte sottostante della figura troviamo l'analisi relativa a come varia la dimensione media delle diverse applicazioni per ciascuna versione minima di sistema operativo richiesta. Questa analisi potrebbe risultare in contrapposizione a quella svolta precedentemente (sezione 2.4), ma, analizzando attentamente le etichette, si vede che per i sistemi operativi *Android 7*, *Android 6* e *Android 8*, ovvero gli ultimi sviluppati, si conferma il fatto che nel corso degli anni l'aspetto di ottimizzazione di memoria è venuta leggermente meno a fronte di hardware sempre più performante e disponibilità di storage molto più elevate.

2.9 Analisi Costi

Nella seguente analisi, figura Fig.2.13, abbiamo intrapreso quella che è un'indagine sul mercato delle applicazioni, considerando diversi aspetti, come: il costo delle applicazioni, la possibilità o meno di fare acquisti, la presenza o meno di annunci all'interno delle applicazioni e il guadagno stimato per ogni categoria di applicazioni (considerando il costo e il numero di installazioni).

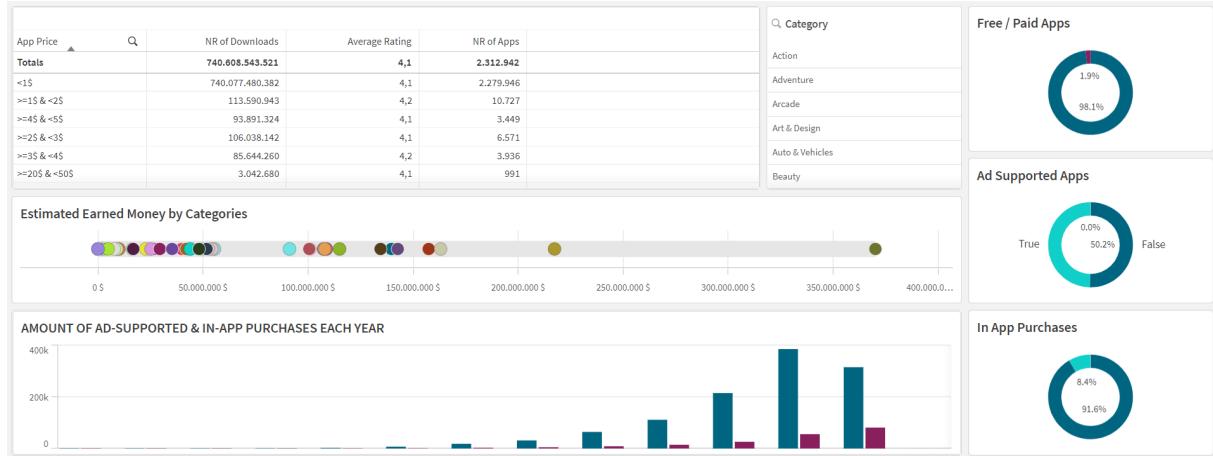


Figura 2.13: Analisi Costi

Osservazioni

Come si può osservare dai vari pie-chart a lato, sono prevalentemente presenti applicazioni free, con una minoranza di applicazioni a pagamento. Questo indica che da un lato l'utente ha la possibilità di scegliere applicazioni free, dall'altro canto, diverse applicazioni risultano dedicate a nicchie di mercato, le quali sono disposte a pagare per ottenere funzionalità aggiuntive. Sempre grazie ai pie-chart è emerso come è opportuno evidenziare il numero di applicazioni che supportano o meno gli annunci pubblicitari; infatti si può vedere che non c'è una distinzione netta, anzi le due percentuali sono pressoché uguali. L'ulteriore aspetto emerso è che il numero di applicazioni che non supportano gli acquisti *in-app* è di molto superiore rispetto al numero di applicazioni che forniscono acquisti in app. L'ultima informazione legata al bar-chart posto in basso nella figura 2.13 ci permette di osservare che nel decennio 2010 – 2021 il numero di applicazioni che supportano gli annunci pubblicitari hanno avuto un trend positivo, mentre il trend legato alle applicazioni che supportano gli acquisti *in-app* è sempre positivo, ma è molto più flebile. Infine, sempre tramite la stessa tabella possiamo dedurre, dal numero di installazioni, come la maggior parte degli utenti preferisca applicazioni con prezzi bassi o nulli.

2.10 Analisi Varie

Al termine delle nostre analisi abbiamo deciso di aggiungere alcuni ulteriori studi, su parametri marginali al nostro progetto. In particolare abbiamo analizzato come le applicazioni si distribuiscono in base a **Content Rating Category**, parametro che ci permette di comprendere se le applicazioni sono utilizzabili o meno da diverse categorie di utenti.

Inoltre, è stato riportato il numero di applicazioni che risultano essere "Scelte dall'editore", ovvero un indicatore per giustificare l'ottimalità dell'applicazione.

Infine, si sono riportati i primi 10 sviluppatori per numero di applicazioni pubblicate nel *Play Store di Google* e le prime 10 applicazioni per numero di installazioni.

Abbiamo aggiunto un filtro sulle categorie per rendere le analisi, eventualmente, più esplicative. Tali analisi sono rappresentati nella figura seguente Fig.2.14.

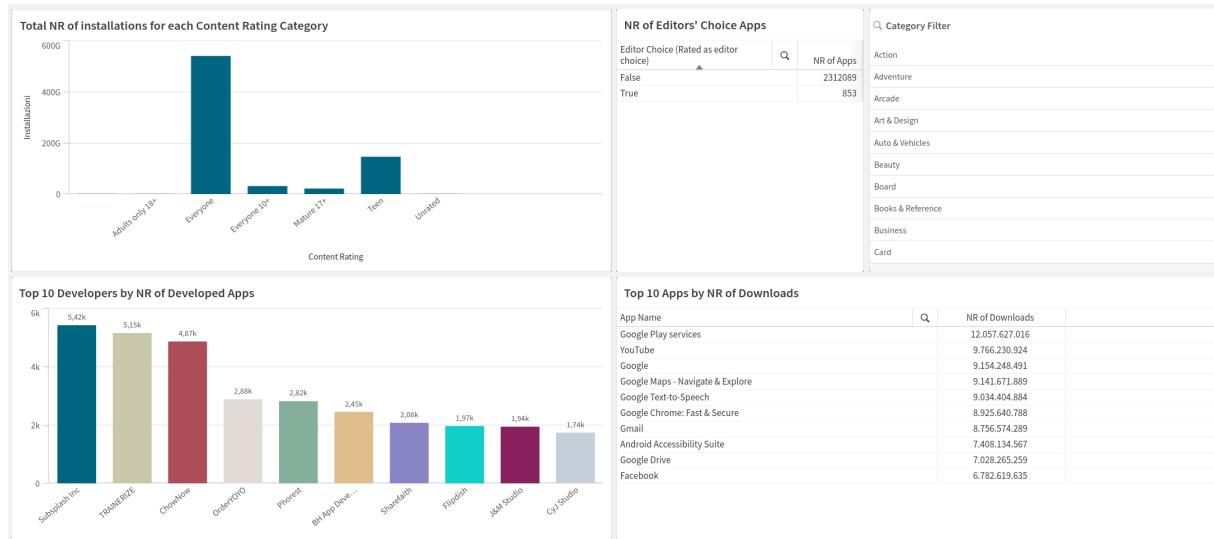


Figura 2.14: Ulteriori Analisi

Osservazioni

Guardando la figura Fig. 2.14 una prima osservazione che si può fare è che la maggior parte delle applicazioni appartengono alla categoria di *Content Rating Everyone*, sono quindi adatte ad utenti di tutte le età.

Un'altra osservazione da fare è sul numero di applicazioni che risultano essere "Scelte dagli editori", queste infatti rappresentano un numero irrisorio rispetto alla totalità di applicazioni analizzate.

Analizzando le applicazioni principali per numero di installazioni vediamo come, le applicazioni *Google* la fanno da padrone. Nelle prime posizioni troviamo infatti applicazioni come *Google Play Services*, *YouTube*, *Google*, *Google Maps*.

Analizzando invece i principali sviluppatori per numero di applicazioni rilasciate vediamo che emergono tre principali sviluppatori nell'insieme, con un numero di applicazioni sviluppate che si distacca di molto rispetto agli altri. Questi tre sviluppatori sono *Subsplash*, *Trainerize*, *ChowNow*. Cercando informazioni su tali sviluppatori si è visto che il primo sviluppa applicazioni *custom* per le chiese, il secondo sviluppa applicazioni per il *personal training*, mentre l'ultimo per il settore della ristorazione. Questa analisi dimostra quindi che i principali sviluppatori si occupano di applicazioni di categorie specifiche, che in un primo momento si potrebbe pensare non rappresentare la maggioranza, cosa invece smentita da tale analisi.

Capitolo 3

Tableau

Analogamente per quanto svolto per il tool **QLik** anche in questo capitolo vedremo come utilizzare e soprattutto come svolgere le analisi viste precedentemente andando ad aggiungere le nuove funzionalità messe a disposizione da **Tableau** come, ad esempio, le stime.



Figura 3.1: Logo Tableau

3.1 Misure

Un’ulteriore possibilità, messa a disposizione da Tableau, è quella di creare delle misure ad hoc, specificando una certa formula da cui ricavare la nuova misura. Nella figura Fig. 3.2 è visionabile l’interfaccia grafica che permette ciò.

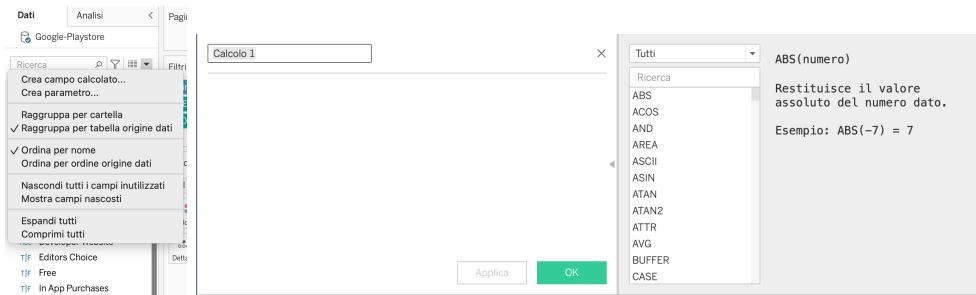


Figura 3.2: Interfaccia Grafica per la creazione di misure ad Hoc

Misura per Size Nella figura Fig.3.3 siamo andati a raffigurare la **misura** adottata tramite Tableau al fine di estrarre la dimensione delle applicazioni.

In particolare, si sono presi i valori di size diversi da 0 ed, essendoci nel dataset valori espressi con unità di misura differenti, questi sono stati convertiti tutti in MB.

Misure per il calcolo del Rating Nella figura Fig.3.4 possiamo osservare la misura utilizzata per estrarre soltanto i valori di rating non nulli.

```

1 IF NOT [Size] = '0' THEN
2   IF FIND([Size], 'M')>1 THEN FLOAT(replace(SPLIT([Size], 'M',1), '.',',',','))*
3     ELSEIF FIND([Size], 'k')>1 THEN FLOAT(replace(SPLIT([Size], 'k',1), '.',',',','))/
4       1000
5     ELSEIF FIND([Size], 'G')>1 THEN FLOAT(replace(SPLIT([Size], 'G',1), '.',',',','))*
6       1000
7     ELSE FLOAT(REPLACE([Size], '.',',',','))/1000
8   END
9 END

```

Figura 3.3: Misura per il calcolo della dimensione delle Apps

```

1 IF NOT [Rating] = 0 THEN [Rating] END

```

Figura 3.4: Misura per il calcolo del rating

Misura per aggregare i prezzi Misura utilizzata al fine aggregare correttamente le applicazioni in base al prezzo, così da avere delle didascalie utili per una migliore rappresentazione.

```

1 IF [Price] >= 0 AND [PRICE] < 1 THEN '[0,0.99]$'
2 ELSEIF [Price] >= 1 AND [PRICE] < 2 THEN '[1,1.99]$'
3 ELSEIF [Price] >= 2 AND [PRICE] < 3 THEN '[2,3.99]$'
4 ELSEIF [Price] >= 3 AND [PRICE] < 4 THEN '[3,3.99]$'
5 ELSEIF [Price] >= 4 AND [PRICE] < 5 THEN '[4,4.99]$'
6 ELSEIF [Price] >= 5 AND [PRICE] < 10 THEN '[5,9.99]$'
7 ELSEIF [Price] >= 10 AND [PRICE] < 20 THEN '[10,19.99]$'
8 ELSEIF [Price] >= 20 AND [PRICE] < 50 THEN '[20,49.99]$'
9 ELSE '[50, inf]$'
10 END

```

Figura 3.5: Misura per il calcolo del prezzo delle Apps

3.2 Analisi Temporale

Nella sezione che ora andiamo ad introdurre siamo andati ad osservare l'evoluzione che hanno avuto le applicazioni analizzate e quanto ne consegue relativamente al loro utilizzo nel decennio 2010-2021. In particolare tramite il tool **Tableau** nella schermata in figura Fig. 2.7 siamo andati a raffigurare quelle che sono le analisi relative a:

1. Dimensione media delle applicazioni rilasciate in ogni anno
2. N° di applicazioni che sono state rilasciate in ogni anno
3. N° di installazioni totali per ogni anno di rilascio della applicazioni
4. N° di applicazioni rispetto la data di ultimo aggiornamento

Note Utili Le analisi 1. e 2. si trovano nella parte superiore della figura Fig.2.7, mentre le restanti si trovano nella parte sottostante.

Un ulteriore menzione da fare è inherente alle analisi 2. e 4. sulle quali abbiamo applicato la funzionalità di calcolo delle stime al fine di osservare come si evolveranno queste ultime nel corso dei prossimi anni.



Figura 3.6: Overtime Analysis

Osservazioni

Dalla figura Fig. 3.6 possiamo osservare quanto già detto per le stesse analisi realizzate su Qlik, dove: si ha un numero crescente di applicazioni sviluppate in ciascun anno, una dimensione media delle applicazioni crescente per ciascun anno di rilascio, un numero di installazioni che oscilla per i vari anni (per il 2021 si ha un valore basso in quanto il dataset contiene dati incompleti per tale anno) e dall'ultimo istogramma possiamo osservare che le applicazioni aggiornate, per l'ultima volta, prima del 2019 sono una piccola quantità per anno, ma che nel complessivo rappresenta un importante e significativo quantitativo.

Delle osservazioni interessanti possono essere fatte sulla stime introdotte in queste analisi grazie alla funzionalità offerta da Tableau.

Dall'istogramma che riporta il numero di applicazioni rilasciate in ciascun anno si osserva che le stime seguono il trend che può essere individuato anche dagli anni precedenti, infatti si ha un numero di applicazioni sviluppate crescente per il 2021 rispetto all'anno precedente e ancora maggiore nel 2022.

Per la stima applicata all'istogramma che riporta il numero di applicazioni con "ultimo" aggiornamento, si nota che anche in questo caso che c'è un aumento del numero di applicazioni sia nel 2021 che nel 2022, confermando la previsione del trend positivo riguardante la stima delle applicazioni che verranno prodotte, in quanto anche queste ultime avranno ovviamente la loro specifica data di aggiornamento più recente.

3.3 Analisi produttività

Nell'analisi della produttività si è analizzato il comportamento dei vari sviluppatori, in particolare oltre all'aspetto economico abbiamo deciso di analizzare anche la loro produttività espressa sia in termini di longevità delle loro applicazioni che in termini di numero di download delle medesime.

Note Utili

- Ciascun singolo sviluppatore può anche aver messo sul mercato applicazioni in anni diversi e quindi non necessariamente sempre presenti nel corso del decennio. Ecco quindi che l'analisi in questione pone l'attenzione solo sul numero di sviluppatori attivi, ovvero si considerano effettivamente per ciascun anno quegli sviluppatori che hanno pubblicato nello store un'applicazione prodotta.
- La stima del profitto è calcolata considerando il prezzo base di acquisto delle app, escludendo quindi gli acquisti in-app.
- Per ciascun sviluppatore, la durata di vita rappresenta la massima durata di vita fra le varie durate di vita delle applicazioni da loro sviluppate. Tale valore è stato calcolato come la differenza fra la data di ultimo aggiornamento e la data di rilascio dell'applicazione. Questa misura ci dà quindi un'idea sul numero di anni di vita dell'applicazione più longeva per ogni specifico sviluppatore, e di conseguenza anche l'attività prolungata negli anni dei creatori di quelle specifiche app.
- Nella sezione "Developers Appreciation by Apps Average Price" della figura Fig.3.8 è fissato un limite di Average Price sui 50 euro, per evitare outlier di casi "estremi" per applicazioni molto distanti e poco significative alla visualizzazione.

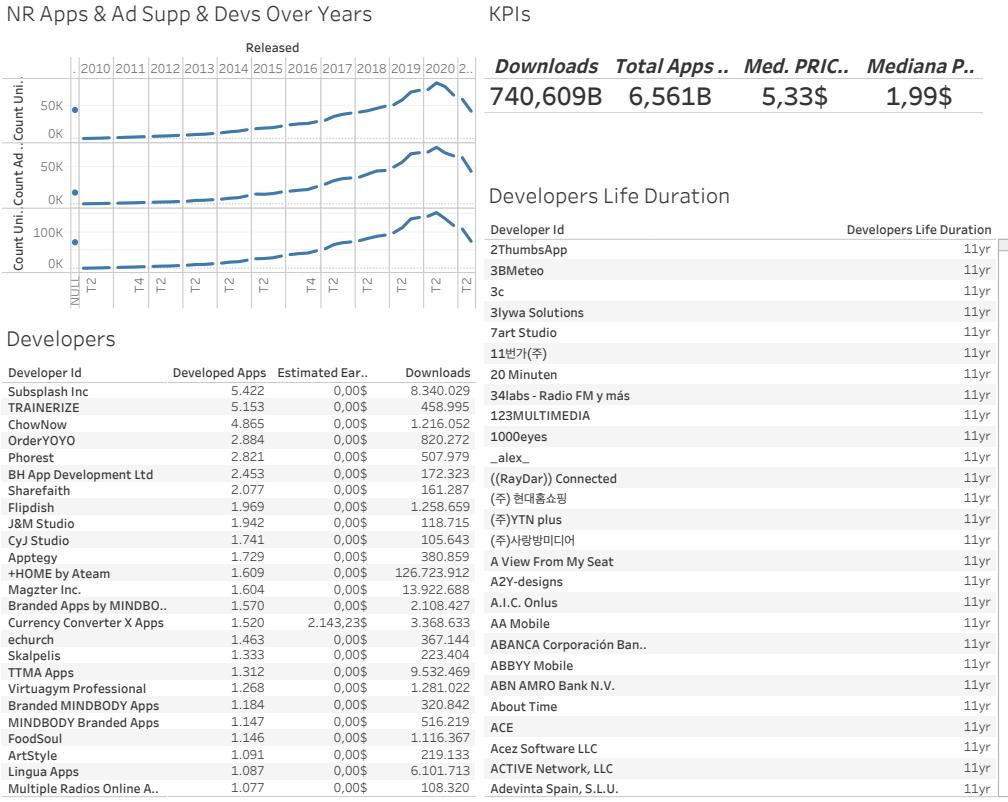


Figura 3.7: Analisi produttività

Successivamente, in aggiunta alle analisi espresse nel paragrafo precedentemente sono stati introdotti ulteriori studi visibili nella figura Fig.3.8. Le analisi qui introdotte sono pressoché le medesime, si è però deciso di esprimerle attraverso ulteriori tipologie di grafici al fine di garantire una migliore fruizione del contenuto espresso da esse.

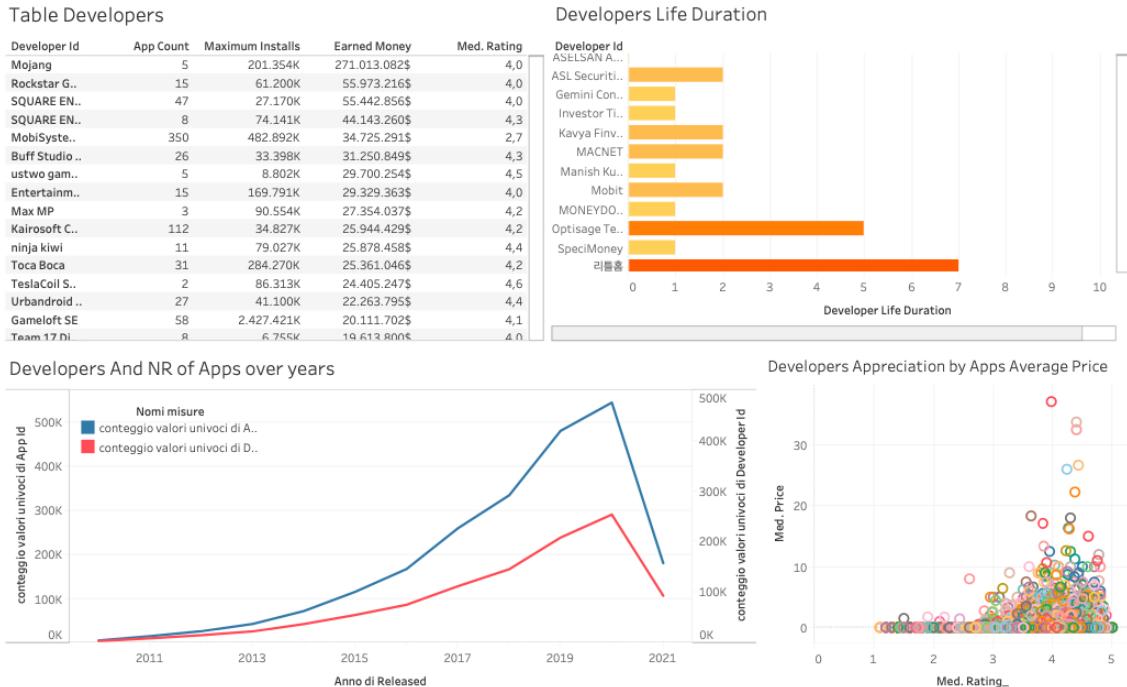


Figura 3.8: Analisi produttività pt.2

Osservazioni

Dalle figure Fig.3.7 e Fig.3.8, possiamo concludere che le osservazioni fatte in seguito alle analisi riportate Qlik rimangono invariate. In estrema sintesi, possiamo concludere che il mercato relativo allo sviluppo di applicazioni è un mercato con un trend positivo nel decennio 2010-2021, sia per i fruitori che per i creatori. Per quanto concerne invece le applicazioni estremamente longeve possiamo osservare come il numero di tali applicazioni sia nettamente esiguo, ma con valori elevati, (infatti ci sono pochissime applicazioni che hanno una longevità effettiva di 10/11 anni di vita).

Dall'analisi "Developers Appreciation by Apps Average Price" possiamo osservare come ciascun sviluppatore, rappresentato da un cerchio, si posizioni considerando il prezzo medio delle applicazioni sviluppate e il valore di rating medio. Si osserva che la maggior parte degli sviluppatori si posizionano in valori minori di 10 per prezzo medio. C'è poi una concentrazione di sviluppatori fra i valori di rating compresi fra 3,5 e 5. Si deduce quindi che ci sono molti sviluppatori che hanno un valore di rating medio/alto e sviluppano maggiormente applicazioni free/con basso costo.

3.4 Analisi Categoria

Nella fase relativa all'analisi delle categorie, abbiamo ripetuto la roadmap fatta per quanto concerne il tool Qlik. Si è quindi andati ad analizzare l'evoluzione del mercato delle applicazioni per ciascuna categoria. Siamo andati quindi a scovare quelle categorie che sono maggiormente presenti sul mercato, ponendo l'attenzione su:

1. Il N° di Applicazioni a pagamento e non
2. Il N° di Applicazioni che consentono gli acquisti al loro interno
3. Il costo medio per per ciascuna categoria
4. Il N° di Applicazioni installate per ciascuna categoria.¹

Il tutto è osservabile nella figura Fig.3.9

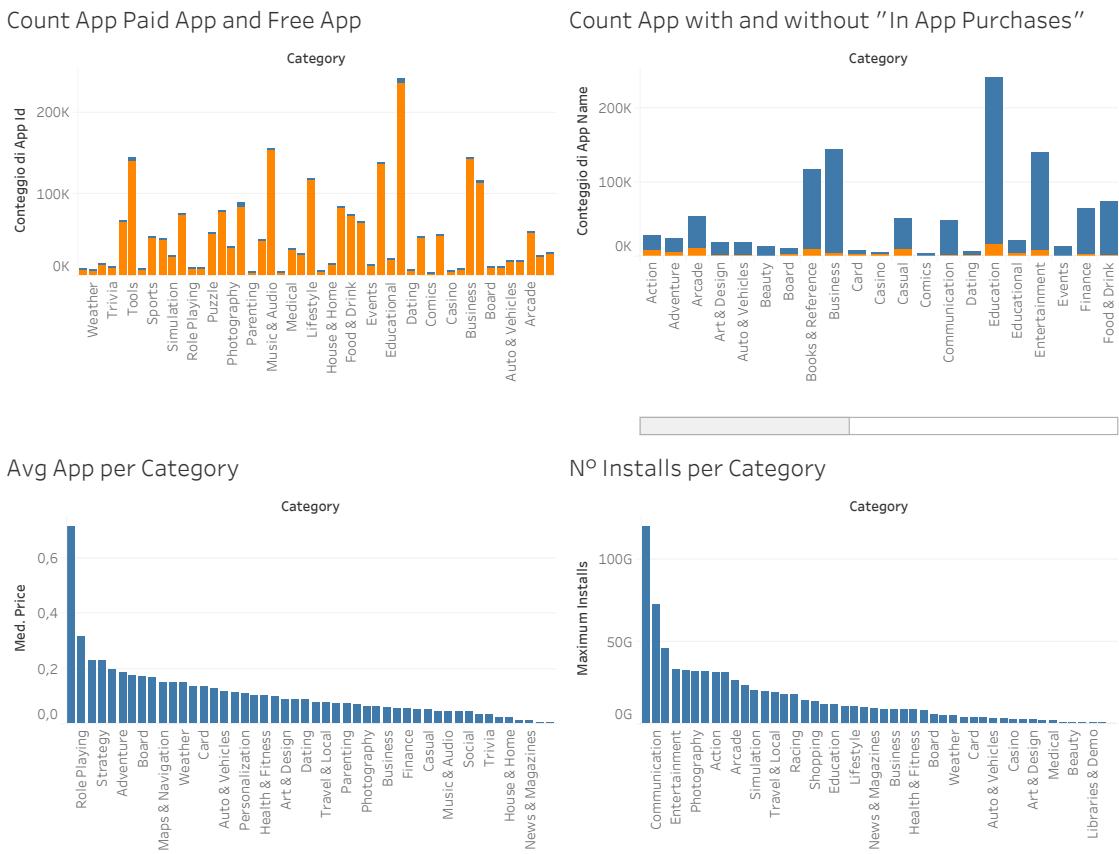


Figura 3.9: Analisi delle diverse Categorie

Infine, tenendo sempre come focus principale le categorie, si è intrapresa un'ulteriore analisi relativa all'andamento temporale di queste ultime. Ecco quindi che nella figura. Fig. 3.10. abbiamo rappresentato tramite due pie chart la divisione delle applicazioni sviluppate in ogni anno e il numero di applicazioni appartenenti a ciascuna categoria. Mentre nella parte sottostante siamo andati a analizzare come si è evoluto il mercato delle applicazioni per ciascuna categoria, analizzando anche quello che sarà il trend futuro espresso tramite le aree colorate.

¹Per tale stima, rifarsi alla nota 2.3

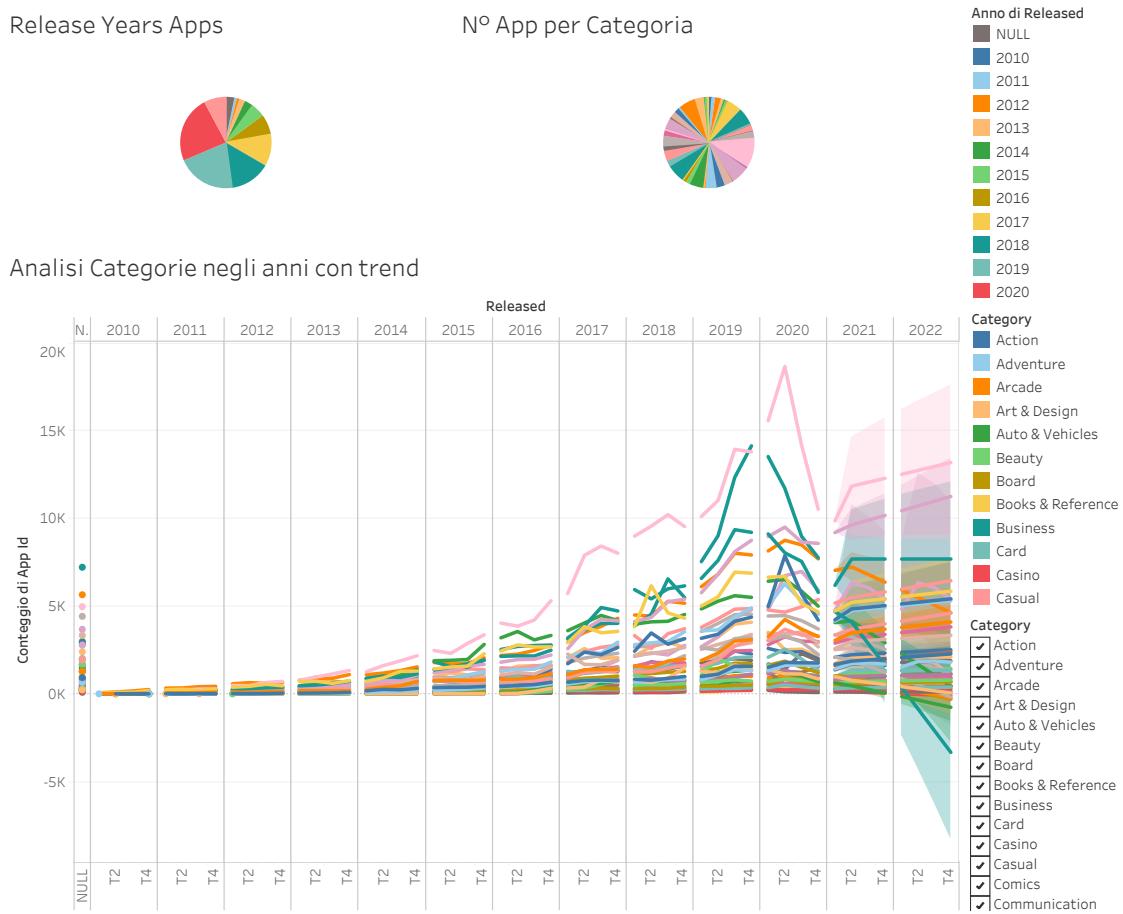


Figura 3.10: Analisi delle diverse Categorie pt.2

Osservazioni

Analogamente a quanto già espresso nella relazione del tool Qlik, le osservazioni possono essere ripetute. Tramite la figura Fig.3.9, notiamo la prevalenza di applicazioni free per ciascuna categoria.

Dall'analisi sul numero di applicazioni che supportano gli acquisti *in-app* possiamo invece osservare come, principalmente, le categorie contengano un numero di applicazioni maggiore che non supportano acquisti *in-app*.

Infine, per quanto concerne il numero di *downloads*, abbiamo una notevole disparità tra la prima categoria, ovvero i *Tools* e le successive categorie, fatto già osservato in Qlik. La figura Fig.3.10 esprime i settori che sono stati maggiormente trainanti nel corso degli anni: *Travel*, *Photography*, *Educational*, *Music & Audio*, etc....

Si tenga presente che per questo ultimo grafico di andamento temporale delle categorie, si è suddiviso il periodo temporale 2010-2021 in trimestri.

3.5 Analisi Longevità

Nella seguente analisi, figura Fig. 3.11, si è analizzato come le diverse applicazioni, presenti all'interno del *dataset*, vengono manutenute dagli sviluppatori.

Note Utili Longevità \vdash Differenza (espressa in anni) tra la data di ultimo aggiornamento e la data di rilascio dell'applicazione.

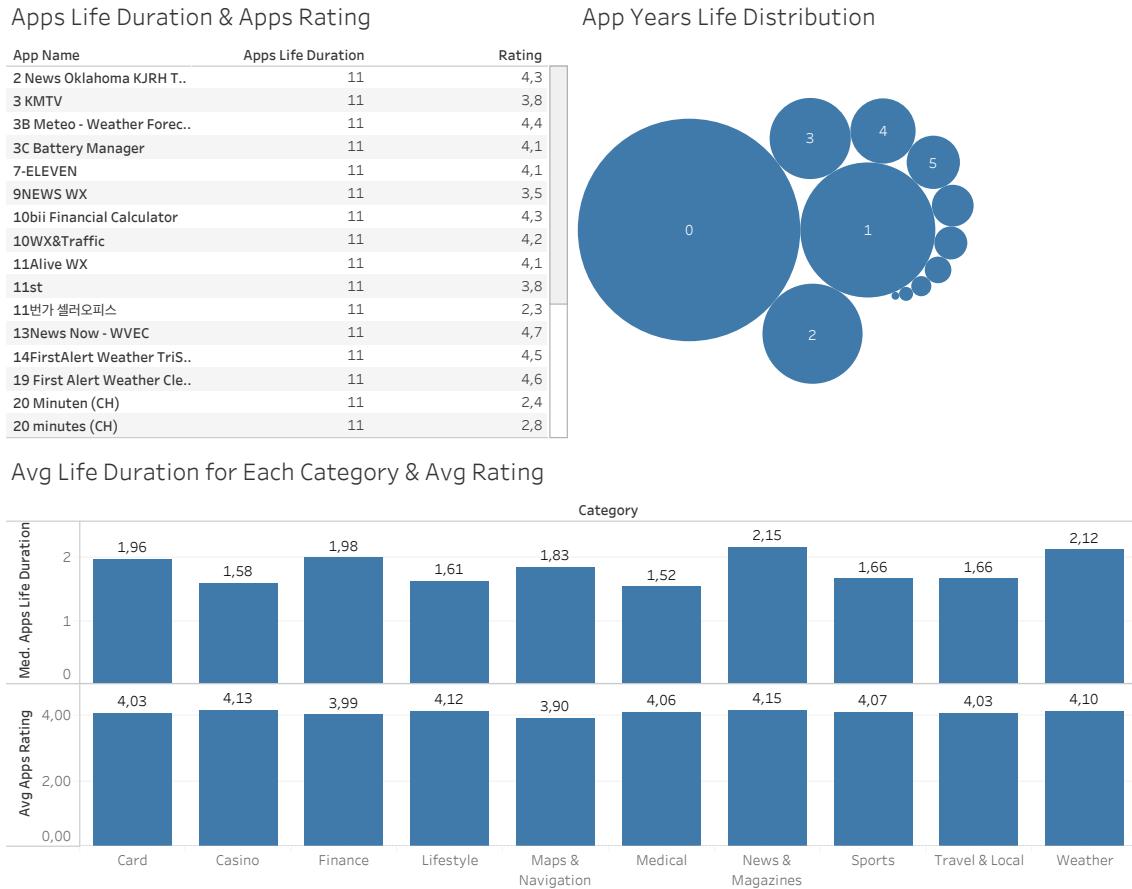


Figura 3.11: Analisi Longevità

Osservazioni

Le osservazioni che si possono fare per tali analisi sono quelle già fatte per le analisi fatte in Qlik. In estrema sintesi, dalla figura Fig. 3.11 è osservabile che il numero di applicazioni "abbandonate" nei primi anni successivi, ma soprattutto quelle nello stesso anno, è assai elevato; specularmente, il numero di applicazioni con elevata longevità è nettamente esiguo.

Infine, si può osservare che le categorie che contengono applicazioni in media più longeve sono le categorie *Weather*, *News & Magazines*.

3.6 Analisi Requisiti e Dimensioni

Nella figura Fig. 3.12 abbiamo riportato le *Minimum Android Version Analysis*, ponendo quindi l'attenzione sul sistema minimo di *Android* supportato dalle applicazioni prese in analisi.

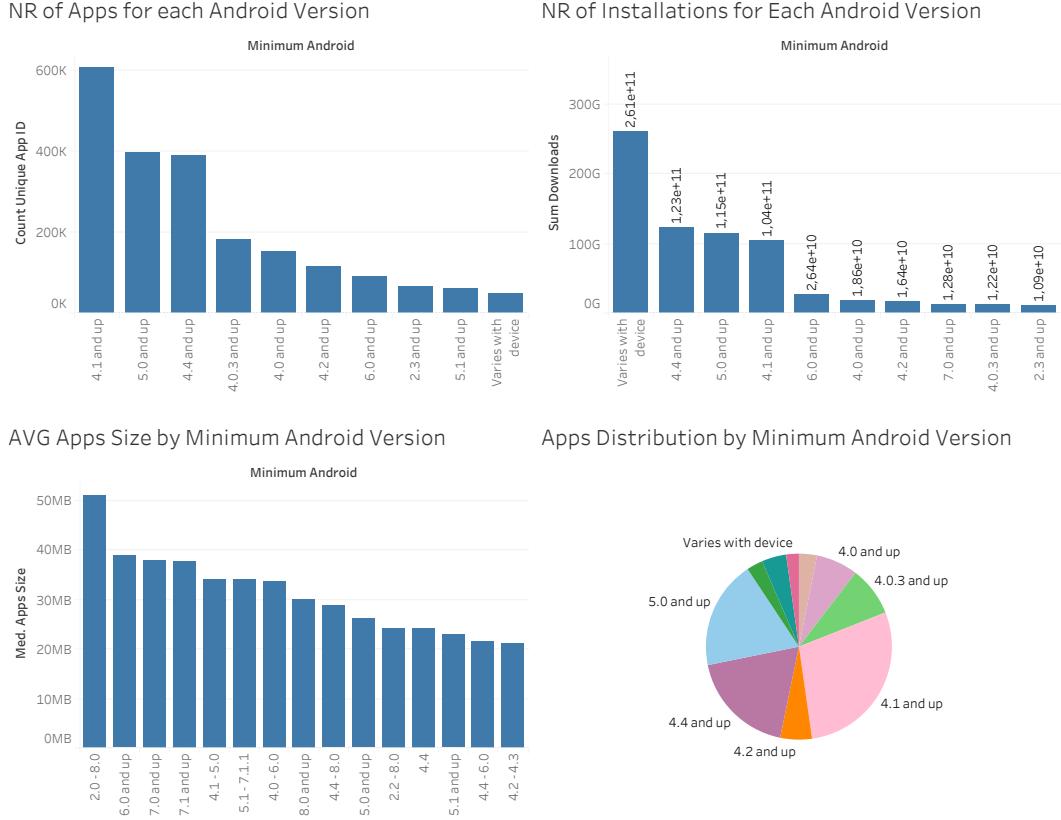


Figura 3.12: Analisi Requisiti e Dimensioni

Osservazioni

Come osservabile dalla figura Fig. 3.12 la maggior parte delle applicazioni richiedono come versione minima *Android 4.1*. Mentre nella parte sottostante della figura riportiamo la dimensione media delle diverse applicazioni per ciascuna versione minima di sistema operativo richiesta. Anche per tali analisi si hanno le stesse osservazioni già riportate nelle analisi di Qlik.

3.7 Analisi Costi

Nella seguente analisi, figura Fig.3.13, si è intrapresa un'indagine sul mercato delle applicazioni, considerando diversi aspetti, come: il costo delle applicazioni, la possibilità o meno di effettuare acquisti all'interno delle applicazioni, la presenza o meno di annunci all'interno delle applicazioni e il guadagno stimato per ogni categoria di applicazioni (considerando il costo e il numero di installazioni).

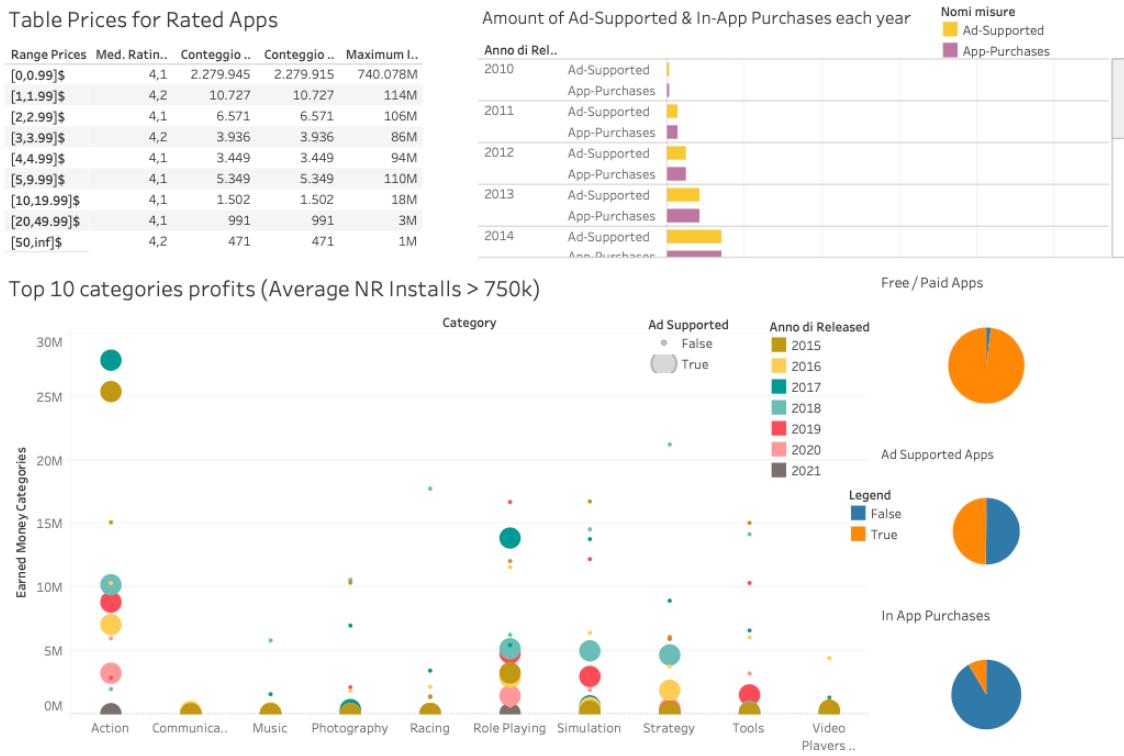


Figura 3.13: Analisi Costi

Nota utile Nell'analisi "Top 10 categories profits (Average NR Installs > 750k)", visibile nella figura Fig.3.13, gli anni sono stati scelti sopra il 2015 e le categorie top sono state scelte per numero medio di installazioni > 750k, questo per facilitare la lettura e la comprensione dell'analisi effettuata.

Osservazioni

Anche in questo studio, le osservazioni rimangono invariate a quelle già espresse in Qlik. Si può osservare come il numero di applicazioni con un prezzo basso (< 1\$) è nettamente maggiore rispetto alle altre fasce di prezzo.

Possiamo, ancora una volta, osservare come il numero di applicazioni a pagamento, così come il numero di applicazioni che supportano acquisti in-app è in numero minore rispetto alle altre.

Dall'analisi in basso possiamo osservare come, a parte nel caso della categoria Action, le altre categorie hanno realizzato un profitto maggiore con le applicazioni che non supportano le pubblicità. Questo può essere giustificato dal fatto che queste sono in numero maggiore ed inoltre, non supportando la pubblicità, hanno presumibilmente un prezzo base di installazione maggiore rispetto a quelle che contengono pubblicità. Inoltre, dal grid chart, è osservabile che la categoria

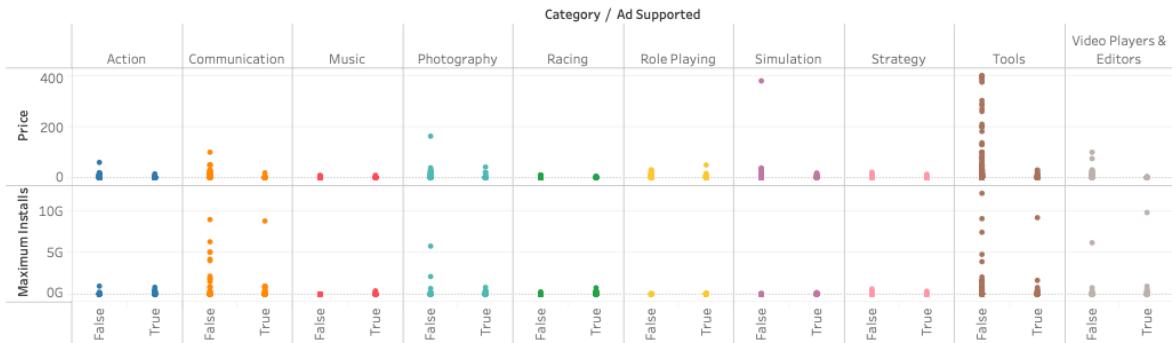
Action ha avuto un picco di incassi nel 2015 e successivamente nel 2017, mentre negli altri anni, il suo range di guadagno è stato intorno ai 10M\$.

Dall'analisi che riporta il numero di applicazioni che supportano la pubblicità e gli acquisti in app per ciascun anno è osservabile che all'aumentare degli anni aumentano il numero di queste applicazioni, questo è giustificato dalla volontà di effettuare maggiori guadagni con tecnologie che permettono, ad esempio, di introdurre pubblicità in maniera sempre più semplice e meno impattante per l'applicazione (essendoci stata un'evoluzione anche sulle dimensioni degli schermi dei dispositivi).

3.8 Box & Whisker

Per una maggiore completezza e comprensione del tool si è inoltre implementato anche un'ulteriore tipologia di plot chiamata **Box & Whisker**. Tramite questo studio abbiamo osservato come le varie categorie² si sono distribuite in termini degli indici "price" e "num installs". Dopotichè per valutare l'impatto derivante dalle pubblicità nelle rispettive categorie si fa quindi la distinzione nell'asse delle ascisse tra Applicazioni *Ad Supported* True/False. Si noti che nella Dashboard di figura Fig. 3.14 l'unica differenza tra il grafico sopra e sotto è l'aver limitato (nel grafico sotto) l'analisi alle sole applicazioni che hanno un massimo di 2,5B di installazioni.

Box & Whisker Categories 1



Box & Whisker Categories 2



Figura 3.14: Box & Whisker

²Solo le categorie con media Max Installs > 750.000

Osservazioni

In questa analisi si è cercato di studiare se ci fossero sostanziali differenze nella top 10 delle categorie preferite dagli utenti (scelte in base al numero di installazioni).

Si tenga presente che la top 10 delle applicazioni preferite è stata selezionata agendo sulle sheet di figura Fig. 3.14 attraverso lo stesso filtro utilizzato per le altre top 10 della tesi in questione. Nel dettaglio, si è utilizzato un filtro per selezionare le sole categorie che presentano una media del numero di installazioni superiore a 750.000 (750k), così da considerare quelle aree che, nell'insieme totale, sono apprezzate dalla maggioranza degli utenti, e quindi quelle categorie che sotto l'aspetto degli applicativi mobili sono quelle più globalmente distribuite ed apprezzate.

Si osservi come, per quanto riguarda l'andamento generale tra le varie categorie, sia per l'aspetto dei prezzi che del numero di installazioni, gli sheets non presentano notevoli differenze, fatta eccezione per la categoria *Tools* che possiede un numero nettamente maggiore di applicazioni nel mercato. Inoltre, questa categoria risulta alquanto distribuita su un vasto range di prezzi, il che indica la volontà degli sviluppatori di differenziare il più possibile i loro applicativi come punto di forza per la concorrenza nel mercato.

In secondo piano, per le categorie come *Communication*, *Photography*, *Video Players & Editors* si notano con gli stessi principi ma con numero di applicazioni e distribuzione meno accentuati. Come risulta quindi dai grafici, l'elenco appena fatto, conferma le attitudini globali degli ultimi anni, in quanto gli sviluppatori tendono molto più frequentemente sulle tipologie di applicativi più in voga in questo passato decennio, e che in quello a venire porterà senz'altro ad una crescita in numero e distribuzione di prezzo, al fine di favorire molteplici gerarchie sociali.

Infine, per quanto riguarda la differenza legata all'attributo *Ad Supported*, vediamo come la tendenza degli utenti è ancora molto chiara nella scelta delle applicazioni. Infatti si dimostra che, per quanto riguarda l'aspetto dei costi, le app che tendono ad avere pubblicità interne cercano di mantenere i costi dell'installazione base della specifica App bassi ed accessibili ad un numero più elevato di utenti, così da non rischiare di avere poco apprezzamento e mantenere il ciclo di vita dell'app il più duraturo possibile.

Dall'altro canto, gli stessi fattori incidono allo stesso modo sul numero di installazioni, così che per applicazioni "libere" da pubblicità si hanno numeri di download leggermente più elevati. A differenza di quanto detto per i costi però, rimane chiaro che, se una applicazione risulta essere ben fatta, gli utenti tendono comunque a mantenere elevati i numeri in Download.

3.9 Analisi Varie

Alla conclusione dei nostri precedenti studi, si è deciso di integrare il lavoro fin qui svolto con ulteriori analisi marginali al fine di avere una più completa panoramica sul dataset utilizzato. Si sono analizzate le applicazioni considerando parametri non menzionati precedentemente, in particolare: **Content Rating Category**, parametro che ci permette di comprendere se le applicazioni sono utilizzabili o meno da diverse categorie di utenti; **Editor Choice** ovvero un indicatore per giustificare l'ottimalità dell'applicazione. Infine, si sono riportati gli sviluppatori principali per numero di applicazioni pubblicate nel *Play Store di Google* e le applicazioni principali per numero di installazioni. Tali analisi sono rappresentate nella figura seguente Fig.3.15.

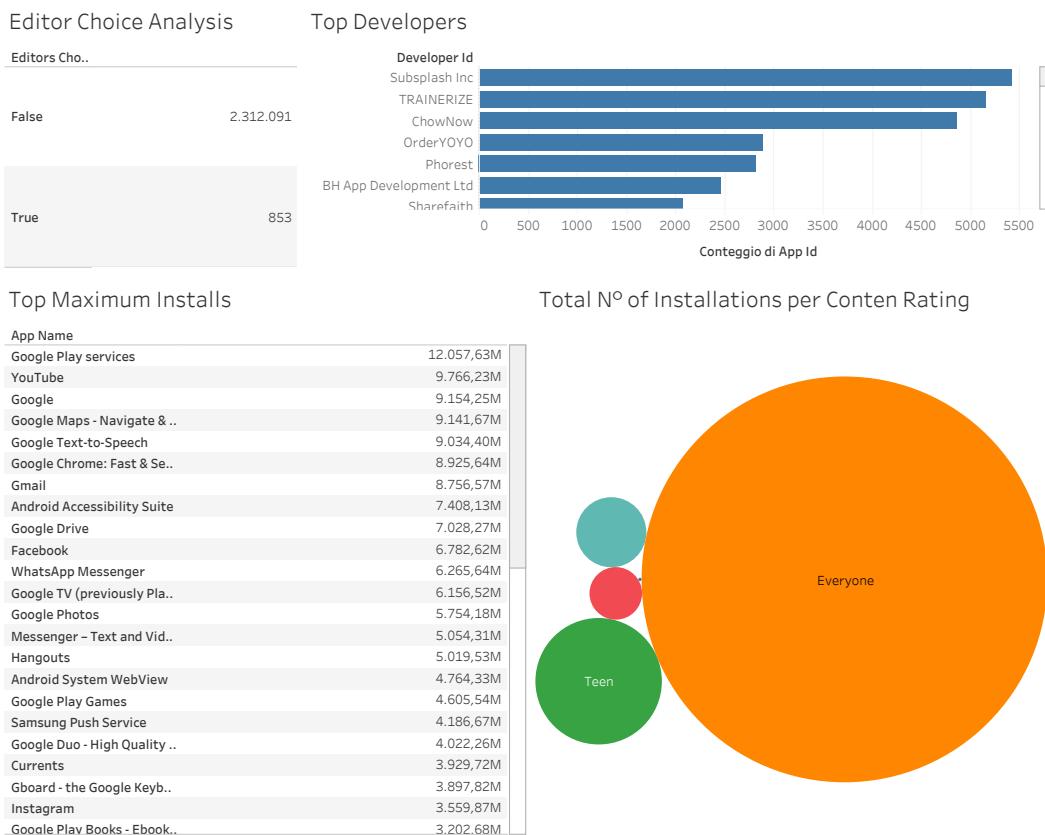


Figura 3.15: Ulteriori Analisi

Osservazioni

Guardando la figura Fig. 3.15 possiamo brevemente riportare le osservazioni già effettuate nelle analisi svolte tramite il tool **Qlik**. In estrema sintesi: la maggior parte delle applicazioni appartengono alla categoria di *Content Rating Everyone*, il numero di applicazioni che risultano essere "Scelte dagli editori" rappresentano un numero irrisorio rispetto alla totalità delle applicazioni. Le applicazioni principali per numero di installazioni appartengono all'ecosistema *Google* ed infine tra i principali sviluppatori per numero di applicazioni rilasciate vi sono tre principali sviluppatori che hanno un numero di applicazioni sviluppate nettamente maggiore rispetto agli altri, indice che come strategia di marketing puntano sulla quantità piuttosto che sulla poca numerosità accompagnata da un dettaglio interno più elevato.

Capitolo 4

Power BI

Infine, in questo capitolo apprenderemo il funzionamento dell'ultimo strumento visto in questo studio, ovvero **PowerBI**. Tramite questo, siamo andati a svolgere le analisi che vedremo di seguito. Inoltre, attraverso il medesimo applicativo siamo andati a sviluppare dei nuovi "oggetti visivi" tramite il linguaggio R e, per una più completa vista sull'applicativo, abbiamo introdotto ulteriori *Plot* non disponibili nativamente ma scaricabili ed utilizzabili tramite il *marketplace* del tool.



Figura 4.1: Logo PowerBI

4.1 Misure & Pre-processing

Il primo passo che si è svolto è stato il *pre-processing* dei dati. Dal momento che PowerBI offre la possibilità di utilizzare lo strumento *Power Query*, si è deciso, a differenza dei tool utilizzati precedentemente, di andare ad effettuare un pre-processing prima di importare i dati e di non utilizzare, in seguito, formule per l'estrazione dei dati nel formato voluto.

Le prime trasformazioni effettuate sono state le **trasformazioni dei tipi**, si sono modificati cioè, dove necessario, i tipi delle colonne presenti nel *dataset*. Le trasformazioni di tipi effettuate sono le seguenti:

- *Rating* convertito in numero decimale, gli spazi vuoti presenti in alcune righe sono stati convertiti in 0.0;
- *Rating Count* convertito in numero intero, gli spazi vuoti presenti in alcune righe sono stati convertiti in 0;
- *Minimum e Maximum Installs* convertiti in numeri interi;
- *Price* convertito in valuta;

- *Free/Ad Supported/In App Purchases/Editors Choice* convertiti in valori booleani;
- *Last Updated* convertito in formato data;

Alcune trasformazioni più complesse, rispetto alla semplice trasformazione di tipo, sono quelle riportate di seguito:

- *Installs*, i valori (stringhe che terminavano il simbolo "+") sono stati modificati eliminando il simbolo "+" e convertendo i valori in numeri interi;
- *Size*, alcuni valori erano espressi in MB ed altri in KB, si è creata una nuova colonna in cui si sono convertiti tutti i valori in MB ed effettuata la trasformazione di tipo in numeri decimali;
- Alcuni valori della colonna *Realease* erano nulli, si sono settati questi valori pari al valore della colonna *Last Updated*, in modo che nelle analisi di longevità delle applicazioni risultasse al minimo pari a 0 e non con valori negativi. Dopo aver effettuato questa trasformazione si è convertito il tipo della colonna *Realease* in formato data.

Una volta effettuate queste trasformazioni si è ottenuto un dataset che contiene i dati nel formato da noi desiderato; questo, come detto precedentemente, ci ha permesso di realizzare analisi senza dover utilizzare formule di conversione, IF, ecc. Cosa che invece si era resa necessaria per gli strumenti visti precedentemente.

4.2 Analisi Temporale

Nello studio relativo all'analisi temporale, ci si è posti come *main focus* l'osservazione dell'evoluzione relativa alle applicazioni presenti all'interno del dataset e quanto è più inerente al loro utilizzo nel decennio 2010 – 2021. Nello specifico, con l'ausilio del software **PowerBI** siamo andati a graficare tali studi nella figura Fig. 4.2, quest'ultimi sono legati a:

1. N° Apps che sono state rilasciate per ogni anno
2. La dimensione media delle applicazioni rilasciate per ogni anno
3. N° Relativo alle installazioni per ogni anno
4. N° di Apps rispetto la data di ultimo aggiornamento

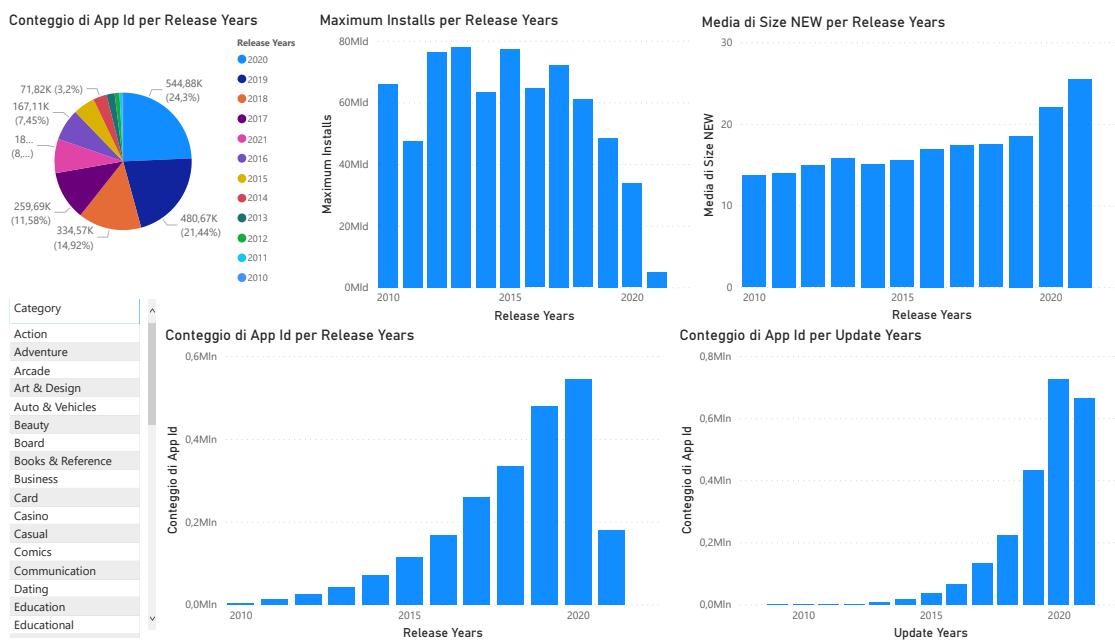


Figura 4.2: Overtime Analysis

Osservazioni

Osservando la figura precedente (Fig. 4.2), anche in questo caso valgono le considerazioni già dette precedentemente durante l'utilizzo degli altri tool. In estrema sintesi possiamo affermare che con il passare degli anni, il numero di applicazioni che vengono rilasciate hanno un trend positivo (da notare il periodo 2019 – 2020), giustificabile in parte dal periodo pandemico, il quale ha fornito un boost al mercato al fine di garantire sia intrattenimento che servizi, poco fruibili in prima persona durante tale periodo. Il calo relativo al 2021 dipende dal aggiornato del dataset l'ultima volta a metà 2021, quindi contiene dati incompleti per tale anno.

Altre osservazioni possono essere fatte in merito al fatto che i valori relativi al numero di installazioni delle applicazioni oscillano leggermente, ad eccezioni degli ultimi anni per cui, essendo le applicazioni più recenti, si ha, per ora, un numero relativamente basso di installazioni. Questo potrebbe indicare il fatto che, nei primi anni del decennio analizzato, essendo l'aspetto di applicativi per cellulari mobili una innovazione che in precedenza era quasi inesistente, prendeva

inizialmente tanta voglia tra gli utenti così da portare a numeri spropositati, anche se magari per un breve lasso di tempo, soprattutto considerando la ridotta qualità delle applicazioni di allora. Si ha inoltre un picco relativo alle applicazioni che sono state aggiornate l'ultima volta tra il 2019 e 2021. Mentre per quanto riguarda la dimensione media delle applicazioni le osservazioni svolte precedentemente rimangono invariate, cioè avendo a disposizione più risorse si può osservare un aumento delle dimensioni delle applicazioni negli ultimi anni rispetto ai primi.

4.3 Analisi Produttività

Nell'analisi della produttività il focus è stato posto sul comportamento dei vari sviluppatori, considerando sia l'aspetto economico che la loro produttività espressa in termini di longevità delle loro applicazioni e in termini di numero di *download* delle medesime.

Note Utili

- Ciascun singolo sviluppatore può anche aver messo sul mercato applicazioni in anni diversi e quindi non necessariamente sempre presenti nel corso del decennio. Ecco quindi che l'analisi in questione pone solo il numero di sviluppatori attivi, cioè quelli che hanno effettivamente pubblicato nello store applicazioni in un certo anno .
- La stima del profitto si è svolta considerando il prezzo di acquisto delle app, escludendo quindi gli acquisti in-app.
- Per ciascun sviluppatore, la longevità è stata calcolata come la massima durata di vita fra le varie durate di vita delle applicazioni da lui sviluppate. La durata di vita di un'applicazione si è calcolata come la differenza fra la data di ultimo aggiornamento e la data di rilascio dell'applicazione. Questa misura ci dà quindi un'idea sul numero di anni di vita dell'applicazione più longeva per ogni specifico sviluppatore.

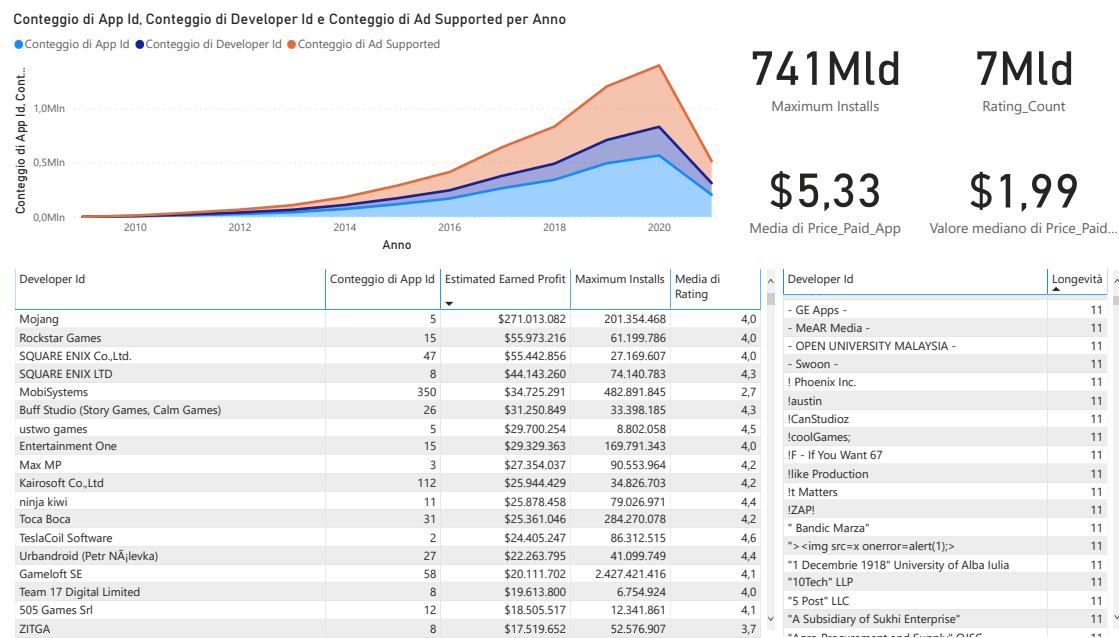


Figura 4.3: Analisi produttività

Osservazioni

Tramite la figura Fig.4.3 possiamo concludere che le osservazioni fatte in seguito alle analisi riportate in Qlik e Tableau rimangono invariate. Possiamo concludere che il mercato relativo allo sviluppo di applicazioni è un mercato con un trend positivo nel decennio 2010 – 2021, sia per i fruitori che per i creatori.

Per quanto concerne invece le applicazioni longeve possiamo osservare come ci siano limitate applicazioni che hanno durata elevata negli anni presi in esame.

Guardando la tabella con gli sviluppatori si può osservare come, quelli che hanno realizzato un guadagno stimato più grande, hanno sviluppato in generale un numero esiguo di applicazioni mobile. Questo ci conferma l'affermazione che per poter realizzare guadagni, bisogna sviluppare e manutenere poche applicazioni e ben realizzate, rispetto allo sviluppare molte applicazioni che saranno poi poco apprezzate dagli utenti.

4.4 Analisi Categoria

Riprendendo nuovamente lo studio relativo all'analisi delle categorie, abbiamo svolto i medesimi steps fatti precedentemente tramite gli altri tool già descritti ampiamente. Lo studio si è posto come focus principale l'analisi relativa alle preferenze degli utenti nel mercato delle applicazioni, in particolare per ciascuna categoria. Ci siamo quindi dati come goal quello di scovare le categorie maggiormente preferite sul mercato, ponendo l'attenzione su:

1. Il N° di Applicazioni Ad Supported per rispettiva categoria
2. Il N° di Applicazioni a pagamento per ciascuna categoria
3. Il costo medio per per ciascuna categoria
4. Il N° di installazioni totali per ciascuna categoria

Ciascuna delle appena elencate analisi, sono tutte state poi legate alle categorie per stilarne una classifica delle migliori in ciascuna analisi. Il tutto è osservabile nella figura Fig.4.4 .

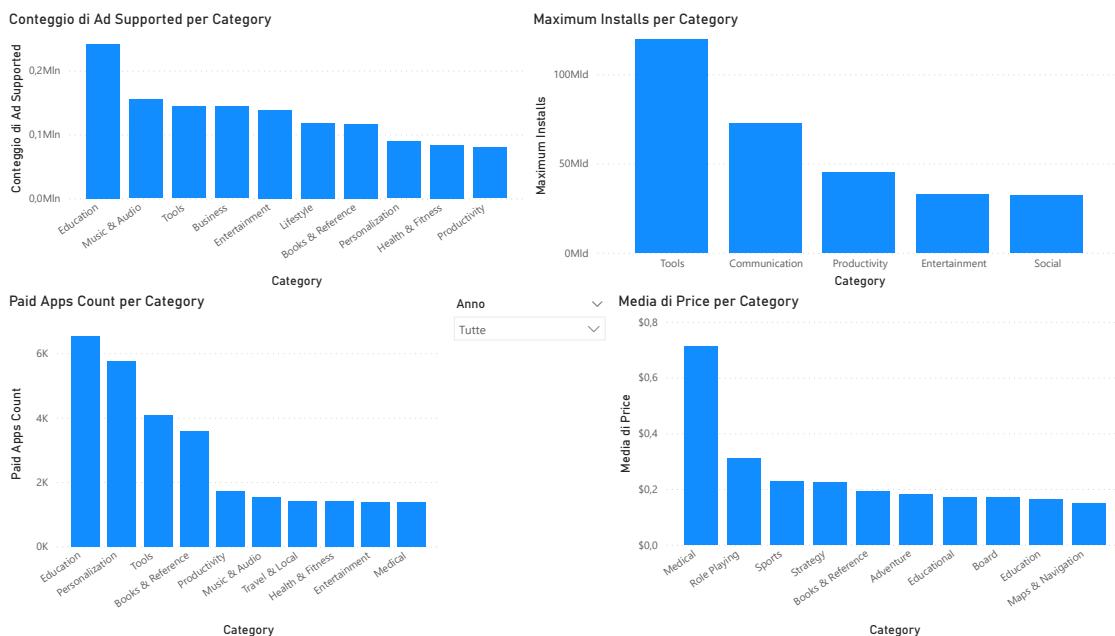
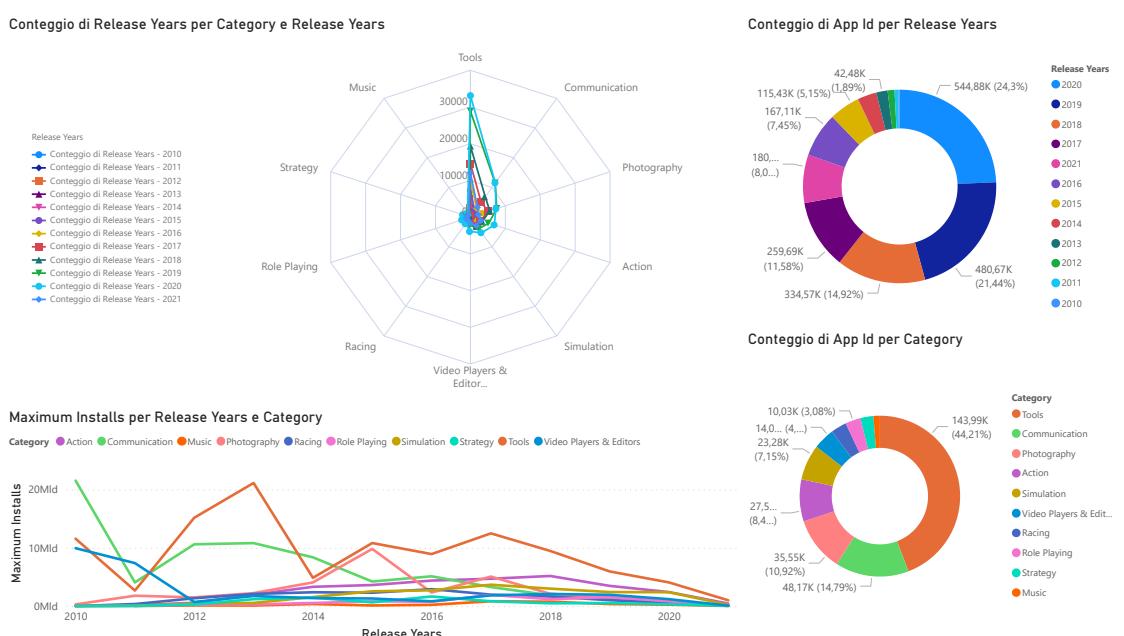
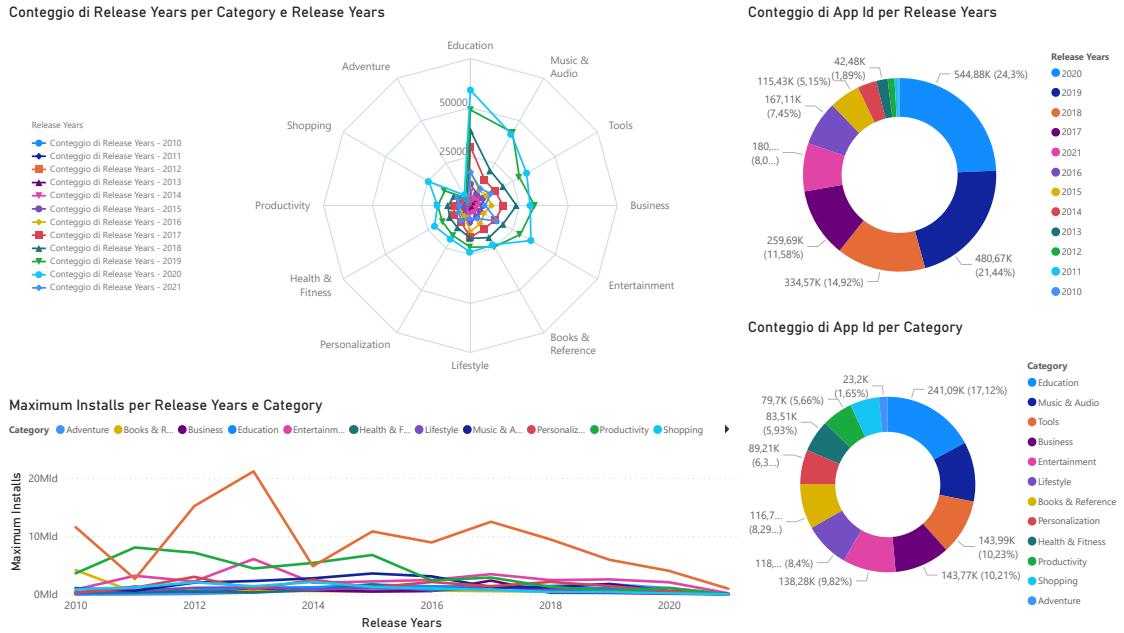


Figura 4.4: Analisi Categoria

Infine, tenendo sempre come stesso focus le categorie, si sono intraprese ulteriori analisi relative all'andamento temporale di queste ultime. Ecco quindi che nelle figure Fig. 4.5 e Fig.4.6 vengono introdotte due analisi simili ma che differenziano per aspetti specifici. Utilizzando ulteriori "oggetti visivi" come il *radar chart* ed il *line chart* si è osservato come nel corso degli anni si sia sviluppato il mercato delle applicazioni. Tali studi sono stati svolti sia nell'ottica del rilascio delle applicazioni che nell'ottica di utilizzo, o meglio, del numero di installazioni totali per ciascuna categoria, core di questo studio.



Osservazioni

Lo studio qui riportato, e le analisi visibili nelle varie figure, dimostrano come le osservazioni fin qui fatte nei precedenti tool rimangano valide. In sintesi, nei *Line chart* risaltano informazioni utili come la top 3 delle Applicazioni maggiormente preferite dagli utenti sono provenienti da categorie Tools piuttosto che Communication (Line Chart sottostante) e Productivity (Line Chart

soprastante), sintomo che gli applicativi dedicati al lavoro e all'interazione tra soggetti ed utenti ha una rilevanza al giorno d'oggi assai importante, con dei probabili incrementi futuri data la digitalizzazione sempre più in espansione globalmente, e lo Smart Working che sta assumendo sempre più importanza. Altre informazioni utili che provengono dai grafici possono indicare allo stesso tempo come la categoria Educational abbia degli scopi di lucro per i produttori ed investitori, tendendo probabilmente a pubblicizzare infatti quelle App di istruzione che sicuramente posseggono una platea di utenti istruiti e magari anche molto predisposti alla considerazione e a porre attenzione su di esse. Quest'ultimo aspetto rappresenterebbe dei profitti per gli investitori, che ritengono possano avere più successo sponsorizzando specifiche tipologie di App, con frequenze di utilizzo più elevate, piuttosto che applicazioni che molti più utenti posseggono ma che allo stesso tempo usano poco (esempio ne è la categoria Tools). Infine si osserva un'altra informazione importante, ovvero che le categorie con prezzo maggiore sono quelle come App Medical, Business, ma anche categorie diverse come il Dating che derivano da strategie diverse di profitto, puntando a offrire un servizio di qualità, ma con una base di costo di installazione più elevata.

Per quanto riguarda le due restanti analisi, figure Fig. 4.5 e Fig.4.6, si è voluta osservare la differenza tra la analisi delle top 12 categorie per N° totale di applicazioni prodotte e rilasciate, parallelamente a quella delle top 10 delle categorie per N° totale di installazioni. I risultati interessanti (provenienti dal Radar Chart) indicano come la quasi totalità delle categorie con il numero più elevato di applicazioni sul mercato non necessariamente implica che le stesse abbiano un indice di apprezzamento e quindi un numero totale di installazioni elevato. Infatti, si vede come le 10 categorie più installate (Radar Chart sottostante) sono quasi interamente diverse rispetto alla top 12 categorie dove si considerando invece il numero di applicazioni prodotte (Radar Chart soprastante). Questo indica come spesso, la quantità non implica necessariamente la qualità, e che allo stesso tempo, alcune categorie hanno indici di apprezzamento che sono largamente maggiori rispetto molte possibili aree di interesse alternative.

4.5 Analisi Longevità

In questa sezione siamo andati ad osservare quella che è l'analisi della longevità delle diverse applicazioni presenti all'interno del nostro dataset. In particolare il focus dello studio si è incentrato sul parametro *App Life Duration* in relazione con il parametro relativo alle reviews, ovvero quello di *Rating*. Il tutto è nuovamente osservabile nella figura Fig.4.7.

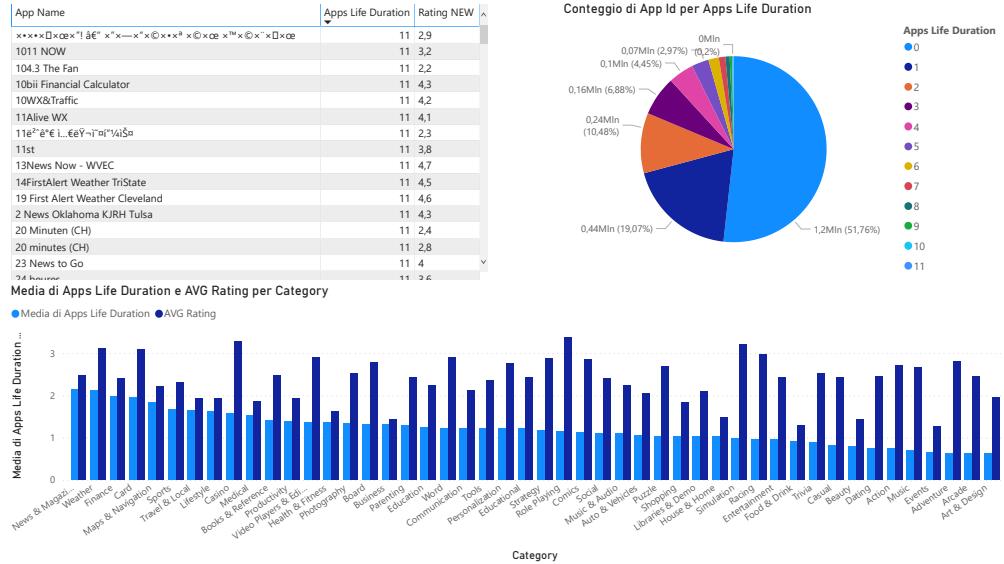


Figura 4.7: Analisi Longevità

Osservazioni

Nuovamente anche in questo studio le osservazioni precedenti in merito all'analisi longevità vengono confermate. È osservabile come la maggior parte delle applicazioni *nasce e muore* nel medesimo anno di sviluppo.

Tra le possibili cause dell'abbandono dell'applicazione da parte dello sviluppatore possono risultare l'utilità dell'applicazione venuta meno nel corso degli anni, lo sviluppo di una nuova applicazione con nuove features evitando quindi di riprendere in mano lo sviluppo di applicazioni datate sia per motivi tecnologici legati alla complessità che di tempistiche necessarie all'adattamento della medesima applicazione a nuovi standard o nuove funzionalità. Va però evidenziato come applicazioni inerenti all'informazione, applicazioni meteorologiche etc., ovvero applicazioni le quali nascono con un focus "poco mutabile" resistono nel corso del tempo e vengono costantemente aggiornate. Questo potrebbe essere legato anche alle disponibilità economiche di chi detiene i diritti di queste ultime e quindi facendo venire meno le motivazioni espresse precedentemente.

4.6 Analisi Costi

Nel seguente studio, visibile nella figura Fig.4.8, il focus dell'indagine si è spostato sull'aspetto economico legato al mercato delle applicazioni, osservando: il costo delle applicazioni, la possibilità o meno di fare acquisti in-app, la presenza o meno di annunci all'interno delle applicazioni e il guadagno stimato per ogni categoria di applicazioni (considerando il costo e il numero di installazioni).

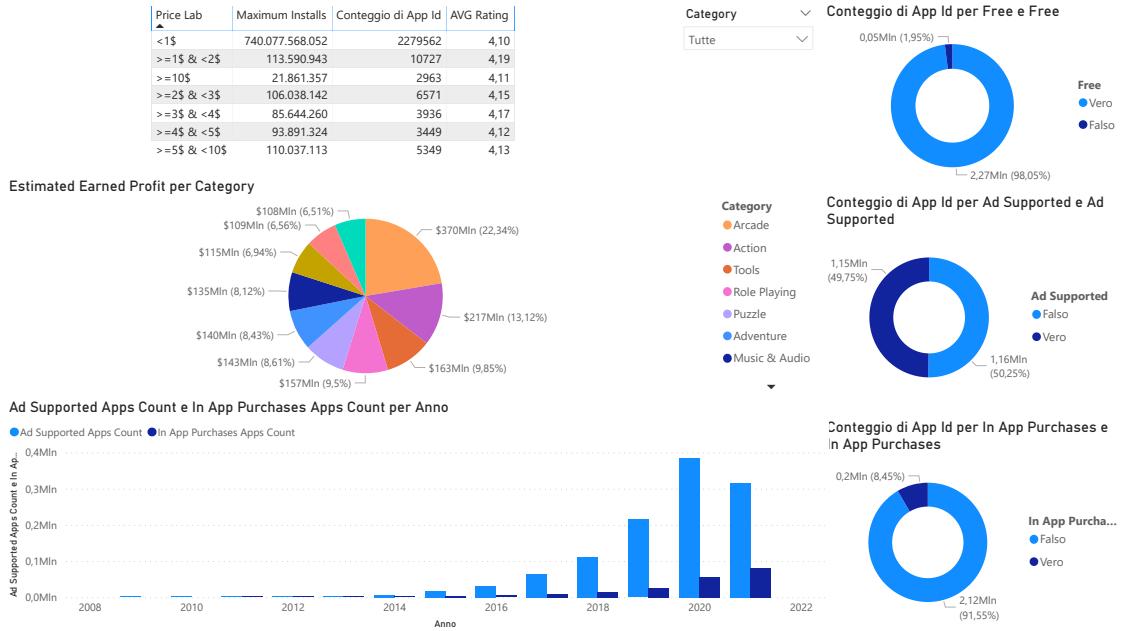


Figura 4.8: Analisi Costi

Osservazioni

Anche nella analisi rappresentate in figura Fig. 4.8 sviluppate tramite il tool **PowerBI**, le osservazioni rimangono invariate a quelle già espresse in Qlik e Tableau. In sintesi: le applicazioni gratuite sono in numero nettamente superiore rispetto a quelle a pagamento, il numero di applicazioni che supportano gli acquisti in app è nettamente inferiore rispetto al numero di applicazioni che non li supportano e, infine, le applicazioni che contengono annunci pubblicitari sono in proporzione quasi uguale a quelle che non li contengono.

Dalla tabella in alto si può osservare come la gran parte delle applicazioni si posizionano nella fascia di prezzi nulli/bassi, mentre un numero esiguo di applicazioni si posizionano nelle fasce di prezzo superiori.

Dall'istogramma in basso si può osservare come negli anni ci sia stato un aumento del numero di applicazioni che supportano gli acquisti in-app così come c'è stato un aumento del numero di applicazioni che supportano gli annunci pubblicitari.

4.7 Box & Whiskers

Al fine di rispettare la road map seguita per Tableau, e quindi anche per una maggiore completezza e comprensione dei tools, abbiamo nuovamente implementato i **Box & Whisker** plot, questa volta però in modo molto più diretto e meno complesso. Tramite questo studio abbiamo osservato come le varie categorie¹ si sono distribuite in termini degli indici "price" e "num installs". Dopodiché per valutare l'impatto derivante dalle pubblicità nelle rispettive categorie si è fatta la distinzione nell'asse delle ascisse tra Applicazioni *Ad Supported* True/False.

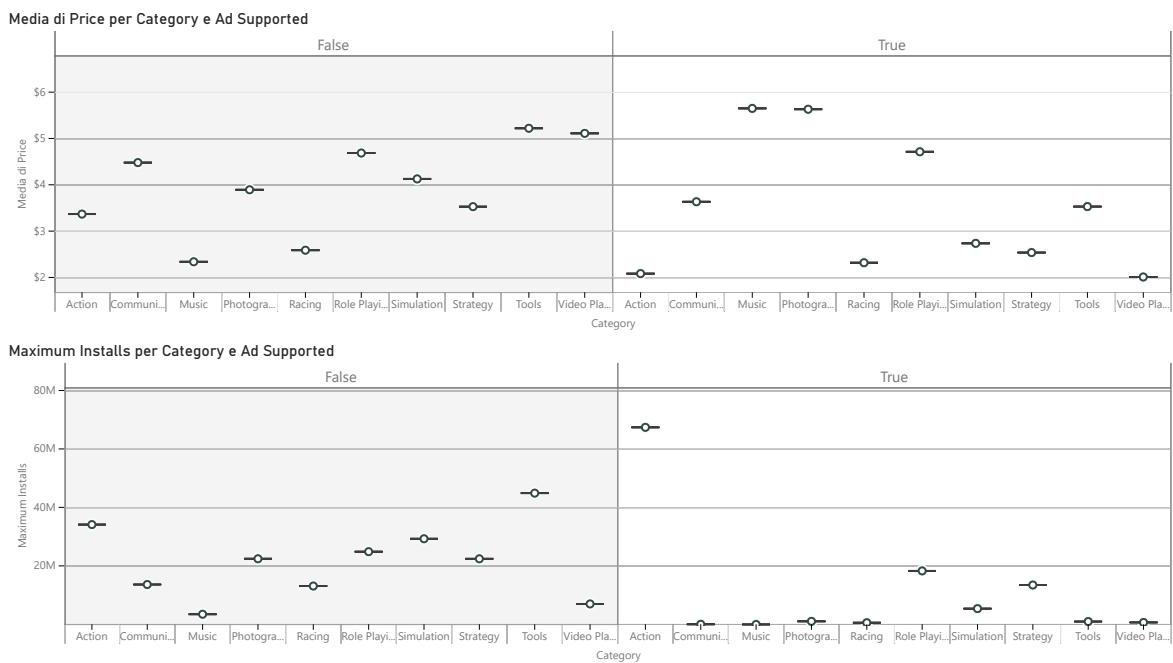


Figura 4.9: Box & Whisker

Osservazioni

In questa analisi si è cercato di studiare se ci fossero casi diversi dagli altri nella top 10 delle categorie preferite dagli utenti.

Si tenga presente che la top 10 delle applicazioni preferite è stata selezionata agendo sulle sheet di figura Fig. 4.9 attraverso lo stesso filtro utilizzato per le altre top 10 della tesi in questione, come accennato nella parte introduttiva.

Si osservi come, per quanto riguarda l'andamento generale tra le varie categorie, per l'aspetto dei prezzi, c'è un'effettiva disparità di prezzo medio per diverse categorie, il che indica che molteplici sviluppatori tendono ad abbassare il prezzo base dell'app, cercando allo stesso tempo di ottenere profitti da pubblicità interne, e quindi adottando una particolare strategia di marketing. Infatti si osservi questo comportamento per categories come *Video Players*, *Tools*, piuttosto che *Action* e *Communication*. Questo avviene probabilmente per evitare la perdita di un eccessivo numero di utenti, a cui potrebbe pesare il duplice aspetto del costo dell'app, e la frequenza delle pubblicità all'interno.

Infine, sempre per la parte dei prezzi medi, si osservano due interessanti casi opposti, ovvero

¹Solo le categorie con media Max Installs > 750.000

le categorie *Music* e *Photography* che al contrario dei soliti andamenti dimostra come queste tipologie di App, tendono comunque a mantenere costi elevati per Applicazioni Ad Supported rispetto quelle che ne sono prive. Questo indice conferma l'importanza che stanno avendo globalmente queste tipologie di app, diventate aree di business per un grande numero di compagnie, probabilmente influenzate dall'avvento di Social Media in cui Editing e interazione con Brani musicali è sempre più abbondante.

Concludendo con la parte riguardante il numero di installazioni, invece, abbiamo che, generalmente, il numero di installazioni per le stesse categorie prive di pubblicità è ancora nettamente maggiore rispetto alle stesse che però le hanno all'interno.

Questo sintomo può derivare dal fatto che ancora, le generazioni di utenti attuali, tendono a dare ancora poco importanza alle pubblicità interne agli applicativi, così da portare investitori a pubblicizzare in piattaforme diverse come Canali televisivi piuttosto che Social Media o canali come Youtube/Twitch.

Sicuramente, nelle future generazioni, con applicazioni sempre più interattive per una maggiore platea di utenti, comporterà l'incremento di Companies che tenderanno ad investire su pubblicizzazioni In App, portando quindi a bilanciare il numero tra il grafico di destra e di sinistra.

Come ultima osservazione interessante si analizzi come, l'unico caso opposto all'andamento appena descritto è proveniente dalla categoria *Action*, probabilmente derivante da qualche particolare applicazione di successo che, pur avendo le pubblicità internamente, ha riscosso tanto successo da portare la categoria ad avere un comportamento anormale rispetto le altre. Questo è un risultato interessante che dimostra come, una applicazione di qualità e ben fatta, porta a numeri di installazioni e introiti elevati e può essere quindi una base di appoggio per le pubblicizzazioni molto importante per le Companies esterne.

4.8 Analisi Requisiti e Dimensioni

Nella figura Fig.4.10 abbiamo riportato l'analisi relativa al supporto del sistema operativo *Android* per quanto riguarda le applicazioni presenti all'interno del dataset, siamo andati quindi a visualizzare attraverso diversi plot, quelli che sono i sistemi operativi maggiormente adottati per lo sviluppo e quindi di conseguenza maggiormente utilizzati a livello del mercato delle applicazioni. Al main focus abbiamo aggiunto anche l'ottica relativa alla dimensione delle applicazioni e la sua evoluzione nel corso del decennio 2010 – 2021

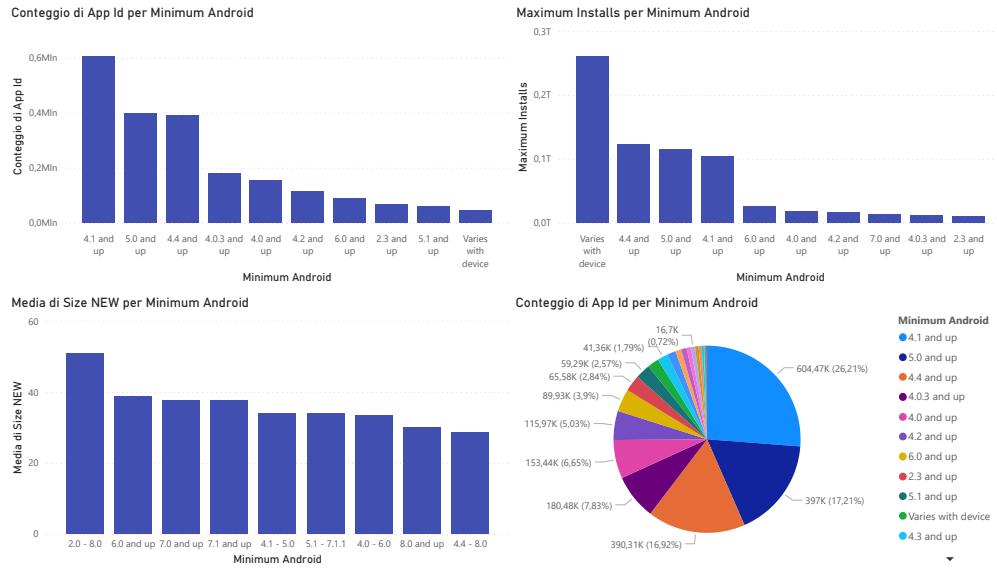


Figura 4.10: Analisi Requisiti e Dimensioni

Osservazioni

Come evidenziato nelle analisi svolte tramite gli altri tool, la maggior parte delle applicazioni richiedono come versioni minime sistemi operativi non poi così recenti, notiamo infatti la netta prevalenza *Android 4.1*. Mentre nella parte sottostante della figura Fig.4.10 riportiamo quella che è la dimensione media delle diverse applicazioni. Come osservato precedentemente queste analisi potrebbe ingannare, affermando una possibile contrapposizione a quella svolta precedentemente (sezione 4.2), ma ponendo maggiore attenzione alle diverse labels presenti, si può concludere che per i sistemi operativi *Android 7*, *Android 6* e *Android 8*, ovvero gli ultimi android operating system sviluppati, l'aspetto di ottimizzazione è venuta leggermente meno a fronte di hardware sempre più performante.

4.9 Analisi Varie

Seguendo la medesima Road Map utilizzata per i tool precedenti, abbiamo nuovamente ripetuto l'analisi relativa ai parametri marginali del nostro dataset, al fine di avere un'ulteriore conferma di ciò che è già emerso. In particolare abbiamo nuovamente ripetuto le analisi relative al *Content Rating Category*. Infine, si sono riportati i primi 10 sviluppatori per numero di applicazioni pubblicate nel *Play Store di Google* e le prime 10 applicazioni per numero di installazioni. Tali analisi sono rappresentati nella figura seguente Fig.4.11.



Figura 4.11: Various Analysis

Osservazioni

Si può affermare che le analisi viste in figura Fig.4.11 sono le medesime di quelle esposte nei capitoli precedenti. In conclusione, il Content Rating afferma l'esistenza di vastità di applicazioni destinate a tutti. Il parametro Editor Choice, non ci fornisce utili informazioni sul mercato delle applicazioni, se non sul fatto che è un parametro di ottimalità delle stesse e quindi un certificato di valore.

Tra le applicazioni più utilizzate risultano presenti invece in numero molto elevato le applicazioni relative all'ecosistema Google, questo è dimostrabile in quanto già installate nella gran parte dei smartphone con sistema operativo Android.

Osservando l'istogramma in basso possiamo nuovamente osservare come ci siano pochi sviluppatori che hanno sviluppato un elevato numero di applicazioni rispetto al resto delle applicazioni.

4.10 Utilizzo di R in PowerBI

Come già evidenziato precedentemente, in PowerBI è possibile utilizzare il linguaggio R per creare nuovi "oggetti visivi". A tal punto si è intrapresa la realizzazione di un semplice plot, che ci permettesse di osservare le varie applicazioni sviluppate nel corso degli anni. Nella figura Fig. 4.12 è possibile osservare il codice utilizzato a tale scopo. Mentre nella figura Fig. 4.13 è osservabile il plot realizzato.

```

1 #Il codice seguente consente di creare un dataframe e rimuovere righe duplicate
  e viene sempre eseguito e funge da preambolo per lo script:
2 #dataset <- data.frame(App_Id, Anno, Trimestre, Mese, Giorno)
3 #dataset <- unique(dataset)
4 #Incollare o digitare qui il codice dello script:
5
6 group_sum <- aggregate(dataset$'App_Id', list(dataset$Anno), length)
7 colnames(group_sum) <- c("Anno", "Num_App")
8 #plot(group_sum)
9
10 # Create the data for the chart
11 H <- group_sum$'Num_App'
12 M <- group_sum$'Anno'
13 # Create the Bar Plot
14 barplot(H, names.arg=M, xlab="Year", ylab="Num_App", col="green", main="N App
  Released per Year", border="yellow")

```

Figura 4.12: R Code

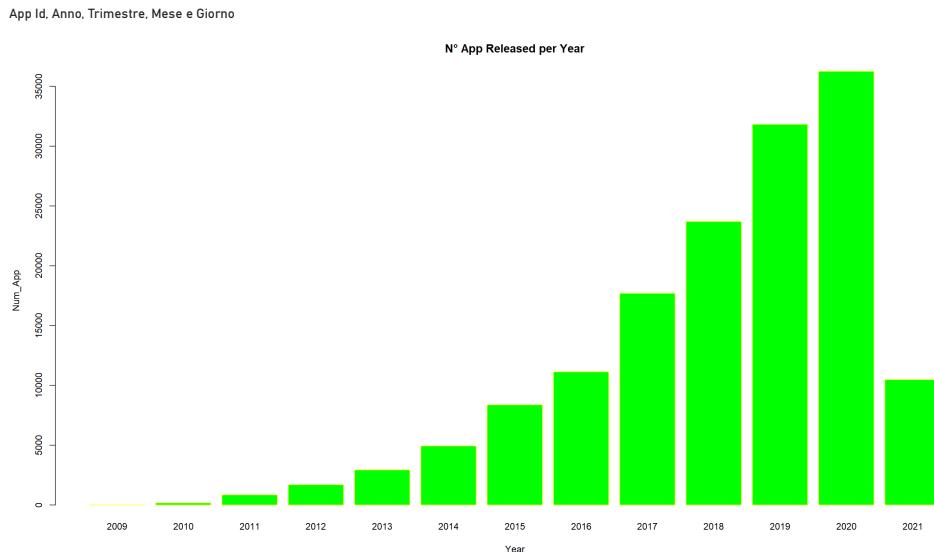


Figura 4.13: Bar Plot sviluppato in R

Osservazioni

Dal grafico di figura Fig. 4.13 possiamo osservare, come già detto nelle analisi precedenti, che negli ultimi anni c'è stato un netto aumento del numero di applicazioni sviluppate. Questo

può essere dovuto anche al periodo di pandemia nel quale si è reso necessario anche l'ausilio di applicazioni mobile per offrire servizi che prima erano forniti di persona.

Il valore esiguo per l'anno 2021 è dovuto ai dati parziali per tale anno, il dataset risulta infatti aggiornato per l'ultima volta a metà 2021.

4.11 ARIMA

Il metodo della media mobile autoregressiva integrata (*AutoRegressive Integrated Moving Average - ARIMA*) ci permette di modellare il passo successivo nella sequenza come funzione lineare delle diverse osservazioni e degli errori residui ottenuti dal calcolo della media nelle fasi precedenti. Ecco quindi che arima combina sia il modello di Autoregressione (AR) che quello di media mobile (MA), a cui aggiunge una fase di pre-processing, chiamata integrazione (I), che ha l'obiettivo di rendere stazionaria la sequenza. La notazione per il modello comporta la specificazione dell'ordine per i modelli AR(p), I(d) e MA(q) come parametri di una funzione ARIMA; ad esempio ARIMA(p, d, q). Nel nostro caso

$$ARIMA : (p, d, q) \rightarrow (0, 1, 2) \quad (4.1)$$

Va inoltre osservato che **ARIMA** è adatto per serie temporali univariate con trend e senza componenti stagionali, come nel nostro caso di studio. Il tutto è osservabile nella figura Fig. 4.14

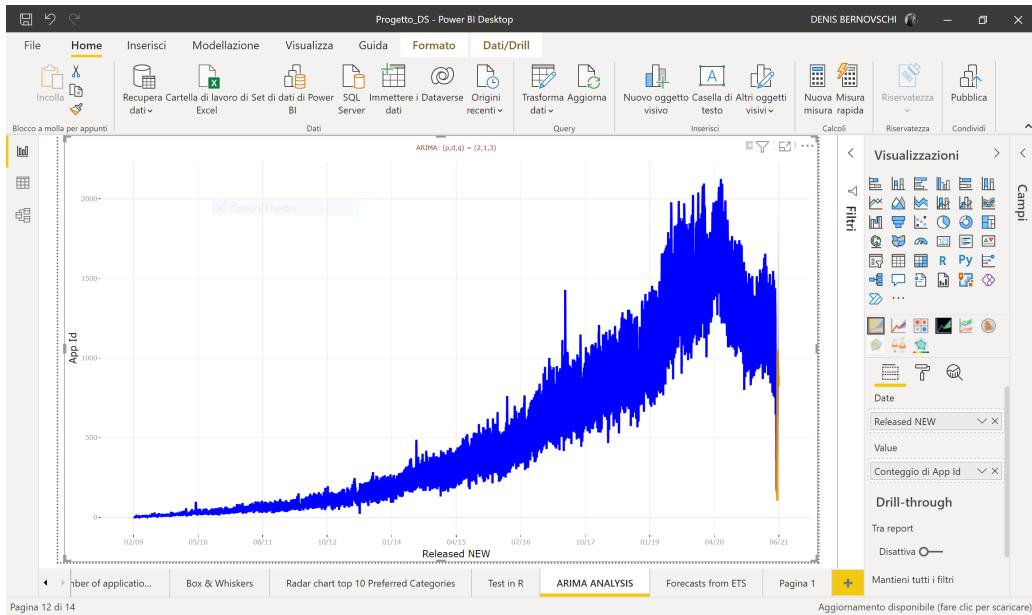


Figura 4.14: Analisi con ARIMA

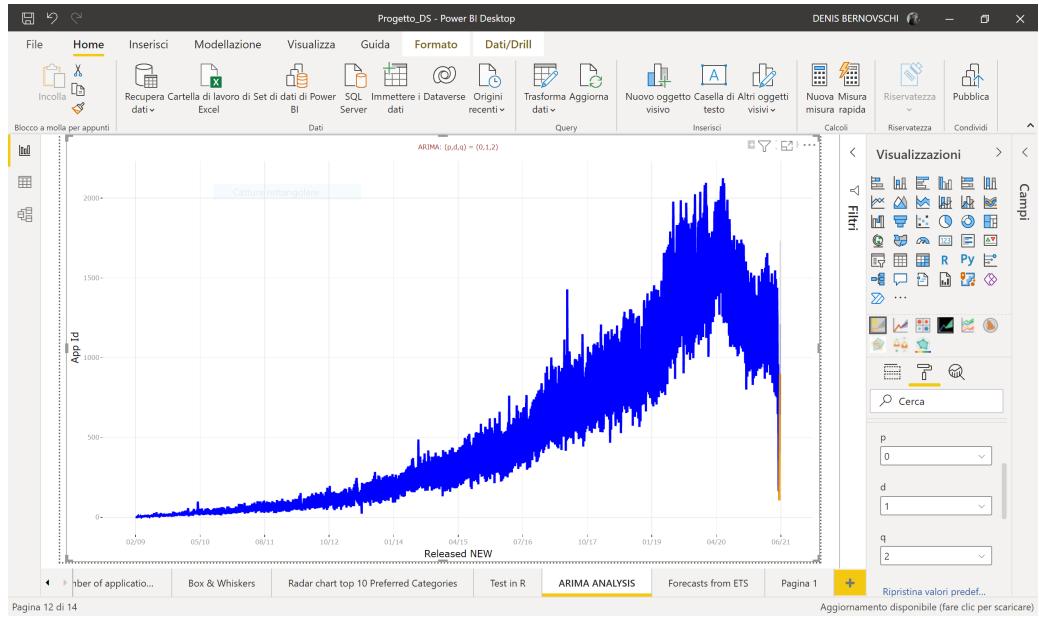


Figura 4.15: Analisi con ARIMA pt.2

Osservazioni Osservando l'analisi nelle figure Fig.4.14 e Fig.4.15 possiamo evidenziare come non ci sia una componente stagionale legata allo sviluppo delle applicazioni. Un'ulteriore osservazione può essere fatta considerando i parametri impostati per tali analisi, abbiamo quindi optato per una duplice verifica, impostando da prima i parametri in maniera automatica (calcolati dal tool **PowerBI**), e poi, successivamente abbiamo attuato un *fine tuning* di questi parametri, ma lo studio qui approfondito riporta le medesime considerazioni.

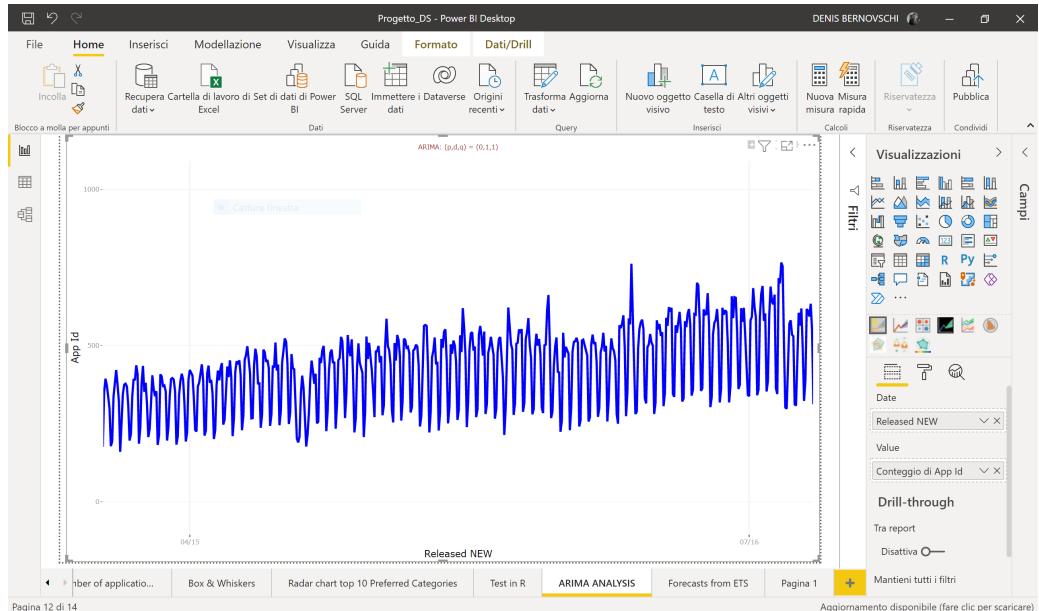


Figura 4.16: Dettagli Analisi con ARIMA

Osservando la figura Fig.4.16, il mercato legato allo sviluppo di applicazioni non ha nessuna componente stagionale e inoltre presenta una serie temporale non stazionaria, abbiamo continu

picchi positivi e negativi in un determinato range di valori. Concludendo, nel complessivo nella figura Fig.4.15 è deducibile che il trend è positivo.

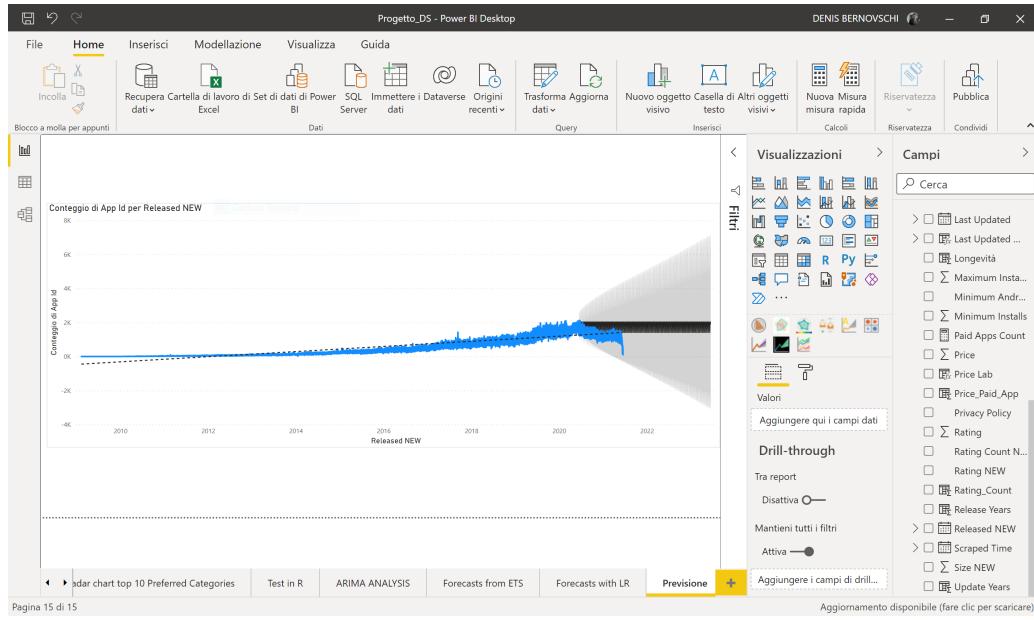


Figura 4.17: Previsione 2021-2024

A fine di dimostrare quanto detto, nella figura Fig.4.17 abbiamo effettuato delle previsioni per poter osservare come si evolverà nel futuro tale mercato. Tale studio è sviluppato con un intervallo di confidenza del 85% ed escludendo i dati relativi all'ultimo anno, per i motivi già espressi precedentemente. Possiamo quindi dire che il mercato dello sviluppo di applicazioni *mobile* è un mercato fiorente e che nel decennio analizzato ha un trend positivo. Inoltre, considerando il continuo percorso di digitalizzazione e di innovazione, possiamo affermare che nel corso dei prossimi anni, il trend non potrà che aumentare.

4.12 Analisi Correlazioni

In questa sezione è riportata l'analisi realizzata per osservare eventuali correlazioni fra i dati delle varie applicazioni *mobile* analizzate.

Questo ci consentirà di dire se e quali fattori influenzano e sono influenzati dagli altri.

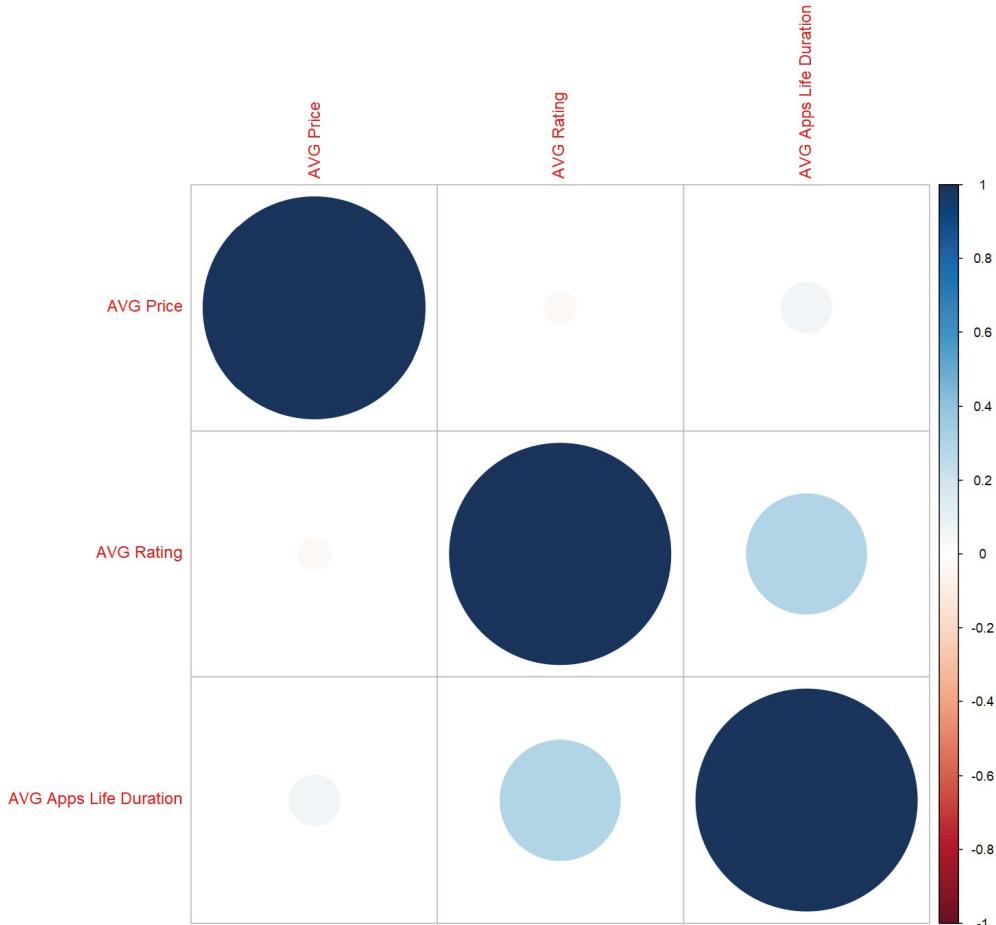


Figura 4.18: Correlazioni

Osservazioni Come può essere osservato dalla figura 4.18 esiste una correlazione diretta fra il *rating* medio e la durata di vita media delle applicazioni e fra il prezzo medio e la durata di vita media delle applicazioni, ma con intensità minore. C'è invece una relazione inversa fra il prezzo medio e il rating medio delle applicazioni.

Il fatto che si ha una correlazione diretta di entità maggiore fra il *rating* medio e la durata di vita media delle applicazioni è un aspetto molto interessante, di fatto le applicazioni con un valore medio di *rating* maggiore sono propense ad avere una durata di vita media più elevata. Questo può trovare conferma dal fatto che un utente fra applicazioni che forniscono servizi simili è più propenso ad installare l'applicazione con un rating medio più elevato e questo determinerà sicuramente una durata di vita dell'applicazione maggiore. Inoltre, eventuali investitori saranno più propensi ad investire su applicazioni valutate positivamente dagli utenti rispetto le altre.

Di entità leggermente minore è la correlazione presente fra il prezzo medio e la durata di vita media, questo conferma il fatto che anche il prezzo influenza, anche se con forza minore, la durata di vita delle applicazioni. Può trovare conferma dal fatto che gli sviluppatori di applicazioni a

pagamento hanno più risorse economiche per portare avanti lo sviluppo delle stesse per più anni. La correlazione inversa fra il rating medio e il prezzo medio ci potrebbe suggerire che gli utenti che acquistano un'applicazione hanno aspettative più elevate rispetto alle applicazioni free, quindi rilasciano un rating più neutro/negativo. Questo fenomeno aumenta con l'aumentare del prezzo medio.

Capitolo 5

Conclusioni Qlik, Tableau e Power BI

Il main goal di questa relazione è permetterci di conoscere e utilizzare i maggiori tool di business intelligence con lo scopo di analizzare quello che è il mercato delle Android Apps. Alla conclusione di questo progetto è possibile affermare che l'obiettivo prefissato è stato raggiunto.

Ogni tool ci ha permesso di condurre diverse analisi sul dataset in esame, a partire da analisi più descrittive tramite il tool **Qlik**, ad analisi sempre più elaborate come le previsioni tramite **Tableau**, fino ad arrivare al **PowerBI** software, il quale ci ha permesso di integrare anche linguaggi di programmazione come *R* e *Python* per creare e manipolare diversi oggetti visivi al suo interno.

Inoltre, al fine di comprendere e conoscere meglio i software sotto esame, la fase relativa al *pre-processing* è stata svolta internamente al tool tramite metriche e/o misure. Il che potrebbe aver reso la strada in salita ma al contempo ci permette di sfruttare a pieno le funzionalità messe a disposizione dai vari software. Va inoltre osservato come del trio dei tool analizzati PowerBi è quello che offre maggiori potenzialità e una maggior *customizzazione* dei vari plot realizzabili al suo interno sia tramite il market place che tramite i linguaggi di programmazione. Tra le differenze utili allo sviluppo di DashBoard, Racconti e Presentazioni possiamo annoverare la possibilità di lavorare online (QLik) e offline (PowerBI e Tableau)¹. È inoltre possibile organizzare dashboard e plot in modo separato e/o unito. Come aspetto aggiuntivo, gli applicativi offrono la possibilità di applicare tecniche di filtraggio internamente tramite un qualsiasi plot chart presente, oppure applicando direttamente un oggetto visivo "filtro" che svolge tale compito. In sintesi, se dovessimo organizzare i software secondo un classifica che tenga conto sia delle prestazioni che delle funzionalità, al primo posto troveremmo *PowerBI*, mentre al secondo posto ci sarà *Tableau* ed infine *QLik*². Alla conclusione di questo progetto possiamo affermare che nonostante le differenze e le diverse sfaccettature dei tools fin qui visti, essi ci hanno permesso di raggiungere il nostro obiettivo, ovvero estrarre conoscenza e informazione dal dataset preso in esame.

¹é possibile farlo anche con loro, ma gli step da fare e attuare sono molto più macchinosi e costosi (e.g. PowerBI for Mac ha un costo elevato)

²La classifica non tiene conto delle prestazioni computazionali in quanto soggettive e dipendenti dalla macchina su cui sono svolte tali analisi

Capitolo 6

Python

Per lo sviluppo del progetto di Data Science oltre ai tool precedentemente analizzati, abbiamo posto il focus sul linguaggio Python. Esso, risulta tra i linguaggi maggiormente utilizzati nell'ambito della Data Science, seguito a breve distanza dal linguaggio introdotto in **PowerBI** ovvero *R*. Python mette a disposizione una serie di librerie particolarmente ampie ed efficaci come *Pandas*, *Seaborn*, *Matplotlib*, *Sklearn*, *Statsmodel*, e tante altre. Le librerie menzionate precedentemente sono totalmente **free** e godono di un ottimo supporto da parte della community. Nel progetto qui sviluppato vedremo l'utilizzo delle diverse librerie, a partire dai task iniziali ma fondamentali come l'ETL¹. La versione di **Python** utilizzata è la 3.7.12



Figura 6.1: Logo Python

6.1 Serie Temporali

6.1.1 Dataset

Il dataset², utilizzato per operare con le serie temporali, contiene le informazioni sul consumo elettrico della Repubblica di Panama.

In particolare, all'interno del set di dati utilizzato sono racchiuse le informazioni relative al consumo elettrico nazionale e i parametri meteorologici come l'umidità, la temperatura, le precipitazioni e la ventosità di tre principali città di Panama. Inoltre vi sono diversi parametri per indicare il periodo di festività, l'identificatore relativo alla festività e se è un periodo scolastico o meno.

Struttura del Dataset

La struttura del dataset come precedentemente accennato, contiene le informazioni relative al consumo elettrico nazionale di Panama e i parametri meteorologici di tre principali città. Durante

¹Extract, Transform, Load - Processo di estrazione, trasformazione e caricamento dei dati in un sistema

²<https://www.kaggle.com/ernestojaguilar/shortterm-electricity-load-forecasting-panama?select=continuous+dataset.csv>

```

    TQL_dav'   T2M_san'      W2M_dav'
    year'      school'  QV2M_dav'   Holiday_ID'
    'T2M_dav'  'T2M_toc'   QV2M_san'   TQL_toc'
    W2M_toc'  timeseries' quarter'  nat_demand' holiday'
    QV2M_toc'          'W2M_san'   data'   TQL_san'

```

Figura 6.2: WordCloud Plot

la fase di importazione tramite la libreria **Pandas** e la funzione **read csv** è possibile specificare i *dtypes* per ciascuna colonna del file CSV³. Questa opzione ci permette quindi di evitare operazioni di ETL più complesse in seguito all'importazione.

Tra le principali operazioni svolte all'inizio sul dataset, per adattarlo alle esigenze richieste, possiamo citare:

1. Scelta opportuna dei dtypes;
2. Manipolazione temporale (trasformazione da Timestamp a Datetime) tramite opportune operazioni di splitting e casting;
3. Creazione dell' index opportuno contenente le informazioni relative all'anno e al quartile

```

1 dtypess={
2     'datetime' : 'str',
3     'nat_demand' : 'float',
4     'T2M_toc' : 'float',
5     'QV2M_toc' : 'float',
6     'TQL_toc' : 'float',
7     'W2M_toc' : 'float',
8     'T2M_san' : 'float',
9     'QV2M_san' : 'float',
10    'TQL_san' : 'float',
11    'W2M_san' : 'float',
12    'T2M_dav' : 'float',
13    'QV2M_dav' : 'float',
14    'TQL_dav' : 'float',
15    'W2M_dav' : 'float',
16    'Holiday_ID' : 'int64',
17    'holiday' : 'boolean',
18    'school' : 'boolean'
19 }
20
21 dataframe = pd.read_csv('/content/drive/MyDrive/ProgettoDS/continuous dataset.csv', sep=',', header=0, dtype=dtypess)
22 print('N° Elementi : {}'.format(len(dataframe)))
--
```

Figura 6.3: Utilizzo di dtypes

Il risultato finale è visionabile nella tabella Tab.6.1 . Invece, nella figura Fig.6.4 è osservabile il medesimo dataset a seguito delle varie modifiche accennate precedentemente

	nat_demand	T2M_toc	QV2M_toc	TQL_toc	W2M_toc	T2M_san	QV2M_san	TQL_san	W2M_san	T2M_dav	QV2M_dav	TQL_dav	W2M_dav	Holiday_ID	holiday	school	timeseries	year	quarter	data
2015-01-03 01:00	970.3450	25.865299	0.018576	0.016174	21.850544	23.482446	0.017272	0.001855	10.328949	22.662139	0.016562	0.096100	5.364148	0	False	False	2015Q1	2015	1	2015-01-03
2015-01-03 02:00	912.1755	25.899255	0.018653	0.016418	22.166944	23.399255	0.017265	0.001327	10.681517	22.578943	0.016509	0.087646	5.572471	0	False	False	2015Q1	2015	1	2015-01-03
2015-01-03 03:00	900.2688	25.937280	0.018768	0.015480	22.454911	23.343530	0.017211	0.001428	10.874924	22.531039	0.016479	0.078735	5.871184	0	False	False	2015Q1	2015	1	2015-01-03
2015-01-03 04:00	889.9538	25.957544	0.018890	0.016273	22.110481	23.238794	0.017128	0.002599	10.518620	22.512231	0.016487	0.068390	5.883621	0	False	False	2015Q1	2015	1	2015-01-03
2015-01-03 05:00	893.6865	25.973840	0.018881	0.017281	21.186068	23.075403	0.017059	0.001729	9.733589	22.481650	0.016456	0.064365	5.611724	0	False	False	2015Q1	2015	1	2015-01-03

Figura 6.4: Dataset post-preprocessing

³CSV ≡ Comma Separated Values

Column Name	Description	Unit
Date time	Datetime corrispondente di Panama time-zone (UTC-05:00)	Datetime
Nat Demand	Carico Elettrico Nazionale	MWh
TM2_toc	Temperatura a 2m (sul livello del mare) in Tocumen	°C
QV2M_toc	Umidità a 2m in Tocumen	%
TQL_toc	Precipitazioni a Tocumen	l/m^2
W2M_toc	Velocità del vento a 2m in Tocumen	m/s
TM2_san	Temperatura a 2m in Santiago	°C
QV2M_san	Umidità a 2m in Santiago	%
TQL_san	Precipitazioni a Santiago	l/m^2
W2M_san	Velocità del vento a 2m in Santiago	m/s
TM2_dav	Temperatura a 2m in David	°C
QV2M_dav	Umidità a 2m in David	%
TQL_dav	Precipitazioni a David	l/m^2
W2M_dav	Velocità del vento a 2m in David	m/s
Holiday ID	Identificatore della Festività	int
Holiday	Parametro Booleano per indicare o meno il periodo di festività	booleano
School	Parametro Booleano per indicare o meno il periodo scolastico	booleano

Tabella 6.1: Struttura del Dataset

6.1.2 Visualizzazione dei Dati

Una volta conclusa la fase di *pre-processing*, abbiamo intrapreso la fase di *visualization* dei dati in esame, ecco quindi che per una maggior comprensione dei dati utilizzati, siamo andati a creare alcuni *plot* tramite le librerie **matplotlib** e **seaborn**.

Period Range

```

1 df_time['datetime'] = pd.to_datetime(df_time['datetime']).dt.to_period('H')
2 idx = pd.period_range(df_time['datetime'].min(),
3                     df_time['datetime'].max(), freq='H')
4 df_time = df_time.set_index('datetime').reindex(idx, fill_value=0)

```

Nella parte di listato superiore abbiamo introdotto quella che è la parte fondamentale dell'operato che segue nelle prossime sezioni. Si è creato l'index del nostro dataset, mantenendo le informazioni relative ai vari parametri meteorologici. Il parametro *H* specifica che la serie temporale ha una frequenza oraria.

Visualizzazione dei dati meteorologici per ciascuna città

Nella figura Fig.6.5 abbiamo rappresentato i vari parametri meteorologici di ciascuna città presente nel dataset. A partire dall'alto troviamo le informazioni della città di Tocumen, passando per la città di Santiago e concludendo con la città di David.

Note Utili Al fine di contenere tutti i parametri all'interno di un singolo plot, i dati relativi all'umidità e alle precipitazioni risultano pressoché nulli, Infatti nella città di Tocumen l'umidità media è di circa **0.02%** e le precipitazioni sono circa **0.08l/m²** e nelle altre città tali valori non si discostano particolarmente.

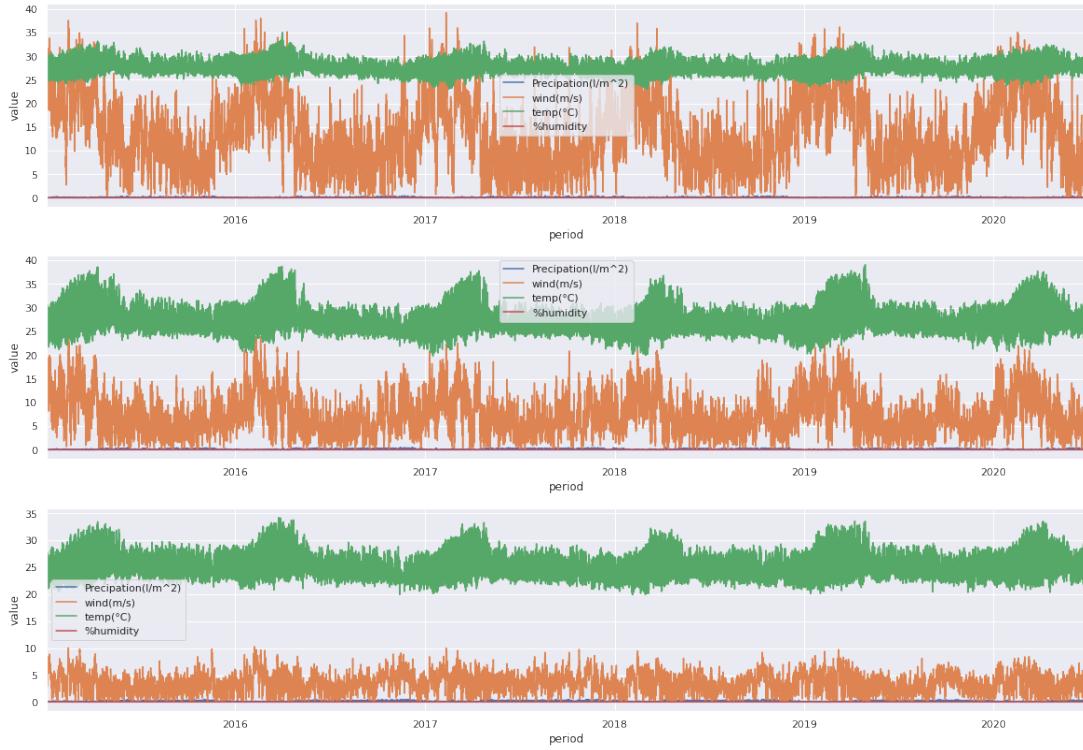


Figura 6.5: Parametri Meteorologici di Tocumen, Santiago, David

Visualizzazione del carico elettrico della Repubblica di Panama

Nel grafico mostrato in figura Fig.6.6 abbiamo rappresentato quello che è il carico elettrico di Panama in ordine temporale. Va precisato che le misurazioni del carico elettrico panamense è misurato giornalmente con 24 misurazioni puntuali, una per ogni ora della giornata. Ecco quindi chiarito il motivo dei continui picchi *UP/DOWN*.

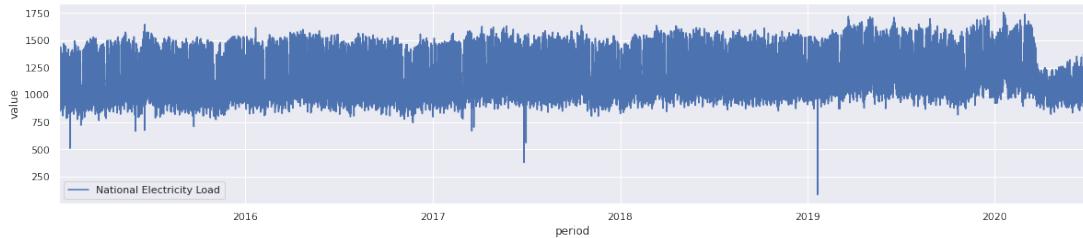


Figura 6.6: Carico Elettrico della repubblica Panamense

Visualizzazione del nat demand su base Mensile, Giornaliera e Oraria Al fine di comprendere al meglio come il carico elettrico nazionale varia in base al giorno della settimana, quindi di riuscire a comprendere se vi è una variazione tra periodo feriale e periodo festivo, ovvero di capire come le festività influenzano il carico, abbiamo deciso di intraprendere un'analisi tramite l'ausilio di plot di tipologia **box-plot**. Ecco quindi che nella figura Fig.6.7 e Fig.6.8 si è visualizzato il giorno della settimana (1 ≡ Lunedì) etc. e il relativo carico, distinguendo tra periodo festivo e periodo feriale, analogo il discorso per il carico mensile.

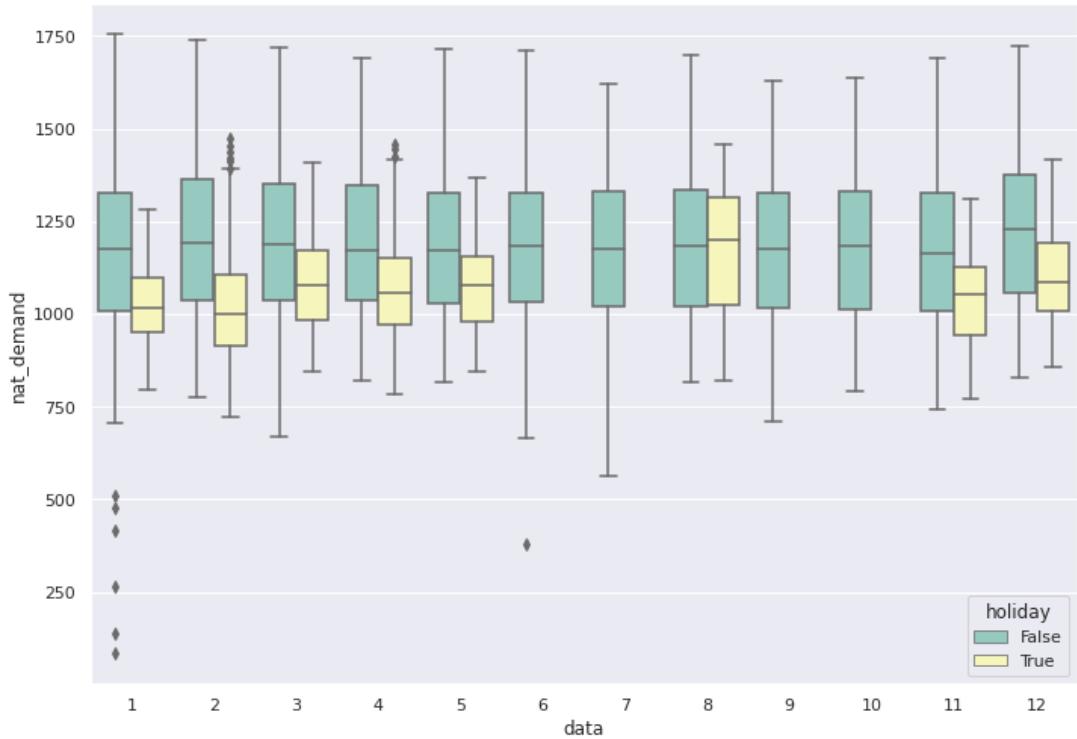


Figura 6.7: Carico Elettrico su base mensile

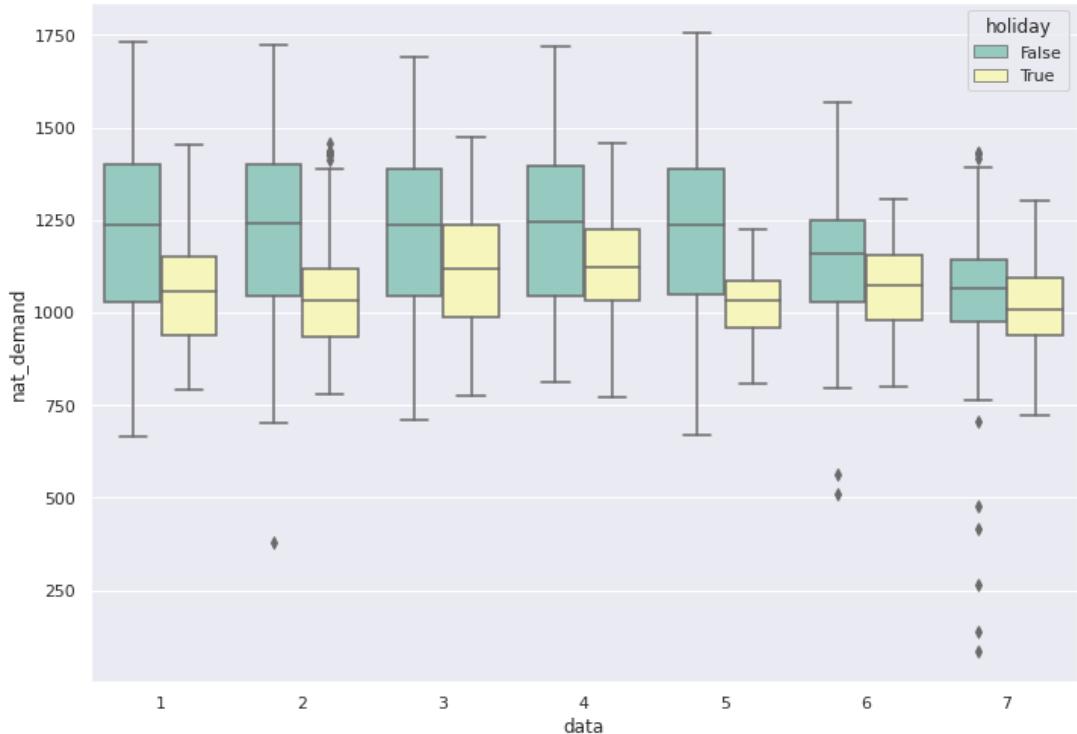


Figura 6.8: Carico Elettrico su base settimanale

Infine, per ulteriore livello di dettaglio, abbiamo analizzato il carico nazionale panamense su base oraria, facendo le medesime suddivisioni introdotte precedentemente. Il tutto è osservabile

nella figura Fig. 6.9.

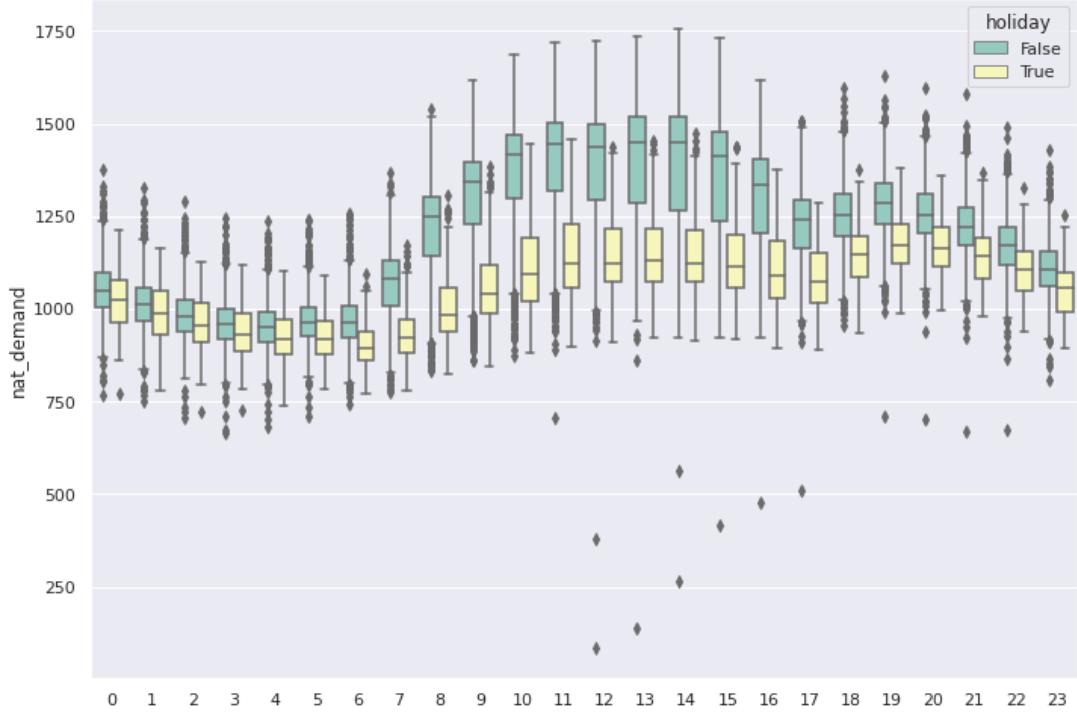


Figura 6.9: Carico Elettrico su base oraria

Osservazioni Come possiamo notare dalle figure soprastanti l’andamento del carico elettrico nazionale è pressoché costante. In particolare se guardiamo la figura Fig.6.7 il suo andamento nel corso dell’anno oscilla leggermente in un range ben definito che è di circa $800 \div 1500$ MWh. Si osserva che vi è un chiaro andamento nel corso della singola giornata, di cui nel periodo notturno $00 : 00 \div 06 : 00$ il carico elettrico scende toccando il picco di minimo; dopodichè nella fascia $07 : 00 \div 17 : 00$ si ha un andamento crescente nella prima metà e successivamente tende a scendere nella seconda metà; mentre nell’ultima parte della giornata tendiamo nuovamente ad un trend che scende lentamente. Questo tipo di andamento è simile a quello che ci aspettavamo, considerando che le strutture pubbliche come ospedali, enti pubblici, stazioni di polizia, etc.., hanno un impatto sul carico nazionale costante, mentre la variabilità è data dalle imprese, dai privati e dai locali di vita mondana. Va inoltre osservato come questo andamento è paragonabile alle offerte che molte compagnie di energia elettrica offrono ai propri consumatori, essi infatti prevedono un prezzo per ciascuna fascia oraria. Una fascia nel pieno della giornata avrà un costo molto più elevato rispetto al costo previsto per una fascia oraria ove il carico nazionale registrato è nettamente inferiore.

6.1.3 Correlazioni

Al termine dell’analisi descrittiva dei vari parametri all’interno del dataset si è intrapreso lo studio delle possibili correlazioni tra le variabili meteorologiche e il carico elettrico nazionale. Tramite la libreria **seaborn** abbiamo utilizzato il metodo *pairplot* che ci ha permesso di osservare a due a due le possibili correlazioni. Nel dettaglio, nelle figure Fig.6.10 Fig.6.11 si è riportato suddiviso il dataset sulla base delle città e dei relativi parametri meteorologici. Anche qui è stata fatta una distinzione sul periodo feriale e periodo festivo.

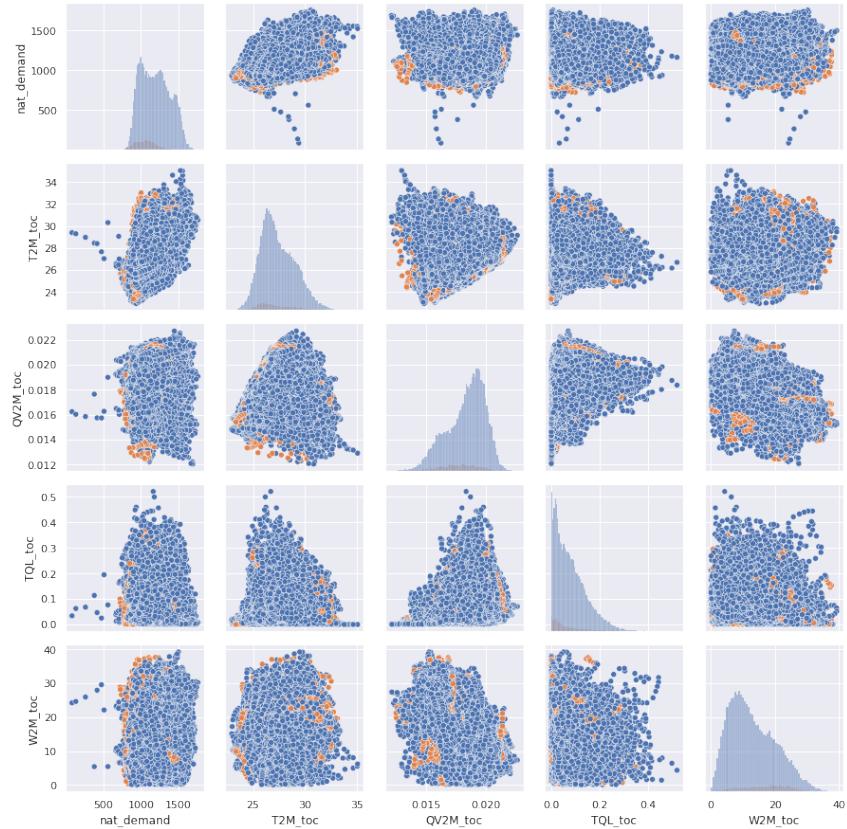


Figura 6.10: PairPlot relativo alla città di Tocumen

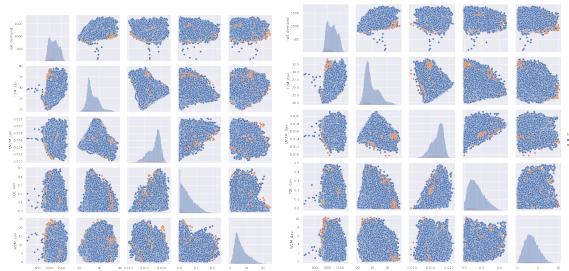


Figura 6.11: PairPlot relativo alle città di Santiago e David

Metodi per il calcolo della correlazione

Per un maggior dettaglio siamo andati ad analizzare nello specifico la correlazione tra i vari parametri espressi precedentemente attraverso ulteriori metodi di correlazione come la correlazione di *Pearson*, *Spearman* e *Kendall Tau*.

```

1 import scipy.stats
2 pearson_r = scipy.stats.pearsonr(df_time['nat_demand'], df_time['T2M_dav'])
3 print('Pearson R : {} '.format(pearson_r[0]))
4
5 spearmanr = scipy.stats.spearmanr(df_time['nat_demand'], df_time['T2M_dav'])
6 print('Spearman R : {}, P-Value {}'.format(spearmanr[0], spearmanr[1]))
7
8 kendalltau = scipy.stats.kendalltau(df_time['nat_demand'], df_time['T2M_dav'])
9 print('Kendall Tau : {}, P-Value {}'.format(kendalltau[0], kendalltau[1]))

```

Successivamente abbiamo adottato un ulteriore metodo, ovvero il metodo **corrcoef** della libreria **numpy**, il quale unito al metodo **heatmap** della libreria **seaborn** ci ha fornito un'ottima visualizzazione grafica per osservare nel dettaglio la correlazione tra le variabili analizzate. Il tutto è osservabile nella figura Fig. 6.12.

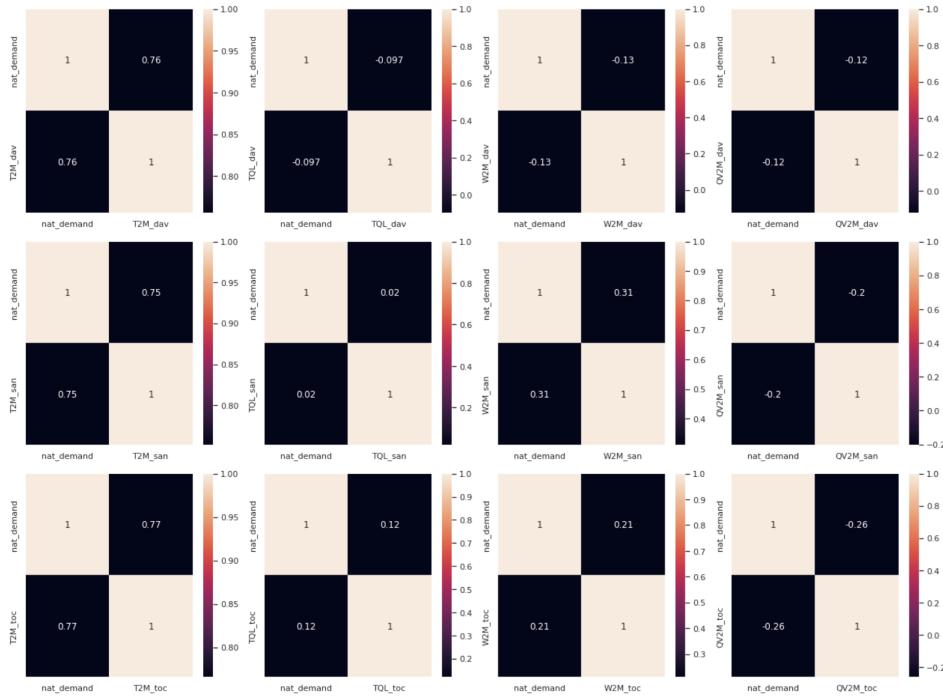


Figura 6.12: Analisi della corellazione tramite il metodo CorrCoef

Al fine di dimostrare la *presenza o meno* della correlazione, abbiamo utilizzato un ulteriore metodo messo a disposizione da *seaborn*, ovvero lo **scatterplot**. In particolare tramite il listato seguente, abbiamo analizzato sia la media che la varianza delle singole variabili utilizzate poi nello scatter plot. Nelle figure Fig. 6.13 e Fig. 6.14 è osservabile quanto detto sopra.

Note Utili Per brevità riportiamo solo alcuni dei plot

```

1 #-----std & mean-----
2 print('mean : {} stdv : {}'.format(np.mean(df_time['nat_demand']), np.std(
3 df_time['TQL_dav'])))
4 #-----plot-----
4 sns.scatterplot(x=df_time['TQL_dav'], y=df_time['nat_demand'])

```

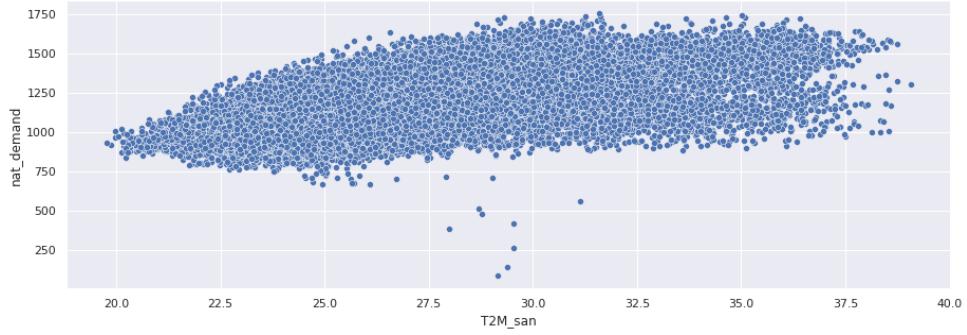


Figura 6.13: Scatter Plot Temperatura e Net Demands

La figura Fig.6.13 ci permette di osservare come vi sia una forte correlazione tra la temperatura e il carico elettrico (fino a circa 32.5 gradi), questo è deducibile dal fatto che osservando attentamente la figura Fig.6.13, lo scatter plot o meglio il *lag plot* rappresenta un andamento tendenzialmente lineare con un trend positivo. Viceversa nella figura Fig.6.14 il fatto che non vi sia una correlazione tra le precipitazioni e il carico elettrico, è altresì osservabile considerando il *lag plot* presente in figura. Va però detto che, in caso in cui vi siano casi dubbi e i metodi già citati precedentemente non permettono di trarre le conclusioni desiderate, è possibile attuare uno *smoothing* del segnale, utilizzando un ulteriore metodo fornito anch'esso da **seaborn** chiamato **lowess** (visibile nella figura Fig.6.15). Esso, prendendo in pasto dati che a primo impatto non sembrano correlati, ci fornisce una visualizzazione grafica di quale è la funzione di correlazione. La complessità di questo modello sta nello specificare correttamente il parametro *frac*.

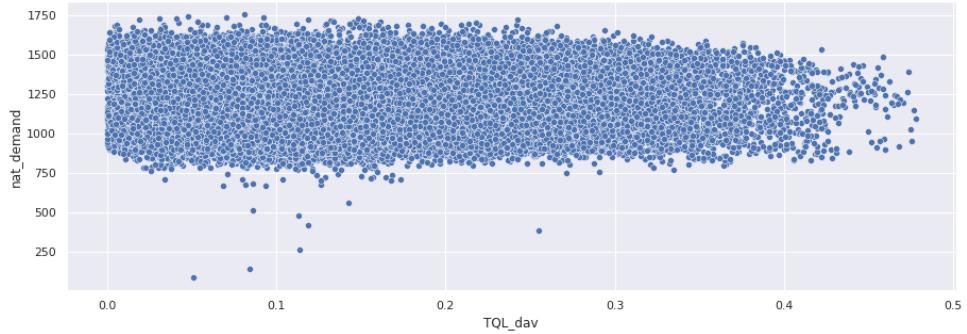


Figura 6.14: Scatter Plot per Precipitazioni e Net Demands

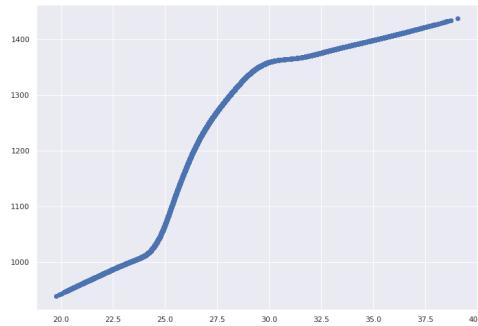


Figura 6.15: Lowess Plot per Temperatura e Net Demands

Osservazioni Nel nostro caso, come abbiamo già più volte detto, la ricerca di una possibile correlazione con il carico elettrico nazionale della repubblica panamense e i fattori ambientali non ha fornito grossi risultati, pur utilizzando i metodo citati prima. L'unica informazione utile estrapolata è quella relativa alla forte correlazione tra temperatura e il carico elettrico. Le altre correlazioni analizzate sono correlazioni deboli e poco utili a comprendere il reale nesso con il carico elettrico.

6.1.4 StatsModels

Tramite la libreria Stats Models abbiamo analizzato l'aspetto temporale del nostro dataset, in particolare tramite modelli come ARIMA e SARIMAX si è osservato quello che è il comportamento della variabile dipendente "nat demand" (national electricity load). Tramite il modello ARIMA ben configurato si è così in grado di fornire delle previsioni sui futuri carichi energetici nazionali. Nella figura Fig.6.6 è possibile osservare l'andamento del carico elettrico nazionale panamense. Una volta verificata l'assenza di dati e la correttezza dell'unità di misura utilizzata, si può ritenere conclusa la fase di analisi dei dati e la loro correttezza.

$$ARIMA \rightarrow (AR, I, MA) \rightarrow (p, q, d) \quad (6.1)$$

Si è partiti con lo studio dei parametri espressi nell'equazione (6.1), nello specifico il parametro d relativo alla stazionarietà della serie è deducibile usando l'*Augmented Dickey–Fuller test*, mentre per quanto riguarda i parametri q e p essi sono deducibili da ulteriori valutazioni di criteri aggiuntivi come l' AIC (Akaike's Information Criterion) o il BIC (Bayesian Information Criterion), che ci permettono di valutare sia la bontà del modello che la sua complessità, e alcuni metodi grafici che ci permettono di valutare l'autocorrelazione e la autocorrelazione parziale. Ecco quindi che osservando la figura Fig.6.16 è deducibile che il parametro d utilizzato sarà $d = 1$, che permette di rendere la serie stazionaria, senza però perdere troppe informazioni. Successi-

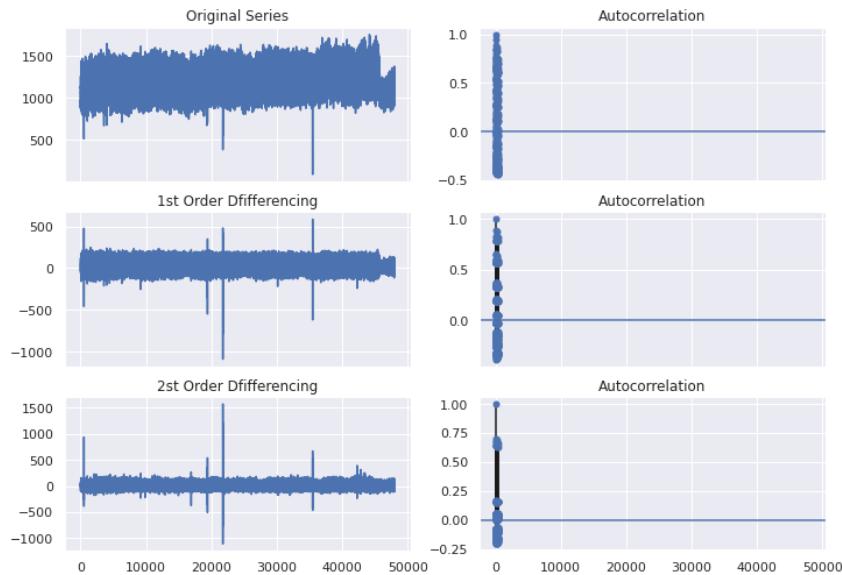


Figura 6.16: Plot di Autocorrelazione e derivate successive

vamente tramite l'autocorrelazione parziale (visibile nella figura Fig.6.17) e la verifica di AIC e BIC siamo andati a impostare il nostro modello di ARIMA, nella fattispecie i parametri scelti sono espressi nell'equazione (6.2)

$$ARIMA \rightarrow (p = 1, q = 0, d = 1) \quad (6.2)$$

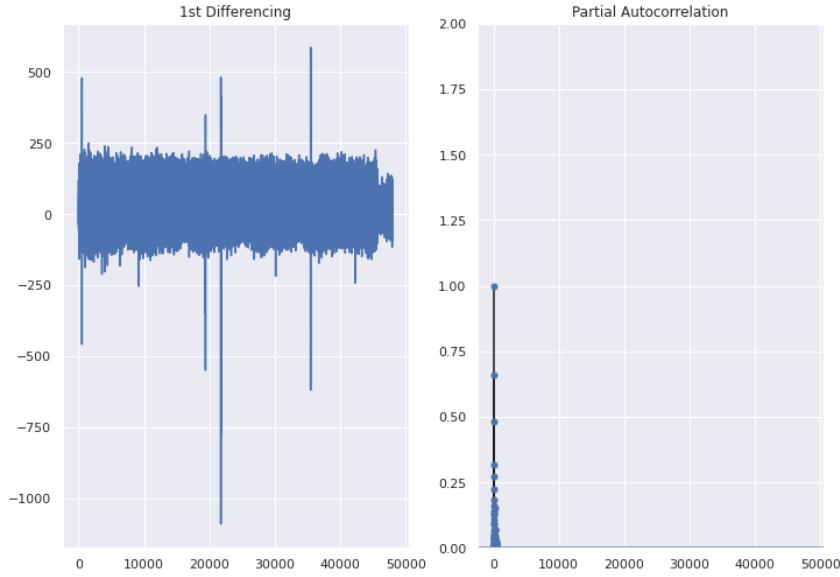


Figura 6.17: Plot di Partial Auto Correlation

6.1.5 Arima

Una volta concluso lo studio dei parametri da passare al modello Arima, siamo andati a configurare quest'ultimo e a verificarne la bontà. In particolare, come si può notare dalla figura Fig.6.18, il modello funziona correttamente e fornisce ottimi risultati sia in termini di bontà che di complessità del modello.

```

1 # fit ARIMA model
2 fig = plt.figure(figsize=(20,8))
3 model = ARIMA(df_time['nat_demand'].values, order=(1,0,1))
4 ax = plt.gca()
5 results = model.fit()
6 plt.plot(df_time['nat_demand'].values)
7 results.plot_predict
8 plt.plot(results.fittedvalues, color='red')
9 ax.legend(['Nat Demand', 'Forecast'])
10 results.summary()

```

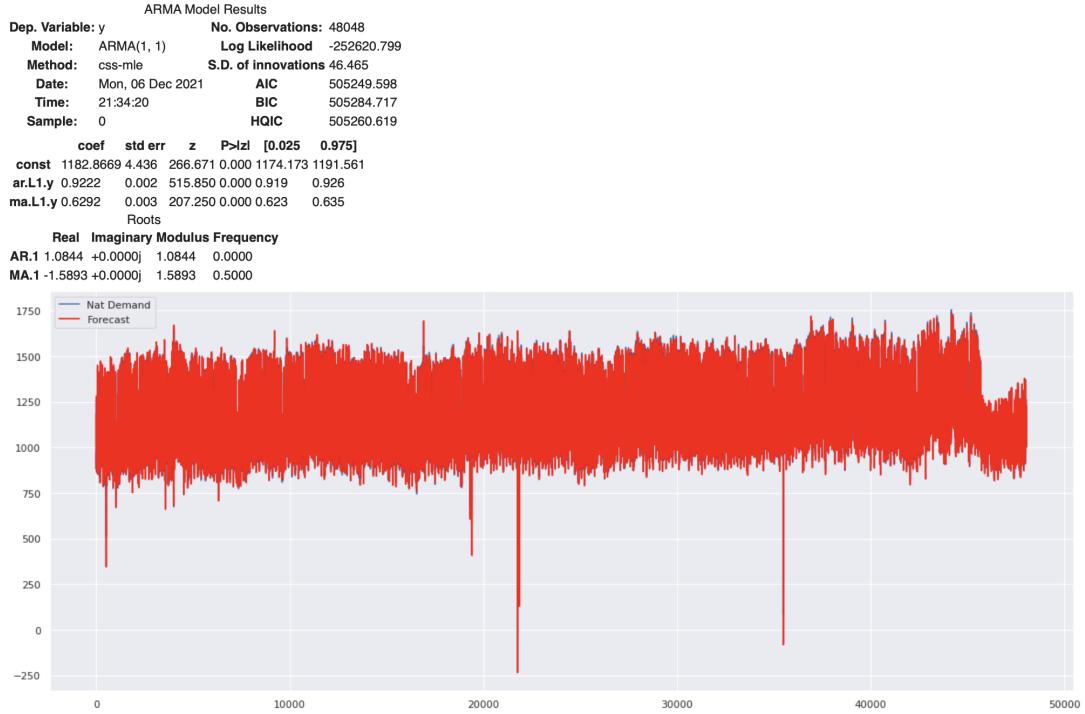


Figura 6.18: Arima Models & Results

Al fine di verificare ulteriormente il corretto funzionamento del modello, abbiamo intrapreso lo studio dei residui. Come si può notare dalla figura Fig.6.19, essa infatti mostra la "variabilità" del modello allenato rispetto alla serie. Si può notare come, eccetto un picco sul range $[-1000, 1000]$, i valori si distribuiscono nel range $[-200, 200]$, con una media sostanzialmente pari a 0, come è possibile vedere anche dal KDE plot mostrato a fianco.

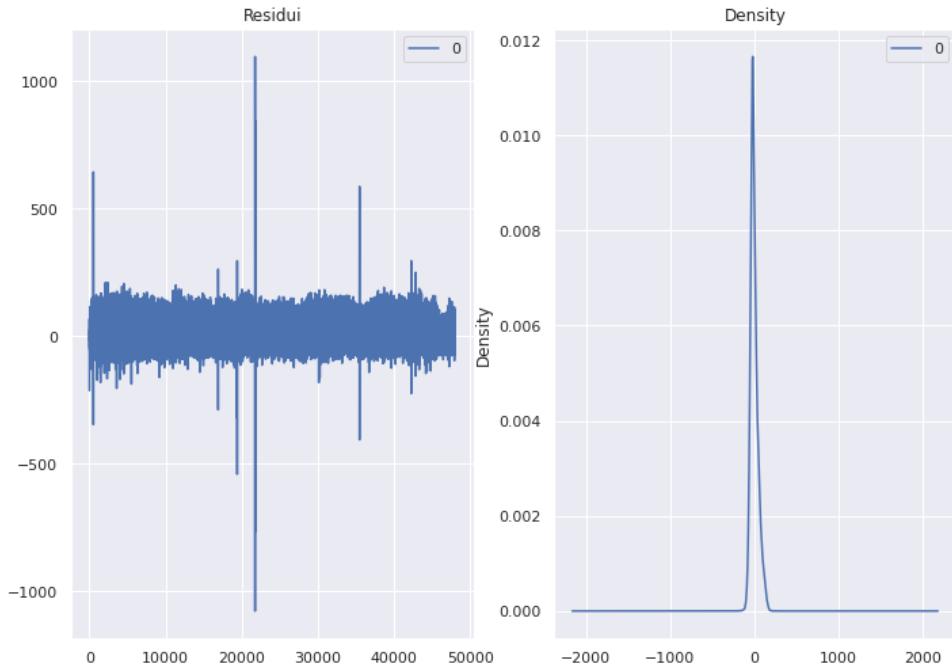


Figura 6.19: Plotting dei Residui

6.1.6 SARIMA

Successivamente si passa alle analisi tramite il modello SARIMAX, il quale tiene conto anche della stagionalità. *Nel nostro caso non è possibile affermare che vi sia una stagionalità mensile, e nemmeno annuale, in quanto i valori sono altamente variabili.* Ecco quindi che nuovamente eseguendo diversi test e verificando via via i parametri BIC e AIC, abbiamo trovato il giusto compromesso tra complessità e bontà dei risultati. Dalle figure seguenti è possibile osservare un leggero incremento del consumo nel corso dei prossimi anni. In figura Fig.6.20 è riportato il risultato finale che giustifica quanto detto.

Note utili Il modello SARIMA è stato utilizzato solo per una maggior completezza dell'elaborato, in quanto, come già detto, è stato parametrizzato in modo analogo al modello ARIMA precedentemente introdotto.

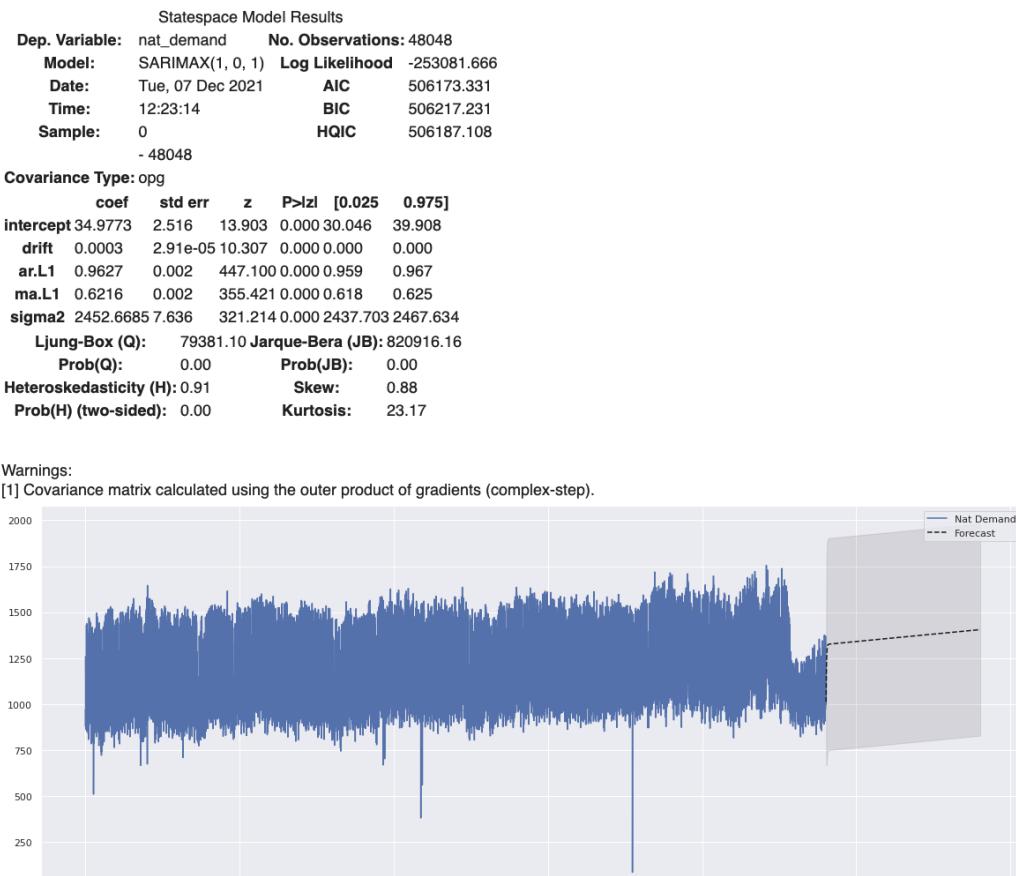
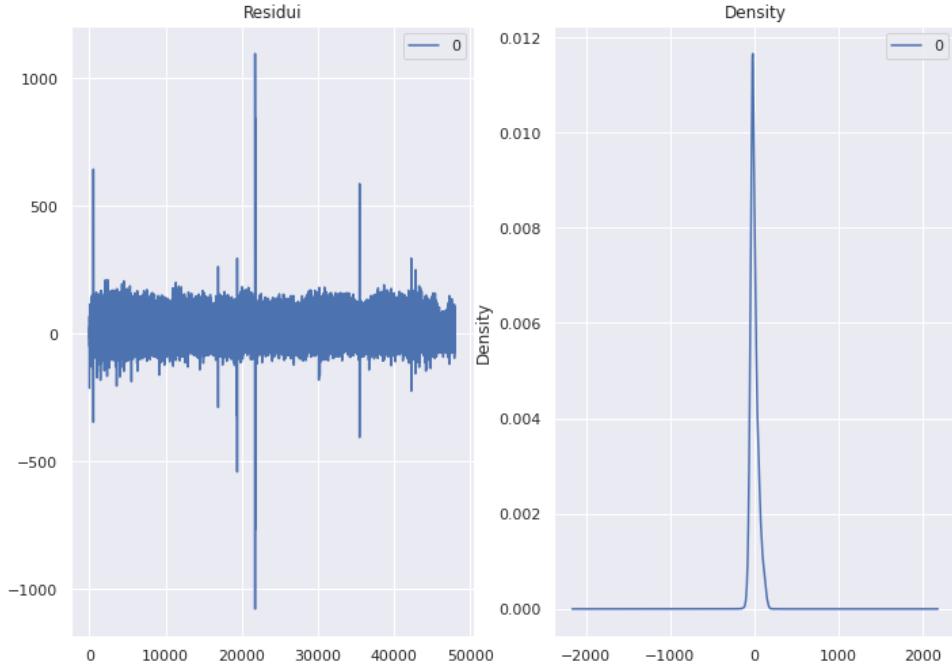


Figura 6.20: Forecasting con SARIMAX

Anche qui, si è intrapreso lo studio dei residui. Come si può notare dalla figura Fig.6.21 la varianza è pressoché costante ed esigua, come nel caso precedente, e quindi conferma di risultati accettabili.

Figura 6.21: *Plotting* dei Residui

6.1.7 Out-of-time cross validation

Per una valutazione più accurata del nostro modello, siamo andati a svolgere la tecnica dell'out-of-time cross validation. Essa richiede la suddivisione della serie di partenza in due subset, uno necessario al training del modello e l'altro necessario alla fase di valutazione, ovvero quella di testing. Nel nostro caso la suddivisione è un $70 \div 30$ ove 70% per la parte di training e 30% per il testing. Tramite questi due subset abbiamo addestrato il modello. Al fine di ottenere un'elevata accuratezza si sono svolti diversi test analizzando i parametri di valutazione citati precedentemente.

Nella figura Fig.6.22 è osservabile quanto detto, in particolare in arancione troveremo la serie predetta, mentre in blu troveremo la serie originale.

Metriche

Per valutare i modelli citati precedentemente siamo andati ad introdurre metriche di valutazione come il *Mean Error*, *Mean Absolute Error*, etc... Per una miglior sintesi riportiamo qui di seguito una tabella (Tab.6.2) contenente i risultati ottenuti.

Error Type	Value
MAPE	0.2570
ME	236.3881
MAE	260.3691
MPE	0.2407
RMSE	305.6469
CORR	0.1121
MINMAX	0.1877

Tabella 6.2: Risultati delle Metriche tramite il modello Out-Of-Time Cross Validation

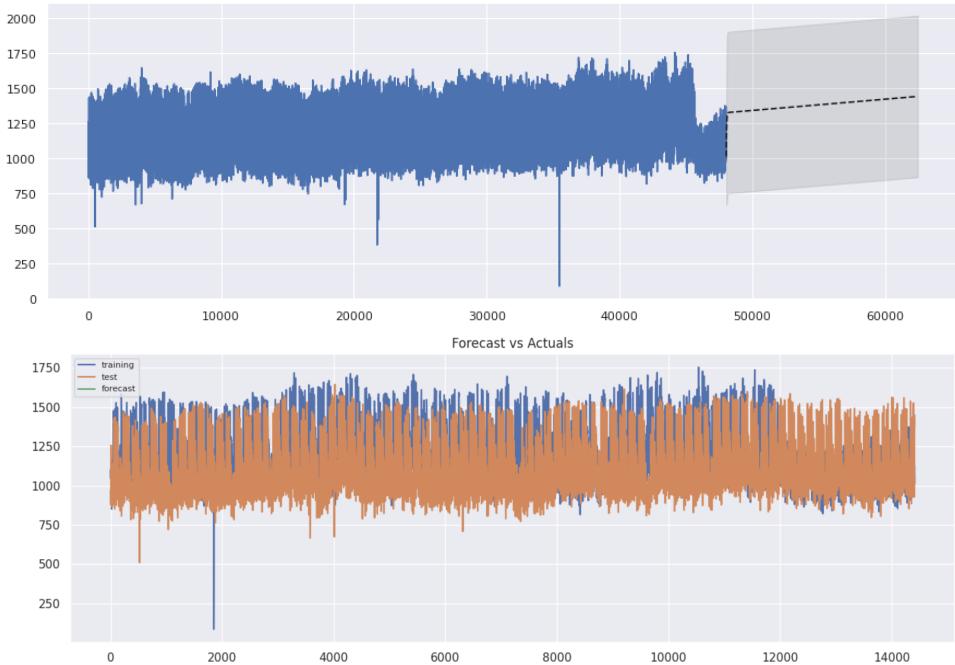


Figura 6.22: SARIMA with out-of-time cross validation

Osservazioni Dati i risultati, è interessante osservare come la correlazione tra i dati predetti e i dati reali risulti essere particolarmente bassa, questo è giustificabile considerando che il valore prevedibile del carico nazionale elettrico è altamente variabile (modifiche di decimali e millesimi). Sono stati svolte anche operazioni di pre-processing come lo *scaling* delle features in range di valori, rendendo però così il contenuto informativo poco significativo.

Inoltre, al fine di mantenere il modello bilanciato il più possibile tra bontà e complessità si è preferito non modificare ulteriormente il modello. A dimostrazione di tale tesi, anche i parametri BIC, AIC sono in linea con i modelli precedentemente citati, questo ci permette quindi di confermare le conclusioni tratte. Quindi, considerando nuovamente le motivazioni già citate (come l'altissima variabilità del dataset), si confermano i risultati ottenuti con il ME e il MAE.

In conclusione, il modello è in grado di prevedere il carico elettrico nazionale futuro, con però un'accuratezza esigua, la quale però permette comunque di affermare come il trend risulti sicuramente non decrescente, ma allo stesso tempo pressoché stazionario nel tempo.

6.2 Cluster

La clusterizzazione è una tecnica non supervisionata che permette di raggruppare gli elementi di un *dataset* in gruppi che contengano elementi quanto più possibile omogenei fra loro.

6.2.1 Dataset

Per realizzare il task di *clustering* si è deciso di utilizzare un *dataset* differente rispetto al task precedente⁴.

Tale *dataset* contiene degli annunci di immobili presenti nello stato della Colombia. In particolare, si è utilizzato il file CSV che contiene gli annunci di appartamenti in vendita nella città di *Baranquilla*.

Tale *dataset* contiene vari campi sugli annunci immobiliari, come: latitudine e longitudine dell'immobile, la categoria dell'immobile, il prezzo, il numero di stanze, ecc...

Struttura Dataset

Nella tabella seguente (Tab.6.3) viene riportata la struttura del *dataset* utilizzato per realizzare il task corrente.

Column Name	Description
lat	Latitudine geografica dell'immobile
long	Longitudine geografica dell'immobile
id	Identificativo dell'immobile
date	Data di acquisizione del dato
category	Categoria dell'immobile (appartamento o appartamento/studio)
location	Zona della città in cui l'immobile è posizionato
price	Prezzo dell'immobile
details	Dettagli dell'immobile
description	Descrizione dell'immobile
surface	Superficie dell'immobile
rooms	Numero di stanze dell'immobile
baths	Numero di bagni dell'immobile
park	Numero di posteggi dell'immobile

Tabella 6.3: Struttura Dataset

Pre-Processing

Prima di effettuare il task di *clustering* è stato necessario realizzare un *pre-processing* al *dataset* precedentemente descritto.

In particolare si sono eseguite le seguenti operazioni:

- Assegnare al campo locazione (stringa) un valore intero, così da ottenere per ciascun valore della locazione un intero ad esso associato;
- Convertire il campo prezzo da stringa a float;

⁴<https://www.kaggle.com/mlangel/real-state-colombia-sale?select=2021-06-20-baranquilla-apartamento-venta-FR-details.csv>

- Assegnare al campo categoria (stringa) un valore intero, così da ottenere per ciascun valore della categoria un intero ad esso associato;
- Convertire il campo superficie da stringa a float.

6.2.2 K-Means Clustering

Uno degli algoritmi utilizzati per il clustering è il *K-means Clustering*. Tale algoritmo fa parte dei **metodi di partizionamento**: dato un dataset di n oggetti e dato $k \leq n$ il numero di cluster da costruire, un algoritmo di partizionamento organizza gli oggetti in k partizioni. I cluster sono costruiti con il fine di ottimizzare un criterio di partizionamento oggettivo, spesso denominato **funzione di similarità**. In breve, gli oggetti all'interno di un cluster devono essere simili fra loro, mentre gli oggetti di cluster differenti devono essere diversi fra loro.

Nel *K-means Clustering* ogni gruppo viene identificato mediante un centroide o punto medio. L'algoritmo determina casualmente k punti di riferimento, e poi segue iterativamente le seguenti fasi:

- Associa ogni elemento del dataset (sample) al punto di riferimento più vicino formando così k cluster
- Per ogni cluster ottenuto calcola il centroide, che diventa il nuovo punto di riferimento, la procedura prosegue finché la posizione dei centroidi non converge.

Per trovare il numero k ottimale si è fatto uso del metodo del gomito (***Elbow method***), in cui si analizza e si va a memorizzare, per valori di k crescenti, uno specifico indice detto *Inertia* che è definito come la somma delle distanze al quadrato dei campioni dal loro centro del cluster più vicino e rappresenta quanto i cluster sono uniformi tra loro. Il miglior k si colloca, infatti, in corrispondenza del punto del grafico in cui è visibile una variazione di pendenza repentina.

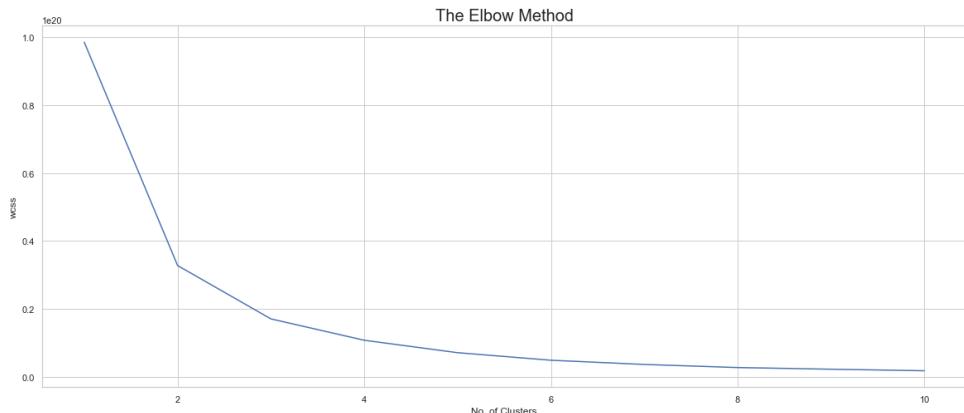


Figura 6.23: Elbow Method

La figura Fig.6.23 riporta il grafico ottenuto dall'*Elbow method* applicato. Si osservano delle variazioni di pendenza in corrispondenza dei valori di ascissa 2, 3, 4 e 6. Le variazioni di entità maggiori sono quelle che si hanno in corrispondenza dei valori 2 e 3.

Si è scelto un valore di k pari a 3 in quanto, fra 2 e 3, si è preferito scegliere un numero maggiore di cluster. Nella figura Fig.6.24 è mostrata la clusterizzazione ottenuta dal *K-means Clustering* nello spazio definito dagli attributi Location e Prezzo del dataset. Nella figura i centroidi dei diversi cluster calcolati dall'algoritmo sono riportati in blu.

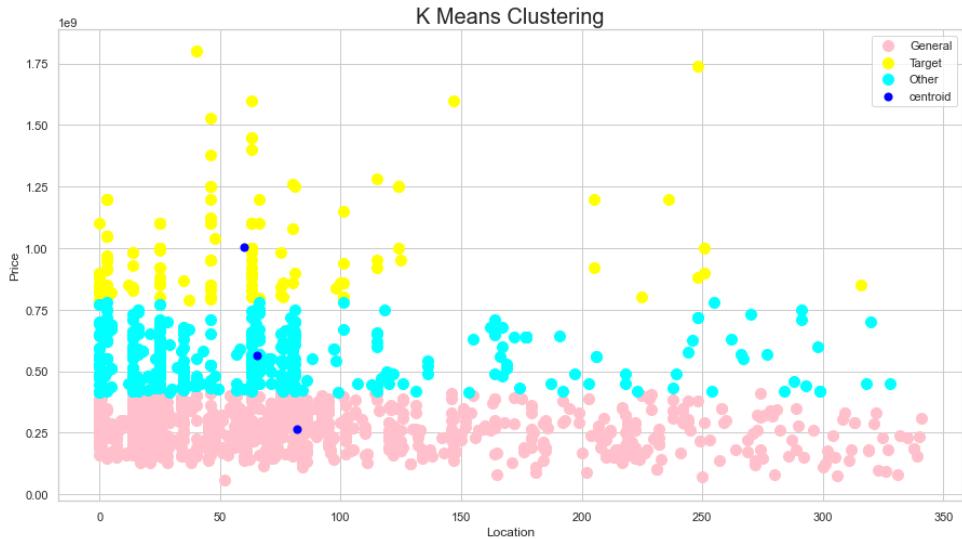


Figura 6.24: Clustering Price - Location

Osservazioni Dal cluster precedente si può osservare come non ci sia una correlazione evidente fra la locazione e il prezzo. Si hanno infatti valori di prezzo variabili per tutte le locazioni. Si osserva che in alcune locazioni si hanno un numero minore di punti, il che significa che sono presenti meno abitazioni in vendita.

Dal grafico precedente si osserva come gli annunci possano essere divisi in tre cluster, un primo cluster (in rosa) che contiene le abitazioni con prezzo più basso e in diverse locazioni. In azzurro troviamo le abitazioni con prezzo medio, anch'esse sparse nelle varie locazioni. Infine, in giallo, troviamo le abitazioni con prezzo maggiori, anche queste divise nelle varie locazioni ma in numero decisamente minore rispetto agli altri due cluster.

6.2.3 Hierarchical Clustering

Il metodo di clustering *Hierarchical* è uno dei metodi di clustering gerarchico che lavora raggruppando gli oggetti in alberi di cluster. In particolare è un metodo di **clustering gerarchico agglomerativo**. A differenza del *K-Means Clustering* che utilizzava un numero k di clustering ben definito, in questo caso i metodi agglomerativi iniziano con un numero n di cluster e poi sequenzialmente combinano i cluster simili per ottenerne uno solo, il numero di cluster k in cui l'algoritmo termina è specificato dall'utente.

Come metrica per calcolare il *linkage*, cioè il collegamento fra i cluster si utilizza la modalità standard, cioè la **distanza euclidea**. Come criterio di *linkage* si è utilizzato il *ward*, il criterio di *linkage* determina quale distanza utilizzare fra due cluster. Utilizzando la *ward* si tende a minimizzare la varianza di cluster che vengono fusi assieme.

Nella figura Fig.6.25 viene riportato il clustering prodotto dall'*Hierarchical Clustering*.

Osservazioni Dal clustering ottenuto in questa sezione si può osservare come i risultati confermano quanto ottenuto già precedentemente con il K-means Clustering. In breve si distinguono tre cluster nei quali si individuano immobili di fasce di prezzi differenti.

6.2.4 Clustering Price - Surface

In questa sezione si analizzeranno i risultati ottenuti dal clustering considerando le feature *Price* e *Surface*.

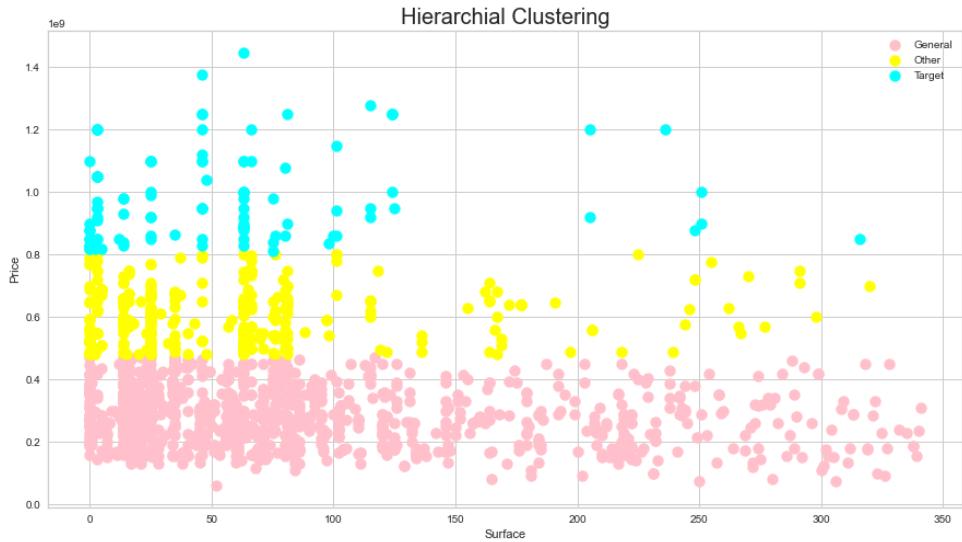


Figura 6.25: Hierarchical Clustering

Anche in questo caso per la scelta del numero k di cluster da utilizzare si è utilizzato l'*Elbow Method*. Nella figura Fig.6.26 viene riportato il risultato ottenuto applicando tale metodo.

Come si può osservare dalla figura si ha una variazione di pendenza rilevante per il valore di k pari a 3, si è scelto quindi questo valore come numero di cluster.

Nella figura Fig.6.27 è riportato il risultato ottenuto, si è usato il *K-means Clustering*.

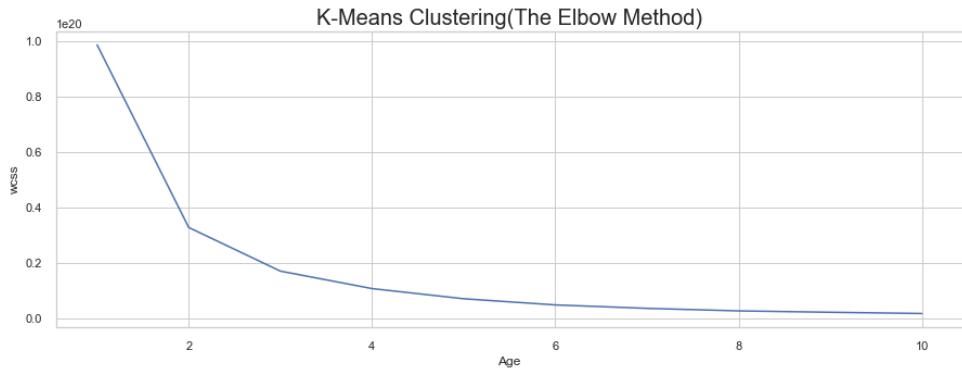


Figura 6.26: Elbow Method

Osservazioni In questo caso si può osservare una correlazione diretta fra prezzo e superficie, infatti per valori di superficie crescenti il valore del prezzo cresce.

Possiamo in questo caso distinguere tre differenti cluster in cui troviamo rispettivamente, gli immobili con superficie e prezzo di valori contenuti, poi con valori medi e infine gli immobili che possiedono un valore elevato di superficie ed hanno un prezzo maggiore rispetto agli altri.

6.2.5 Cluster All

L'ultima clusterizzazione si è realizzata utilizzando tutte le *features* a disposizione, in particolare si sono utilizzate le seguenti *features*:

- Superficie

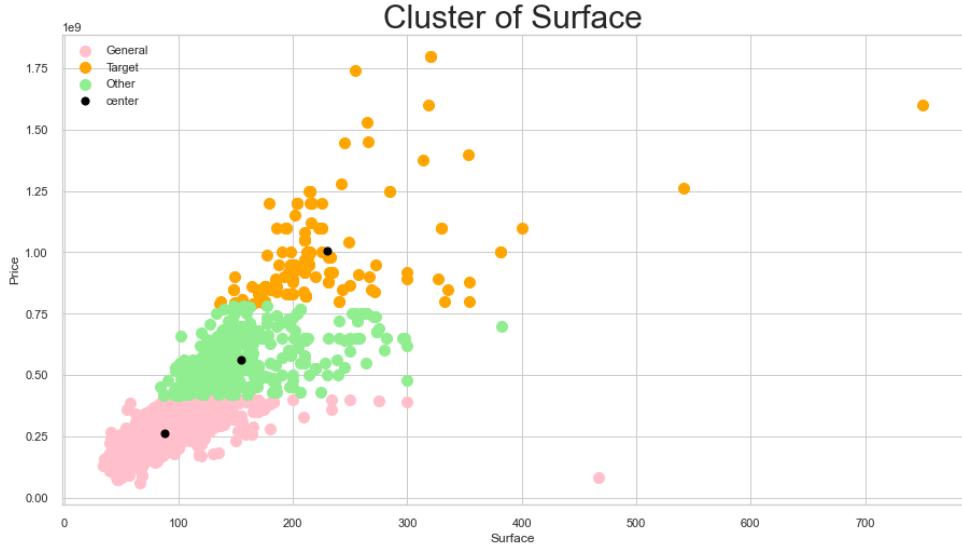


Figura 6.27: Clustering Price - Surface

- Numero di stanze
- Numero di bagni
- Numero di posteggi
- Locazione
- Prezzo
- Categoria

In questo caso si è reso necessario l'utilizzo della tecnica PCA(Principal Component Analysis) per avere una proiezione dalle 7 features a due features, così da poter graficare il risultato ottenuto.

Per stabilire il numero k di cluster da individuare si è utilizzato anche il questo caso l'*Elbow Method*. Ottenendo il risultato riportato nella figura Fig.6.28. Si è scelto k pari a 3.

I risultati ottenuti in questo clustering sono riportati nella figura Fig.6.29, si è utilizzato anche in questo caso il *K-means Clustering*.

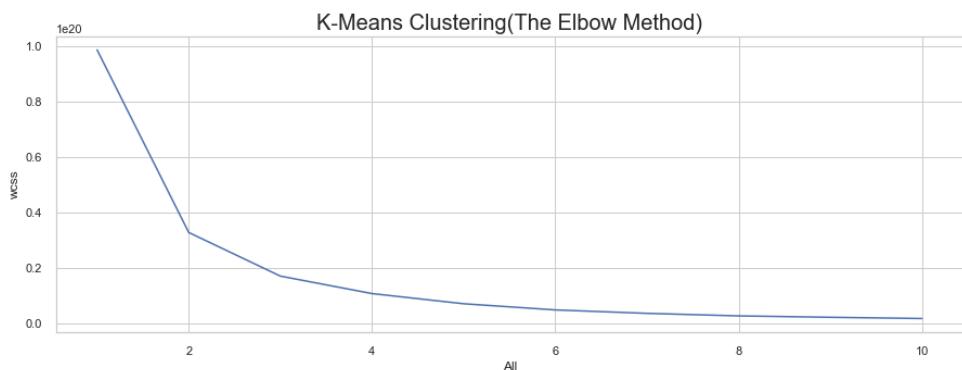


Figura 6.28: Elbow Method

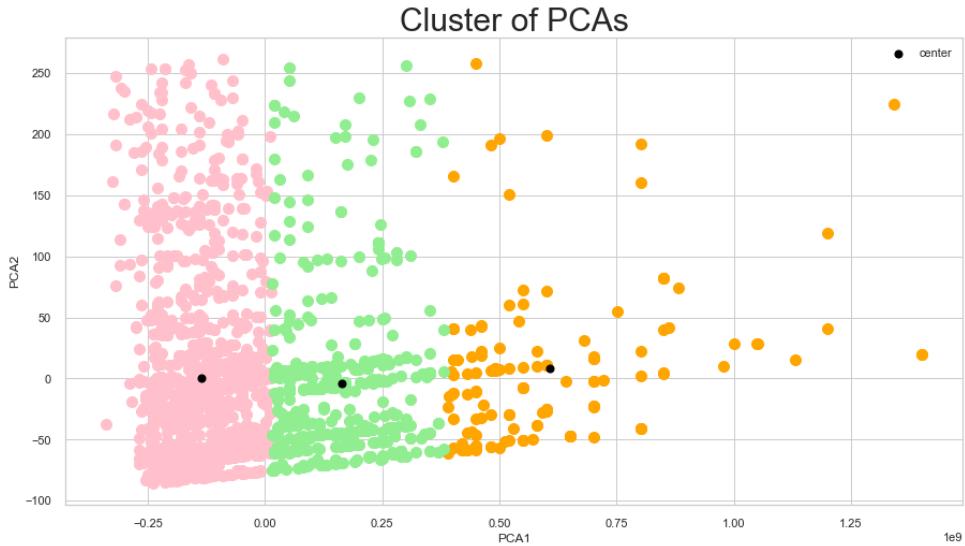


Figura 6.29: Clustering

Osservazioni Come possiamo notare, anche con l'utilizzo della PCA sono stati individuati i 3 cluster dal dataset di partenza. Il numero di cluster è stato scelto, come detto sopra, dopo la valutazione dell'*Elbow Method*.

Non risulta però chiaro come i parametri siano stati valutati, dalle 7 feature di partenza. In seguito però si è analizzata la *Silhouette* per valutare la bontà dei cluster ottenuti.

Silhouette Per valutare il clustering precedente si è calcolata la *Silhouette*. La *Silhouette* ci permette di vedere se i diversi campioni presi in considerazione sono clusterizzati opportunamente o, viceversa, si troverebbero meglio in altri cluster. I valori della *Silhouette* variano da -1 a 1 . Dalla figura Fig.6.30 si osserva quanto ottenuto. Si ha che per le label 0 e 2 la maggior parte delle osservazioni si trovano correttamente nel loro cluster, si ha infatti che la maggior parte dei valori si trovano sopra il valor medio di *Silhouette*. Si osserva inoltre che il cluster con label 1 contiene meno osservazioni rispetto agli altri cluster ed inoltre si hanno valori leggermente minori rispetto a quelli ottenuti negli altri due.

In definitiva si ha comunque la maggior parte delle osservazioni con valori sopra la media e la quasi totalità dei valori positivi, questo sottolinea il fatto che si è ottenuta una buona clusterizzazione.

6.2.6 DBSCAN

Il DBSCAN è un altro metodo per effettuare la clusterizzazione del dataset basato sulla densità dei punti. Questo effettua la clusterizzazione secondo la densità dei punti in determinate regioni dello spazio. Parametri importanti da settare per tale metodo sono:

- Epsilon, definisce la soglia sotto la quale se si trova la distanza di due punti questi vengono considerati vicini;
- Numero minimo di *samples*, numero minimo di vicini che deve avere un punto per essere considerato *core point*;
- Metrica utilizzata per calcolare la distanza fra i punti.

Tutti i punti core che si trovano a distanza minore di epsilon fra loro vengono considerati appartenenti allo stesso cluster.

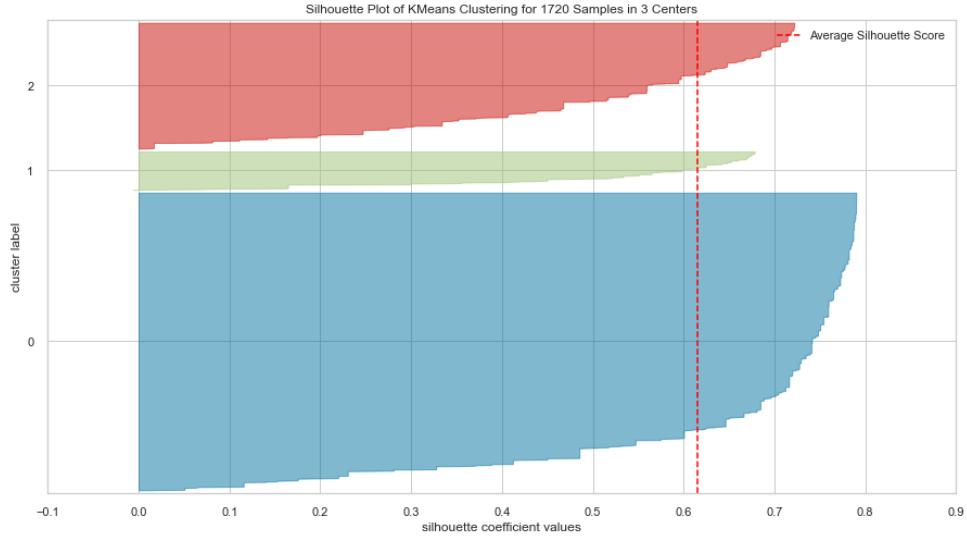


Figura 6.30: Silhouette

Per trovare il parametro epsilon ottimale si è calcolata la distanza degli n vicini per ciascun punto, ordinando e graficando i risultati ottenuti. Di questo grafico si considera il punto in cui si ha una netta variazione di pendenza, il valore della y corrispondente risulta essere la epsilon ottimale.

Il grafico utilizzato per ottenere il valore della epsilon ottimale è mostrato nella figura 6.31. Esso ha un andamento anomalo, inizialmente si hanno infatti valori costanti terminando poi con un picco improvviso, senza mostrare quindi nessun comportamento monotono crescente.

Questo andamento comporta un epsilon di valore molto basso che porta a classificare quasi tutti i punti come rumore, rendendo di fatto inutilizzabile tale procedura. Effettuando il clustering

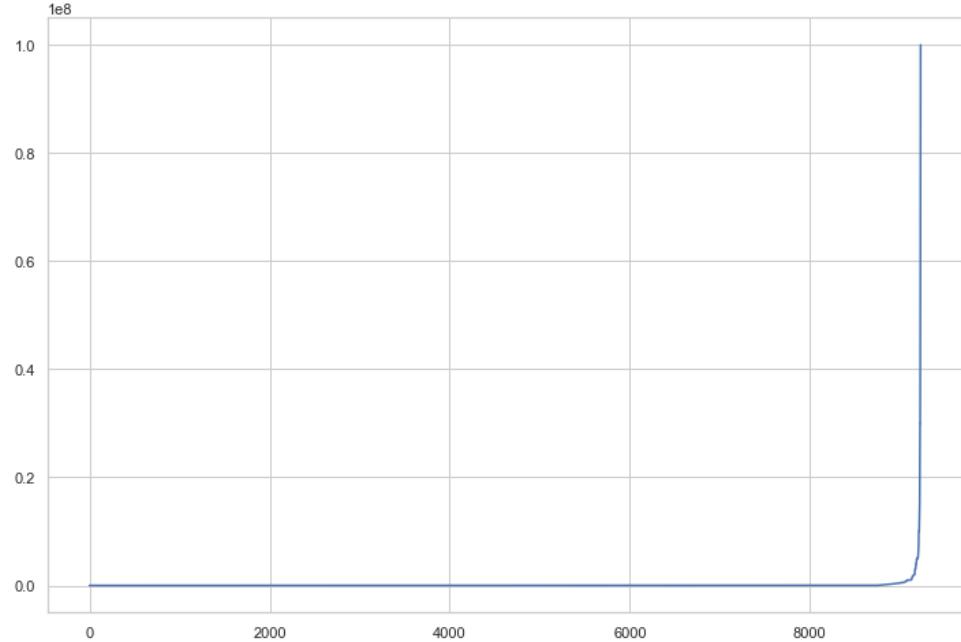


Figura 6.31: Nearest Neighbors

utilizzando il metodo DBSCAN, settando la epsilon al valore 0.1 ed n al valore 8, si ottiene il

risultato riportato nella figura Fig. 6.32.

Si può osservare che si è ottenuto quanto previsto, cioè la maggior parte dei punti sono considerati rumore e quindi rende tale metodo non utilizzabile.

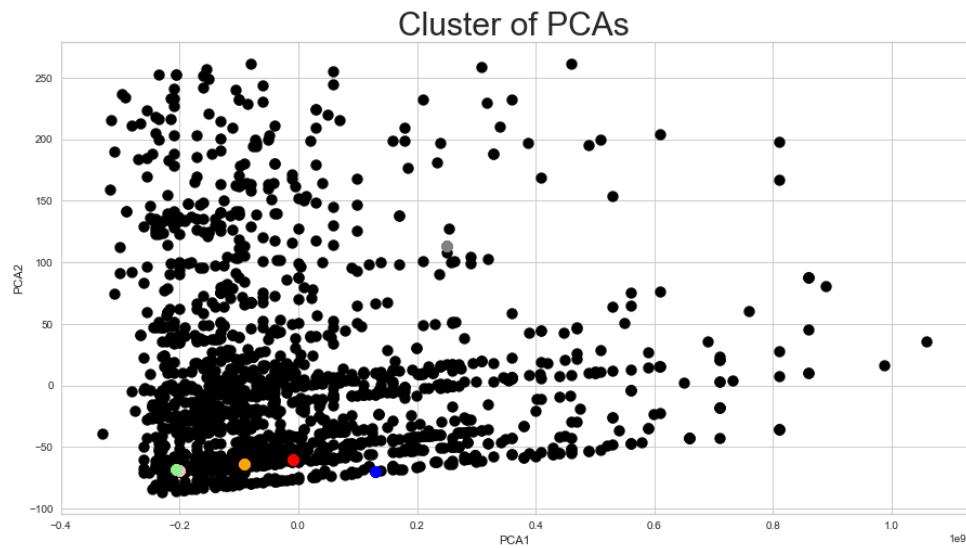


Figura 6.32: Clustering DBSCAN

6.3 Classificazione

6.3.1 Dataset

A differenza del dataset utilizzato per le parti precedenti dell’elaborato di Python, per quanto riguarda il task di classificazione, si è deciso di utilizzare un differente set di dati⁵ per la parte di classificazione e predizione degli stessi. In particolare, si tratta di un insieme di dati inerenti al campo immobiliare Russo, dove si trova l’insieme degli immobili messi in vendita dall’anno 2018 fino a agli ultimi mesi del 2021. Nel dettaglio, si possono trovare all’interno molteplici campi, tra cui data, regione, prezzo, ma anche dettagli più specifici inerenti a ciascun singolo immobile come numero di stanze, piuttosto che la metratura della casa, o piano dell’appartamento nel caso si tratti di un condominio con molteplici piani. Il processo di classificazione verrà formato attraverso librerie come **sklearn**, con l’apporto anche di altre librerie utili all’esposizione visiva dei risultati come **matplotlib** e **seaborn**. Si procede, nel seguito, all’illustrazione specifica di tutti i campi a disposizione per l’elaborazione della parte della tesina, al fine di effettuare classificazione sulle fasce di prezzo di vendita degli immobili, di cui si esplicerà maggiormente nelle sezioni a venire.

6.3.2 Struttura Dataset

La composizione complessiva del Dataset precedentemente introdotto, contiene una totalità di tredici campi, di seguito illustrati all’interno della tabella Tab.6.4:

Column Name	Description	Unit
Price	Prezzo di vendita dello specifico immobile	Int
Date	Data di inserimento nel listino vendite	Date
Time	Ora di inserimento nel listino vendite	Time
Geo_lat	Latitudine geografica dell’immobile	Float
Geo_lon	Longitudine geografica dell’immobile	Float
Region	Regione Russa in cui si trova l’immobile	Int
Building_type	Tipologia di costruzione	Int
Level	Piano in cui si trova l’immobile	Int
Levels	Numero totale di piani del complesso in cui si trova l’immobile	Int
Rooms	Numero di stanze totale dell’immobile	Int
Area	Metratura dell’immobile	Float- m^2
Kithcen_area	Metratura della cucina	Float- m^2
Object_type	Nuova costruzione - Costruzione pre-esistente	Int

Tabella 6.4: Struttura del Dataset Immobiliare

Si tenga presente che la tabella non presenta una intensa prima fase di pre-processing se non per le colonne riguardanti Date, Time, che inizialmente erano di tipo Stringa; mentre Object type è stato mantenuto a Intero con due soli valori possibili, considerando che prima dell’allenamento dei classificatori verranno trasformati tutti i dati per rientrare in un range [0,1]. È stata poi effettuata una seconda fase di pre-processing degli stessi al fine di adattarli al contesto di interesse della tesina. Nel dettaglio, si elencano le manipolazioni effettuate.

⁵<https://www.kaggle.com/mrdaniilak/russia-real-estate-20182021>

1. Conversione di prezzo da Rubli a Euro - Coefficiente di cambio : 83.33 (1 RUBLO = 0.012 EURO)
2. Rimozione delle istanze che presentano valori negativi o nulli nella colonna del prezzo dell'immobile
3. Rimozione delle colonne Date, Time, Levels in quanto ritenute poco utili al task di classificazione
4. Rimozione delle istanze che presentano metrature dell'immobile inferiore a $10m^2$, in quanto ritenuti poco realistici

Per terminare la manipolazione dei dati, si è poi effettuata l'estrazione delle istanze di una singola regione Russa, in quanto la totalità del dataset presenta 5mln di istanze, un numero eccessivo di elementi da gestire nella sua totalità per il task di classificazione. Così si è optato per estrarre le istanze della regione che presentava più istanze di vendita nel triennio 2018-2021, con una totalità di 1mln circa di immobili, per una corrispondenza quindi di 1/5 della totalità. Successivamente si sono filtrati diversi Range di prezzi effettuando quindi Undersampling delle classi maggioritarie, al fine di rendere l'insieme delle istanze per la classificazione bilanciate tra tutte le classi da predire, senza incappare in favoreggiamenti per i Range di prezzi più bassi, che come si può immaginare, presentano un numero di immobili molto superiore rispetto i range di prezzi più elevati. Infine, dopo aver estratto lo stesso numero di istanze (10.000) da ciascun range, si sono concatenati gli insiemi così da ottenere un singolo dataframe di istanze da utilizzare nei task successivi. Il tutto viene illustrato in figura Fig.6.33:

```
'''TRAIN'''
major_region = d[d["region"]== 9654] #REGIONE CON PIU CASE IN VENDITA
print("regione con piu case in vendita")
print(major_region)

above_100k = major_region[(major_region['price'] > 100000)]
between_80k_100k = major_region[(major_region['price'] > 80000) & (major_region['price'] <= 100000)]
between_60k_80k = major_region[(major_region['price'] > 60000) & (major_region['price'] <= 80000)]
between_40k_60k = major_region[(major_region['price'] > 40000) & (major_region['price'] <= 60000)]
between_20k_40k = major_region[(major_region['price'] > 20000) & (major_region['price'] <= 40000)]
below_20k = major_region[major_region["price"] <= 20000]

...
above_100k_t = above_100k.sample(n=10000, random_state=rnd_state)
between_80k_100k_t = between_80k_100k.sample(n=10000, random_state=rnd_state)
between_60k_80k_t = between_60k_80k.sample(n=10000, random_state=rnd_state)
between_40k_60k_t = between_40k_60k.sample(n=10000, random_state=rnd_state)
between_20k_40k_t = between_20k_40k.sample(n=10000, random_state=rnd_state)
below_20k_t = below_20k.sample(n=10000, random_state=rnd_state)

d = pd.concat([above_100k_t, between_80k_100k_t, between_60k_80k_t, between_40k_60k_t, between_20k_40k_t, below_20k_t], ignore_index=True)
```

Figura 6.33: Estrazione delle istanze per il set di Train

6.3.3 Visualizzazione del dataset di partenza

Terminata la fase di pre-processing dei dati, si illustra nella figura Fig.6.34 il dataset a seguito delle varie modifiche effettuate, e da cui si parte per effettuare i task successivi.

6.3.4 Pre-processing - Fase 3

Come ultima fase di manipolazione dei dati, si è resa necessaria la codifica di 3 delle colonne del dataset in specifici Range, come precedentemente accennato nelle sottosezioni precedenti. In esempio, considerando che la variabile da predire (prezzo) aveva una collezione di numerose

	price	geo_lat	geo_lon	region	building_type	level	rooms	area	kitchen_area	object_type	
0	116404.656186	55.017710	82.944786	9654.0		3.0	2.0	3.0	127.4	16.70	1.0
1	108004.320173	55.040379	82.943840	9654.0		3.0	7.0	3.0	88.0	13.00	1.0
2	123604.944198	55.022695	82.916263	9654.0		3.0	6.0	1.0	85.5	15.90	1.0
3	120004.800192	54.771749	83.071630	9654.0		3.0	3.0	4.0	241.7	34.50	1.0
4	135605.424217	55.008170	82.964735	9654.0		0.0	7.0	1.0	75.2	7.64	11.0

Figura 6.34: Dataset Immobiliare post-preprocessing

e variabili possibilità, si è sviluppata una codifica in Range di prezzi, gli stessi utilizzati nella sottosezione precedente nella fase di filtraggio dei campioni per bilanciare il dataset. La stessa considerazione viene fatta per features come Area e Kitchen_area. Si mostra sotto quanto detto in figura Fig.6.35.

Al termine di questa manipolazione, si ottiene il dataset definitivo con la nuova forma di figura Fig. 6.36.

Infine si è effettuata una semplice normalizzazione dei dati per codificarli tutti entro un range di valori [0,1], utili ai classificatori per allenare il meglio possibile. Si elude l'inserimento illustrativo di questo passaggio in quanto risulta avere facile interpretazione.

6.3.5 Correlazione

Per concludere il processo di pulizia del dataset, si è elaborata, tramite il metodo *heatmap* della libreria *seaborn*, una matrice di correlazione al fine di osservare eventuali correlazioni "forti" tra le varie features, così da eliminare quelle che hanno un legame intenso tale da influenzare i risultati finali di predizione. Per quanto detto, si illustrano i risultati ottenuti in figura Fig.6.37

Come si può notare dal grafico, l'attributo *Rooms* ha un forte legame con l'attributo *Range-area*, quindi è stato deciso di eliminarla dal dataset.

Si tenga conto che la colonna *Range-price* è stata inserita solo a scopo informativo e per motivo di studio del problema, infatti anch'essa è stata poi eliminata prima di iniziare la fase di classificazione vera e propria.

Con tutti i passaggi sopra definiti si è ottenuto così un dataset bilanciato, rappresentativo di ciascuno dei range di prezzo da predire e con valori di correlazione tra le features basse.

In questo modo si raggiungeranno dei risultati più puliti possibile in fase di addestramento dei classificatori.

6.3.6 Classificatori

Gli algoritmi di classificazione scelti sono i seguenti:

- LogisticRegression, ha lo scopo di descrivere e stimare la probabilità di una variabile dipendente categoriale da uno o più variabilità indipendenti. L'algoritmo consente di stimare la probabilità di occorrenza di un evento mediante l'inserimento di dati in una funzione logistica;
- GradientBoostingClassifier, rappresenta un modello ensemble che si basa su una combinazione di singoli modelli semplici, detti weak learner, che insieme creano un nuovo modello più potente chiamato strong learner, dove questo risulta più efficace dei singoli. Ad ogni ciclo di allenamento il weak learner viene costruito e le sue previsioni confrontate con il risultato corretto. La distanza tra osservazione e previsione rappresenta il tasso di errore. Lo scopo dell'algoritmo è ridurre al massimo la funzione di perdita tramite uno strumento

```

def range_prices(pricee):
    if pricee<=20000:
        val = 0
    elif pricee >20000 and pricee<=40000:
        val = 1
    elif pricee >40000 and pricee<=60000:
        val = 2
    elif pricee >60000 and pricee<=80000:
        val = 3
    elif pricee >80000 and pricee<=100000:
        val = 4
    elif pricee >100000:
        val = 5
    else:
        print(pricee)
    return val

def range_areas(areaa):
    if areaa<=40:
        val = 0
    elif areaa >40 and areaa<=50:
        val = 1
    elif areaa >50 and areaa<=60:
        val = 2
    elif areaa >60 and areaa<=70:
        val = 3
    elif areaa >70 and areaa<=80:
        val = 4
    elif areaa >80 and areaa<=90:
        val = 5
    elif areaa >90 and areaa<=100:
        val = 6
    elif areaa >100 and areaa<=150:
        val = 7
    elif areaa >150 and areaa<=200:
        val = 8
    elif areaa >200:
        val = 9
    else:
        print(areaa)
    return val

def range_areas_kitchen(areaa):
    if areaa<=10:
        val = 0
    elif areaa >10 and areaa<=20:
        val = 1
    elif areaa >20 and areaa<=30:
        val = 2
    elif areaa >30 and areaa<=40:
        val = 3
    elif areaa >40 and areaa<=50:
        val = 4
    elif areaa >50:
        val = 5
    else:
        print(areaa)
    return val

```

Figura 6.35: Codificazione dei range per Features Price, Area, Kitchen_area

	price	geo_lat	geo_lon	region	building_type	level	rooms	area	kitchen_area	object_type	range_price	range_area	range_area_kitchen	
0	116404.656186	55.017710	82.944786	9654.0		3.0	2.0	3.0	127.4	16.70	1.0	5	7	1
1	108004.320173	55.040379	82.943840	9654.0		3.0	7.0	3.0	88.0	13.00	1.0	5	5	1
2	123604.944198	55.022695	82.916263	9654.0		3.0	6.0	1.0	85.5	15.90	1.0	5	5	1
3	120004.800192	54.771749	83.071630	9654.0		3.0	3.0	4.0	241.7	34.50	1.0	5	9	3
4	135605.424217	55.008170	82.964735	9654.0		0.0	7.0	1.0	75.2	7.64	11.0	5	4	0

Figura 6.36: Database complessivo finale

chiamato “gradiente”, che rappresenta la derivata parziale della funzione di perdita usata poi per trovare la direzione in cui modificare i parametri del modello al fine di ridurre l’errore alla prossima iterazione;

- DecisionTreeClassifier, il modello è rappresentato attraverso alberi binari. La creazione dell’algoritmo implica decidere quali caratteristiche scegliere, ovvero le variabili di input,

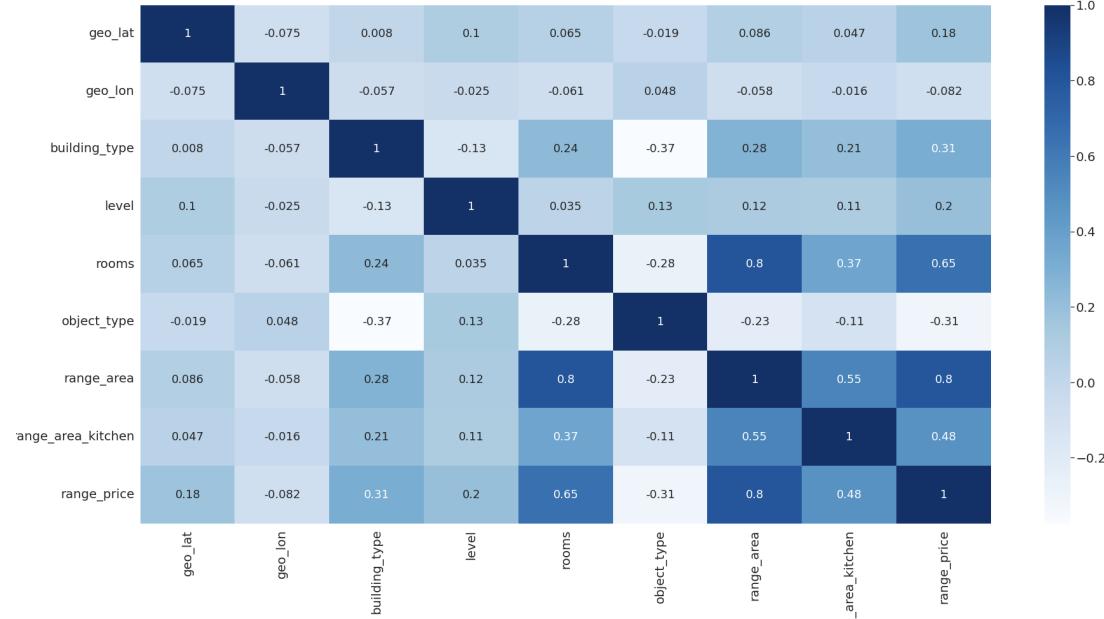


Figura 6.37: Matrice di correlazione delle varie features

quali condizioni utilizzare per la divisione e definire quando fermarsi. Una volta impostato e creato l'albero decisionale è possibile testare un nuovo set di dati partendo dal nodo radice, per effettuare previsioni;

- LinearSVC, implementato in termini di liblinear anziché libsvm, quindi ha una maggiore flessibilità nella scelta delle penalità e delle funzioni di perdita e dovrebbe scalare meglio su un numero elevato di campioni.
- RandomForestClassifier, prevede di costruire una moltitudine di alberi decisionali al momento dell'addestramento al fine di restituire la classe che ottiene più voti. Individualmente, le previsioni fatte dai singoli alberi decisionali potrebbero non essere accurate, ma combinate insieme saranno in media più vicine al risultato;
- SGDClassifier, implementa modelli lineari regolarizzati con apprendimento stocastico di discesa del gradiente (SGD): il gradiente viene stimato ogni campione uno alla volta e il modello viene aggiornato lungo il percorso con un programma di aggiornamento decrescente (aka learning rate). Per ottenere i migliori risultati utilizzando il programma del tasso di apprendimento predefinito, i dati devono avere media e varianza dell'unità pari a zero;
- Perceptron, è una rete neurale progettate per minimizzare un errore di misura al fine di assumere il comportamento di una funzione arbitraria. La funzione mappatura della rete dipende da un numero di pesi che contengono le informazioni della rete e l'addestramento viene fatto andando a scegliere quei pesi che minimizzano l'errore di misura;
- KNeighborsClassifier, predice la classe di appartenenza di un dato punto calcolando la distanza, in questo caso euclidea, da un dato punto di riferimento. In particolare, dentro ad ogni classe vengono associati dei data points o istanze il cui insieme definisce il set di dati. Si visualizzano suddette classi su un asse cartesiano dove gli assi rappresentano le caratteristiche del problema. Più un'istanza è vicina ad un dato point più il KNN li considererà simili. La somiglianza si calcola sulla base della distanza euclidea. Inoltre, si fissa un parametro k, che identifica il numero di data points più vicini; l'algoritmo valuta le k

minime distanze così ottenute e la classe che ottiene il maggior numero di queste distanze è scelta come previsione;

- AdaBoostClassifier, tecnica di ensemble learning che permette di avvalersi di classificatori deboli sequenziali per problemi di classificazione binaria. Questi vengono utilizzati per formare un classificatore forte. Infatti, un algoritmo preso singolarmente può classificare male, ma se si combinano più classificatori con la selezione del set di addestramento ad ogni iterazione e se si assegna la giusta quantità di peso nella votazione finale, si può avere un buon punteggio di precisione per il classificatore generale;
- ExtraTreesClassifier, implementa un meta stimatore che adatta un numero di alberi decisionali randomizzati (noti anche come alberi extra) su vari sottocampioni del set di dati e utilizza la media per migliorare l'accuratezza predittiva e controllare l'overfitting;
- GaussianNB, è sufficiente trovare la media e la deviazione standard di ciascuna probabilità per ogni attributo e per ogni singola classe. Sostituendo tali valori nella funzione di densità di probabilità gaussiana si ricava una probabilità che permette di ricavare le varie probabilità di classe. Il valore di probabilità di classe più alto rappresenta la classe da associare alla nuova istanza che si vuole categorizzare.

Parametrizzazione e scelta degli Hyperparametri

Come primo aspetto si sottolinea la scelta di uno split tra set di Train e Test con un rapporto del $70 \div 30$, così da dare maggiore importanza al test-set in fase di verifica dei classificatori durante le iterazioni del k-fold validation. Inoltre si cerca così di limitare overfitting dei risultati che potrebbe provenire da un'eccessivo numero di dati nel set di Train. La nota più importante riguarda la modalità di classificazione scelta per allenare i classificatori, con strategia adottata denominata **OneVsRestClassifier**, necessaria per la rappresentazione grafica e per un corretto addestramento degli algoritmi, andando a verificare ciascuna specifica classe rispetto le $n - 1$ rimanenti.

Da questi modelli è possibile poi misurarne la qualità di addestramento attraverso metriche di valutazione della classificazione. Come metrica di riferimento per questo elaborato è stata utilizzata la *accuracy*, che rappresenta il rapporto tra il numero di istanze classificate correttamente e il numero di istanze totali. In aggiunta, si è utilizzato il metodo di Cross-Validation, di tipologia Holdout, e che procede suddividendo il set di dati in n partizioni in cui iterativamente uno verrà usato per il set di test, mentre gli altri rimanenti formeranno insieme il set di train.

6.3.7 Addestramento dei classificatori

Valutazione dei classificatori

I risultati ottenuti rispetto l'accuratezza di classificazione per un problema di classificazione multi classe con sei classi univoche sono mostrati in figura Fig.6.38.

Nel dettaglio, si osserva come i risultati migliori vengono ottenuti da 3 classificatori specifici, di cui si elencano i rispettivi risultati in accuratezza nella tabella Tab.6.5:

Classificatore	Accuracy
DecisionTreeClassifier	0.768
RandomForestClassifier	0.801
ExtraTreesClassifier	0.789

Tabella 6.5: Migliori classificatori

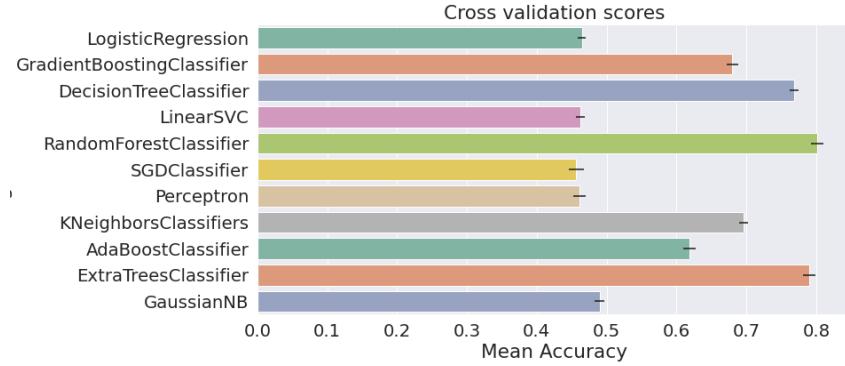


Figura 6.38: Barplot dei risultati

Si vanno ora ad illustrare le ROC-Curve e la Confusion Matrix dell’ algoritmo coi migliori risultati, rispettivamente in figura Fig.6.39 e Fig.6.40.

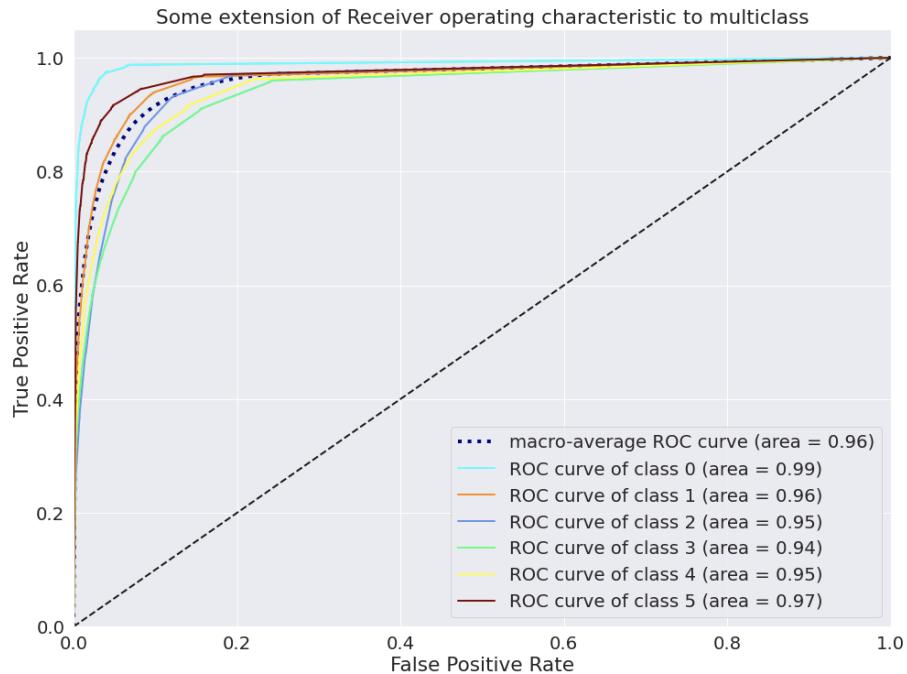


Figura 6.39: Roc-Curve RF-Classifier

OSSERVAZIONI

Si osserva come, dalla Roc-Curve, che il range di classe 0 rispetto i restanti 4 range risulta essere quella più precisa, questo dovuto dal fatto che il range più basso di prezzi contiene pochi outliers, ovvero la quasi totalità delle istanze sono molto simili tra loro senza avere immobili con caratteristiche eccessivamente diverse dalle altre, così da portare a risultati in addestramento migliori rispetto i restanti Range. Le stesse considerazioni si possono effettuare osservando la tabella sottostante (Tab. 6.6), da cui si conferma il fatto che per le classi più esterne si hanno specifiche delle case molto simili tra loro, con risultati molto migliori rispetto range di prezzo interni, in cui le specifiche possono essere molto più variabili, e talvolta rischiano di ricadere in alcuni range vicini, comportando risultati più bassi di qualche decina percentuale.

Nel dettaglio, e per maggiori informazioni, si illustra anche la tabella Tab.6.6 con ulteriori me-

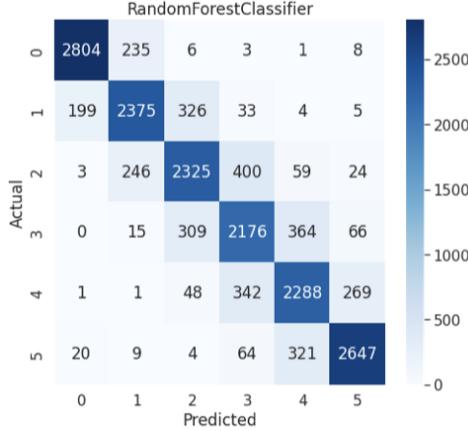


Figura 6.40: CM RF-Classifier

triche come *Recall*, *F1 Score*, *Precision* e *Support*. Nella colonna relativa alle classi trovo posto l'informazione relativa al range di prezzo dell'immobile, nel dettaglio i range di prezzi sono:

- 0 $\models 0 \div 20.000$
- 1 $\models 20.000 \div 40.000$
- 2 $\models 40.000 \div 60.000$
- 3 $\models 60.000 \div 80.000$
- 4 $\models 80.000 \div 100.000$
- 5 $\models \text{over } 100.000$

	Precision	Recall	F1-Score	Support
0	0.93	0.92	0.92	3057
1	0.82	0.81	0.82	2942
2	0.77	0.76	0.77	3057
3	0.72	0.74	0.73	2930
4	0.75	0.78	0.76	2949
5	0.88	0.86	0.87	3065
Accuracy				0.81
Macro Avg	0.81	0.81	0.81	18000
Micro Avg	0.81	0.81	0.81	18000

Tabella 6.6: Classification Report RF-classifier

6.3.8 Classificazione con GridSearch

Si procede ora ad effettuare Fine Tuning di uno degli algoritmi che ha restituito risultati migliori, in particolare si mostrano i risultati ottenuti del RandomForestClassifier, quello che tra i 3 è risultato ottimale al problema in esame, almeno come primo caso di studio. Nel dettaglio, si procede a fare auto-tuning di Iperparametri (Hyperparameters) del classificatore appena richiamato, attraverso questa tecnica di Sklearn, che permette di selezionare il miglior modello

di classificatore, impostando il problema con molteplici parametri, ciascuno dei quali può a sua volta assumere valori differenti. Dopodichè, il metodo permette di effettuare la combinazione di tutti i parametri definiti pre addestramento, e restituire all'utente la configurazione migliore da adottare. Nel caso in esame si sono settati i parametri di figura Fig.6.41 :

```
RF_param = {'n_estimators': [10, 25],
            'max_features': [0.3, 0.7, 1],
            'max_depth': [10, 50, None],
            'bootstrap': [True, False]}
```

Figura 6.41: GridSearch Parameters RF-Classifier

Anche in questo caso si è deciso di mantenere la strategia di splittaggio tra Train e Validation Set di K-Fold, così da osservare il cambiamento mantenendo la stessa procedura di addestramento interna. I risultati ottenuti attraverso questo metodo portano un incremento lieve, ma significativo, dato che certifica la grande importanza che ha la scelta dei parametri al fine di ottenere il miglior modello possibile. Il risultato infatti non risulta numericamente molto diverso:

0.801 → 0.812.

Questo effetto rappresenta un cambiamento importante considerando i soli 4 parametri scelti per testare questa metodologia. Si elencano i migliori parametri individuati nella tabella successiva 6.7.

Parameters	Values
bootstrap	True
max_depth	None
max_features	0.7
n_estimators	25

Tabella 6.7: Miglior Setting di parametri

6.3.9 Model Ensemble

Si procede come conclusione della parte di classificazione con la realizzazione di un Ensemble dei due migliori classificatori ottenuti in questa parte di elaborato, per poi procedere a rappresentare graficamente i risultati ottenuti su un set di test nuovo ed osservare i risultati ottenuti. Tutto questo viene fatto attraverso la tecnica di model ensemble con cui si combinano i classificatori *Random Forest*, *ExtraTrees*.

VotingClassifier

Libreria di `sklearn.ensemble` che permette attraverso la scelta di diversi parametri la combinazione di due modelli. In questo elaborato si procede alla specifica di due unici parametri:

- Estimators: *ExtraTrees*, *RandomForest*, ovvero la definizione dei classificatori da utilizzare
- Voting: unico parametro fondamentale da definire, in cui si sceglie tra le alternative *Hard* e *Soft*, che rispettivamente simboleggiano la scelta di fare ensemble mediando direttamente tra le *labels* delle classi predette per ciascuna istanza e su ciascun classificatore, oppure agendo sulle *probabilità* predette di appartenenza di ciascuna istanza per ciascun classificatore, per poi sommare le probabilità dei vari classificatori da combinare e restituire la *label* con probabilità totale maggiore.

L’alternativa di approccio scelta è quella (*Soft*), considerata la fase precedente di GridSearch affrontata per effettuare Tuning dei parametri, e quindi partendo da classificatori ben calibrati che portano ad una scelta coerente di questa alternativa.

Test del modello finale

Si conclude mostrando i risultati ottenuti testando il modello ricavato attraverso tutte le sezioni descritte in questa parte di classificazione, utilizzando un test set univoco e con istanze diverse rispetto quelle utilizzate per l’addestramento, con dimensione dimezzata rispetto il set per allenare i classificatori.

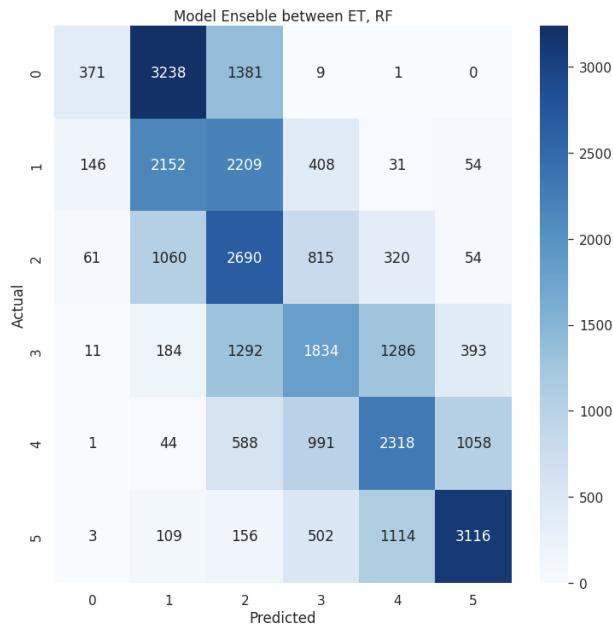


Figura 6.42: CM - Test Ensemble Model

Parameters	Values
2	8316
1	6787
4	5070
5	4675
3	4559
0	593

Tabella 6.8: Count lables Prediction - Ensemble Model

OSSERVAZIONI Dalla immagine e tabelle inserite si può concludere che il risultato, considerato il problema essere una classificazione multiclasse (6 Classi), porta dei buoni risultati finali con accuracy media del **0.42**. A differenza dei risultati ottenuti in precedenza nella fase di addestramento, questo è un chiaro sintomo di overfitting dei dati, ove esso impara quindi a classificare bene, però facendolo da features che poi in alcune delle classi del test finale non ritrova affatto, come osservabile anche dalla classe 0. Allo stesso tempo, osservando i risultati nelle tabelle, si

	Precision	Recall	F1-Score	Support
0	0.63	0.07	0.13	5000
1	0.32	0.43	0.37	5000
2	0.32	0.54	0.40	5000
3	0.40	0.37	0.38	5000
4	0.46	0.46	0.46	5000
5	0.67	0.62	0.64	5000
Accuracy			0.42	30000
Macro Avg	0.47	0.42	0.40	30000
Micro Avg	0.47	0.42	0.40	30000

Tabella 6.9: Classification Report Model Ensemble Test

coglie che questi risultati sono molto influenzati dalle classi centrali che probabilmente differenziano nelle features molto più di quanto fanno le classi esterne, portando la classe 0 a classificare come una di esse. Conferma di questo fatto si osserva nella classe esterna 5, che a differenza della precedente, ha il valore di accuracy più alta tra le altre, questo dovuto al fatto che nel dataset, le features che legano le istanze degli immobili con più valore, hanno features molto simili tra loro, e che allo stesso tempo sono diverse dalle altre classi restanti. Invece, negli immobili di valore sotto i 50k €, le features sono molto simili tra loro e si trovano in un insieme molto ampio di possibilità, portando risultati di misclassificazione (classificazione errata). Inoltre, le istanze della classe 0 sono sicuramente di partenza con features molto discostanti da quelle utilizzate in addestramento, significando anche un possibile errore nell'inserimento delle istanze nel dataset del real-estate in Russia. Inoltre, come nota conclusiva, si può pensare di migliorare il codice al fine di ottenere migliori risultati andando ad agire sulle parametrizzazioni dei classificatori, ma ancora meglio generando dei range sulle Aree dell'immobile ancora più numerosi e di dimensioni ridotte, così da valorizzare ancora meglio le istanze nel loro singolo e cercando quindi di migliorare l'accuratezza di tutti i range di prezzo, in particolare dei range interni che risultano avere accuratezza inferiore.

Capitolo 7

Social Network Analysis

In questo capitolo andremo a presentare la libreria **NetworkX**¹ per il linguaggio **Python**, una delle principali soluzioni adottate al fine di effettuare la Social Network Analisys². La SNA ha come obbiettivo cardine l'analisi delle diverse reti sociali con lo scopo di identificare quelle che sono le entità fondamentali all'interno della rete, le figure di spicco e come essere si relazionano, il tutto con l'ausilio dei grafi, in particolare dei nodi e archi che rappresentano al meglio la social network. La SNA viene quotidianamente applicata nei social media (e non solo) e garantisce ottime opportunità di studio e lucrative.

Va inoltre aggiunto come le funzionalità di interfacciamento con altre librerie di Python come per e.g. Matplotlib, Seaborn etc. ci permette di aggiungere una migliore versatilità per le rappresentazioni grafiche.



Figura 7.1: Logo NetworkX

7.1 Dataset

Il dataset utilizzato per realizzare tale studio è quello inherente alle pagine Facebook di diversi personaggi politici, in particolare le pagine sono quelle verificate da Facebook, ovvero le *blue verified*. Va osservato che i dati raccolti sulle pagine di Facebook risalgono al novembre 2017, quindi molte considerazioni che verranno fatte di seguito dovranno tener conto di questo aspetto temporale. Tornando a parlare in notazione, rispetto ai grafi, i nodi rappresentano le pagine Facebook e gli archi rappresentano i *likes* reciproci delle due pagine Facebook, rappresentate dai nodi collegati dall'arco. Il grafo non è orientato.

Il dataset appena descritto è reperibile al seguente link: <https://networkrepository.com/fb-pages-politician.php>.

¹Networkx è un pacchetto Python per la creazione, la manipolazione e lo studio della struttura, della dinamica e delle funzioni di reti complesse.

²S.N.A. ≡ Social Network Analisys

7.2 ETL

Finora abbiamo introdotto il dataset e fatto un breve accenno di quelli che sono i grafi, senza però creare il link tra le due entità. Ecco quindi che nella fase di ETL, siamo andati a crearlo, per ciascuna coppia di entità dove esso esiste. In particolare nel listato seguente si trovano le operazioni iniziali per creare il grafo.

```

1 edges = []
2
3
4 #with open("fb-pages-politician.txt", "r") as f:
5 with open("/content/fb-pages-politician2.txt", "r") as f:
6     for line in f:
7         s = line.strip()
8         a, b = s.split(",")
9         edges.append((int(a), int(b)))
10
11 graph = nx.Graph()
12 graph.add_edges_from(edges)
13 pos = nx.spring_layout(graph)

```

Listing 7.1: ETL per creazione del grafo

7.3 Rappresentazione

Nello snippet introdotto nel sezione precedente, si è costruita la struttura necessaria al fine di creare e rappresentare il grafo. Ecco quindi che nel listato successivo andiamo ad introdurre le funzioni utili a rappresentare visivamente quanto appena detto.

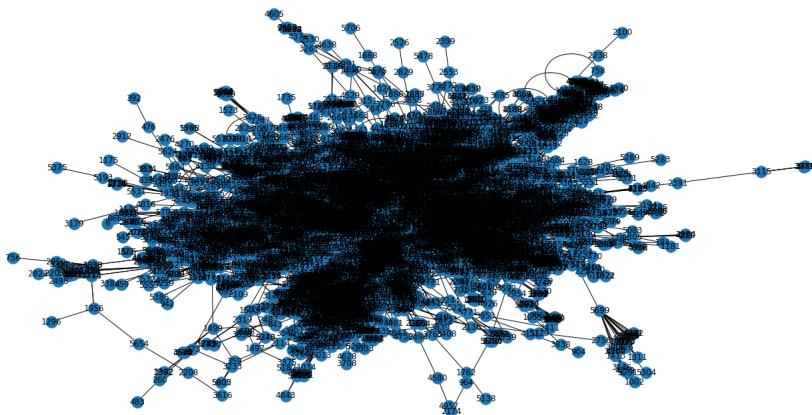
```

1 plt.figure(figsize=(12,8))
2 nx.draw(graph, pos, with_labels=True)
3 plt.show()

```

Listing 7.2: Funzioni utili per la presentazioni del grafo

Nella figura Fig.7.2 vediamo il risultato di quanto detto finora.



7.4 Caratteristiche della rete

La social network fin qui introdotta è rappresentata da 5908 nodi, 41729 archi, e da come è possibile vedere dalla figura Fig.7.2 è difficile apprezzarla e/o analizzarla nella sua completezza. La densità di tale rete è di 0.0024. La rete come si può notare è assai densa, tuttavia alcuni nodi possiedono una maggior centralità rispetto ad altri e questo è visibile, ove c'è una maggior concentrazione di archi. Un'ulteriore osservazione che vi si può notare è il diametro, che in questo studio è pari a 14 (quindi raggio 7); quest'ultima informazione ci permette di confermare la teoria di *"Six Degrees of Separation"*, la quale afferma che ogni persona può essere collegata a qualunque altra persona attraverso una catena di conoscenze e relazioni con non più di 5 intermediari. Per quanto concerne la periferia del grafo, ovvero l'insieme dei nodi che hanno eccentricità pari al diametro, questa analisi restituisce una coppia di nodi formato dal nodo 5706 e dal nodo 3616. L'analisi relativa al clustering riporta come coefficiente di clustering: 0.385, il che indica che all'interno della rete vi sono diversi nodi con una tendenza a creare nuovi cluster. Tra le altre informazioni rilevanti riguardo al nostro grafo, va citato il fatto della connessione, che nel nostro caso riporta un valore positivo.

7.5 Degree Centrality

Tra i principali studi che si possono svolgere in una social network analysis c'è la **Degree Centrality**, una semplice misura di centralità che conta quanti vicini ha un nodo. Nel caso in cui la rete è diretta, va fatta una distinzione tra i collegamenti in arrivo e in uscita. La tesi alla base di questa misura è : *Un nodo è importante se ha molti vicini, o, nel caso diretto, se ci sono molti altri nodi che si collegano ad esso, o se si collega a molti altri nodi.*

La Degree Centrality ci permette quindi di misurare il potere, o meglio l'influenza di alcuni componenti all'interno della rete, prendendo in considerazione il numero di connessioni. Sarà quindi possibile identificare quelle che sono le main entity della rete e le loro relationship con gli altri nodi. **NetworkX** ci fornisce tale metrica normalizzandola per il massimo grado di ogni nodo, con il vantaggio di ottenere un valore percentuale con cui è possibile effettuare confronti anche in sotto-reti distinte.

Top5 Degree Centrality Andando a prendere i 5 nodi con i valori maggiori di *Degree Centrality* otteniamo i seguenti nodi (ordinati secondo il valore di *Degree Centrality*, dal maggiore al minore).

- Manfred Weber - 0.0547
- Joachim Herrmann - 0.0423
- Barack Obama - 0.0394
- Katarina Barley - 0.0388
- Arno Klare MdB - 0.0376

7.5.1 Layouts di Degree Centrality

Il primo layout che andremo a vedere è lo Spring Layout, visibile nella figura Fig.7.4. Esso permette di osservare come vi sia una netta concentrazione di nodi nella parte centrale della rete, che rende difficilmente distinguibile i nodi racchiusi al suo interno. Lo spring layout ci

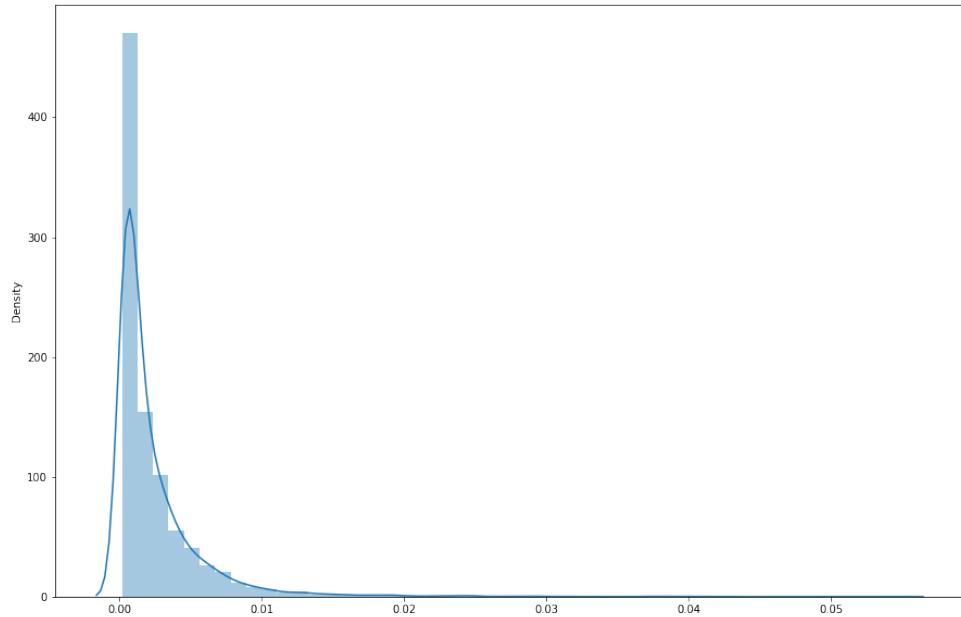


Figura 7.3: Degree Centrality

fornisce una acerba visualizzazione delle comunità come quelle che si possono trovare intorno alla zona centrale. Esse infatti sono collegate alla centro della rete tramite pochi archi, ma formano un piccolo agglomerato di nodi nei dintorni.

Un’ulteriore layout è visibile nella figura Fig.7.5, il quale permette di visionare e comprendere l'estrema versatilità e utilità delle diverse tipologie di *plot* disponibili.

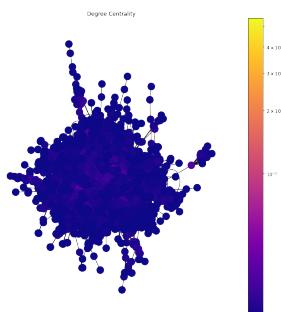


Figura 7.4: Spring Degree Centrality Layout

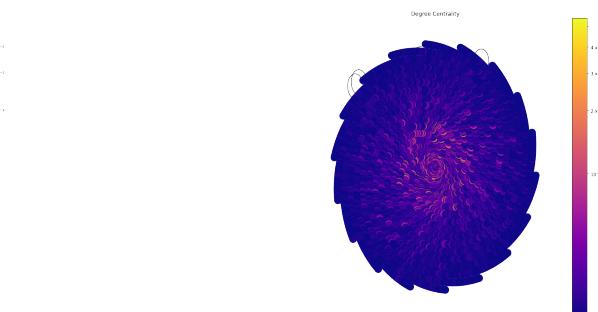


Figura 7.5: Spiral Degree Centrality Layout

Altre due possibilità di plotting sono visibili nelle figure Fig.7.6 e Fig.7.7. Nella fattispecie, nella figura Fig.7.7 è possibile vedere lo Kamada Kawai Layout, che fornisce una maggior informazione sulla distribuzione dei nodi nello spazio. Infine, nella figura Fig.7.6 è visionabile lo Fruchterman Reingold layout, che però non fornisce ulteriori informazioni ma ha il solo scopo di una maggior compresione del tool e della rete in esame.



Figura 7.6: Fruchterman Reingold Degree Centrality Layout
Figura 7.7: Kamada Kawai Degree Centrality Layout

7.5.2 Eigenvector Centrality

La Eigenvector Centrality misura l'importanza di un nodo tenendo conto dell'importanza dei suoi vicini. Per e.g., un nodo con 300 amici relativamente impopolari su Facebook avrebbe una centralità più bassa rispetto a qualcuno con 300 amici molto popolari (come Barack Obama). A volte viene utilizzato per misurare l'influenza di un nodo nella rete. È determinato eseguendo un calcolo di matrice per determinare quello che è chiamato l'autovettore principale usando la matrice di adiacenza. Non solo è utilizzato per determinare l'influenza nelle social network, ma una variante di centralità eigenvector è al centro di un algoritmo Pagerank di Google, che vedremo di seguito. Il main focus è che i collegamenti coi nodi importanti valgono più dei collegamenti coi nodi poco importanti.

Osservazioni Nella figura Fig.7.8 è osservabile la distribuzione dei nodi in funzione del valore della Eigenvector Centrality. Osservando attentamente possiamo concludere che vi sono un elevato numero di nodi con un valore molto basso, indice del fatto che all'interno della rete analizzata non ci sono particolari nodi con la funzione di *eminenza grigia*.

Top5 Eigenvector Centrality Come fatto per la centralità precedente, anche in questo caso si sono presi i primi 5 nodi per valori di *Eigenvector Centrality*.

- Katarina Barley - 0.1741
- Arno Klare MdB - 0.1699
- Katja Mast - 0.1684
- Heike Baehrens - 0.1603
- Christian Petry - 0.1595

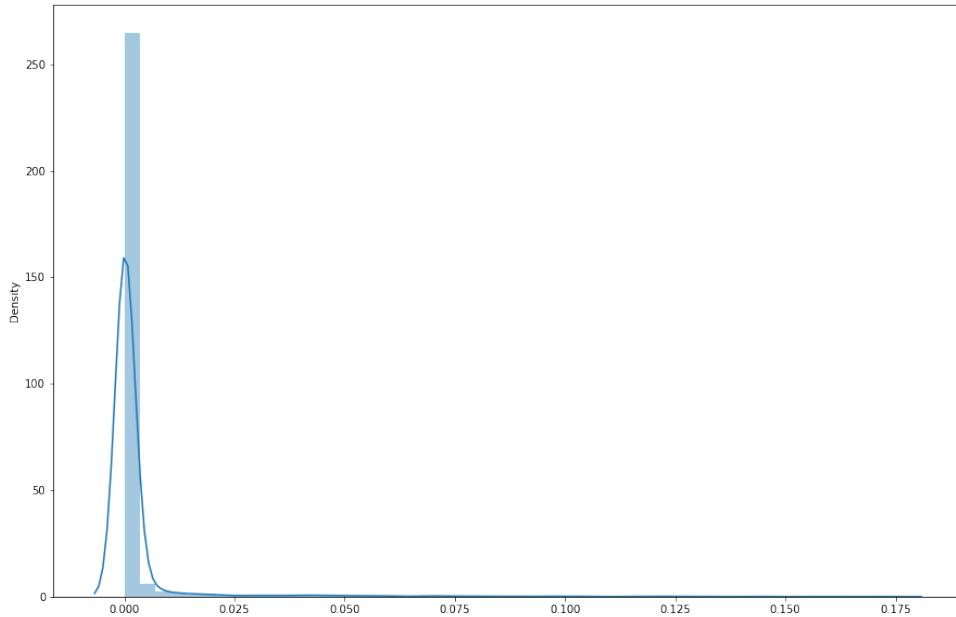


Figura 7.8: Eigenvector Centrality Distribution Plot

7.5.3 Closeness Centrality

La Closeness Centrality misura la posizione di ogni individuo nella rete attraverso una prospettiva diversa dalle altre metriche di rete, catturando la distanza media tra ogni vertice e ogni altro vertice della rete. Una centralità di prossimità bassa significa che una persona è direttamente collegata o "intermediario" dalla maggior parte degli altri nella rete. Al contrario, i vertici in posizioni molto periferiche possono avere punteggi di centralità di prossimità elevati, infatti essi sono lontani e quindi la distanza da percorrere è elevata. Pensate alla vicinanza, paradossalmente, come a un punteggio "distanza". In alcuni casi l'inverso della distanza media rispetto ad altre della rete è usato come misura di centralità di prossimità. In questo caso, valori più alti indicano una posizione più centrale.

Osservazioni Nella figura Fig.7.9 viene riportato il grafico di distribuzione dei nodi all'interno della rete secondo la metrica della closeness centrality. Come si può evincere c'è un elevato numero di nodi con closeness centrality compresa tra 0.15 e 0.30, indice del fatto che parti della stessa rete risultino ben collegate fra loro. Alle estremità dei valori centrali, quindi valori di closeness centrality superiori a 0.30, nuovamente indicatore del fatto che non vi sia un nodo in particolare in grado di trasferire informazioni con un'alta velocità tale da permettergli di raggiungere gran parte della rete stessa. In altre parole, sopra un certo limite di importanza il numero di persone influenti è anche qui molto limitato, e quindi sono pochi i soggetti con collegamenti molto superiori in numero rispetto al valore centrale.

Top5 Closeness Centrality Come fatto per le centralità precedenti, anche in questo caso si sono presi i primi 5 nodi per valori di *Closeness Centrality*.

- Barack Obama - 0.3588
- Mariya Gabriel - 0.3235
- Michael Roth - 0.3205

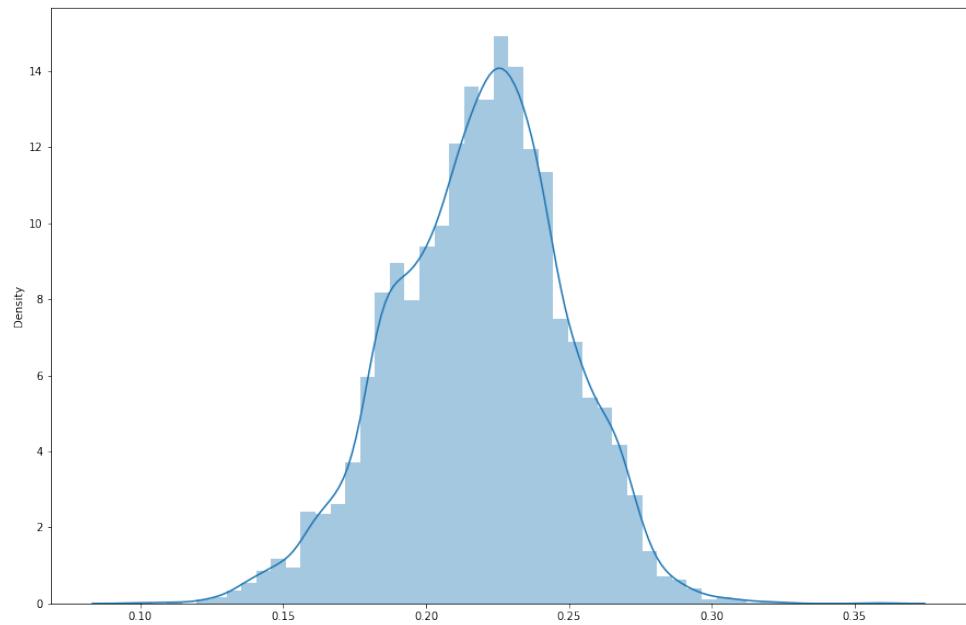


Figura 7.9: Closeness Centrality Distribution Plot

- Niels Annen - 0.3174
- Mariano Rajoy Brey - 0.3125

7.5.4 Current flow closeness Centrality

La Current-flow closeness centrality è una variante della Closeness Centrality basata sulla resistenza effettiva tra nodi in una rete. Questa metrica è anche conosciuta come centralità dell'informazione.

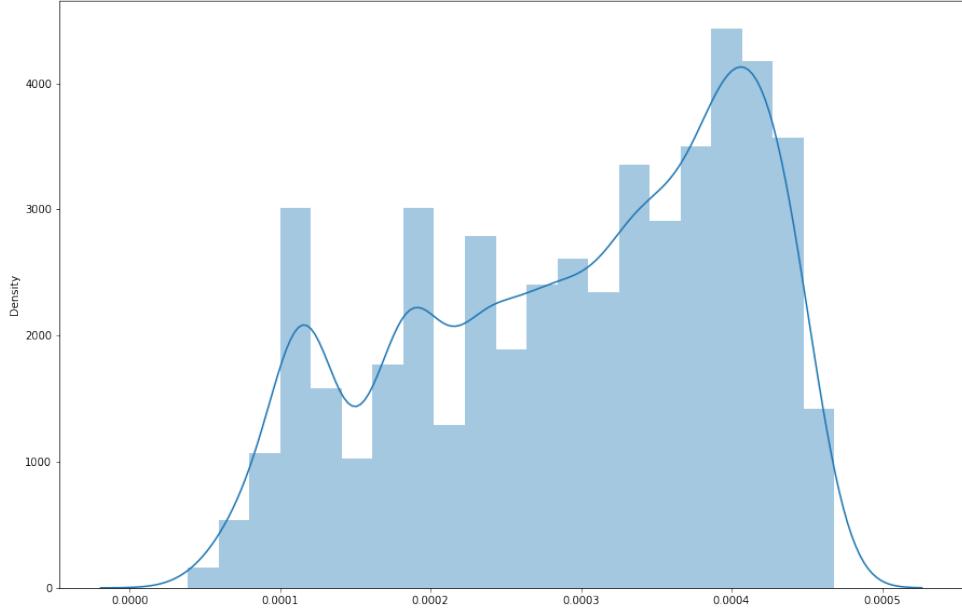


Figura 7.10: Current flow closeness Centrality Distribution Plot

Osservazioni Nella figura Fig.7.10 notiamo come il numero di percorsi risultati essere sempre stimabile in un range di valori estremamente bassi, indice del fatto che la resistenza effettiva tra i vari nodi è estremamente bassa. Infatti, come abbiamo visto nelle precedenti analisi, questo è rappresentato anche dal fatto che la rete risulta ben connessa tra i vari nodi e con un numero elevato di collegamenti.

Top5 Current Flow Closeness Centrality Come fatto per le centralità precedenti, anche in questo caso si sono presi i primi 5 nodi per valori di *Current Flow Closeness Centrality*.

- Manfred Weber - 0.000468
- Barack Obama - 0.000468
- Katja Mast - 0.000467
- Joachim Herrmann - 0.000466
- Katarina Barley - 0.000466

7.5.5 Betweennes Centrality

La betweennes centrality ha come principio il fatto che se un individuo all'interno di una rete presidia quello che può definirsi un collo di bottiglia, è in grado di poter guadagnare maggior potere all'interno della rete, in quanto mette in comunicazione diverse comunità di una rete. Ad esempio nel caso di intermediazione tra due gruppi di politici, il mediatore assume un ruolo chiave nella comunicazione. Togliendolo, si rischia di perdere la comunicazione tra i gruppi.

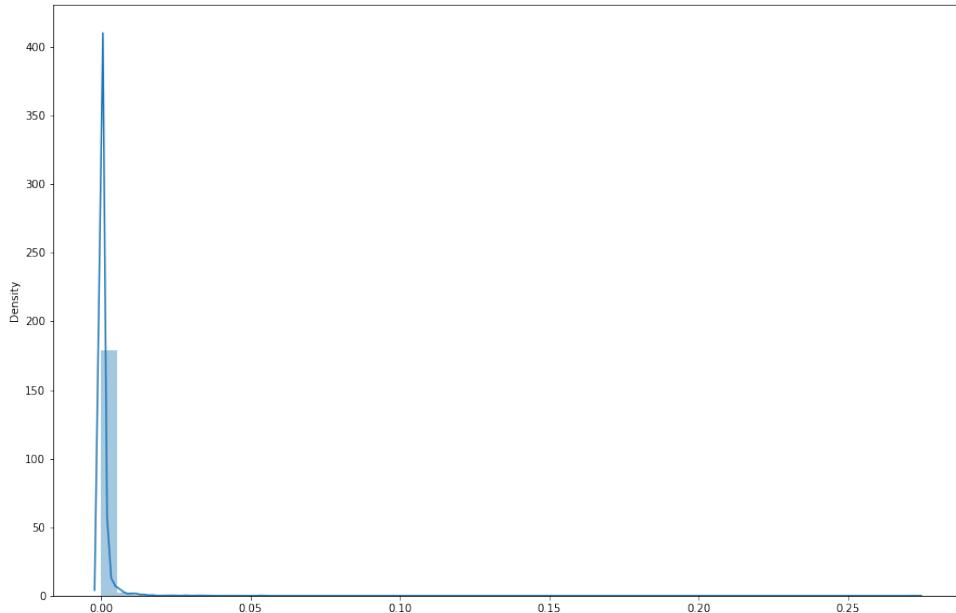


Figura 7.11: Betweennes Centrality Distribution Plot

Osservazioni Nella figura Fig.7.11 è visibile il grafico relativo alla distribuzione dei nodi in funzione del valore di betweennes centrality. Come si evince dal grafico sono moltissimi i nodi, quasi la totalità, che hanno una betweennes centrality molto bassa. Questo è indice del fatto che all'interno della rete non ci sono particolari nodi che fungono da bridge per diverse comunità o che comunque facciano parte di un collo di bottiglia all'interno della rete da cui trarre potere. E questo viene confermato dal fatto che in effetti, solitamente, queste entità sono molto limitate nella maggioranza dei casi di studio standard.

Top5 Betweennes Centrality Come fatto per le centralità precedenti, anche in questo caso si sono presi i primi 5 nodi per valori di *Betweennes Centrality*.

- Barack Obama - 0.2373
- Angela Merkel - 0.0933
- Manfred Weber - 0.0908
- Mitt Romney - 0.0670
- Betinho Gomes - 0.0651

7.5.6 Load Centrality

La *Load Centrality* di un nodo è la frazione di tutti i percorsi più brevi che passano attraverso quel nodo. La centralità del carico è leggermente diversa da quella tra un carico e l'altro. La Load Centrality di Goh et al.'s (riformulata da Brandes (2008)) è una misura simile a quella di un ipotetico processo di flusso. In particolare, si presume che ogni vertice invii un'unità di qualche "merce" l'uno all'altro vertice al quale è collegato (senza vincoli di capacità di bordo o vertice). *Nel nostro caso la merce sono i likes reciproci fra pagine Facebook (i nodi della rete).*

Implementazione Nel listato seguente è possibile osservare l'implementazione della Load Centrality.

```

1 #opzione per edge : edge load centrality
2 start = time.time()
3 load_centrality = nx.load_centrality(graph, v=5440, cutoff=3, normalized=True,
4 weight=None)
5 print(load_centrality)
6 end = time.time()
6 print('Tempo trascorso {}'.format(end - start))

```

Listing 7.3: Implementazione della load centrality relativa al nodo 5440 con un cutoff pari a 3

Risultati Il risultato dell'implementazione appena vista è $1.1466e - 07$, mentre il tempo di esecuzione è di circa 20 secondi. Questo tempo è però soggetto al nodo in questione e in particolar modo al *cutoff*, se infatti impostiamo un *cutoff* maggiore di 3, questo tempo tende ad aumentare in modo esponenziale.

In conclusione possiamo affermare che l'informazione (i *likes* reciproci) veicolata tra i vari nodi è esigua.

Top5 Load Centrality Come fatto per le centralità precedenti, anche in questo caso si sono presi i primi 5 nodi per valori di *Load Centrality*.

- Barak Obama - 0.2562
- Angela Merkel - 0.0532
- Manfred Weber - 0.0528
- Hillary Clinton - 0.0494
- Justin Trudeau - 0.0429

7.5.7 Harmonic centrality

La *Harmonic centrality* (conosciuta anche come centralità stimata) è una variante della centralità di prossimità, inventata per risolvere il problema che la formula originale aveva quando si trattava di grafi non collegati. Come per molti degli algoritmi di centralità, ha origine dal campo dell'analisi delle social network. La centralità armonica è stata proposta come alternativa alla centralità di prossimità, e quindi ha casi d'uso simili. Per esempio, potremmo usarlo se stiamo cercando di identificare in quale città collocare un nuovo servizio pubblico in modo che sia facilmente accessibile per i residenti. Se stiamo cercando di diffondere un messaggio sui social media potremmo usare l'algoritmo per trovare gli influencer chiave che possono aiutarci a raggiungere il nostro obiettivo.

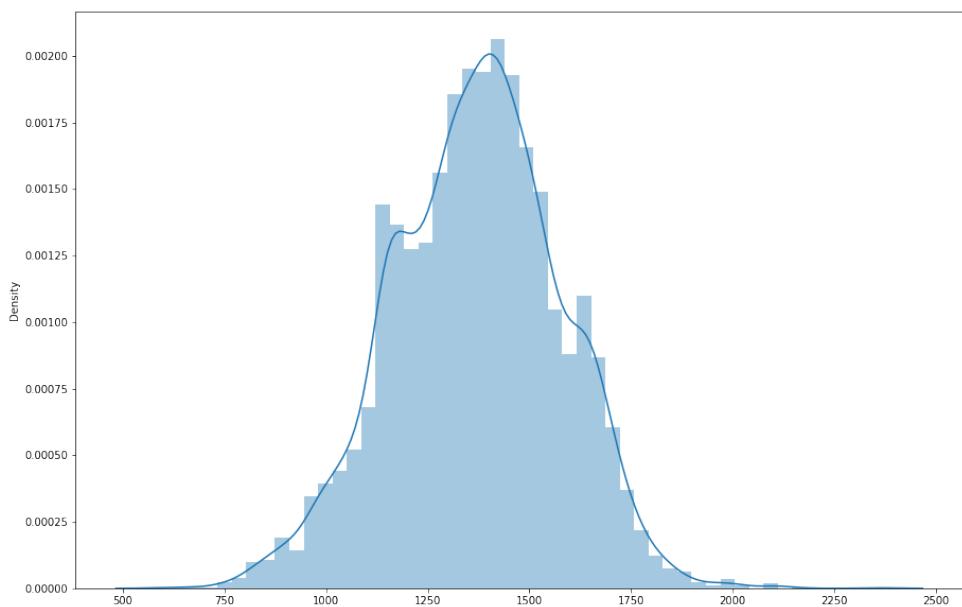


Figura 7.12: Harmonic Centrality Distribution Plot

Osservazioni Come si può osservare dalla figura Fig.7.12 non vi sono nodi o gruppi di nodi definibili "influencer", bensì gran parte dei nodi della nostra rete hanno valori bassi e simili di Harmonic Centrality, indice che nella rete non vi sono nodi rappresentativi o fondamentali da individuare al fine di veicolare quello potrebbe essere un messaggio importante da diffondere all'interno della rete.

Top5 Harmonic Centrality Come fatto per le centralità precedenti, anche in questo caso si sono presi i primi 5 nodi per valori di *Harmonic Centrality*.

- Barak Obama - 2359
- Mariya Gabriel - 2147
- Niels Annen - 2108
- Michael Roth - 2103
- Angela Merkel - 2102

7.5.8 Confronto "Top Five" Nodi

Dai top 5 nodi calcolati per le centralità precedenti possiamo osservare che politici come: Barak Obama, Angela Merkel e Manfred Weber compaiono ripetutamente nei diversi *rank*.

Ciò consente di affermare la loro importanza nel quadro politico mondiale, compaiono infatti come nodi fondamentali in diverse centralità calcolate.

7.5.9 Dispersion

La Dispersion ci permette di calcolare la dispersione tra due nodi u e v in un grafo G . Un collegamento tra due "attori" u e v ha un'elevata dispersione quando i loro legami (*mutual ties*) reciproci (s e t) non sono ben collegati tra loro.

```

1 start = time.time()
2 dispersion = nx.dispersion(graph,
3     u=0, v=4924,
4     normalized=True,
5     alpha=1.0,
6     b=0.0, c=0.0)
7 print(dispersion)
8 end = time.time()
9 print('Tempo trascorso {}'.format(end - start))

```

Listing 7.4: Implementazione del calcolo della dispersione relativa al nodo 0 e il nodo 4924

Osservazioni Il risultato del listato appena introdotto, fornisce come risultato 0.0286 in un tempo pari a 0.004 secondi. In questo specifico caso l'unica informazione che possiamo estrapolare è il grado di connessione tra i vari nodi. Valori bassi come in questo caso ci indicano quindi una buona connessione.

7.5.10 Vote Rank

Voterank - calcola una classifica dei nodi in un grafico G basato su uno schema di voto. Con Voterank, tutti i nodi votano per ciascuno dei suoi vicini e il nodo con i voti più alti viene eletto iterativamente. La capacità di voto dei nodi eletti è diminuita nei turni successivi. Nota: trattiamo ciascun bordo indipendentemente in caso di multi grafi.

```

1 start = time.time()
2 vote_rank = nx.voterank(graph, number_of_nodes=10)
3 print(vote_rank)
4 end = time.time()
5 print('Tempo trascorso {}'.format(end - start))

```

Listing 7.5: Implementazione dell'algoritmo di Vote Rank

Osservazioni Il risultato del listato appena presentato fornisce una lista di dieci nodi, quelli che hanno ottenuto nella classifica finale i primi dieci posti, questo ci permette quindi di osservare i nodi fondamentali più importanti all'interno della rete. **RISULTATO:** [1864, 4874, 5800, 5416, 1595, 4602, 3576, 3008, 1474, 2386] in un tempo di 0.426 secondi.

7.6 Clique, K-core e K-plex

Per quanto concerne lo studio delle Clique, dei K-Core e dei K-Plex, sono state implementate diverse metriche. Nel caso delle Clique, ovvero delle più comunemente note "Cricche", un sotto insieme di nodi del grafo G che hanno una connessione con tutti gli altri nodi del grafo. Mentre i K-Core e i K-Plex riguardano la ricerca delle varie entità maggiormente influenti all'interno della rete, ecco quindi che successivamente andremo a vedere diverse metriche e implementazioni utili al fine di trovare tali soggetti.

7.6.1 Clique

Nel listato che andremo ad introdurre qui di seguito andiamo a visionare quali sono le funzioni messe a disposizione della libreria *networkX* con l'obbiettivo di trovare e enumerare le varie clique.

```

1 #Returns the clique number of the graph
2 print('Number of Clique : {}'.
3       format(nx.graph_clique_number(graph)))
4 #Returns the number of maximal cliques in the graph
5 print('Graph number of Clique : {}'.
6       format(nx.graph_number_of_cliques(graph)))
7 #Returns the size of the largest maximal clique containing each given node.
8 print('Size of the largest maximal clique : {}'.
9       format(nx.node_clique_number(graph)))
10 #Returns the number of maximal cliques for each node
11 print('Number of maximal clique for each node : {}'.
12       format(nx.number_of_cliques(graph)))
13 #Clique containing node
14 node = 1972
15 print('Clique, containing node : {} - {} '.
16       format(node, nx.cliquesContaining_node(graph, nodes=node)))

```

Listing 7.6: Funzioni di NetworkX per le trovare le informazioni principali

Nel listato appena presentato, come già detto, permette di osservare quelle che sono le informazioni principali inerenti alle Clique presenti all'interno della Social Network. Nella fattispecie, tramite esso sappiamo che all'interno della rete stessa vi sono presenti 21 clique, la più grande ha una dimensione pari a 16. Nel medesimo listato c'è anche la possibilità di verificare a quale clique appartiene un determinato nodo. Nel listato successivo invece vi sono le istruzioni necessarie a trovare e enumerare le varie clique presenti all'interno della rete.

```

1 print('clique enumerate')
2 print(list(nx.algorithms.clique.enumerate_all_cliques(graph)))
3
4 print('find cliques')
5 print(list(nx.algorithms.clique.find_cliques(graph)))

```

Listing 7.7: Funzioni di NetworkX per le trovare le Clique

7.6.2 K-Core

Il k-core ci permette di identificare i sottografi massimali che contengono nodi di grado k o superiore. In particolare è una delle metriche utilizzate per identificare le varie entità all'interno della rete. Nel caso di reti pesate, è possibile introdurre un ultimo concetto, ovvero il concetto di F-groups. Gli F-groups restituiscono i gruppi massimali formati da triadi «fortemente transitive» e «debolmente transitive». Nel nostro caso l'implementazione non tiene conto di quest'ultimo concetto introdotto. Il tutto è osservabile nella figura Fig.7.13

```

1 #A k-core is a maximal subgraph that contains nodes of degree k or more.
2 k_core_with_k = nx.k_core(graph, k=20, core_number= core_number)
3 print('K = 20 , k_core : {}'.format(k_core_with_k))
4 plt.figure(figsize=(15,10))
5 plt.title('k corona with k = 20')
6 nx.draw(k_core_with_k, with_labels = True)

```

Listing 7.8: Funzioni di NetworkX per le K-core

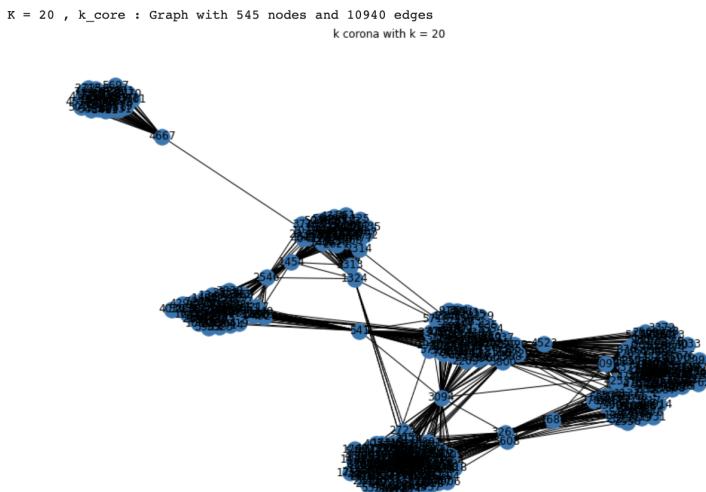


Figura 7.13: K-Core

Osservazioni Come si osserva dalla figura Fig.7.13, impostando il grado $k = 20$ è possibile ottenere una rappresentazione che ci permette di visualizzare come vi siano sei k-core, messi in collegamento da alcuni nodi o in alcuni casi, piccoli gruppi di nodi, che permettono quindi la comunicazione tra vari k-core. Nel caso in cui si andasse a rimuovere tali nodi, la rete perderebbe questa funzionalità di comunicazione.

7.6.3 K-Shell

La k-shell è il sottografo indotto da nodi con *core number* k . Cioè, nodi nel k-core che non sono nel $(k+1)$ core. Il tutto è visibile nella figura Fig.7.14.

```

1 k_ = 20
2 k_shell_with_k = nx.k_shell(graph, k=k_, core_number= core_number)
3 print('K = {} , k_shell : {}'.format(k_, k_shell_with_k))
4 plt.figure(figsize=(15,10))
5 plt.title('k shell with k = {}'.format(k_))
6 nx.draw(k_shell_with_k, with_labels = True)

```

Listing 7.9: Funzione K-Shell

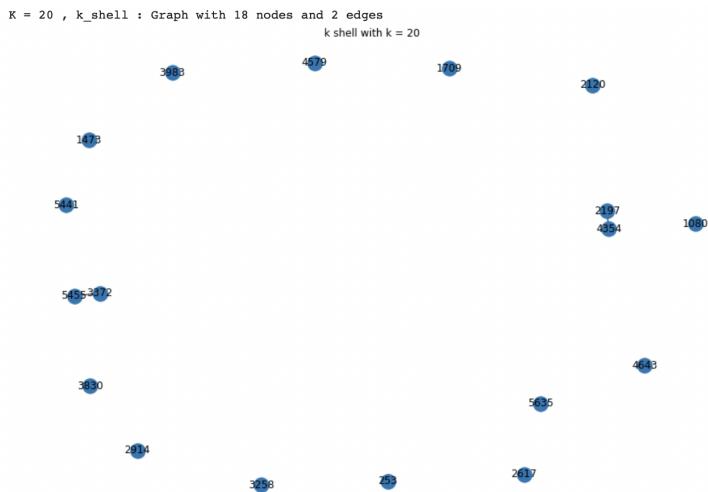


Figura 7.14: K-Shell

Osservazioni Uno studio recente mostra che l'accuratezza del metodo k-shell nel determinare la "corazza" dei nodi in un processo di diffusione è in gran parte influenzata dall'esistenza di un gruppo simile al nucleo, che ha un indice k-shell di grandi dimensioni ma una bassa efficienza di diffusione. Sulla base dell'analisi della struttura dei gruppi simil-core nelle reti del mondo reale, scopriamo che i nodi nel gruppo simil-core sono reciprocamente collegati con pochissimi collegamenti esterni dal gruppo (guarda sezione 7.6.2). Definendo una misura di importanza di diffusione per ogni bordo in base al numero di collegamenti di uscita delle sue due estremità, siamo in grado di identificare i collegamenti ridondanti nel processo di diffusione, che hanno un'importanza di diffusione relativamente bassa, ma che portano a formare un gruppo simile al nucleo collegato localmente e densamente. Dopo aver filtrato i collegamenti ridondanti e applicato il metodo k-shell alla rete residua, si ottiene una rinnovata coreness ks per ogni nodo che è un indice più accurato per indicare la sua importanza di localizzazione e diffusione influenza nella rete originale. Inoltre, troviamo che le prestazioni degli algoritmi di ranking basati sulla rinnovata corazza sono anche notevolmente migliorate. I nostri risultati aiutano a scomporre più accuratamente la struttura centrale della rete e identificare i nodi influenti nei processi di diffusione. Tale nodi sono rappresentati nella figura Fig.7.14.

7.6.4 K Crust

Il k-crust è il grafico G con i bordi del k -core rimossi e i nodi isolati trovati dopo la rimozione dei bordi. Nel listato seguente andiamo a vedere l'implementazione di tale analisi.

```

1 k_crust_with_k = nx.k_crust(graph, k=20, core_number= core_number)
2 print('K = 20 , k_crust : {} '.format(k_crust_with_k))
3 plt.figure(figsize=(15,10))
4 plt.title('k crust with k = 20')
5 nx.draw(k_crust_with_k, with_labels = True)
```

Listing 7.10: Funzione K-Crust

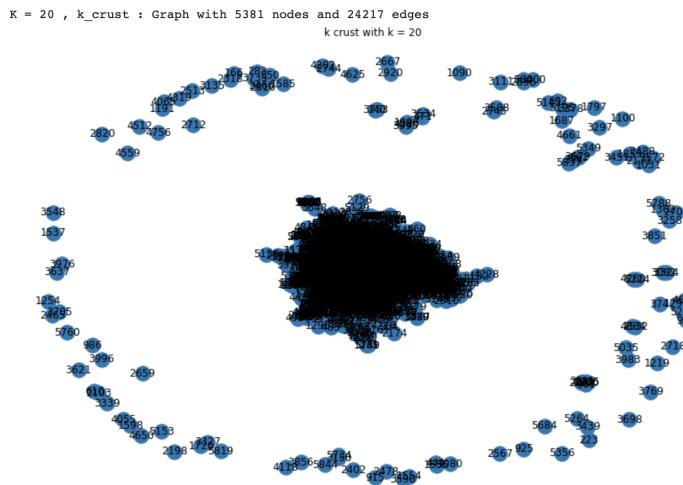


Figura 7.15: K-Crust

Osservazioni Nella figura Fig.7.15 è osservabile il risultato dell'analisi appena introdotta. Come si può osservare il k-crust permette di comprendere la struttura data dall'unione di tutte le k "corazze" con indici più piccoli o uguali a k . Per quanto riguarda i valori di indice esattamente uguali a k bisogna rifarsi all'analisi delle k -corone.

7.7 K-Corona

La k-corona è il sottografo dei nodi nel k-core che hanno esattamente k vicini nel k-core. Nel listato seguente c'è il codice relativo alla analisi delle k-corone, mentre nella figura Fig.7.16 è visionabile il risultato di tale implementazione.

```

1 k_corona_with_k = nx.k_corona(graph, k=20, core_number= core_number)
2 print('K = 20 , k_corona : {}'.format(k_corona_with_k))
3 plt.figure(figsize=(15,10))
4 plt.title('k corona with k = 20')
5 nx.draw(k_corona_with_k, with_labels = True)

```

Listing 7.11: Funzione K-Corona

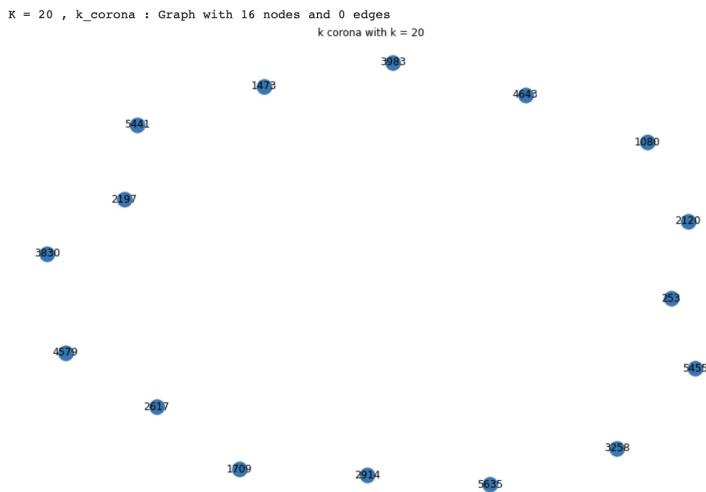


Figura 7.16: K-Corona

7.7.1 K-Truss

il K-Truss è il sottografo massimale indotto di G che contiene almeno tre vertici dove ogni bordo è incidente ad almeno $k - 2$ triangoli.

Note utili Una k-clique è un $(k-2)$ -truss e un k-truss è un $(k+1)$ -core.

```

1 k_truss_with_k = nx.k_truss(graph, k=20)
2 print('K = 20 , k_truss : {}'.format(k_truss_with_k))
3 plt.figure(figsize=(15,10))
4 plt.title('k truss with k = 20')
5 nx.draw(k_truss_with_k, with_labels = True)

```

Listing 7.12: Funzione K-Truss

```
K = 20 , k_truss : Graph with 126 nodes and 2688 edges  
k truss with k = 20
```

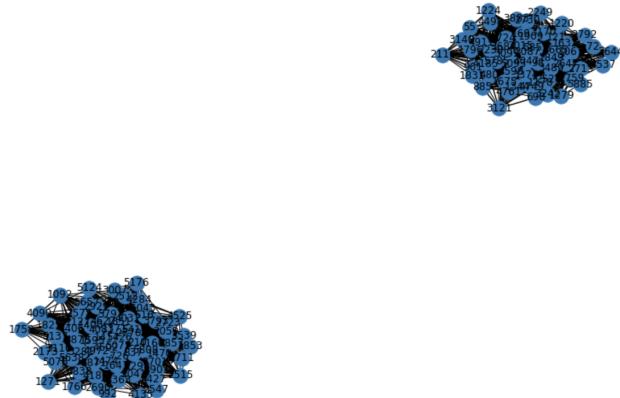


Figura 7.17: K-Truss

Osservazioni Alcuni studiosi sostengono che ciò che manca da molte delle definizioni di gruppi coesivi è la forza di avere percorsi multipli che si uniscono agli attori per una data relazione sociale. Questo è il punto di partenza che si è usato per definire un "traliccio". In particolare, si può fare questa osservazione sulle strutture sociali: se due attori sono fortemente legati, è probabile che condividano legami anche con altri. Creeremo qualcosa chiamato k-truss, in cui un legame tra A e B è considerato legittimo solo se supportato da almeno $k - 2$ altri attori che sono legati ad A e B . Il traliccio può essere considerato un'ulteriore estensione della cricca. Si può vedere la cricca di ordine k come un sottografo in cui ogni bordo si unisce a due vertici che hanno almeno $k - 2$ vicini comuni. Detto in un altro modo, ogni bordo è rinforzato da $k - 2$ coppie di bordi che fanno un triangolo con il bordo originale. Nel nostro caso impostando il parametro $k = 20$, è osservabile che vi sono due macro "tralicci" composti da 126 nodi e 2688 archi.

7.8 Ego Networks, Triadi e Articulation Points (Cut-Vertices)

7.8.1 Ego Networks

Le *Ego Networks* sono sottoreti che sono centrate su un certo nodo. Su Facebook e Linkedin, questi sono semplicemente descritti come "la tua rete". Avere un insieme di dati più grande ci permette di esaminare e confrontare le reti dell'ego di varie persone.

Per ricavare le ego-networks si esegue una ricerca di "primo respiro" (descritta in Breadth-First Traversal), e limitando la profondità della nostra ricerca tramite il parametro del raggio di rete ad un piccolo valore, di solito non più di 3. Nel caso invece della BFS tradizionale costruiamo un albero di collegamenti che abbiamo seguito per arrivare ad un determinato nodo. Per produrre una ego network andiamo a catturare tutti i collegamenti tra i vicini dei nodi.

```
1 ego_graph = nx.ego_graph(graph, 5440, radius=3, center=True, undirected=False,  
    distance=None)  
2 plt.figure(figsize=(10,10))  
3 pos = nx.spring_layout(ego_graph)  
4 nx.draw(ego_graph, pos, with_labels=True)  
5 plt.show()
```

Il risultato di tale implementazione è visibile nella figura Fig.7.18

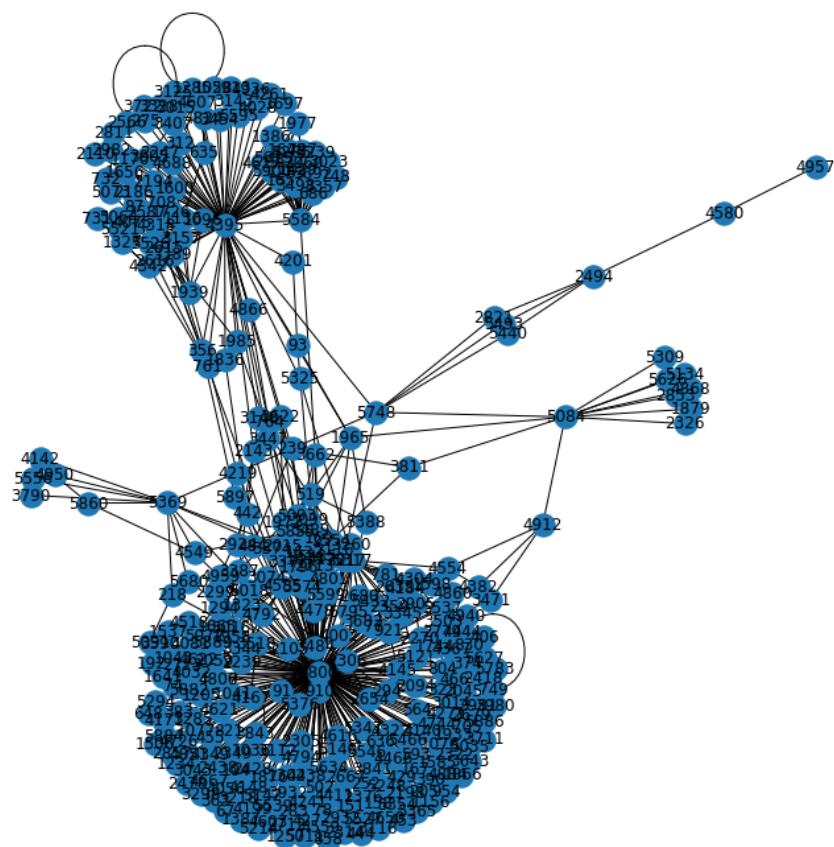


Figura 7.18: La Ego-Network Relativa al nodo 5440

7.8.2 Triadi

Le triadi possono essere viste come il più piccolo insieme di persone che formano una società, ecco quindi che una triade non è altro che un insieme di tre nodi, questi possono essere di quattro tipologie principali: Chiuse, Aperte, Connesse e Non connesse. Come si può notare, le prime due hanno tutti i loro nodi interconnessi e, pertanto, presentano un interesse significativo. Va inoltre aggiunto che durante uno studio sociologico è stato dimostrato che: i vincoli più stabili sono quelli, simmetrici, quelli più duraturi nel tempo sono le triadi e i meno stabili sono i vincoli asimmetrici.

Un'ultima osservazione va fatta per quanto riguarda le triadi proibite e i buchi strutturali, dove i primi sono il classico esempio in cui una triade non può chiudersi. Il classico esempio di tale tipologia è l'intermediazione immobiliare, nella quale il venditore e il compratore non devono mai mettersi in contatto. Mentre i buchi strutturali hanno anche il compito di mettere in collegamento diverse comunità.

Tramite il listato seguente è possibile svolgere quello che è un censimento delle varie triadi. Nel nostro caso non è stato implementato in quanto il grafo deve essere diretto.

```
1 triadi = nx.triadic_census(graph)
```

7.8.3 Articulation Points (Cut-Vertices)

Un vertice in un grafo connesso non diretto è un Articulation Points (o Cut-Vertices) se rimosso (e i bordi attraverso di esso) si disconnette il grafo. I punti di articolazione rappresentano vulnerabilità in una rete connessa - singoli punti il cui fallimento dividerebbe la rete in 2 o più componenti. Sono utili per la progettazione di reti affidabili.

Per un grafo non diretto disconnesso, un punto di articolazione è una rimozione di vertici dal grafo principale ma che aumenta il numero di componenti collegati.

```
1 articulation_points = nx.articulation_points(graph)
2 print('N of articulation_points {}'.
3     format(len(list(nx.articulation_points(graph)))))
4 print(list(nx.articulation_points(graph)))
```

Il listato appena introdotto fornisce gli *Articulation Points* e il numero di essi.

Risultati N° of articulation points: 448

Lista di punti: [191, 1734, 833, 2604, 476, 5864, 1956, 2208, ...]

I risultati mostrati sopra, riportano parte dell'elenco degli *articulation points* della rete e il loro numero complessivo.

7.9 Community Detection

L'idea cardine alla base della social network analysis è comprendere, misurare e rappresentare le relazioni tra le diverse entità facenti parti della rete, sia a livello individuale che a livello di gruppi di individui. Ecco quindi che in questa sezione andremo a rappresentare la problematica della community detection. Quest'ultima problematica ci permette di rappresentare i vari nodi in sottoinsiemi chiamati appunto comunità.

Nella fattispecie, i nodi appartenenti alle diverse comunità, rappresentano un sottoinsieme coeso, che condivide con tutti i nodi un insieme di features fondamentali, nel nostro caso una delle feature è rappresentata dai link con altri nodi. Ma l'insieme di features che stanno alla base di una comunità non sempre è formato dalla singola informazione inerente al link dei diversi nodi, ma può contenere anche informazioni di genere, età, nazionalità etc.. Tornando però al nostro caso di studio, possiamo concludere che l'informazione rilevante rimane il collegamento di diversi nodi (o meglio pagine). Si va quindi a definire una comunità in base alle relationship delle varie pagine mutualmente connesse.

Per svolgere tale studio, si usando un approccio *Greedy "Goloso"*, in particolare tramite l'algoritmo "Greedy Modularity Communities" sviluppato da Clauset A., Newman M. E., & Moore C., esso ci permette di attuare una strategia nella quale ad ogni iterazione viene fatta la "scelta" più golosa, ovvero la scelta che ha un miglior comportamento rispetto alle possibili alternative. L'algoritmo appena citato, è uno degli algoritmi messi a disposizione dal pacchetto NetworkX. Vi sono però ulteriori alternative, come quelle presentate nell' elenco seguente:

- Girvan-Newman algorithm: L'algoritmo Girvan-Newman rileva le comunità rimuovendo progressivamente i bordi (edges) dalla rete originale.
- Fluid Communities algorithm: Questo algoritmo si basa sulla semplice idea di fluidi che interagiscono in un ambiente, espandendosi e spingendosi a vicenda.
- Label Propagation algorithm: La propagazione delle etichette è un algoritmo di apprendimento automatico semi-supervisionato che assegna le etichette a punti di dati precedentemente non marcati.
- Clique Percolation algorithm: L'algoritmo trova comunità k-clique in un grafo usando il metodo della percolazione.
- Kernighan-Lin algorithm: Questo algoritmo divide una rete in due insiemi scambiando iterativamente coppie di nodi per ridurre il taglio del bordo (edge) tra i due insiemi.

La scelta è però ricaduta sul primo in quanto nonostante la limitazione dei "weights" non considerati, in quanto la rete utilizzata non presenta alcun tipo di peso. Risultava altresì un'ottima alternativa anche in termini di complessità computazionale.

Nel listato successivo andiamo quindi ad osservare una sua implementazione, mentre nella figura Fig.7.19 andiamo a vedere il risultato di tale implementazione.

```
1 from networkx.algorithms.community import greedy_modularity_communities
2 communities = list(greedy_modularity_communities(graph))
```

Listing 7.13: Implementazione del Girvan-Newman algorithm

Sul listato appena presentato, esso ci permette come "main task" di trovare le varie comunità. In particolare la sua esecuzione restituisce un insieme di "frozen sets" ciascuno dei quali rappresenta una comunità. Nel nostro studio, usando la network già descritta, ci permette di trovare 36 comunità di diverse dimensioni dalla più grande che considera 1400 nodi circa, alle più intermedie con circa un centinaio di nodi, fino ad arrivare alle più piccole che contengono a malapena 3

nodi. Un'osservazione sulla figura Fig.7.19 va fatta sul fatto che le comunità rappresentate sono le principali quattro per numero di nodi, nonostante ciò c'è una netta problematica relativa alla separazione delle diverse comunità, in particolare esse sembrano interconnettersi tra loro.

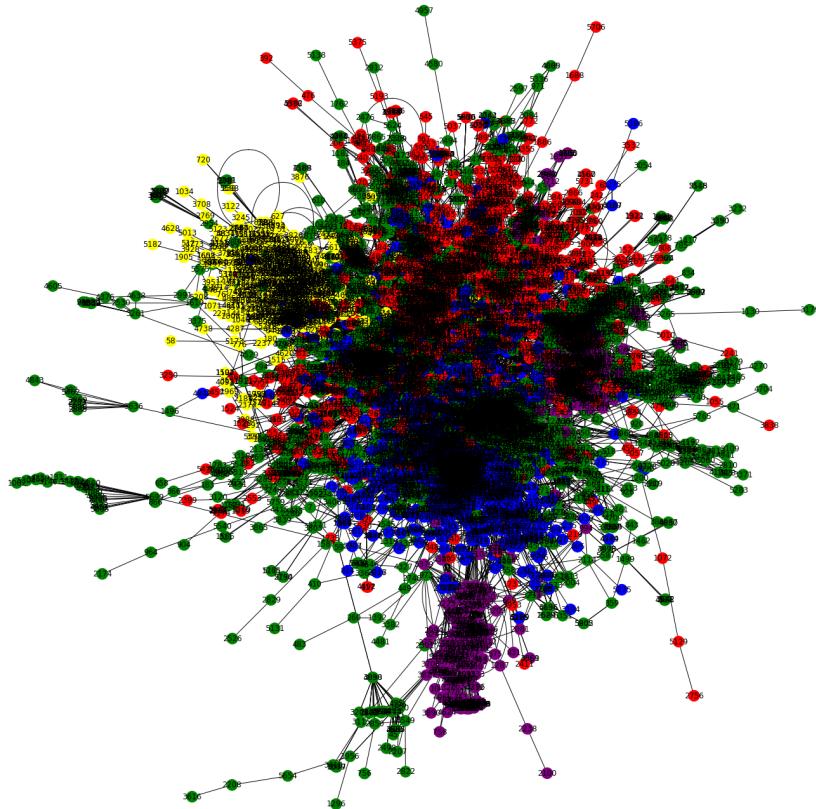


Figura 7.19: Le principali Comunità all'interno della Network

Colore	N° Nodi	Comunità
Blu	1405	Comunità 1
Rosso	1106	Comunità 2
Giallo	568	Comunità 3
Viola	477	Comunità 4
Verde	436	Comunità 5

Tabella 7.1: Leggenda comunità e numero nodi

Ecco quindi che nella figura Fig.7.20 abbiamo optato per uno dei layout presentati precedentemente, al fine di ottenere una visualizzazione più corretta. Tra le possibili rappresentazioni, vi sono diverse alternative che abbiamo testato, per poter migliorare quello che è l'aspetto grafico e che al contempo ci fornisce una maggior interpretabilità. Visti però i scarsi risultati ottenuti abbiamo preferito non aggiungere tali test all'interno di questa relazione, oltre a quelli già visti.

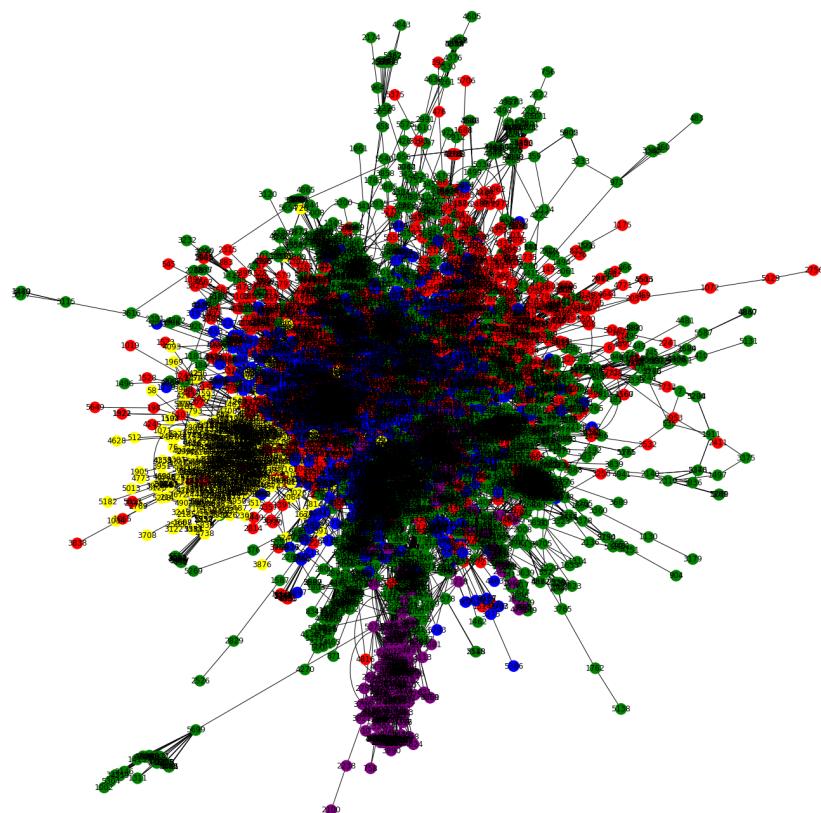


Figura 7.20: Kamada Kawai Layout con le diverse comunità

7.10 Link Prediction

Link prediction è la tecnica di base dei vari recommender system applicati alla social network analysis. Un recommender system è un software di filtraggio dei contenuti che crea delle raccomandazioni personalizzate specifiche per l'utente così da aiutarlo nelle sue scelte. Viene utilizzato per diversi prodotti, come libri, musica, film, video, notizie e social media. Alcune delle reali applicazioni possono essere viste in Facebook quando consiglia nuovi amici o in Amazon quando predice per poi suggerire degli articoli a cui l'utente può essere interessato. Ecco quindi che reti sociali, anche quelle analizzate in questo studio, permettono una previsione dei collegamenti futuri. Data l'elevata complessità che un semplice task di predizione può raggiungere, considerando i diversi aspetti fondamentali che permettono tutto ciò, abbiamo deciso di trattare questo argomento in modo elementare col fine solo di una maggior completezza dello studio intrapreso in questo capitolo sulle Social Networks.

Per svolgere questo task abbiamo ripescato una delle comunità viste precedentemente. La comunità scelta è stata selezionata tra quelle meno numerose, nella fattispecie, essa è composta da 7 nodi (*comunità 27*) e 9 archi. La comunità esaminata è visibile nella figura Fig.7.21

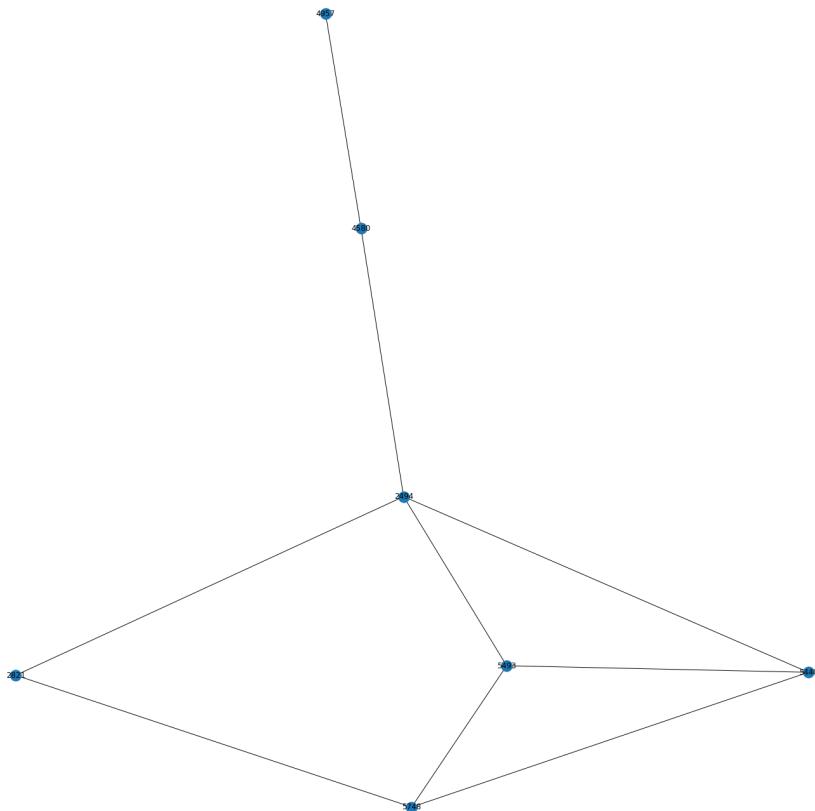


Figura 7.21: Comunità 27

Col fine di verificare il funzionamento dei vari algoritmi e la loro efficienza, abbiamo eliminato degli archi tramite la funzione messa a disposizione da networkX. Nella fattispecie gli archi eliminati sono quelli relativi ai nodi: 5493 e 5440 e i nodi: 5493 e 5748. Nella figura Fig.7.22 è

visibile il risultato di tali operazioni.

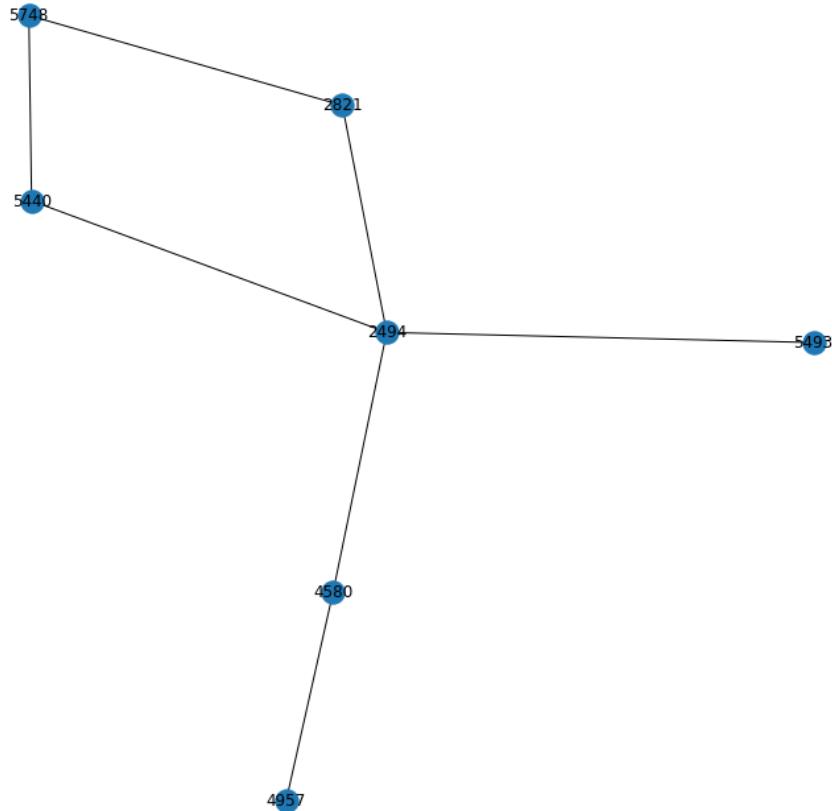


Figura 7.22: Comunità 27 con archi rimossi

Alla base di questo task di studio c'è l'obiettivo di verificare se, dato il grafo di partenza (con cut edges), tramite le previsioni sia possibile ottenere nuovamente il grafo originale (senza cut edges). A seguire vedremo i diversi algoritmi e i loro risultati.

7.10.1 Jaccard Coefficient

L'indice di Jaccard, noto anche come coefficiente di similarità di Jaccard, è un indice statistico utilizzato per confrontare la similarità e la diversità di insiemi campionari. Il coefficiente di Jaccard misura la similarità tra insiemi campionari, ed è definito come la dimensione dell'intersezione divisa per la dimensione dell'unione degli insiemi campionari. Nella previsione dei collegamenti, tutti i vicini di un nodo vengono trattati come un insieme e la previsione viene eseguita calcolando e classificando la somiglianza dell'insieme adiacente di ciascuna coppia di nodi. L'output dell'algoritmo consiste in una lista di tre elementi (u, v, p) dove u, v è il nuovo arco predetto e p è il coefficiente di Jaccard.

```

1 preds = nx.jaccard_coefficient(graph)
2 for u, v, p in preds:
3     print(f"({u}, {v}) -> {p:.8f}")

```

Listing 7.14: Algoritmo di previsione basato sul Jaccard Coefficient

Output Come possiamo vedere nella equazione seguente, l'algoritmo predice una numero di archi maggiore a quelli rimossi, ecco quindi che l'obiettivo che ci siamo posti diventa molto più complesso. Va anche osservato che gli archi rimossi indicati con * sono predetti con un coefficiente medio bassi.

$$\begin{aligned}
 & (5440, 4957) - > 0.00000000 \\
 & (5440, 4580) - > 0.33333333 \\
 & (5440, 2821) - > 1.00000000 \\
 & *(5440, 5493) - > 0.50000000 \\
 & (4580, 5748) - > 0.00000000 \\
 & (4580, 2821) - > 0.33333333 \\
 & (4580, 5493) - > 0.50000000 \\
 & (2821, 4957) - > 0.00000000 \\
 & (2821, 5493) - > 0.50000000 \\
 & (5748, 4957) - > 0.00000000 \\
 & *(5748, 5493) - > 0.00000000 \\
 & (5748, 2494) - > 0.50000000 \\
 & (5493, 4957) - > 0.00000000 \\
 & (4957, 2494) - > 0.25000000
 \end{aligned} \tag{7.1}$$

7.10.2 Resource Allocation Index

L'algoritmo si basa sull'indice di allocazione delle risorse, misura utilizzata per calcolare la vicinanza dei nodi in base ai loro vicini condivisi. Introdotto da Tao Zhou Linyuan Lü e Yi-Cheng Zhang nel 2009, è parte di uno studio per le previsioni di collegamenti in varie Social Network e non solo. Anche in questo caso l'output dell'algoritmo consiste in una lista di tre elementi (u, v, p) dove u, v è il nuovo arco predetto e p è l'indice delle di allocazione delle risorse.

```

1 preds = nx.resource_allocation_index(graph)
2 for u, v, p in preds:
3     print(f"{{u}}, {{v}}) -> {p:.8f}")

```

Listing 7.15: Algoritmo di previsione basato sull'indice di allocazione delle risorse

Output L'output in questo caso, mediante quindi l'utilizzo dell' algoritmo "Resource Allocation Index" fornisce dei risultati simili a quelli precedentemente esposti. L'osservazione da svolgere è relativa all'indice che indica il coefficiente delle risorse allocate. Anche in questo caso i risultati sono **scarsi**.

$$\begin{aligned}
 & (5440, 4957) - > 0.00000000 \\
 & (5440, 4580) - > 0.25000000 \\
 & (5440, 2821) - > 0.75000000 \\
 & *(5440, 5493) - > 0.25000000 \\
 & (4580, 5748) - > 0.00000000 \\
 & (4580, 2821) - > 0.25000000 \\
 & (4580, 5493) - > 0.25000000 \\
 & (2821, 4957) - > 0.00000000 \\
 & (2821, 5493) - > 0.25000000 \\
 & (5748, 4957) - > 0.00000000 \\
 & *(5748, 5493) - > 0.00000000 \\
 & (5748, 2494) - > 1.00000000 \\
 & (5493, 4957) - > 0.00000000 \\
 & (4957, 2494) - > 0.50000000
 \end{aligned} \tag{7.2}$$

7.10.3 Triadic closure

La Triadic closure è un concetto nella teoria dei social network, studiata per la prima volta dal sociologo Georg Simmel nel 1908, per cui avendo tre nodi A, B e C se esistono le connessioni A-B e B-C allora ci sarà la tendenza a formare una nuova connessione tra A e C. Questo concetto è stato modellato e implementato in un semplice algoritmo che aggiunge nuovi archi andando a creare dei triangoli. L'algoritmo restituisce una lista di nuovi archi predicendoli partendo appunto dal concetto spiegato precedentemente. In estrema sintesi: Triadic closure per un grafico è la tendenza per i nodi che hanno un vicino comune di avere un arco tra di loro. Nel caso in cui più archi siano aggiunti nel grafico, questi sono gli archi che tendono a formarsi.

```

1 e = list(graph.edges())
2
3 def triadic(e):
4     new_edges = []
5
6     for i in e:
7         a, b = i
8
9         for j in e:
10            x, y = j
11
12            if i != j:
13                if a == x and (b, y) not in e and (y, b) not in e:
14                    new_edges.append((b, y))
15                if a == y and (b, x) not in e and (x, b) not in e:
16                    new_edges.append((b, x))
17                if b == x and (a, y) not in e and (y, a) not in e:
18                    new_edges.append((a, y))
19                if b == y and (a, x) not in e and (x, a) not in e:
20                    new_edges.append((a, x))
21
22    return new_edges
23
24 print("The possible new edges according to Triadic closure are :")

```

```
25 print(triadic(e))
```

Listing 7.16: Algoritmo di previsione "Triadic Closure"

Output L'output in questo caso non prevedere alcun coefficiente o indice, ma indica solo le possibili coppie di archi che potrebbero crearsi, basando su quanto *Georg Simmel* modello nel suo algoritmo. In questo caso solo uno dei archi rimossi è predetto (*). Va notato inoltre che le coppie di archi sono speculari, ovvero sono visibili ambo le combinazioni delle varie coppie.

$$\begin{aligned} & [(2494, 5748), (5440, 4580), (5440, 2821), *(5440, 5493), (5748, 2494), (5440, 2821), \\ & (4580, 5440), (2494, 4957), (4580, 2821), (4580, 5493), (4957, 2494), (2821, 5440), \\ & (2821, 4580), (2494, 5748), (2821, 5493), (2821, 5440), (5748, 2494), (5493, 5440), \\ & (5493, 4580), (5493, 2821)] \end{aligned} \quad (7.3)$$

7.10.4 Adamic Adar Index

Questa misura è stata introdotta nel 2003 dall'informatica statunitense Lada Adamic e da Eytan Adar, per prevedere i collegamenti mancanti in una rete, in funzione della quantità di collegamenti condivisi tra due nodi. Esso è calcolato come segue:

$$A.A.Index(X, Y) = \sum_{u \in N(x) \cap N(y)} \frac{1}{\log |N(u)|} \quad (7.4)$$

Nuovamente l'output sarà formato da una tupla composta dai nodi e l'indice appena espresso.

Output In questo caso gli archi rimossi sono stati predetti. L'indice in questo caso risulta però medio basso se rapportato con altri valori predetti di archi che però non c'erano inizialmente.

$$\begin{aligned} & [(5440, 4957, 0), \\ & (5440, 4580, 0.72), \\ & (5440, 2821, 2.16), \\ & *(5440, 5493, 0.72), \\ & (4580, 5748, 0), \\ & (4580, 2821, 0.72), \\ & (4580, 5493, 0.72), \\ & (2821, 4957, 0), \\ & (2821, 5493, 0.72), \\ & (5748, 4957, 0), \\ & *(5748, 5493, 0), \\ & (5748, 2494, 2.88), \\ & (5493, 4957, 0), \\ & (4957, 2494, 1.4)] \end{aligned} \quad (7.5)$$

7.10.5 Preferential Attachment

L'attaccamento preferenziale significa che più un nodo è connesso, più è probabile che riceva nuovi collegamenti. I nodi con grado più elevato diventano più vicini. Anche qui l'output è composto da una terna composta dai 2 nodi e dal coefficiente di preferenza.

Output Le considerazioni da svolgere anche in questo caso sono le medesime di quelle riportate precedentemente. In estrema sintesi gli archi rimossi sono predetti e il valore di *preferenza* risulta medio basso se rapportato con gli altri valori.

$$\begin{aligned} & [(5440, 4957, 2), \\ & \quad (5440, 4580, 4), \\ & \quad (5440, 2821, 4), \\ & \quad *(5440, 5493, 2), \\ & \quad (4580, 5748, 4), \\ & \quad (4580, 2821, 4), \\ & \quad (4580, 5493, 2), \\ & \quad (2821, 4957, 2), \\ & \quad (2821, 5493, 2), \\ & \quad (5748, 4957, 2), \\ & \quad *(5748, 5493, 2), \\ & \quad (5748, 2494, 8), \\ & \quad (5493, 4957, 1), \\ & \quad (4957, 2494, 4)] \end{aligned} \tag{7.6}$$

7.10.6 Tempistiche

Nella figura Fig.7.23 sono riportate le varie tempistiche per l'esecuzione dei codici introdotti nelle sezioni precedenti. Va però osservato che tali risultati sono vincolati al grafo considerato e alle sue dimensioni. Ecco quindi che tale valutazione ha come fine unico una maggior completezza dell'elaborato.

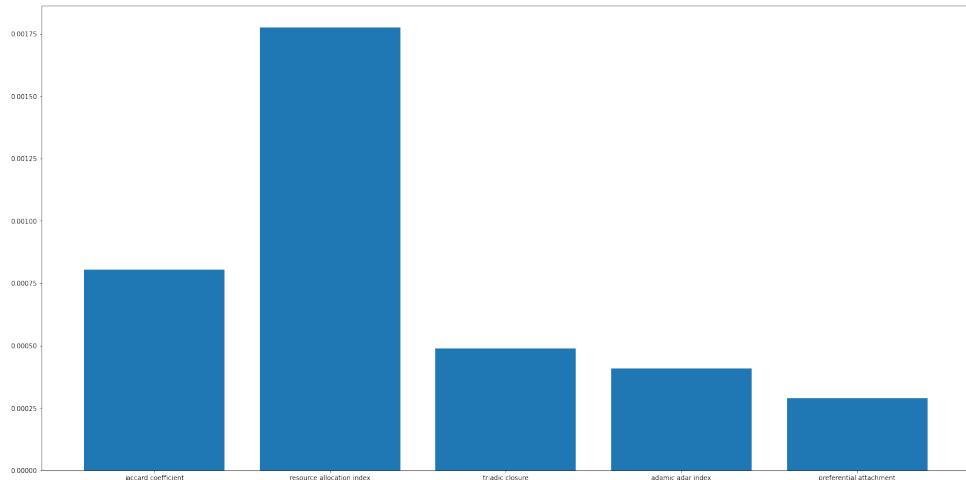


Figura 7.23: Tempi di esecuzione (s) dei diversi algoritmi

Capitolo 8

Chatbot

Per quanto concerne l'ultimo progetto sviluppato all'interno del corso di Data Science, ci si è dati l'obiettivo di realizzare un *chatbot* di un generico ristorante con il quale interagire al fine di ottenere dei consigli su quali portate siano più adatte in seguito ad alcune stringhe che l'utente può specificare in fase di comunicazione con il bot. Il chatbot permetterà quindi di intrattenere una conversazione col fine di fornire un servizio. La loro popolarità negli ultimi anni è aumentata anche grazie agli assistenti vocali come Siri, Alexa, etc.. Un chatbot unisce quindi diverse tecnologie tra cui Natural Language Processing (NLP), Natural Language Understanding (NLU) fino al Natural Language Generation (NLG). Il primo permette di dividere l'input dell'utente in frasi e parole. Il NLU aiuta il chatbot a capire ciò che l'utente ha detto usando sia oggetti linguistici generali che specifici del dominio come i costrutti lessici, sinonimi e temi. Infine il NLG fornisce un'esperienza significativa e personalizzata al di là delle risposte previste. Infine, si può concludere che vi sono oggi giorno due tecnologie principali per quanto riguarda i chatbot:

1. Chatbot basati su regole;
2. Chatbot AI.

La prima tipologia, possiamo paragonarla a delle FAQ interattive, e le loro capacità sono di base. La seconda invece, quelli basati su A.I. risultano essere più conversazionali, guidati dai dati e predittivi. Questi tipi di chatbot sono generalmente più sofisticati, interattivi e personalizzati. Infine, una precisazione doverosa da fare è riguardante lo studio di Gartner, il quale secondo alcune stime prevede che il 40% delle applicazioni chatbot/assistente virtuale di prima generazione lanciate nel 2018 saranno state abbandonate entro il 2020. Tale dato è da considerarsi come causa di alcuni limiti delle tecnologie disponibili fino ad oggi, ovvero una mancanza di interazione intelligente da parte del chatbot e la confusione sulla proprietà dei dati. Tra i limiti principali possiamo citare:

1. Difficoltà di creare un appeal globale
2. Normative per la protezione dei dati
3. Scarsa comprensione nelle conversazioni
4. Mancanza di dati di training

8.1 Tecnologie Utilizzate

Per lo sviluppo del progetto sono state utilizzate librerie di Python già viste nei capitoli precedenti come Pandas, Numpy, etc., la distribuzione Anaconda e il social network Telegram, per una

maggior integrazione. Tra gli editor utilizzati, troviamo VS CODE e alcuni tool relativi all'yaml validator¹. Infine, il tutto è stato sviluppato utilizzando il framework **RASA**. Esso infatti è uno dei tool *OPEN-SOURCE* maggiormente utilizzato, anche da aziende blasonate come *T-Mobile*, *Adobe* etc.. Tra i principali numeri che possiamo citare per dare un'idea relativa al framework utilizzato, possiamo citare oltre 10M di download, oltre 10k utenti attivi e oltre 600 collaboratori attivi.

8.2 Framework Rasa

Nella figura Fig.8.1 è possibile visionare quella che è la struttura base dei nostri file. Brevemente, oltre alle cartelle che vedremo di seguito nella relazione, la cartella models come suggerisce il nome contiene i modelli allenati tramite il comando **rasa train**, successivamente in test c'è il file relativo alle *test story's* che ci permette tramite il comando **rasa test** di verificare la bontà del nostro modello, ed infine alcuni file ".yml" tra cui config e credentials che ci permettono anche l'integrazione con alcune piattaforme social come Facebook, Telegram etc. Nel dettaglio, nella

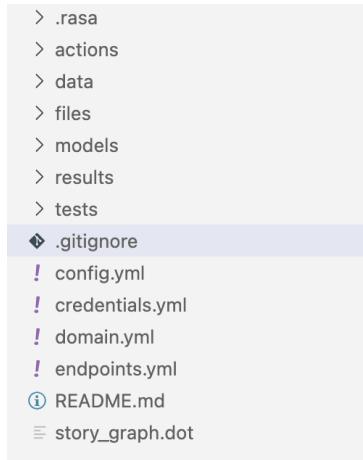


Figura 8.1: Framework Rasa

cartella actions Fig.8.2 è visibile il file **actions.py** il quale come vedremo di seguito ci permette di svolgere alcune action particolari tramite il linguaggio Python, azioni in risposta a degli intenti dell'utente in input. Infine, nella cartella *data* c'è il core del nostro chatbot, infatti come si vede nella figura Fig.8.3 vi sono i tre file principali che ci permettono di sviluppare le "regole" e le frasi con cui è possibile comunicare con il chatbot.

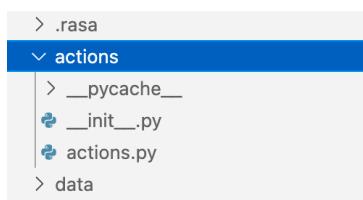


Figura 8.2: Framework Rasa "Actions"

¹ad esempio, <https://codebeautify.org/yaml-validator>



Figura 8.3: Framework Rasa "Data"

8.2.1 NLU

Come già detto nella presentazione del framework, all'interno della directory Data, abbiamo il primo file del nostro core, ovvero il file *NLU.yml*, dove al suo interno come possiamo vedere dalla figura Fig.8.4 è possibile specificare i vari **intent** del chatbot. Come è visibile, oltre a specificare il nome dello specifico intent, è possibile anche specificare una serie di esempi utili al chatbot per apprendere e comprendere al meglio l'intento dell'utente.

```

1   version: "3.0"
2   nlu:
3     - intent: greet
4       examples: |
5         - hey
6         - hello
7         - hi
8         - hello there
9         - good morning
10        - good evening
11        - moin
12        - hey there
13        - let's go
14        - hey dude
15        - goodmorning
16        - goodevening
17        - good afternoon
18

```

Figura 8.4: Intent NLU

Intent L'intent si riferisce all'obiettivo che il cliente ha in mente quando comunica con il chatbot.

```

99   - intent: order_something
100  examples: |
101    - I'd like to order {pasta}(food)
102    - I want to order [rice](food)
103    - Can I order a [pizzal](food)?
104    - I'd like to order a plate of {pasta}(food)
105    - Order a [pizzal](food)
222  - lookup: food
223  examples: |
224    - Pasta
225    - Pizza
226    - Rice
227    - Soup
228    - Meat
229    - Salmon
230    - Tuna
231    - Vegetables
232    - Eggs
233    - Drinks
234

```

Figura 8.5: Lookup Table

Lookup Table Per poter comunicare con il chatbot è probabile che sia necessario inserire anche qualche informazione aggiuntiva, come un piatto, un ingrediente etc.. Ecco quindi che al fine di evitare di dover ripetere le medesime frasi di esempio definiamo delle entità. Gli esempi per le

entità è possibile specificarli utilizzando le lookup table. Questo ci permette anche in futuro di non dover modificare l'intento, ma solo i possibili valori delle entità contenuti all'interno della lookup table. Nella figura Fig.8.5 è visibile quanto appena detto.

8.2.2 Slots

Le lookup table possono essere integrate con le slots che a tutti gli effetti sono la memoria del bot. Essi sono settati nel file **domain.yml**. Questa combinazione ci permette inoltre di processare i possibili input dell'utente tramite le varie actions definite nel file **action.py** con l'ausilio dei trackers.

```

79 |     diets:
80 |     type: text
81 |     mappings:
82 |     - type: from_entity
83 |       entity: diets
84 |
85 |     intolerance:
86 |     type: text
87 |     mappings:
88 |     - type: from_entity
89 |       entity: intolerance
90 |
91 |     regions:
92 |     type: text
93 |     mappings:
94 |     - type: from_entity
95 |       entity: regions

```

Figura 8.6: Slots

8.2.3 Rules

All'interno del file **rules.yml** è possibile specificare una serie di regole, per poter interagire con il chatbot. Nell'esempio in figura Fig.8.7 sono visibili le regole che ci permettono di iniziare, terminare una conversazione e fornire aiuto all'utente.

```

- rule: Start the conversation
  steps:
    - intent: start
    - action: utter_start

- rule: End the conversation
  steps:
    - intent: stop
    - action: utter_stop

- rule: Help the user
  steps:
    - intent: help
    - action: utter_help

```

Figura 8.7: Rules

8.2.4 Story

All'interno del file **story.yml** è invece possibile impostare quelle che sono delle vere e proprie storie di possibili comunicazioni fra utente e chatbot. In particolare possiamo specificare le

interazioni tramite una serie di intent e action che l'utente potrà richiedere e soprattutto le azioni con cui il chatbot deve rispondere nel caso in cui tale richiesta avverrà. Nella figura Fig. 8.8 è riportato un esempio di quanto appena detto.

```
186 - story: preferences ingredients story
187 steps:
188 - intent: greet
189 - action: utter_greet
190 - intent: preference_ingredients
191 - action: action_preference_ingredients
192 - intent: order_something_generic
193 - action: utter_order_info
194 - intent: order_something
195 - action: utter_order
196 - action: utter_time
197 - intent: choose_time
198 - action: action_save_order
199
```

Figura 8.8: Story's

8.2.5 Action

Nella figura Fig.8.9 è possibile definire le *custom actions*, ovvero delle possibili azioni che il nostro bot può svolgere in modo del tutto automatico. Nella figura è possibile vedere una delle azioni implementate che permette di salvare l'ordine dell'utente all'interno di un file CSV².

```

26  class ActionSaveOrder(Action):
27
28      def name(self) -> Text:
29          return 'action_save_order'
30
31      def run(self, dispatcher: CollectingDispatcher,
32             tracker: Tracker,
33             domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
34
35          # get slots values
36          food = str(tracker.get_slot('food'))
37          time = str(tracker.get_slot('time'))
38          # create a random order ID
39          order_id = f'AA{random_with_N_digits(5)}'
40
41          # check if orders file exists
42
43          if os.path.exists(orders_filename):
44              # append if already exists
45              file = open(orders_filename, 'a', newline='')
46          else:
47              # make a new file if not
48              file = open(orders_filename, 'w', newline='')
49              writer = csv.writer(file)
50              writer.writerow(['Order ID', 'Time', 'Plate'])
51
52              writer = csv.writer(file)
53              writer.writerow([order_id, time, food])
54              file.close()
55              dispatcher.utter_message(text=f'Your order is safe and sound! The order ID is {order_id}.')
56
57          return []
58

```

Figura 8.9: Actions

8.2.6 Pipeline

In Rasa i messaggi scritti dagli utenti al bot vengono processati da una sequenza di componenti. Tali componenti possono essere specificati attraverso la pipeline che si trova nel file **config.yml**. Scegliere una pipeline ci permette di personalizzare il nostro modello, adattandolo al *dataset*. Si è fatta la scelta di utilizzare la pipeline di default a cui poi si è aggiunto un componente, il **RegexEntityExtractor**, per cercare di estrarre le entità basandosi sui pattern dei caratteri delle entità (cioè utilizzando le Regex). Nella figura Fig. 8.10 viene riportata la configurazione utilizzata per il componente che si è aggiunto rispetto alla pipeline di default.

```

- name: RegexEntityExtractor
  case_sensitive: False
  use_lookup_tables: True
  use_regexes: True

```

Figura 8.10: RegexEntityExtractor

²CSV ≡ Comma Separated Value

8.2.7 Integrazione con Telegram

Tramite il file **credentials.yml** è possibile attuare l'integrazione con i diversi social media, tra cui facebook, slack, etc... Nel nostro caso realizzeremo l'integrazione con Telegram. Ecco quindi che sarà necessario avere a disposizione un account Telegram, tramite il quale attraverso il bot **BotFather** è possibile creare il nostro bot. Sarà sufficiente quindi specificare il nome del bot e il suo username, così da avere l'access token da impostare all'interno del file di configurazione citato precedentemente.

Per realizzare tale integrazione si è reso necessario l'utilizzo di **ngrok**, il quale ha permesso di realizzare il *tunneling*, ovvero esporre al mondo esterno un server interno alla rete, occupandosi quindi sia dei NAT che dei possibili firewall, garantendo così una massima sicurezza. Nella figura Fig. 8.12 è possibile vedere la conversazione con il bot realizzata direttamente su Telegram.

```
1 ngrok http 5005
```

Listing 8.1: comando ngrok per attivare il tunnel

```
ngrok by @inconshreveable
Session Status           online
Session Expires          1 hour, 59 minutes
Version                  2.3.40
Region                   United States (us)
Web Interface             http://127.0.0.1:4040
Forwarding               http://33b3-95-252-172-177.ngrok.io -> http://localhost:5055
                         https://33b3-95-252-172-177.ngrok.io -> http://localhost:5055
Connections              ttl     opn     rt1     rt5     p50     p90
                         0       0       0.00    0.00    0.00    0.00
```

Figura 8.11: NGROK

```
1 telegram:
2   access_token: 5056600425:AAHRj_VWwY7NA0e53m2TTvm30nVM9KFpFA
3   verify: FoodOrdering_bot
4   webhook_url: "https://fb90-79-30-60-85.ngrok.io/webhooks/telegram/webhook"
```

Listing 8.2: configurazione integrazione con telegram

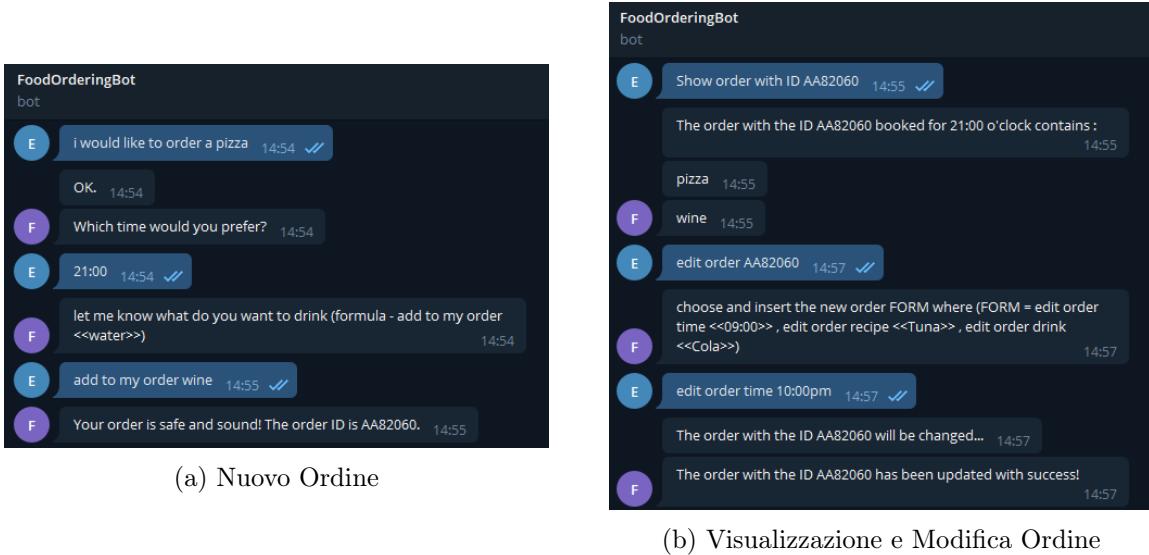


Figura 8.12: Chat Telegram

8.3 Funzionalità Aggiuntive

In questa sezione vengono riportate le funzionalità aggiuntive, implementate nel nostro progetto, rispetto alle esercitazioni del corso.

8.3.1 Test Stories

Un’ulteriore implementazione è stata quella di inserire delle storie di test, così da valutare il chatbot realizzato su storie specifiche. Le storie di test sono state realizzate specificando l’input dell’utente e l’intent che si sta realizzando, specificando poi le azioni che ci aspettiamo il bot realizzi.

Sulla base di queste storie di test è stato possibile valutare il bot a seguito dei vari training effettuati, senza dover ripetere manualmente le storie in modalità shell.

Il comando che ha permesso di valutare le storie di test è *rasa test*.

Nell’immagine di figura 8.13 è possibile vedere una delle storie di test che abbiamo implementato.

Risultati

Dopo aver inserito delle storie di test il passo successivo è stato quello di lanciare il comando *rasa test*, così da validare il modello precedentemente allenato su tali storie e valutarne i risultati con appositi grafici e metriche.

Di seguito vengono riportati alcuni dei grafici ottenuti dopo il procedimento di test del modello. Nella figura Fig. 8.14 è riportata la *Intent Confusion Matrix* che riporta il numero, e con quali altri, alcuni intent sono stati classificati erroneamente o meno.

Come si può vedere abbiamo che la quasi totalità dei valori si trovano sulla diagonale della matrice, sono stati quindi classificati correttamente dal modello. Sono presenti alcuni valori con un colore più chiaro rispetto ad altri, in futuro si potrebbe aggiungere ulteriori storie di test così da verificare più volte i vari intent ed avere una *confusion matrix* ancora più affidabile (e colori più scuri per i diversi intent).

Nella figura Fig.8.15 viene riportato un istogramma che permette di visualizzare la confidenza per tutte le varie predizioni mostrando quelle corrette e quelle errate (rispettivamente, in verde e in rosso).

```

23   - story: order path
24     steps:
25       - user: |
26         | Can I order a [pizza]{"entity": "food"}?
27         intent: order_something
28       - action: utter_order
29       - action: utter_time
30       - user: |
31         | [11:00PM]{"entity": "time"}
32         intent: choose_time
33       - action: utter_drink
34       - user: |
35         | add to my order [cola]{"entity": "drink"}
36         intent: add_drink_to_order
37       - action: action_save_order
38

```

Figura 8.13: Esempio di Test Story

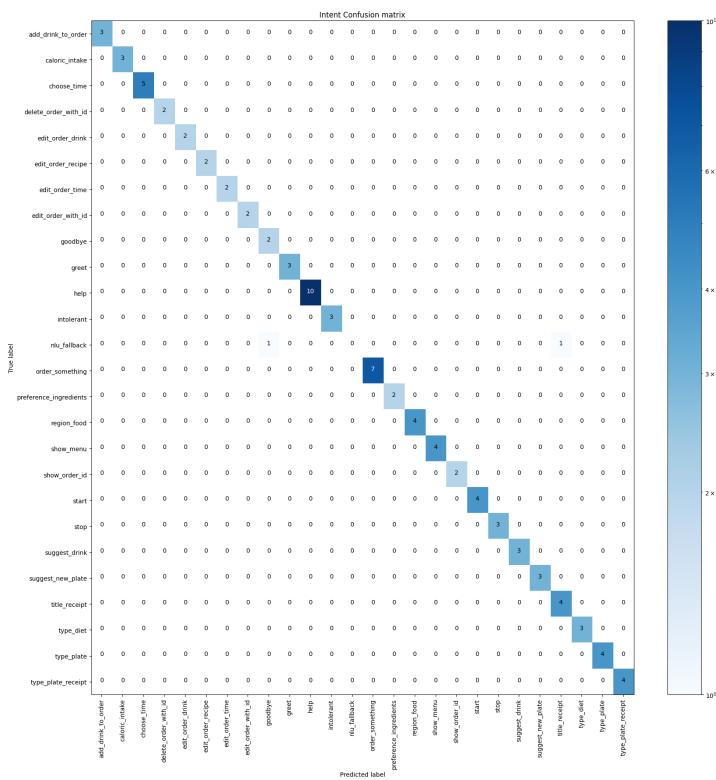


Figura 8.14: Intent Confusion Matrix

Come si può osservare abbiamo la maggior parte delle predizioni corrette nell'area in alto, questo indica che per quelle predizioni si hanno valori elevati di confidenza. Viceversa, abbiamo un numero esiguo di predizioni errate e con un basso valore di confidenza. Nel complessivo, si sono ottenuti dei buoni risultati per il modello testato.

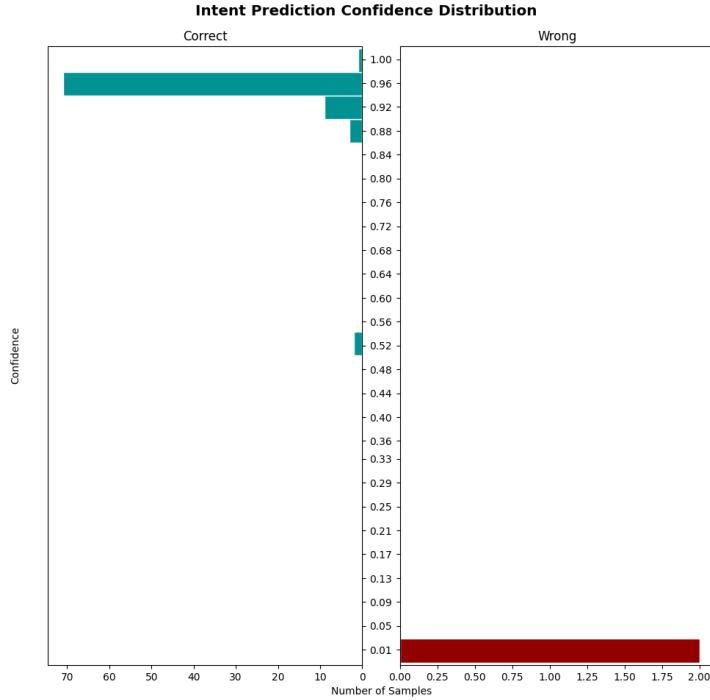


Figura 8.15: Intent Prediction Confidence Distribution

8.3.2 File CSV

Una funzionalità aggiuntiva che si è utilizzata nel progetto è il supporto di file CSV per la lettura e il salvataggio persistente di alcuni dati.

In particolare, si sono utilizzati alcuni file CSV per le varie tipologie di diete. Nelle *actions* poi è stato possibile leggere tali file per poi proporre le diverse tipologie di piatti agli utenti.

Un ulteriore file CSV è stato utilizzato per il salvataggio degli ordini, questo ci ha permesso poi di andare a leggere gli ordini presi per poterli eventualmente modificare o annullare.

Infine, un file CSV è stato utilizzato per scrivere i suggerimenti da parte degli utenti. Qui ci troveremo i piatti consigliati dagli utenti, con associate le tipologie (piatto principale, dessert, ecc.). Si è ipotizzato in questo contesto che il proprietario del ristorante e di questo *chatbot* andrà poi a verificare quante volte uno stesso piatto compare in questo file e se il numero è sufficientemente alto potrebbe decidere di inserirlo nel proprio menù. Nota importante da tenere in considerazione è che al momento i piatti da poter suggerire si trovano in una possibile lista fornita sempre dai ristoratori, e quindi stampata come risposta iniziale dal chatbot quando si fa richiesta di suggerimento per il menù.

Per la lettura/modifica dei file CSV si sono utilizzati i pacchetti Python **pandas** e **csv**.

8.4 Il nostro FoodOrderingBot

8.4.1 Help Command

Per una maggior semplicità di fruizione del chatbot abbiamo inserito la funzionalità di HELP, ovvero è possibile chiedere al bot quali sono le funzionalità che permettono l'interazione con lui e quali sono delle frasi di esempio che il bot riconosce.

```

1 Your input -> help
2 You're welcome. Here is the list of things you can do by us:
3 #####
4 - Visualize the Menu -> (e.g.: Show me the entire menu)
5 #####
6 - Order a dish -> (e.g.: I would like to order a <<pizza>>)
7 #####
8 - Show a specified order -> (e.g.: Show order with ID <<AA12457>>)
9 #####
10 - Edit a specified order -> (e.g.: Edit order with ID <<AA36985>>)
11 #####
12 - Delete a specified order -> (e.g.: Delete order with ID <<AA25479>>)
13 #####
14 - Visualize dishes for a specified category (i.e.: vegetarian, vegan, omnivorous
     ) -> (FORM: I'm <<vegan>>)
15 #####
16 - Visualize dishes of a specified type (i.e.: main dish, second dish, dessert,
     side dish) -> (FORM: I want a <<main dish>>)
17 #####
18 - Visualize acceptable dishes from intolerance (i.e.: gluten, lactose, nuts) ->
     (e.g.: I'm intolerant to <<lactose>>)
19 #####
20 - Visualize typical dishes from a specified Region (i.e.: emilia romagna,
     toscana, marche, ...) -> (e.g.: I want to try something from <<emilia
     romagna>>)
21 #####
22 - Asking a suggestion to you for a plate with chosen ingredient -> (e.g.: I'd
     like to eat something with <<tomato>>)
23 #####
24 - Ask a suggestion of a drink for the dish you would like to take (BEFORE
     ORDERING A DISH!) -> (e.g.: Suggest me a drink for <<Pizza>>)
25 #####
26 - Knowing the Caloric Intake of a SPECIFIED dish (BEFORE ORDERING A DISH!) -> (e
     .g.: I would like to know the caloric intake for <<Pizza Diavola>>)
27 #####
28 - Suggest a new dish for the menu (to insert in the future by the owners) -> (e.
     g.: Can I propose a new dish?)
29 #####
30 - Quit the appli

```

8.4.2 Suggerimento Piatto

- Inserire un ingrediente a proprio piacimento e il bot risponderà con le possibili alternative.

```
1 Your input -> i'd like to eat something with cheese
```

```

1 We have plates adapt you, please wait a second
2 Name: Pallotte cacio e ova
3 Ingredients: [bread raffermo - cheese - eggs - parsley - pe...
4 Name: Fusilli con la mollica
5 Ingredients: [pasta - mollica di bread - acciughe - cheese ...
6 Name: crosta di formaggio romana

```

```

7 Ingredients: [flour - butter - sugar - eggs - salt - cheese...
8 Name: pasta con il ciasculo
9 Ingredients: [pasta - ciasculo - tomato - peper - basilico ...

```

- Specificare se desidera un primo, un secondo, etc. e il bot risponderà con le possibili alternative

```

1 Your input -> I want a dessert

```

```

1 We have plates adapt you, please wait a second
2 Name: parozzo
3 Ingredients: [eggs - sugar - semolino - mandorle - liquor]
4 Name: Chinulidd
5 Ingredients: [flour - eggs - sugar - wine - strutto - casta...
6 Name: pitta mpigliata o nchiusa
7 Ingredients: [flour - eggs - oil di oliva - wine - yeast - ...
8 Name: baba al rum
9 Ingredients: [flour - butter - eggs - latte - sugar - yeas...
10 Name: tagliattele fritte
11 Ingredients: [flour - eggs - rum - limon - sugar - oil di a...
12 Name: Schiacciata con uva fragola
13 Ingredients: [flour - acqua - sugar - oil - yeast - salt - ...
14 Name: tiramisu
15 Ingredients: [eggs - mascarpone - coffee - cacao - savoiar...

```

- Specificare una regione per avere una specialità tipica e il bot risponderà con le possibili alternative

```

1 i want to try something typical from calabria

```

```

1 We have plates adapt you, please wait a second
2 Name: polpette di melanzane
3 Ingredients: [melanzane - eggs - bread - salt - peper - pa...
4 Name: penne provola
5 Ingredients: [penne - tomato - onion - provola silana - wi...
6 Name: lamb del pastore
7 Ingredients: [lamb - cheese - onion - garlic - olive - oil...

```

8.4.3 Intolleranza

L'utente, specificando una particolare intolleranza, può ricevere dal bot consigli sui piatti a lui più adatti.

```

1 Your input -> i'm intolerant to lactoiso
2 We have plates adapt you, please wait a second
3 Name: Pumpkin Curry Soup
4 Ingredients: [Sugar pumpkin - Butternut squash - Olive oil ...
5 Name: gnocchi speck e crema potatoes
6 Ingredients: [speck - potatoes - parsley - salt - gnocchi - ...
7 Name: cestini di brise con salmon e avocado
8 Ingredients: [salmon - brise - avocado]

```

8.4.4 Tipologia di dieta

L'utente, specificando una particolare dieta (ad es. vegana, vegetariana, etc.), può ricevere dal bot consigli sui piatti a lui più adatti.

```

1 Your input -> i'm vegan
2 I will select a plate adapt for you
3 We suggest you: vegan lasagna

```

```

1 Your input -> i'm vegan
2 I will select a plate adapt for you
3 We suggest you: vegan crostino

```

8.4.5 Ordine

Utilizzando il bot, l'utente ha la possibilità di creare un nuovo ordine o modificare un ordine già esistente, ed eliminare quest'ultimo.

- **Realizzare un nuovo ordine**, l'utente può scegliere uno fra i possibili piatti del ristorante da ordinare, specificando l'orario in cui vuole ritirare il suo ordine;

```

1 Your input -> i want to order a pizza
2 OK.
3 Which time would you prefer?
4 Your input -> 19:45
5 let me know what do you want to drink (formula - add to my order <<water>>)
6 Your input -> add to my worder cola
7 Your order is safe and sound! The order ID is AA29741.

```

- **Show del ordine già effettuato** l'utente una volta eseguito il proprio avrà un ID identificativo che gli permetterà di visualizzare il suo ordine

```

1 Your input -> show order with ID AA29741
2 The order with the ID AA29741 booked for 19:45 o'clock contains:
3 pizza
4 cola

```

- **Modificare un ordine già effettuato**, l'utente può modificare un ordine già effettuato, specificando l'ID dell'ordine. In particolare, può decidere di modificare l'orario in cui desidera ritirare l'ordine, la bevanda o il piatto;

```

1 Your input -> edit order with ID AA29741
2 choose and insert the new order FORM where (FORM = edit order time
    <<09:00>> , edit order recipe <<Tuna>> , edit order drink <<Cola>>)
3 Your input -> edit order recipe pasta
4 The order with the ID AA29741 will be changed...
5 The order with the ID AA29741 has been updated with success!
6 Your input -> show order with ID AA29741
7 The order with the ID AA29741 booked for 19:45 o'clock contains:
8 pasta
9 cola

```

```

1 Your input -> edit order with ID AA29741
2 choose and insert the new order FORM where (FORM = edit order time
    <<09:00>> , edit order recipe <<Tuna>> , edit order drink <<Cola>>)
3 Your input -> edit order time 20:45
4 The order with the ID AA29741 is updating...
5 The order with the ID AA29741 has been updated with success!
6 Your input -> show order with ID AA29741
7 The order with the ID AA29741 booked for 20:45 o'clock contains:
8 pasta
9 cola

```

```

1 Your input -> edit order with ID AA29741
2 choose and insert the new order FORM where (FORM = edit order time
    <<09:00>> , edit order recipe <<Tuna>> , edit order drink <<Cola>>)
3 Your input -> edit order drink wine

```

```

4 The order with the ID AA29741 will be changed...
5 The order with the ID AA29741 has been updated with success!
6 Your input -> show order with ID AA29741
7 The order with the ID AA29741 booked for 20:45 o'clock contains:
8 pasta
9 wine

```

- **Disdire un ordine già effettuato**, l'utente può cancellare un ordine già effettuato, specificando l'ID dell'ordine.

```

1 Your input -> delete order with ID AA29741
2 The order with the ID AA29741 has been deleted with success!

```

8.4.6 Suggerire una modifica al menu

Tramite un apposita funzionalità l'utente può effettuare quella che una votazione del piatto che desidera che il ristorante aggiunga al proprio menu.

```

1 Your input -> Can I propose a new dish?
2 let me know what do you want suggest, from this list of elements (FORM: title
   plate suggesting <<pasta ai fichi>>):
3 - pasta ai fichi
4 - pizza ananas
5 - strudel
6 - gnocchetti al cinghiale
7 - gelato fritto
8 - anatra arrosto
9 Your input -> title plate suggesting anatra arrosto
10 Let me know what kind of plates is "Main Dish, Dessert, Second Dish, Side Dish"
    (FORM: type plate suggesting <<Main Dish>>)
11 Your input -> type plate suggesting second dish
12 Title suggest receipt: anatra arrosto - Type Plate Suggest: second dish
13 Thanks for the suggestion.

```

8.4.7 Richiesta suggerimento bevanda

Un'ulteriore funzionalità è stata implementata per richiedere le bevande consigliate per un determinato piatto

```

1 Your input -> suggest me a drink with parozzo
2 Drinks suggested for your choice are:
3 [wine - water - vodka]

```

8.4.8 Informazione apporto calorico

L'ultima funzionalità implementata è quella relativa all'apporto calorico, così che si è in grado di chiedere al bot le kilocalorie di un determinato piatto.

```

1 Your input -> I would like to know the caloric intake for risotto milanese
2 Calories of you recipe:
3 330Kcal

```

8.4.9 Show Entire Menu

Lo "Show Entire Menu" ci permette di visionare il menu del ristorante, nella sua completezza. Per brevità riportiamo solo una parte dell'output.

```
1 Your input -> show entire menu
2 Menu :
3 We have plates adapt you, please wait a second
4 ----- Name: vegan crostino-----
5 Ingredients: [bread - vegetables - tomato - tofu]
6
7 ----- Name: vegan lasagna-----
8 Ingredients: [lentils - pasta - onion - celery - carrots]
9
10 ----- Name: seitan burger-----
11 Ingredients: [seitan burger - hamburger bread - onion - tom...
12
13 ----- Name: spaghetti with avocado-----
14 Ingredients: [spaghetti - avocado - chickpeas - chives]
15
16 ----- Name: patate al forno-----
17 Ingredients: [patate - rosmarino - sale]
18
19 ----- Name: insalata finocchi e arance-----
20 Ingredients: [arance - finocchi - sale - olive]
21
22 ----- Name: Vegetarian Burrito Bowl with Avocado Crema-----
23 Ingredients: [Cauliflower - Bell peppers - Onions - Olive o...
24
25 ----- Name: Pumpkin Curry Soup-----
26 Ingredients: [Sugar pumpkin - Butternut squash - Olive oil ...
27
28 ----- Name: Falafel_burgers-----
29 Ingredients: [Canned chickpeas - Carrot - Red onion - Garli...
```