



UNIVERSITA' POLITECNICA DELLE MARCHE

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

**CORSO DI DATA SCIENCE:
PROGETTO PYTHON**

Caprari David

Cingolani Cristian

Anno Accademico 2021-2022

Sommario

Introduzione.....	3
Clustering e Classificazione	4
Data Visualization	4
Clustering	11
Cluster su BMI e PhysicalHealth.....	11
Cluster di PCA.....	12
Cluster con DBSCAN.....	13
Valutazione con il pacchetto Yellowbricks	14
Classificazione.....	15
Primo approccio.....	16
Secondo approccio.....	22
Serie Temporali	24
Caricamento ed esplorazione dati.....	24
Verifica della stazionarietà	26
ARIMA.....	27
SARIMA	30
Social Network Analysis	32
Caricamento ed esplorazione dati.....	32
Selezione del componente connesso più grande.....	34
Visualizzazioni	35

Introduzione

L'obiettivo di questo progetto è quello di mostrare le capacità elaborative nell'ambiente di analisi dei dati dell'ambiente Python, linguaggio di programmazione e strumenti ad esso associati e scaricabili.

In particolare, il progetto si porrà di utilizzare tale strumento per portare a termine tre diversi tipi di analisi su tre diversi problemi classici della Data Science.

La prima parte di questa relazione utilizzerà un dataset medico per portare esempi di Clustering e Classification. Verrà utilizzato un dataset patologico e si cercherà di verificare se uno strumento software può suggerire o meno la presenza di una patologia specifica a partire dai dati a disposizione.

Nella seconda parte verrà riportata l'analisi di una serie temporale con l'obiettivo di analizzarla, determinarne le caratteristiche, effettuare previsioni. Un dataset riportante l'andamento mensile degli acquisti di un bene ci permetterà di comprendere se il settore relativo è in crescita o se le fluttuazioni sono dovute soltanto alla stagionalità.

Infine, un task di social network analysis esaminerà le caratteristiche di un campione di una rete. Vogliamo apprendere alcune caratteristiche peculiari di un campione di subreddit, nel tentativo di comprendere qualche piccolo dettaglio di comportamento del social network, Reddit appunto.

Clustering e Classificazione

I due task vengono svolti con il supporto di un dataset patologico reperito tramite la piattaforma Kaggle¹.

Il dataset è associato alla malattia cronica diabetica e riporta i dati raccolti dal sondaggio *The Behavioral Risk Factor Surveillance System (BRFSS)*. Quest'ultimo è un sondaggio telefonico che ogni anno raccoglie le risposte fornite da parte di oltre 400'000 americani riguardo a comportamenti di rischio clinico, condizioni legate a malattie croniche e l'uso di servizi preventivi. In particolare, il dataset riporta i dati raccolti nell'anno 2015. Poiché viene utilizzato un sample delle dimensioni dei dati raccolti nel sondaggio, il dataset in nostro possesso è composto da 21 feature, sottosezione delle 330 originali.

Per quanto riguarda lo svolgimento dell'analisi che verrà riportata, per i task di clustering e classificazione vi sono delle attività che potremmo definire ridondanti ed in particolare queste riguardano principalmente la pulizia del dataset ed un primo processo di Data Gathering e Visualization necessario all'analyst per comprendere alcune proprietà del dataset e sue proprie caratteristiche. Dovendo, nella nostra opinione, queste attività essere ripetute per entrambi i task, abbiamo deciso di riportarle solo inizialmente, predisponendo la trattazione in misura il più lineare possibile.

Data Visualization

Dopo l'aver importato trivialmente il dataset all'interno di un costrutto della libreria Pandas, ne riportiamo una breve descrizione, per permettere di comprendere misure e dimensioni dei dati riportati.

- **Diabetes_012:** 0 = no diabetes; 1 = prediabetes; 2 = diabetes
- **HighBP:** 0 = no high Blood Pressure; 1 = high Blood Pressure
- **HighChol:** 0 = no high cholesterol; 1 = high cholesterol
- **CholCheck:** 0 = no cholesterol check in 5 years; 1 = yes cholesterol check in 5 years
- **BMI:** *Body Mass Index*
- **Smoker:** "Have you smoked at least 100 cigarettes in your entire life?" [Note: 5 packs = 100 cigarettes] 0 = no 1 = yes
- **Stroke:** 0 = no; 1 = yes
- **HeartDiseaseorAttack:** *coronary heart disease (CHD) or myocardial infarction (MI)* 0 = no 1 = yes
- **PhysActivity:** *physical activity in past 30 days - not including job* 0 = no 1 = yes
- **Fruits:** *Consume Fruit 1 or more times per day* 0 = no; 1 = yes
- **Veggies:** *Consume Vegetables 1 or more times per day* 0 = no; 1 = yes
- **HvyAlcoholConsumer:** *Heavy drinkers (adult men having more than 14 drinks per week and adult women having more than 7 drinks per week)* 0 = no; 1 = yes
- **AnyHealthcare:** *Have any kind of health care cover, including health insurance, prepaid plans such as HMO, etc.* 0 = no 1; = yes

Fig. 1: Descrizione dei campi del dataset.

¹ Dataset disponibile a [questo link](#).

- **NoDocbcCost:** "Was there a time in the past 12 months when you needed to see a doctor but could not because of cost?" 0 = no; 1 = yes
- **GenHlth:** "Would you say that in general your health is": scale 1-5 1 = excellent; 2 = very good; 3 = good; 4 = fair; 5 = poor
- **MentHlth:** "Now thinking about your mental health, which includes stress, depression, and problems with emotions, for how many days during the past 30 days was your mental health not good?" scale of values 0-30
- **PhysHlt:** "Now thinking about your physical health, which includes physical illness and injury, for how many days during the past 30 days was your physical health not good?" scale of values 0-30
- **DiffWalk:** "Do you have serious difficulty walking or climbing stairs?" 0 = no 1 = yes
- **Sex:** 0 = female; 1 = male
- **Age:** 13-level HighBP category 1 = 18-24; ...; 9 = 60-64; 13 = 80 or older
- **Education:** Education level scale 1-6 1 = Never attended school or only kindergarten; 2 = Grades 1 through 8; ...;
- **Income:** Income scale scale 1-8 1 = less than 10,000; 5 = less than 10,000; 5 = less than 35,000; 8 = \$75,000 or more

È facile notare che la maggior parte dei campi presenta dati binari o dati numerici discreti in un intervallo molto spesso compreso nelle dieci unità. Questa condizione, sebbene aiuti fortemente la raccolta dati, potrebbe incidere negativamente sulla capacità degli algoritmi di Machine Learning che verranno utilizzati di effettuare una corretta inferenza. Ci si aspetta, soprattutto, che gli algoritmi di clustering non riescano a riempire uno spazio bidimensionale omogeneo e, in contrapposizione, che i dati rappresentati si trovino stringati lungo le dimensioni ad intervallo più breve.

Fortunatamente, essendo il dataset utilizzato già una scrematura di un rapporto più imponente, i dati non contengono valori null. Attraverso la funzione `isNull()` fornita da Pandas per i suoi DataFrame, è possibile determinare la somma degli elementi assenti nelle colonne.

Utilizzando le suite `seaborn` e `matplotlib` è possibile ottenere dei banali bar-chart che riportino graficamente la distribuzione dei dati sulla base delle loro possibili misure.

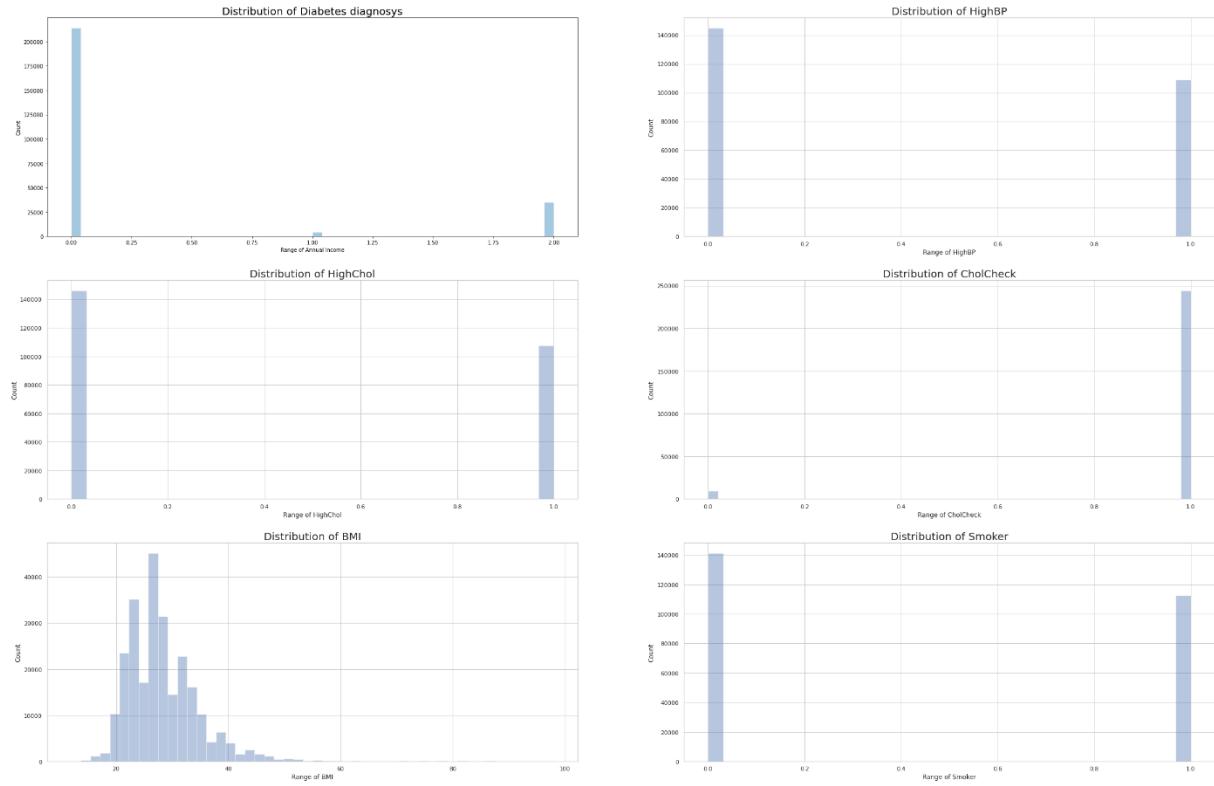


Fig. 2: Primo gruppo di distribuzione delle misure.

Dall'immagine è evidente come la classi dei dati relative alla diagnosi o meno di diabete, compresa la condizione di prediabete, siano fortemente sbilanciate, a rispecchiare quello che ci si aspetterebbe dalla popolazione americana. Risulta chiaro anche a colpo d'occhio come gli unici dati passibili di una distribuzione non binaria siano quelli relativi all'indice di massa corporea.

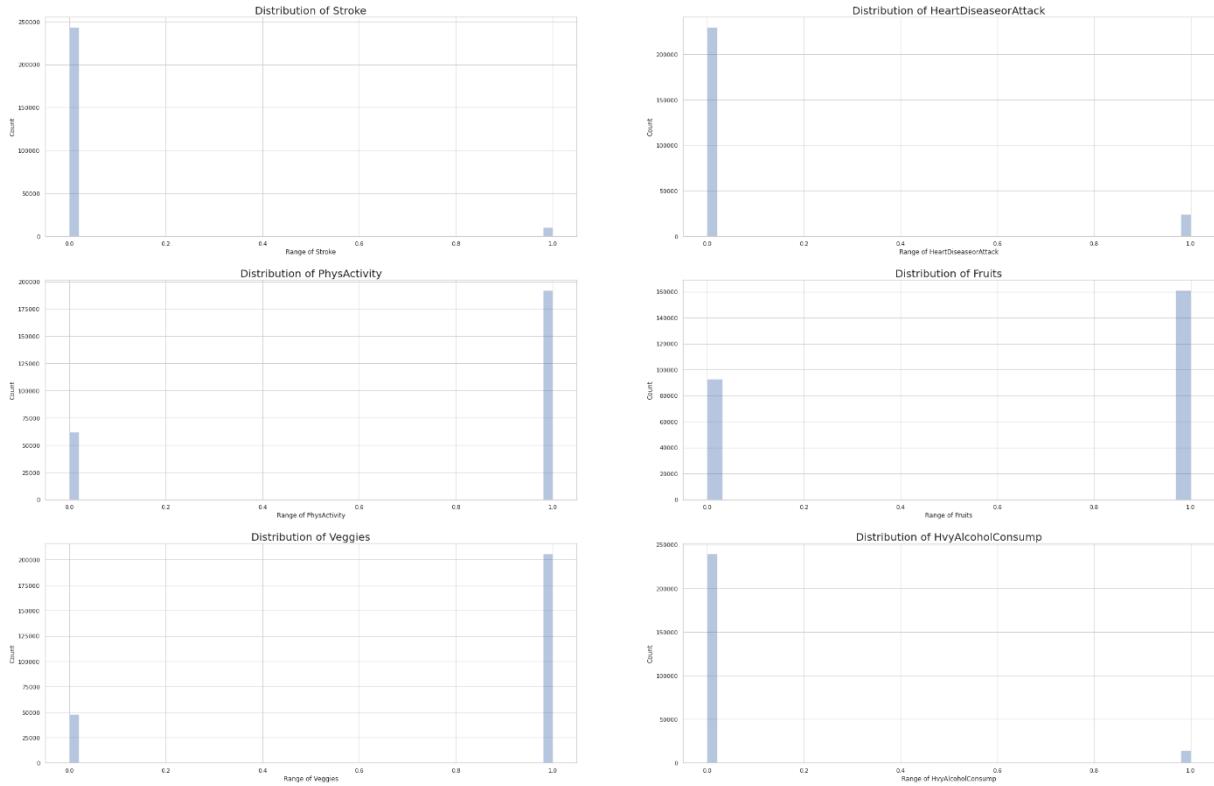


Fig. 3: Secondo gruppo di distribuzione delle misure.

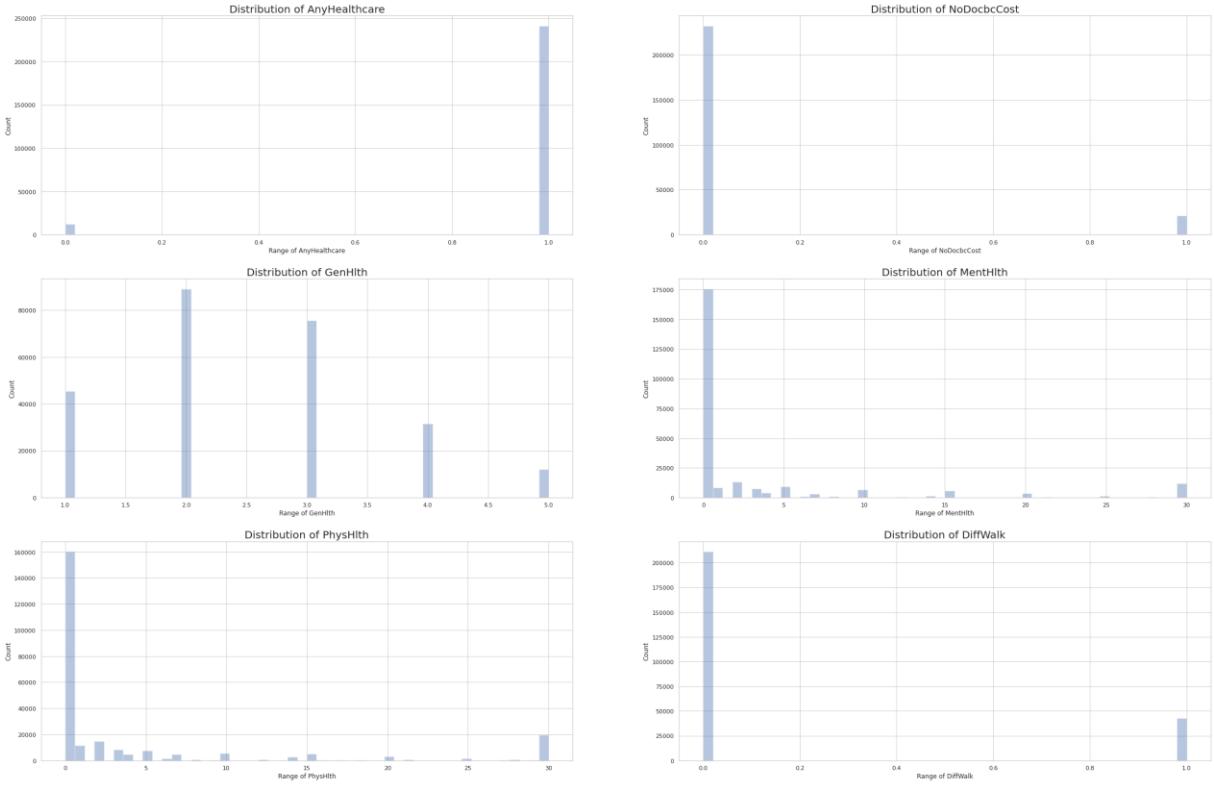


Fig. 4: Terzo gruppo di distribuzione delle misure.

Mentre in *Fig. 3* la distribuzione delle misure è ancora fortemente binaria, già in *Fig. 4* si notano alcune misure leggermente distribuite. In particolare, quelle relative alla salute fisica e mentale, soprattutto dato il metodo di misurazione discreto e fortemente sbilanciato all'assenza di problematiche di questo tipo.

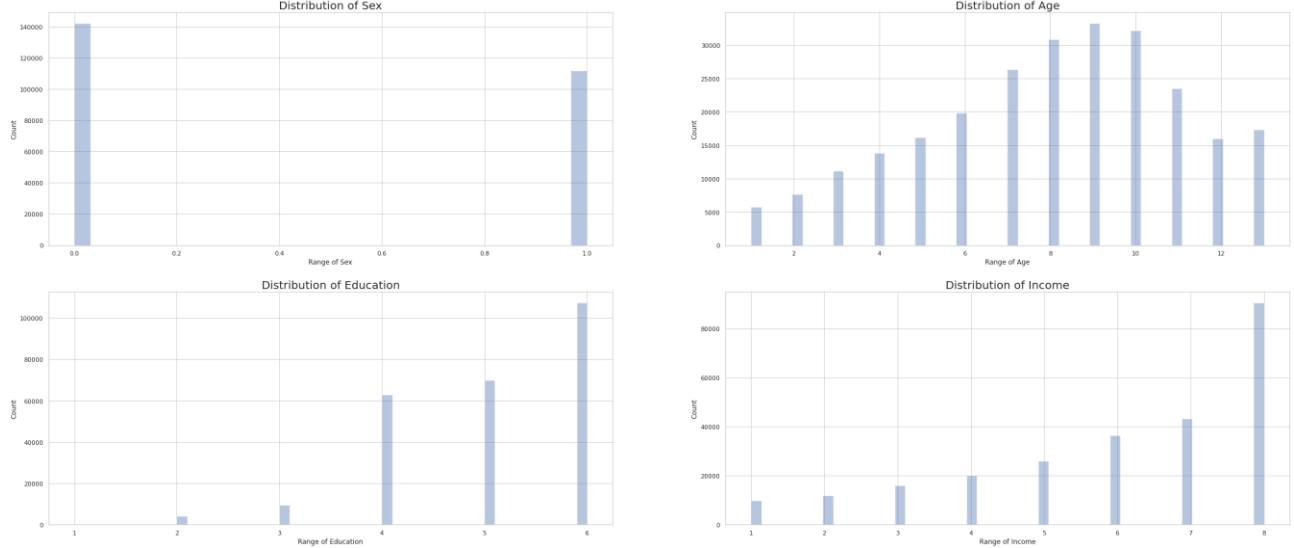


Fig. 5: Ultimo gruppo di distribuzione delle misure.

Fig. 5 riporta i campi meglio distribuiti riguardanti i range di età, educazione e stipendio annuale.

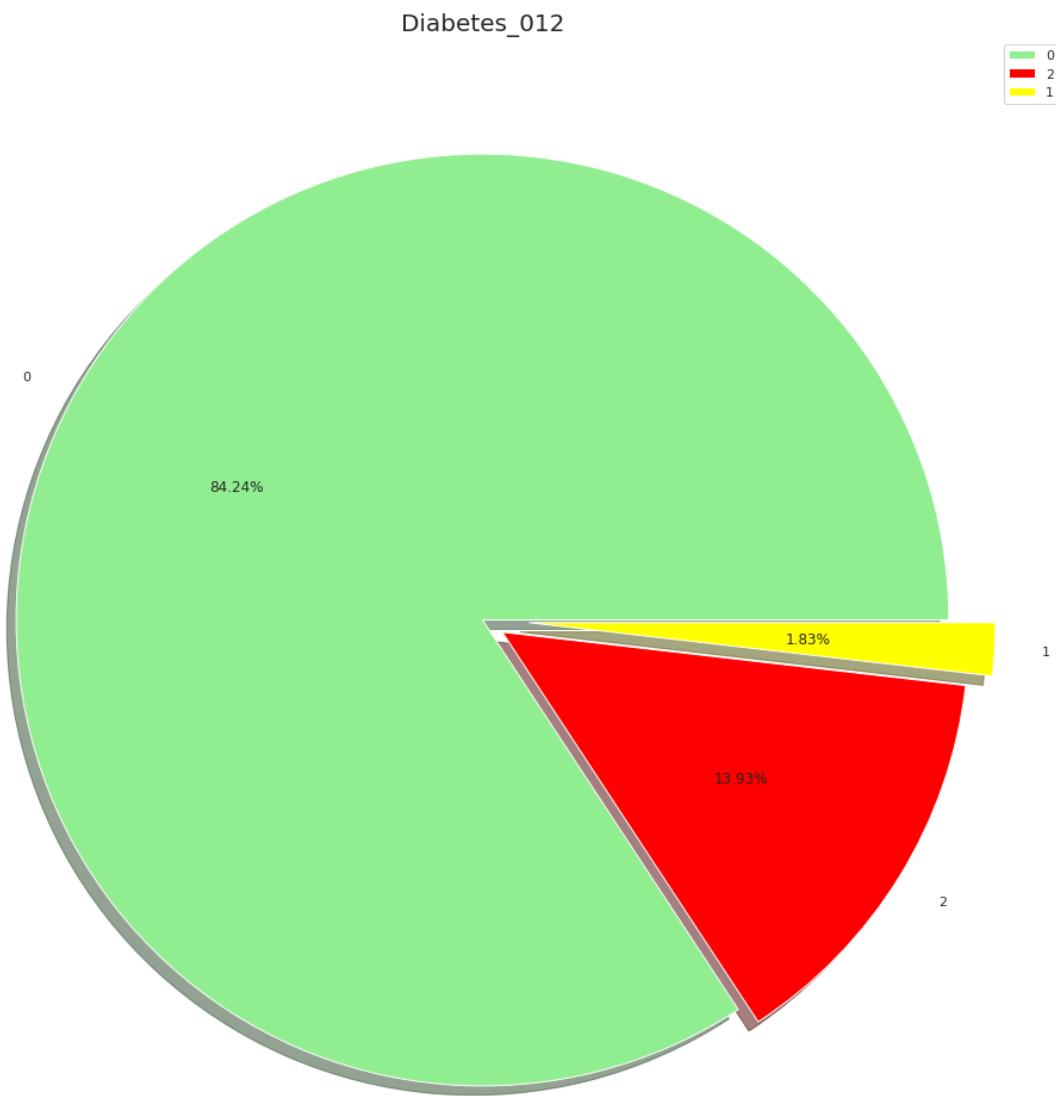


Fig. 6: Riporta il forte sbilanciamento tra le classi di diabete, in cui solo l'1.83% del totale è in condizione di prediabete, il 13.93% ha diagnosi di diabete ed l'84.24% i restanti.

A seguito della prima descrizione di dati si è deciso di selezionare un subset della collezione dei nostri dati, ed estrarre alcune informazioni interessanti.

In particolare, si è scelto di considerare soltanto le persone con diagnosi definitiva di diabete. Da questi sono state estratte diverse informazioni. In particolare, c'è un leggero sbilanciamento di diabete nella popolazione femminile più che in quella maschile.

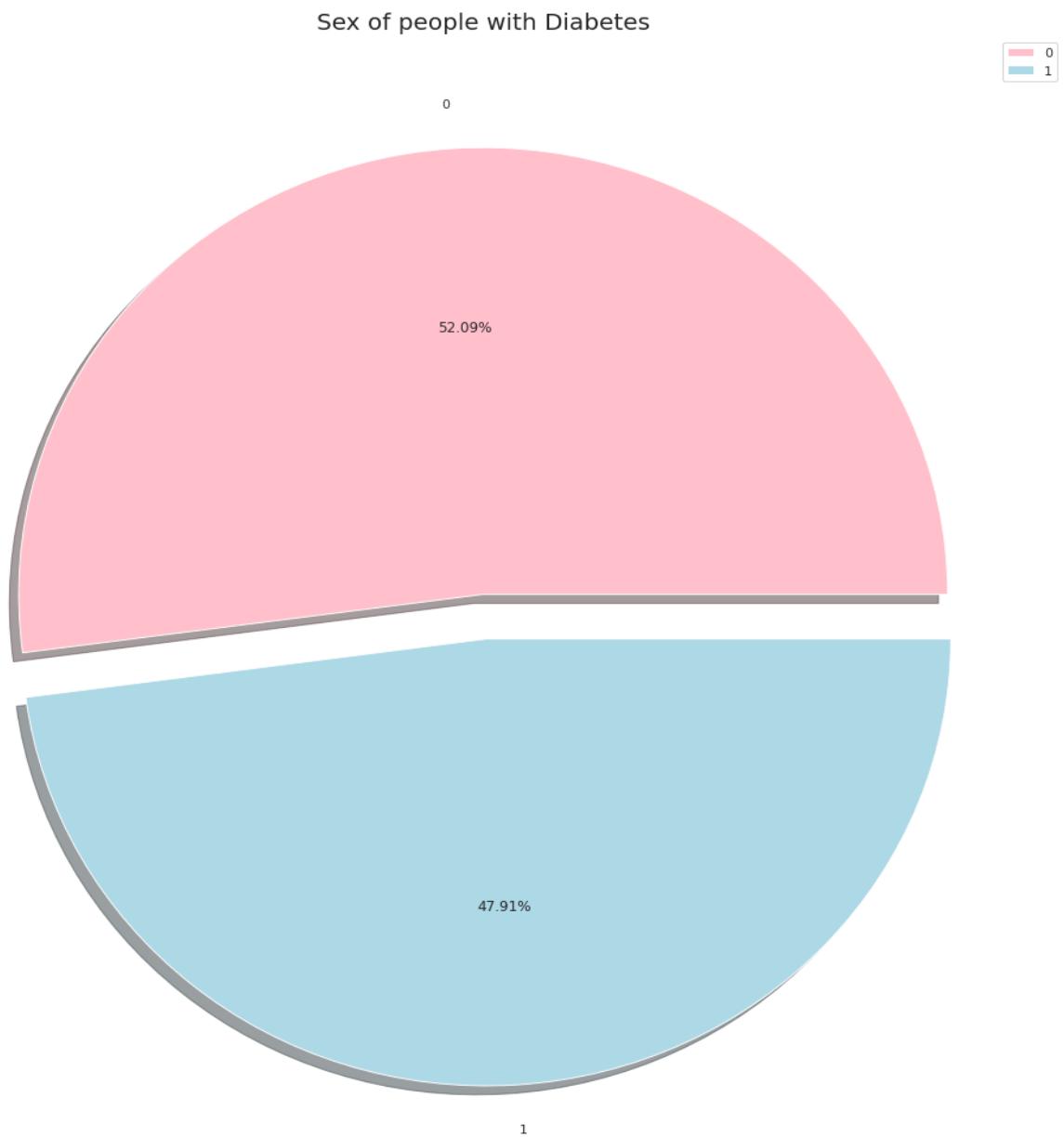
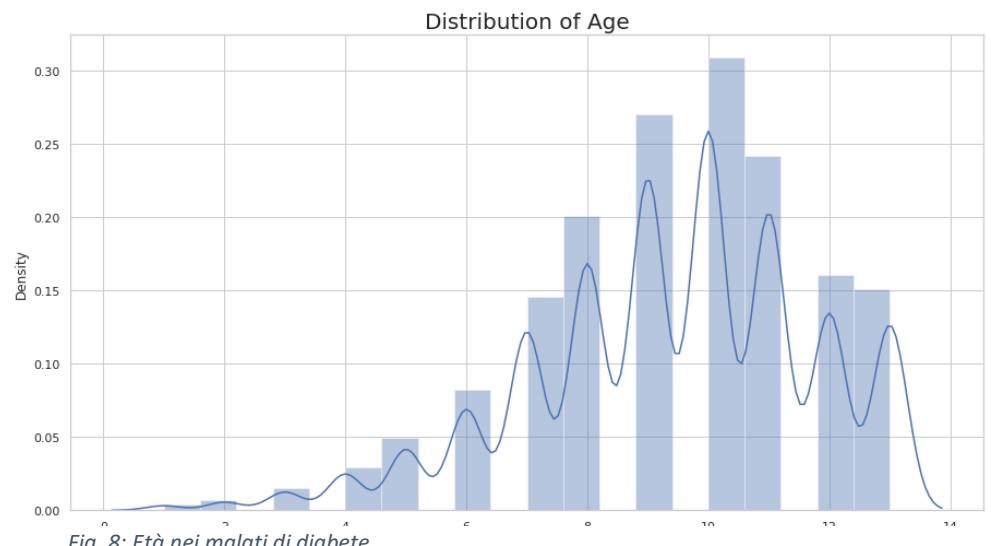


Fig. 7: Sesso nei malati di diabete.

Altra caratteristica notevole è l'analisi per età con relativa distribuzione. Si ricorda, a questo punto, che l'età considerata è divisa in range che non corrispondono all'età reale: i dati sono raggruppati in una scala a 14 valori.

La stessa semplice estrazione è stata fatta per stipendio dei malati di diabete, anche qui, mostrando la curva di distribuzione dei dati.

Nonostante il grafico presenti



una tendenza di crescita della popolazione diabetica alla crescita della sua rendita economica, il dato potrebbe essere influenzato dal fatto che lo stesso comportamento si verifica anche nella popolazione totale.

Considerando i malati di diabete come un campione della popolazione totale, si può assumere lo stesso comportamento, il che suggerirebbe che l'aumento dei diabetici alla crescita del reddito potrebbe non essere una ottima conclusione da trarre.

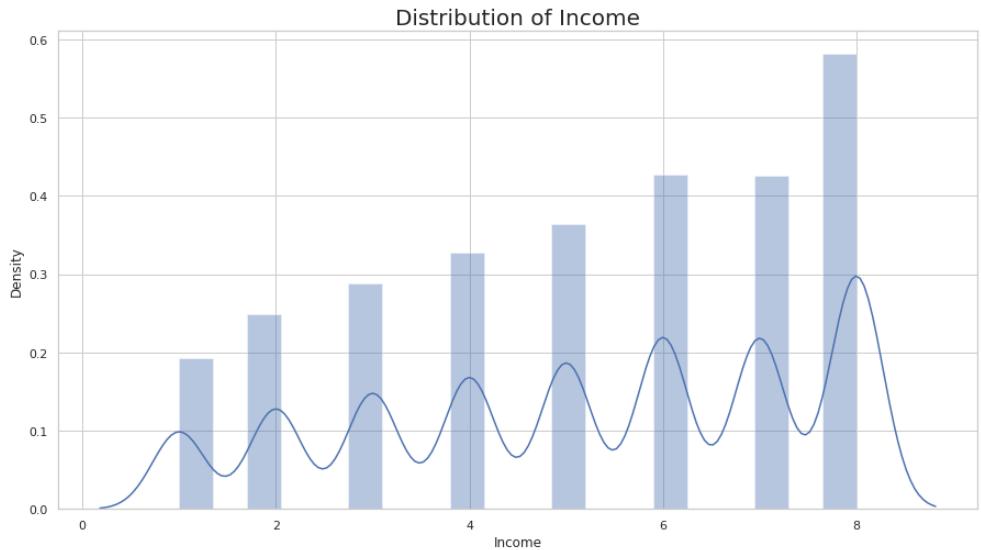


Fig. 9: Stipendio nei malati di diabete.

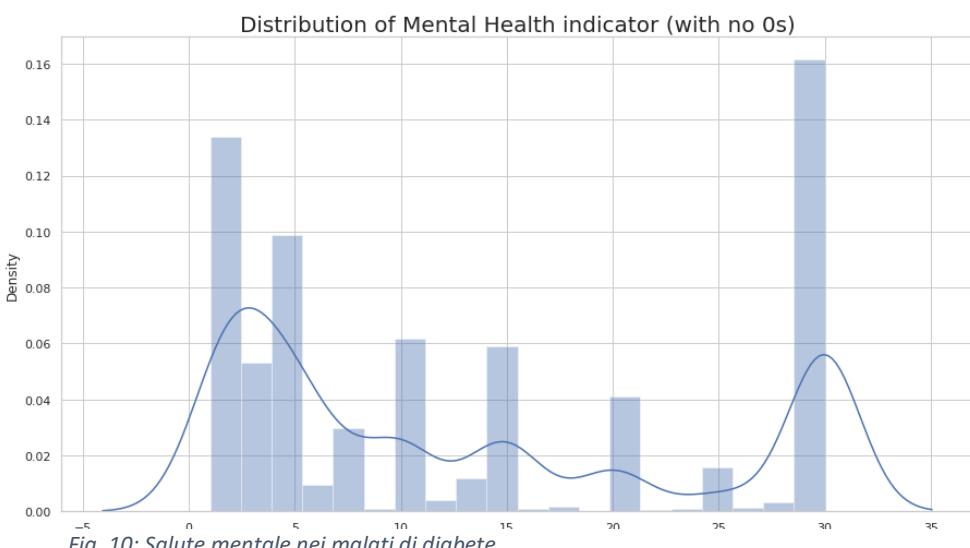


Fig. 10: Salute mentale nei malati di diabete.

L'ultima estrazione da riportare riguarda la distribuzione dei dati di salute mentale. La distribuzione ottenuta mostra che si osservano due picchi corrispondenti all'1 e al 30: i dati permettono di inferire il fatto che il diabete è presente quasi nella stessa proporzione sia per chi soffre raramente di problemi di salute mentale, sia per chi ne soffre ad altissima frequenza. Al fine di rappresentare i dati correttamente scalati, è stato rimosso l'accumulatore per il valore 0, in quanto, parallelamente alla popolazione totale, il numero di elementi del dataset senza salute mentale, essendo la quasi totalità, avrebbe distorto la scala del bar-chart, vanificandone le proprietà grafiche.

Clustering

Cluster su BMI e PhysicalHealth

Al fine di ottenere cluster rilevanti, si è scelto di effettuare ulteriori correzioni e pulizia del dataset.

La prima azione corrisponde al ridurre il numero di elementi con diagnosi di diabete e senza diagnosi, al fine di ridurli allo stesso numero dei campioni in prediabete. Otteniamo quindi, selezionando un campione delle classi più grandi, un dataset finale da dare in pasto al cluster di 4631 elementi.

Come già annunciato, però, gran parte delle dimensioni di quest'ultima selezione sono prettamente dati categorici o con minima escursione su di un range discreto. Al fine di ottenere una migliore rappresentazione grafica dei cluster, si è scelto di selezionare come valori contributori a due dimensioni i campi di indice di massa corporea e di attività fisica. Utilizzando l'algoritmo **KMeans**, ricercando 3 cluster suggeriti dall'elbow method per il nostro campione, otteniamo il seguente risultato:

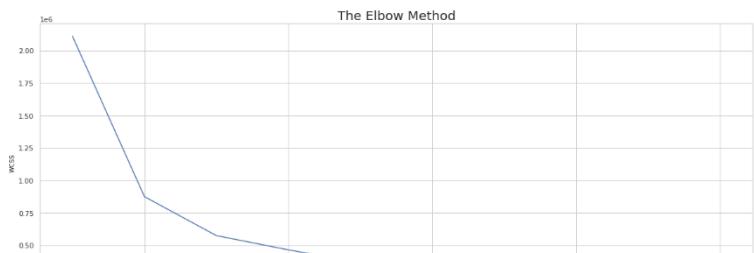


Fig. 11: Elbow Method dei cluster individuati su indice di massa corporea e attività fisica.

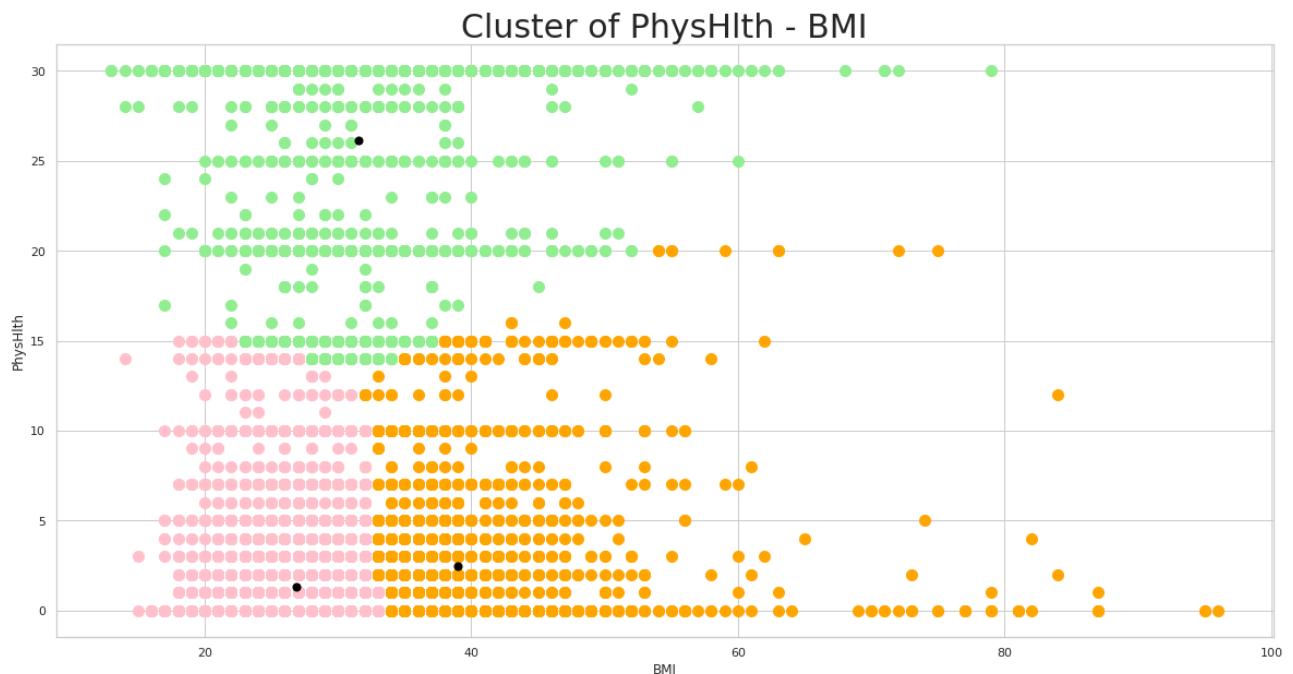


Fig. 12: Cluster dei campi di indice di massa corporea e attività fisica con centroidi in nero.

I tre cluster risultano leggermente stringati sui 30 valori disponibili per i dati di attività fisica, ma individuano un'idea del senso comune: chi ha bassa massa corporea e si allena poco potrebbe corrispondere ad un certo numero di persone sufficientemente magre e possibilmente in salute, chi ha massa corporea alta e si allena poco potrebbe corrispondere a persone sovrappeso con possibilità di problemi di salute, tra cui il diabete, infine, chi si allena molto e ha alta massa corporea potrebbe corrispondere a pesanti atleti, molto spesso in salute.

A questo punto, un'interessante analisi che vogliamo riportare è relativa all'accuratezza dei nostri cluster considerati come se

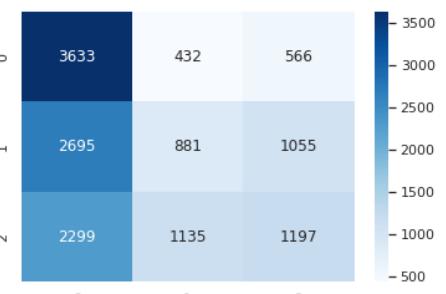


Fig. 13: Matrice di confusione del nostro semplice classificatore basato sui cluster ottenuti.

fossero classificatori, per testare la possibile sovrapposizione presente tra i nostri tre cluster e le tre classi di partenza, diabete, non diabete e prediabete.

Il dato ottenuto è di un'accuratezza del 41%, decisamente maggiore al 33% che si otterrebbe con un classificatore completamente randomico. Quindi, nonostante l'accuratezza sia discretamente bassa, possiamo supporre che l'analisi fatta abbia un minimo di influenza nella popolazione diabetica.

Cluster di PCA

Un'analisi parallela a quella svolta è stata quella ottenuta ricercando i cluster con la totalità delle dimensioni a nostra disposizione. Abbiamo quindi scelto di riprendere il nostro campione con 4631 istanze e mantenerne tutte le 21 colonne. A questo punto abbiamo scelto, per ottenere uno spazio di visualizzazione rappresentabile, di applicare l'analisi spettrale **Principal Component Analysis** (PCA) e riportare i dati in solo due dimensioni significative. Dopo aver replicato l'elbow method con risultati simili sempre indicanti i 3 cluster, otteniamo i seguenti cluster di PCA:

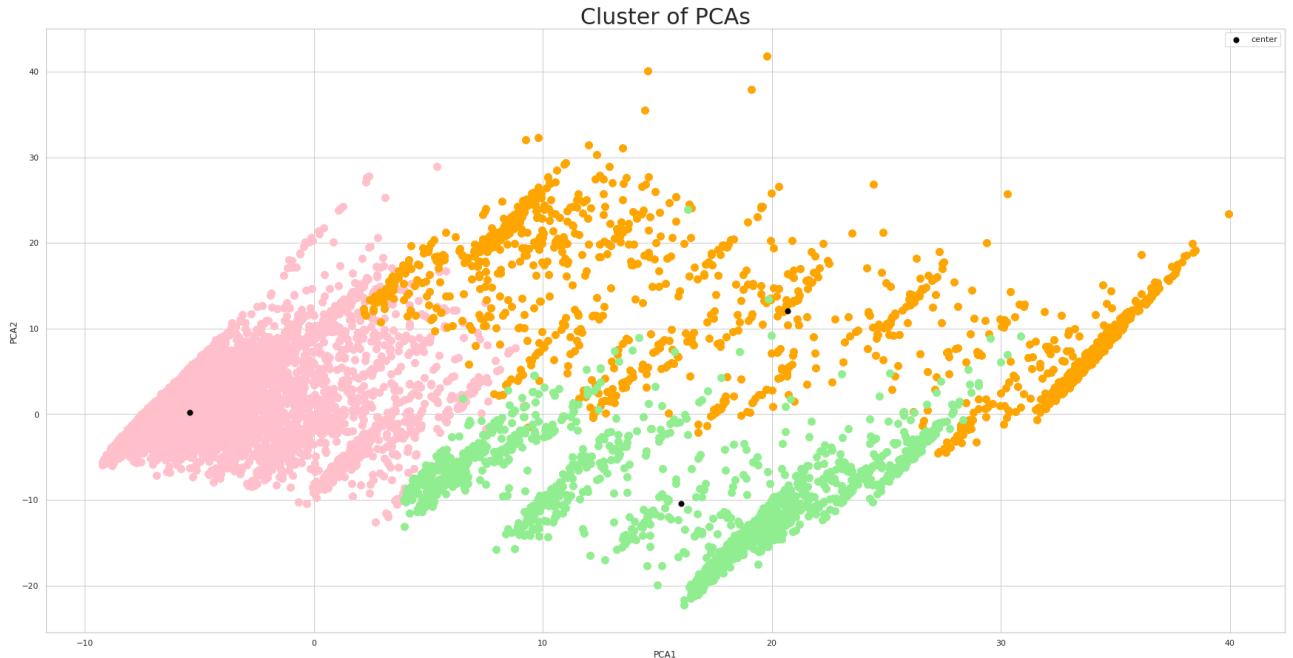


Fig. 14: Cluster dei campi di PCA con centroidi in nero.

I cluster ottenuti, per merito dell'analisi spettrale applicata, sono di più difficile comprensione. Nonostante tutto, replicando l'analisi dell'accuratezza otteniamo una percentuale di 39%, anche questa maggiore della 33% randomica.

Si tiene comunque conto del fatto che in questo tipo di analisi non si sta facendo una vera e propria classificazione e che quindi cluster e classi così ottenute non hanno nessun motivo di corrispondere.

Rimane comunque interessante notare quanto ci sia una minima correlazione tra questi dati, sebbene non basti a giustificare ed a ottenere ottimi confini di classificazione.

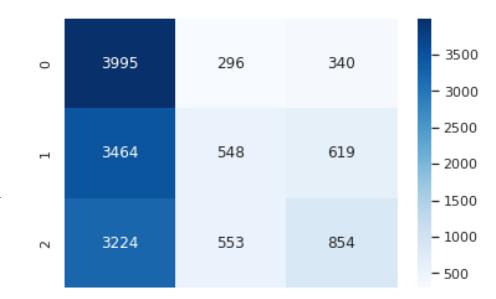


Fig. 15: Matrice di confusione del classificatore ottenuto con l'utilizzo dei cluster di PCA.

Cluster con DBSCAN

Si è scelto di variare sull'approccio del KMeans sostituendolo con il **DBSCAN**, applicandolo alle variabili trasformate con la PCA. Utilizzando la funzione di Nearest Neighbors per la scelta dell'epsilon, inteso come la distanza tra gli elementi, abbiamo ottenuto un valore consigliato di 4. Con questo valore abbiamo poi generato i cluster con DBSCAN.

I cluster ottenuti, in realtà, in questo caso non riportano bene i dati di dominio in quanto l'algoritmo da idea di necessitare di una maggiore ottimizzazione.

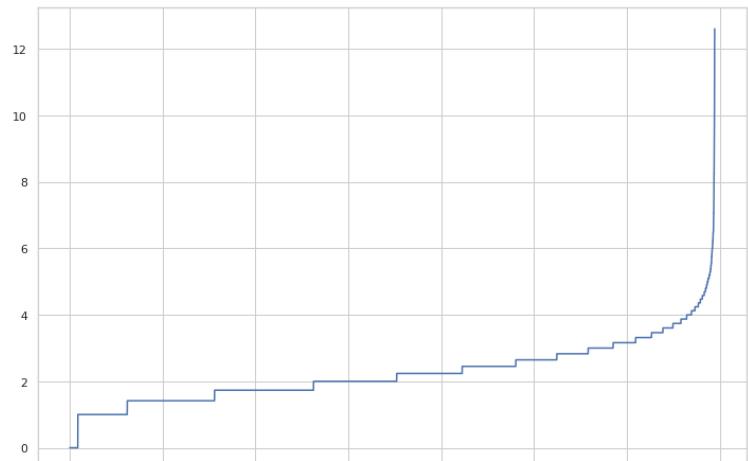


Fig. 16: Nearest Neighbors per l'ottenimento dell'epsilon dai dati di PCA.

Cluster of PCAs

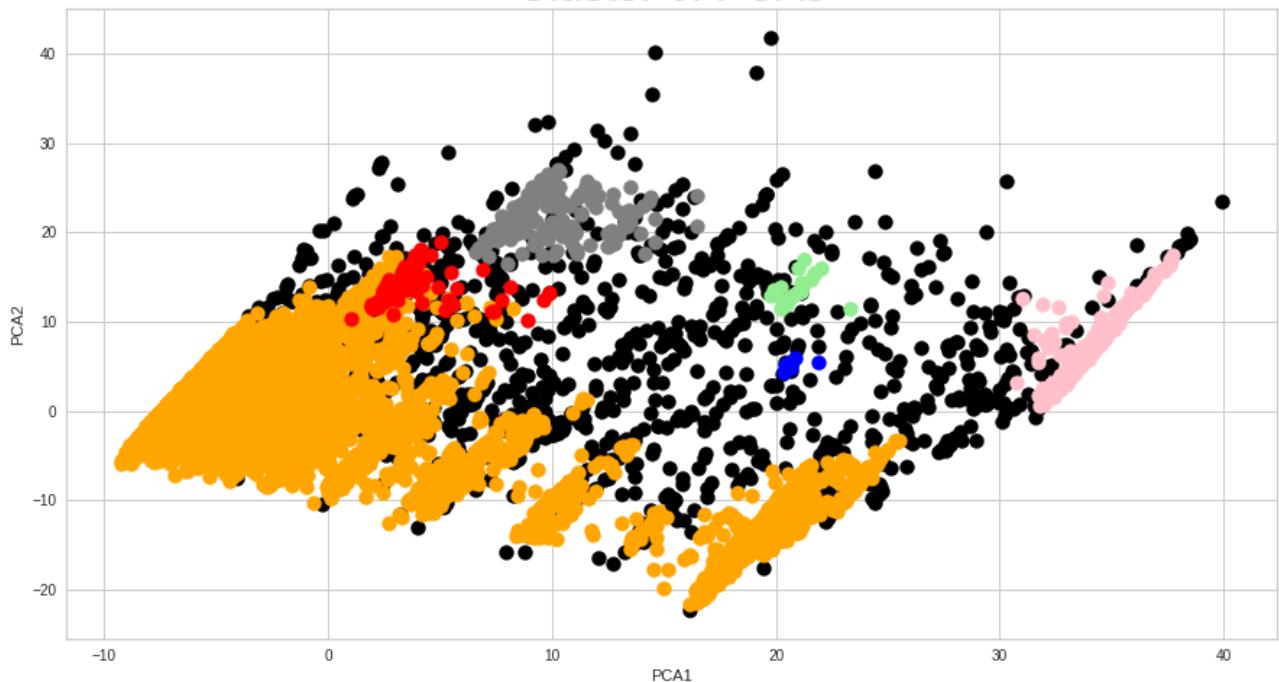


Fig. 17: Cluster di PCA generato dall'utilizzo dell'algoritmo di DBSCAN.

Valutazione con il pacchetto Yellowbricks

Nel caso dei cluster ottenuti con l'algoritmo di KMeans a partire dalla PCA, è stata anche effettuata una valutazione dei cluster stessi con l'utilizzo della libreria Yellowbricks. In particolare, dall'analisi della silhouette dei cluster ottenuti è possibile confermare che si ottengono risultati discreti, questi dati da un discreto numero di istanze positive contenute nel marking oltre la linea rossa tratteggiata.

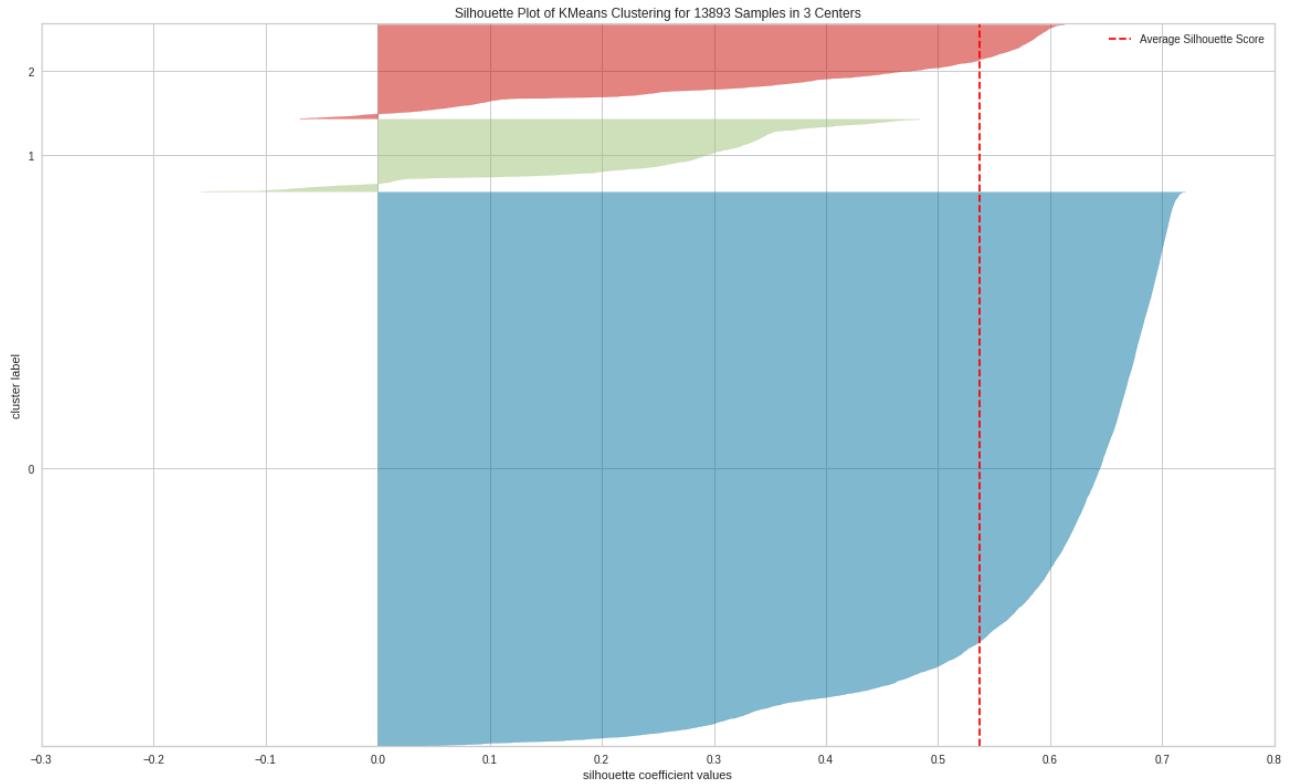


Fig. 18: Valutazione della silhouette dei cluster ottenuta con lo strumento Yellowbricks.

Classificazione

Per ottenere un corretto processo di classificazione, va di nuovo effettuato un preprocessing. Questa volta si è scelto di considerare una semplice classificazione binaria, escludendo l'utilizzo della classe prediabete, per semplificare il processo di classificazione e aumentare i dati di training a disposizione, non dovendo effettuare un resize del dataset su di una classe così tanto più piccola. Il dataset in questa iterazione sarà quindi composto da 70692 istanze per un numero di 22 classi.

Per avere una migliore visualizzazione del comportamento delle classi si è scelto quindi di generare una heatmap della crosscorrelazione tra le colonne.

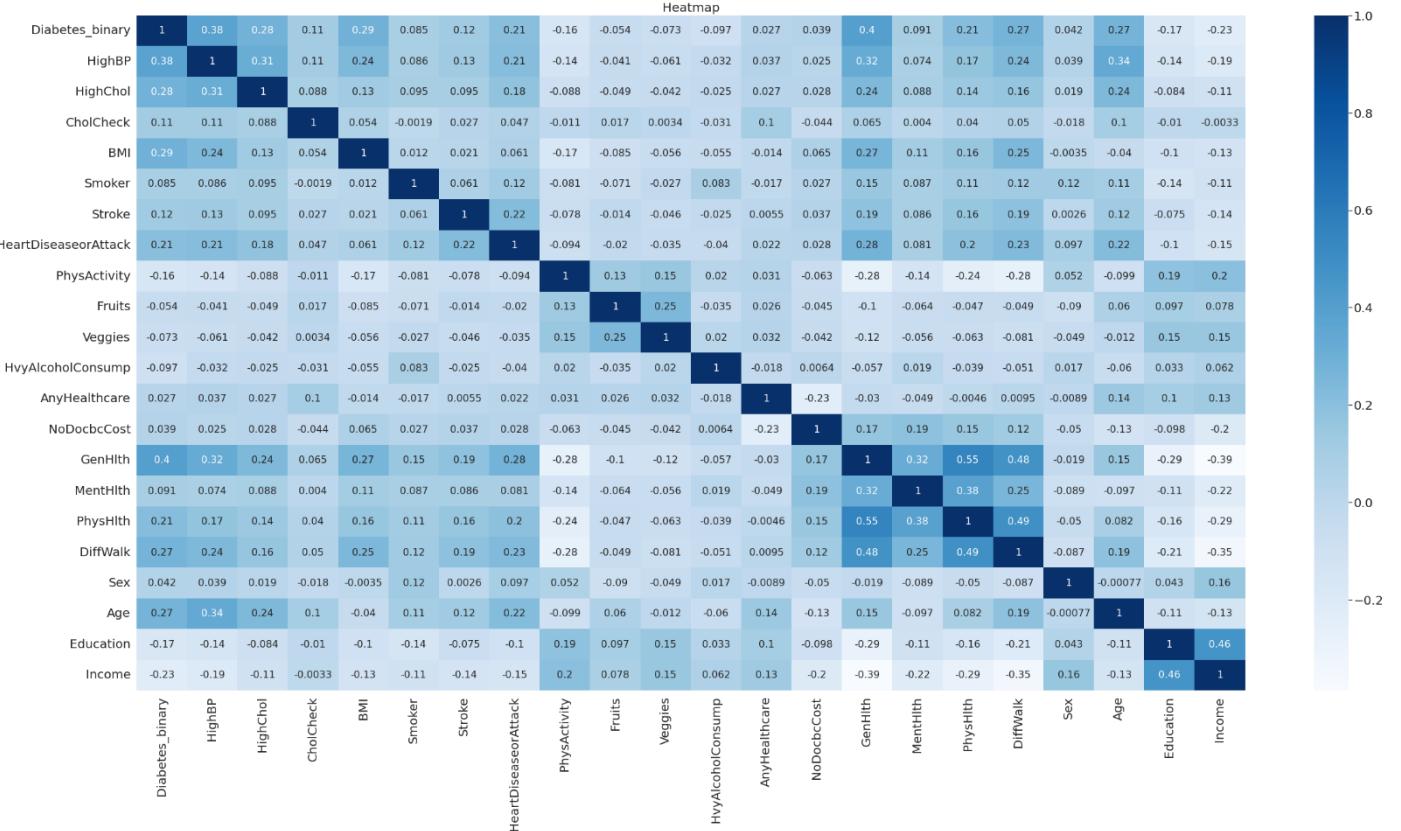


Fig. 19: Heatmap della correlazione tra le dimensioni del dataset.

Si nota discretamente che vi è una maggiore, anche se sufficientemente bassa, correlazione tra i campi relativi l'attività fisica, la salute generale e la salute mentale. Si tende inoltre a specificare quanto i dati in questo caso siano normalizzati. È quindi stato scelto di implementare due diversi task. Il primo, quello che verrà raccontato con maggior dettaglio, consiste nell'utilizzare tutte le colonne del dataframe riportato, il secondo invece, consiste nell'utilizzo di un sottoinsieme delle colonne con correlazione alla classe binaria maggiore della cifra significativa.

Primo approccio

A questo punto, per iniziare il task di classificazione verrà inizialmente separato il dataset di training da quello di testing a partire dalla totalità dei dati ottenuti, ricercando una proporzione di 80%-20% train-test.

Si avranno quindi 56553 istanze di training, dove le label 0-1 sono bilanciate a circa il 50%, e 14139 istanze di testing con stessa frequenza di classi “sano”(0) e “diabete”(1).

Vengono quindi addestrati quattro classificatori:

1. Logistic Regression
2. Decision Tree Classifier
3. SVC
4. Random Forest Classifier

Tutti a ottenuti dalla libreria sklearn.

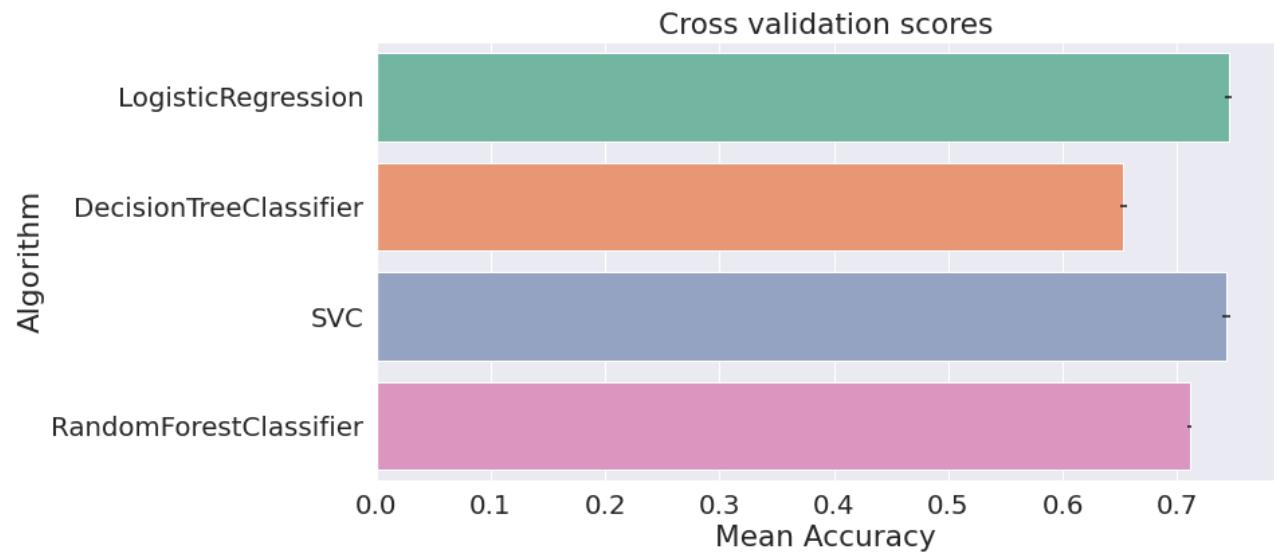


Fig. 20: Output del processo di training e testing dei classificatori.

I dati ottenuti riportano performance di classificazione discrete, in particolare l'accuracy del classificatore che utilizza regressione logistica e dell'SVC si egualgiano al 74%. Di seguito, le matrici di confusione riportano come il classificatore SVC sia il migliore, nel nostro caso, a limitare i False Negative, parametro fondamentale nel caso di studio di un dataset medico in cui è preferibile non diagnosticare nessuna malattia

quando purtroppo questa è presente.

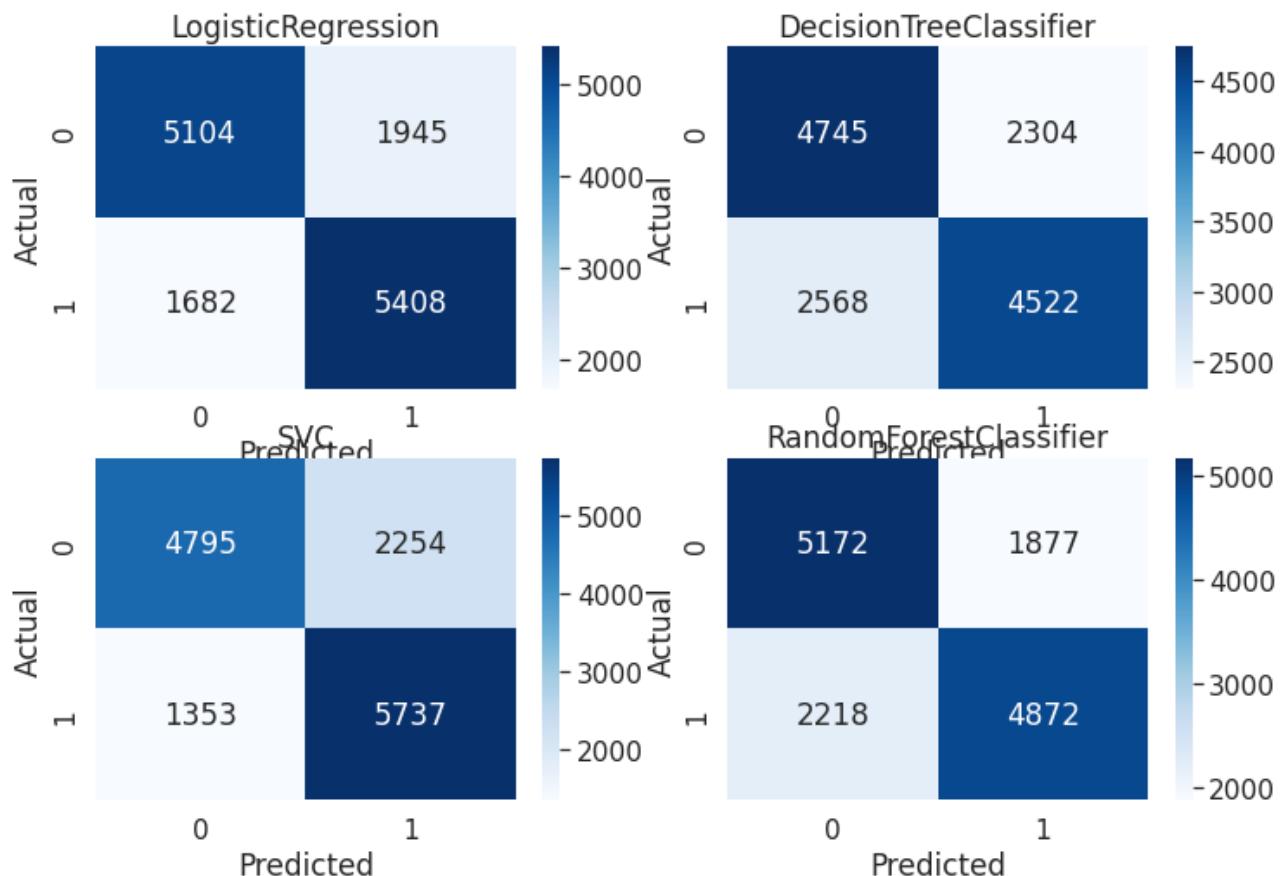


Fig. 21: Matrici di confusione della fase di testing dei classificatori addestrati.

A confermare che i classificatori così ottenuti performano meglio del caso è anche la curva ROC che mette a confronto i classificatori rispetto al loro True Positive rate e al loro False Positive Rate.

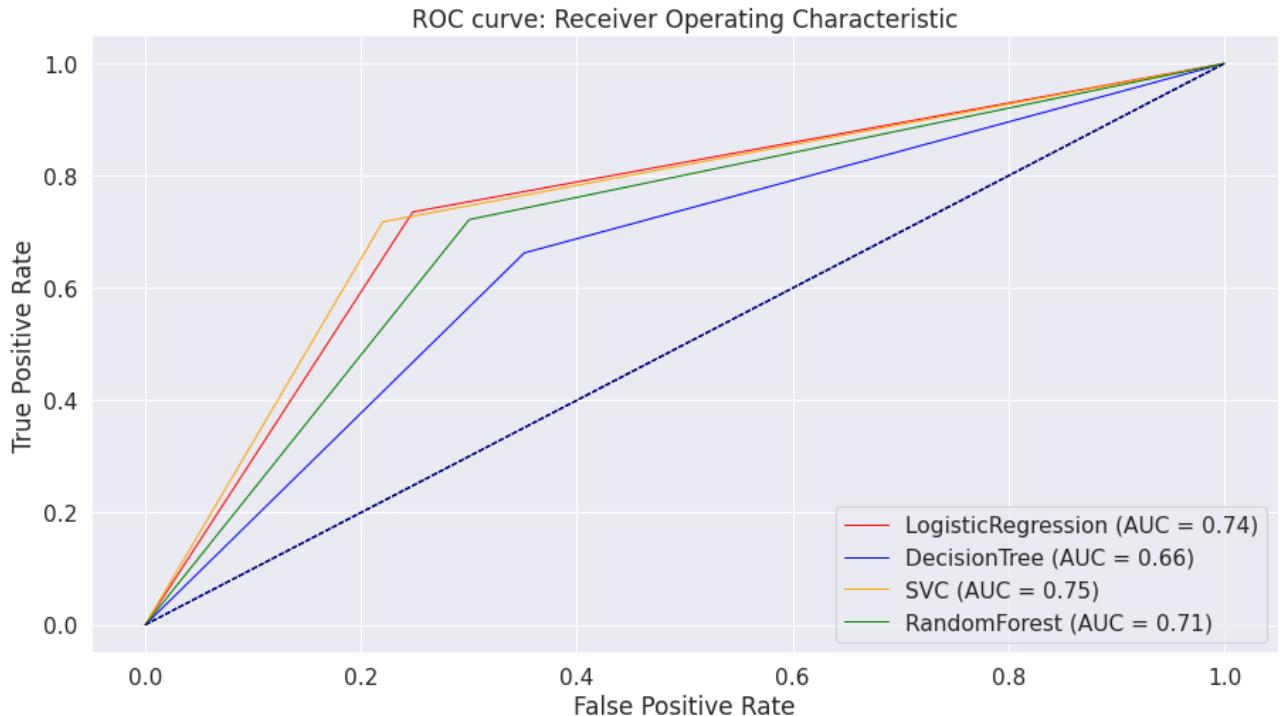
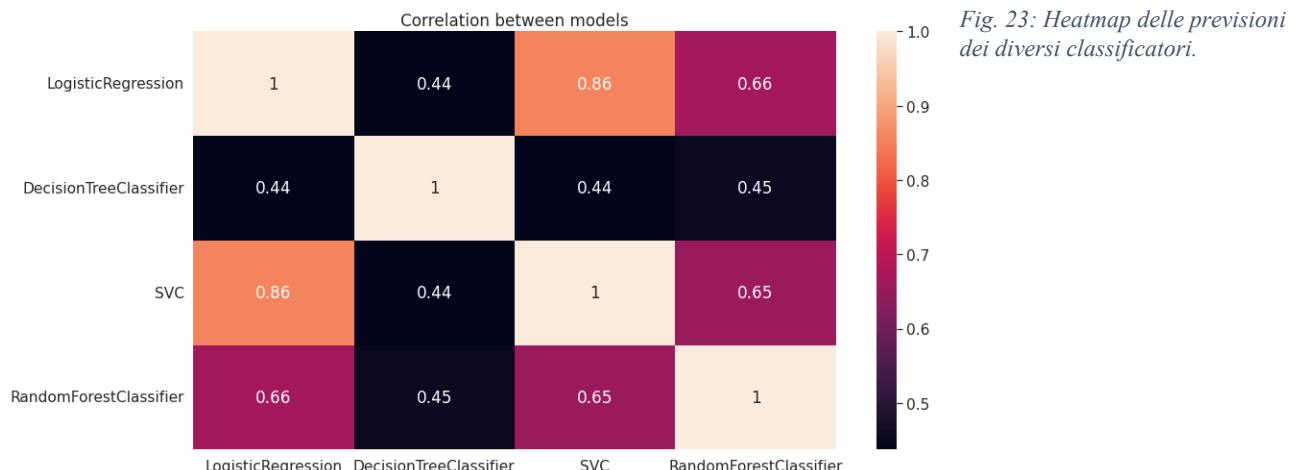


Fig. 22: ROC curve dei classificatori.

La ROC conferma anche l'analisi fatta precedentemente sulla bontà di LogisticRegression e SVC: i due performano discretamente e sono quelli che più si avvicinano al vertice in alto a sinistra corrispondente alla classificazione perfetta.

Un passaggio intermedio che ha portato ad un miglioramento dei due peggiori classificatori è stato quello di implementare una GridSearch che permettesse un primo finetuning dei classificatori, permettendoci di aumentare leggermente gli score di classificazione. In particolare, implementando l'algoritmo sul training multiplo dei classificatori su un range di diverse variabili di configurazione è stato possibile ottenere un miglior punteggio di accuracy del 73,8% per il Decision Tree Classifier assieme al 74,8% del Random Forest Classifier.

Interessante, in coda, andare a verificare la correlazione tra le previsioni dei modelli appena visti. Questo è stato fatto con la seguente heatmap.



A terminare l'analisi dei primi 4 classificatori si è voluto testare l'andamento di un classificatore di tipo Voting, basato sul voto ottenuto dai 4 diversi strumenti, Logistic Regression, SVC e i due ottimizzati Random Forest e Decision Tree.

Il risultato ottenuto è in statisticamente in linea con i risultati di classificazione ottenuti precedentemente dai singoli, sebbene leggermente migliore. In particolare, si ottiene un'accuratezza del 75,7% e la seguente matrice di confusione su di un campione di 1000 elementi bilanciati del dataset principale.

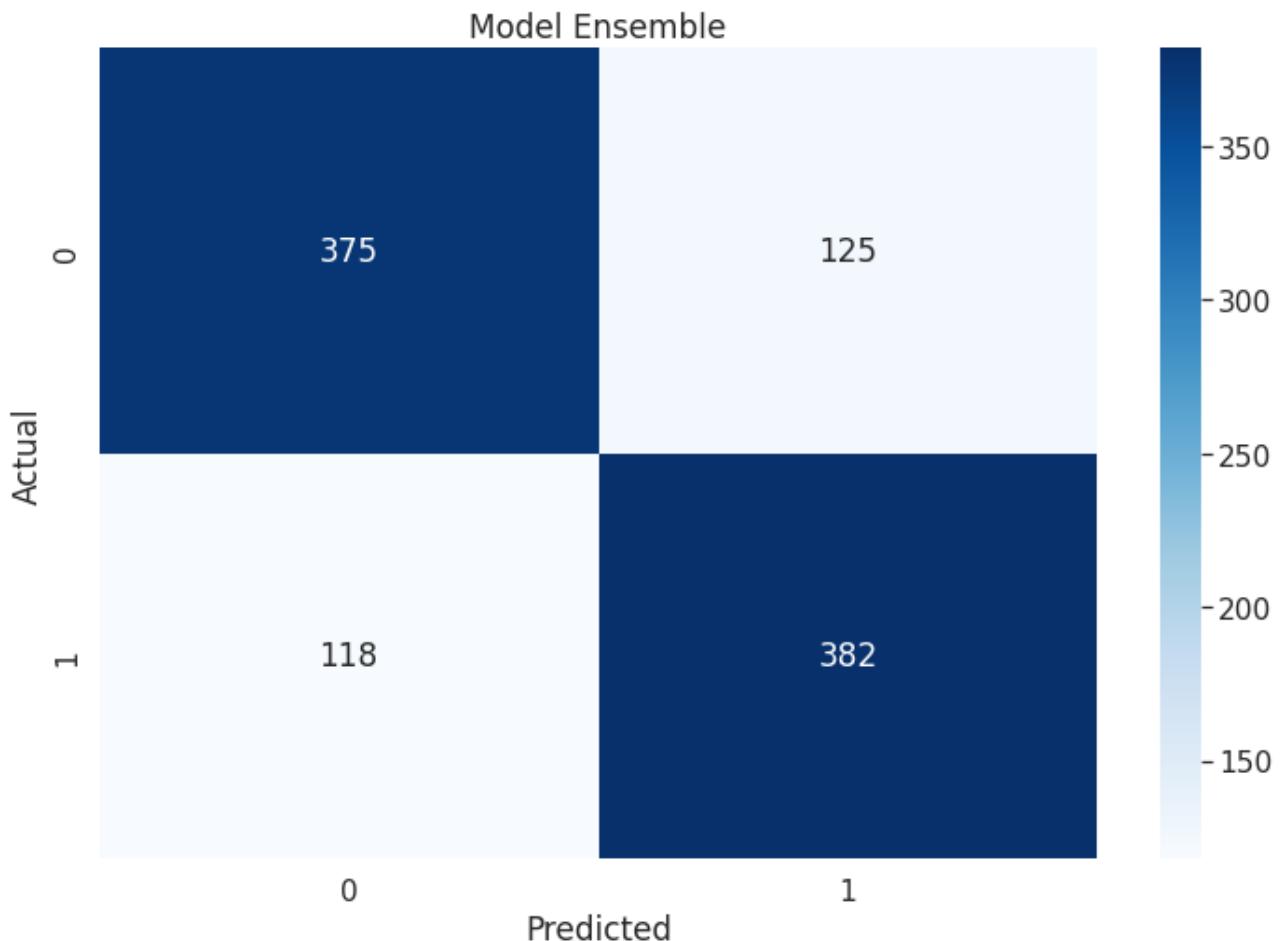


Fig. 24: Matrice di confusione del Voting Classifier su di un campione bilanciato di 1000 elementi.

Mantenendo il classificatore così ottenuto date le sue performance leggermente migliori degli altri, si è scelto di utilizzare nuovamente la libreria Yellowbrick per ottenere interessanti insight sullo strumento appena generato.

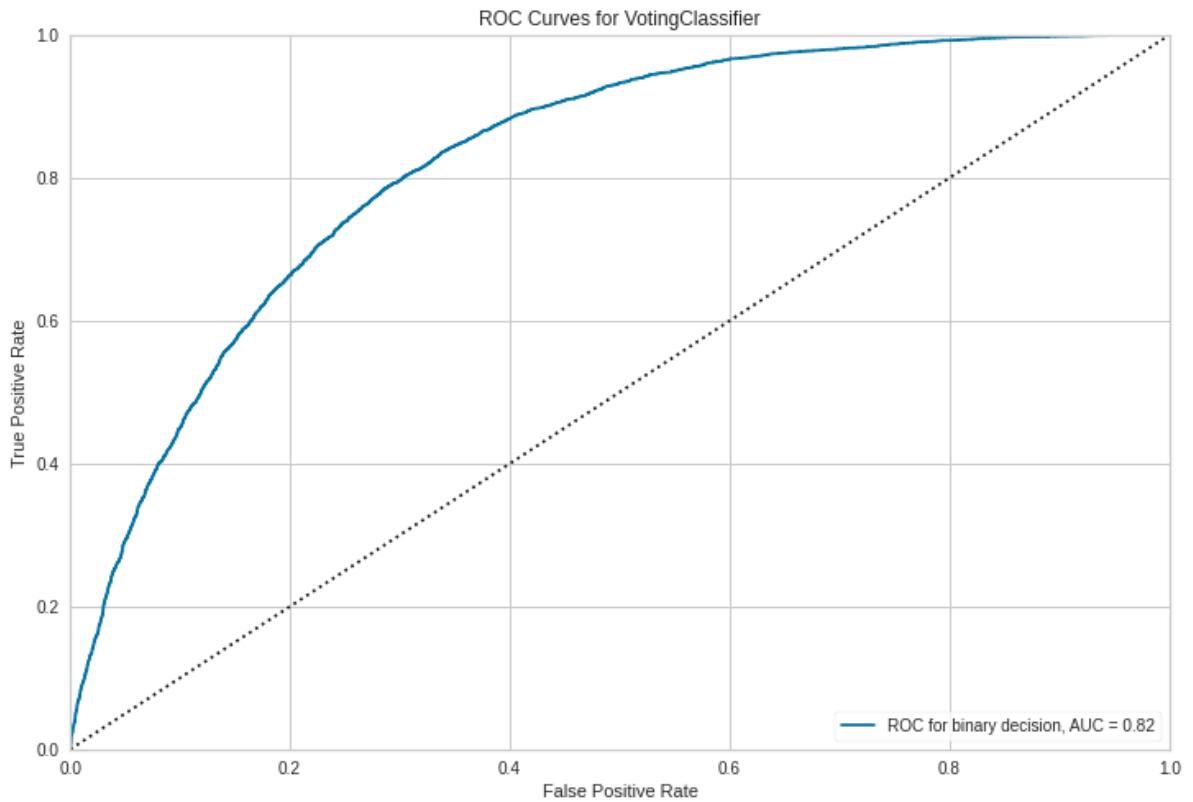


Fig. 25: ROC Curve del Voting Classifier

Come previsto la ROC curve riporta un discreto andamento e permette il calcolo dell’altro parametro notevole, l’Area Under Curve (AUC), l’area compresa al di sotto della ROC sul totale, che ci consegna un buon 82%. Gli stessi ottimi risultati sono evidenziati dalla PR Curve, curva che mette in relazione precision e recall.

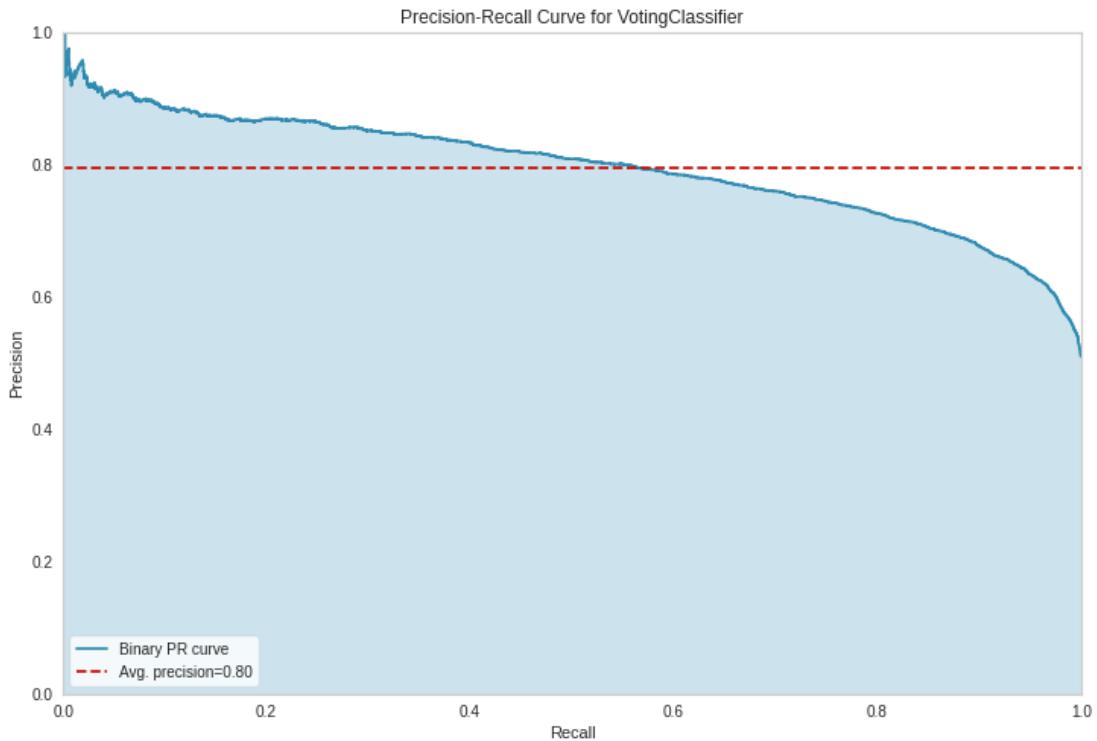


Fig. 26: PR Curve del Voting Classifier.

L'ultima interessante analisi permessa dall'utilizzo della libreria Yellowbrick ci viene data dalla possibilità di generare la Learning Curve. Questa ci mette in relazione l'andamento di training score e cross validation, dove solitamente la loro convergenza all'aumento delle istanze di training è consigliata e permette di percepire la difficoltà che si avrebbe nell'ottenere performance maggiori con l'aumento dei dati di training.

Nel nostro caso, come mostrato dalla figura successiva, training score e cross validation score continuano a convergere, ma è chiaro come per ottenere risultati fortemente migliori sia necessario un più forte aumento dei dati a disposizione.

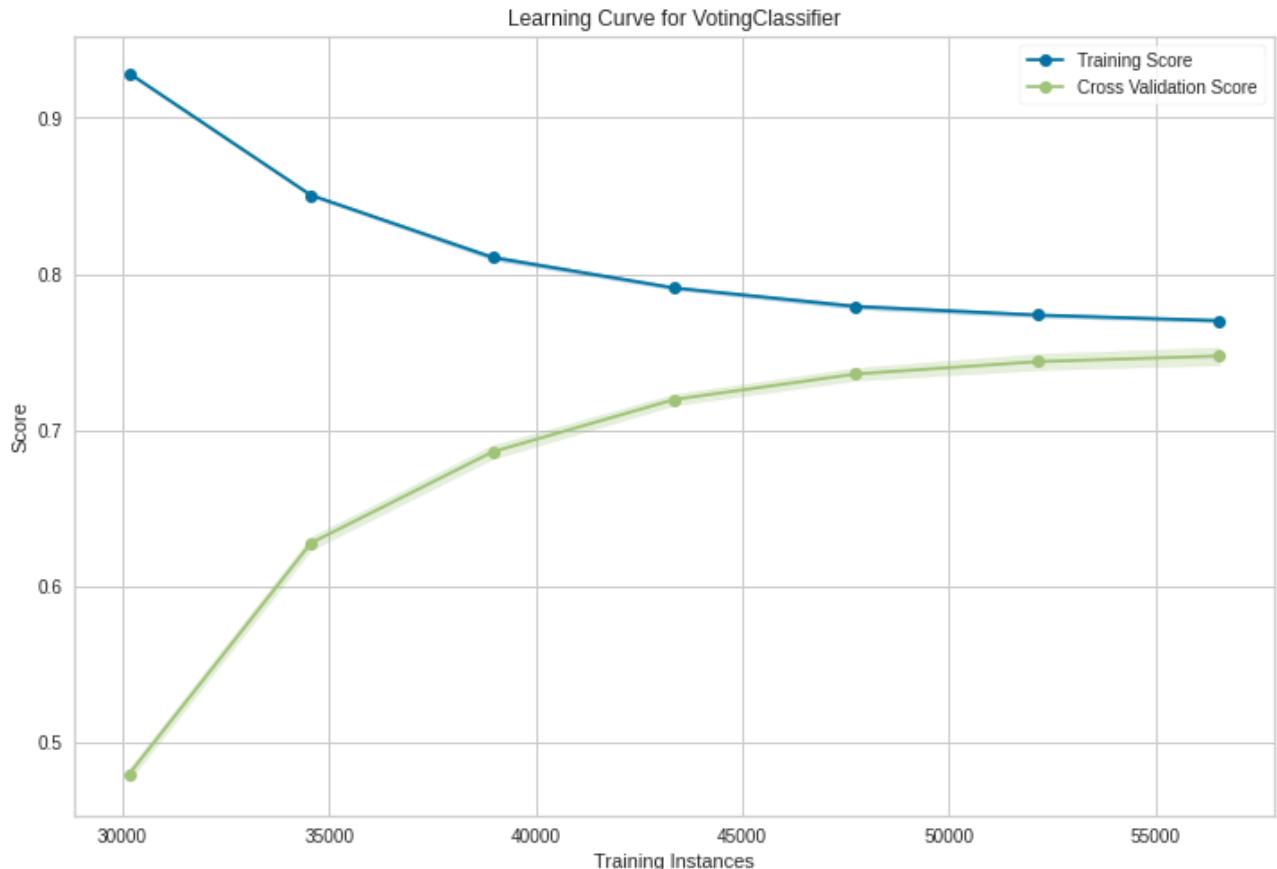


Fig. 27: Learning Curve del Voting Classifier.

Secondo approccio

Lo stesso approccio viene applicato su di un subset delle colonne. In particolare, vengono scartate:

- Smoker
- Fruits
- Veggies
- HvyAlcoholConsump
- AnyHealthcare
- NoDocbcCost
- MentHlth
- Sex

La prima accuracy dei classificatori risulta in linea con i risultati ottenuti precedentemente:

- Logistic Regression: 74%
- Decision Tree: 66%
- SVC: 74%
- Random Forest: 70%

Le stesse matrici di confusione sono confrontabili con le precedenti. Lo sono quindi anche la ROC Curve e la heatmap di correlazione delle previsioni fornite, quindi, per non appesantire ulteriormente la trattazione non verranno riportate.

Il voting classifier ottiene risultati fortemente paragonabili con quello ottenuto precedentemente ottenendo una accuracy score del 74,7%, leggermente inferiore, ma comprensibile nel caso di una fluttuazione statistica.

La chiave di lettura che però giustifica questa analisi è fornita dal grafico della Learning Curve del Voting Classifier.

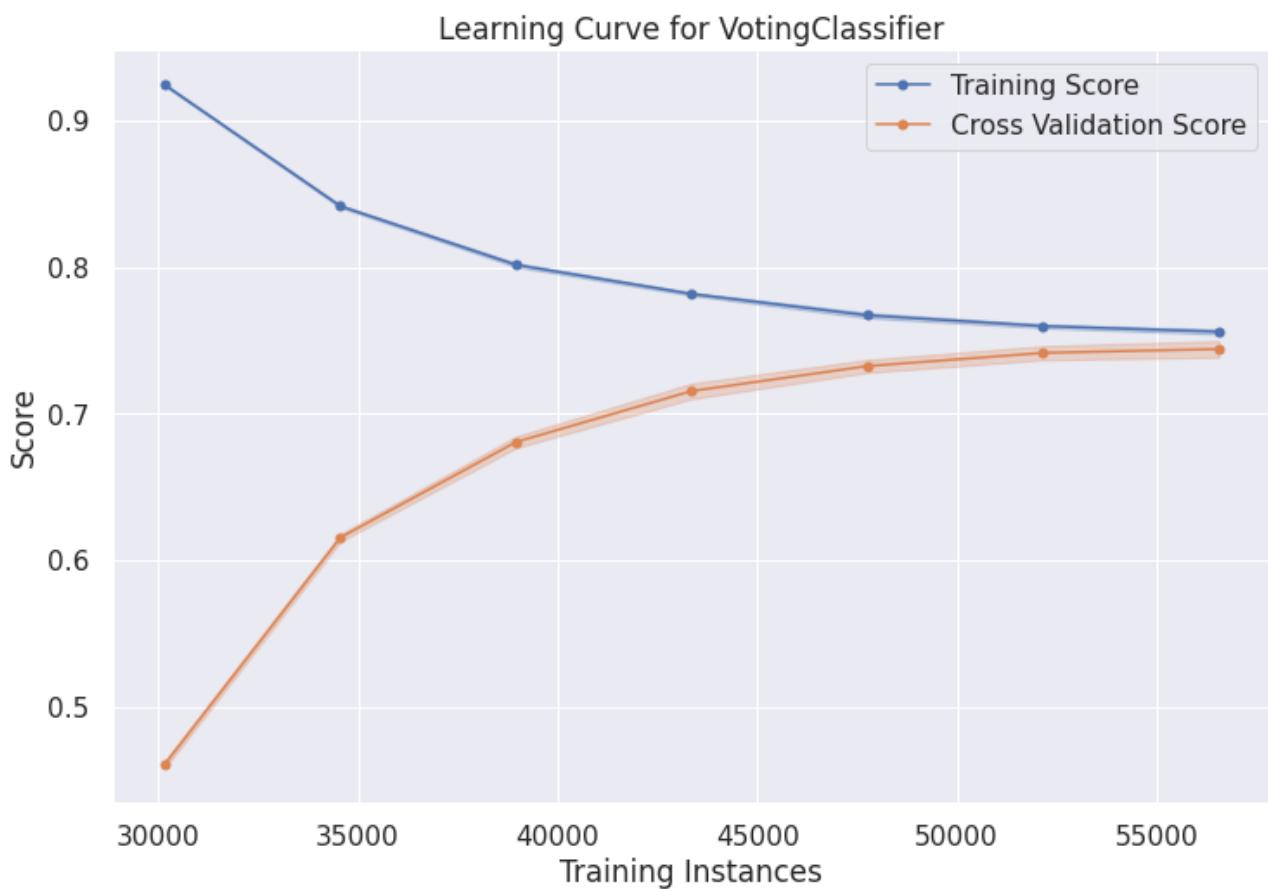


Fig. 28: Learning Curve del Voting Classifier ottenuto escludendo dal training e dal testing le colonne con basso valore di correlazione con il valore delle classi.

Il confronto tra questa Learning Curve e la precedente ottenuta dal training su tutte le 21 colonne del dataset ci fa intuire come una selezione migliore dei dati abbia contribuito ad una maggiore velocità di convergenza delle due curve, segnalando quindi la possibilità di ottenere uno strumento più maturo, con precisione simile, del precedente. Si potranno quindi fare assunzioni sul fatto che il maggior numero di dati di training, se questi non sono decisivi, possono contribuire ad aumentare il gap-informativo che c'è tra lo strumento di machine learning e la verità.

Serie Temporali

Iniziamo questa seconda analisi dando la definizione di serie temporale che corrisponde a quella di una sequenza di osservazioni relative ad una sola variabile registrate ad intervalli di tempo regolari.

Le serie temporali possono essere suddivise in quattro componenti principali: livello di base, trend, stagionalità ed errore. Durante la trattazione vedremo come alcune librerie permettono proprio di identificare tali componenti a partire dal dataset utilizzato.

PS. D'ora in poi la trattazione proseguirà con una logica figura-spiegazione per semplificare la struttura dell'elaborato.

Le librerie utilizzate sono Pandas, per la manipolazione ed analisi dati, NumPy, per aggiungere supporto a grandi matrici o array multidimensionali, Matplotlib, per la creazione di grafici, e Statsmodels, per l'uso di modelli o test statistici. Possiamo osservare tali librerie nella seguente figura:

```
IMPORT LIBRARIES

import pandas as pd
import numpy as np
from numpy import log
from pmdarima import auto_arima
import warnings
import matplotlib.pyplot as plt
import statsmodels.api as sm
from statsmodels.tsa.stattools import acf
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.arima_model import ARIMA
from statsmodels.tsa.seasonal import seasonal_decompose

✓ 118s
```

Caricamento ed esplorazione dati

Il dataset utilizzato è un file csv che riporta le vendite mensile di alcool da gennaio 1992 fino a gennaio 2019.

```
LOAD DATA & DATA EXPLORATION

df = pd.read_csv('Alcohol_Sales.csv')
df.head()
✓ 0.1s

DATE S4248SM144NCEN
0 1992-01-01      3459
1 1992-02-01      3458
2 1992-03-01      4002
3 1992-04-01      4564
4 1992-05-01      4221

df.rename({"S4248SM144NCEN": "Alcohol_Sales", "DATE": "Time"}, axis=1, inplace=True)
df['Time'] = pd.to_datetime(df['Time'])

✓ 0.9s

df.set_index('Time', inplace=True)
df.head()
✓ 0.9s

Alcohol_Sales
Time
1992-01-01      3459
1992-02-01      3458
1992-03-01      4002
1992-04-01      4564
1992-05-01      4221
```

Nella prima parte di codice viene importato il file csv e viene creato un dataframe df. Vengono quindi visualizzate le prime 5 istanze. Possiamo notare la presenza di due colonne, la prima denominata *DATE* riporta appunto la data e la seconda *S4248SM144NCEN* riporta il valore sulle vendite di alcool.

È evidentemente come ci sia da eseguire un po' di preprocessing.

² Dataset disponibile a [questo link](#).

La prima operazione svolta rinomina il campo *DATE* in *Time* ed il campo *S4248SM144NCEN* in *Alchol_Sales*.

La seconda converte il campo *Time* in formato *datetime* in modo che possa essere riconosciuta correttamente come una data.

```
df.index
✓ 0.1s
DatetimeIndex(['1992-01-01', '1992-02-01', '1992-03-01', '1992-04-01',
               '1992-05-01', '1992-06-01', '1992-07-01', '1992-08-01',
               '1992-09-01', '1992-10-01',
               ...
               '2018-04-01', '2018-05-01', '2018-06-01', '2018-07-01',
               '2018-08-01', '2018-09-01', '2018-10-01', '2018-11-01',
               '2018-12-01', '2019-01-01'],
              dtype='datetime64[ns]', name='Time', length=325, freq=None)

df.iloc[0]
✓ 0.1s
Alcohol_Sales    3459
Name: 1992-01-01 00:00:00, dtype: int64
```

Df.Index restituisce gli indici del dataframe e notiamo come viene riconosciuto il formato datetime.

Viene nuovamente confermato che il dataset riporta i valori dal 1992 a gennaio 2019.

Df.iloc[0] visualizza il primo valore del dataframe.

```
df.iloc[-1]
✓ 0.1s
Alcohol_Sales    10718
Name: 2019-01-01 00:00:00, dtype: int64

df.describe()
✓ 0.1s
Alcohol_Sales
count    325.000000
mean    7886.400000
std     2914.269061
min     3031.000000
25%    5231.000000
50%    7481.000000
75%    9977.000000
max    15504.000000
```

Df.iloc[-1], invece, visualizza l'ultimo valore e notiamo già come il valore delle vendite sia praticamente triplicato. Dai valori minimi, massimi e dai percentili vediamo come all'interno della serie i 325 valori a nostra disposizione cambini di molto.

Verifica della stazionarietà

```
STATIONARITY VERIFY

ORIGINAL DATASET

result = adfuller(df.Alcohol_Sales.dropna())
print('ADF Statistics: %f' % result[0])
print('p-value: %f' % result[1])
✓ 0.9s
ADF Statistics: 2.037405
p-value: 0.998720

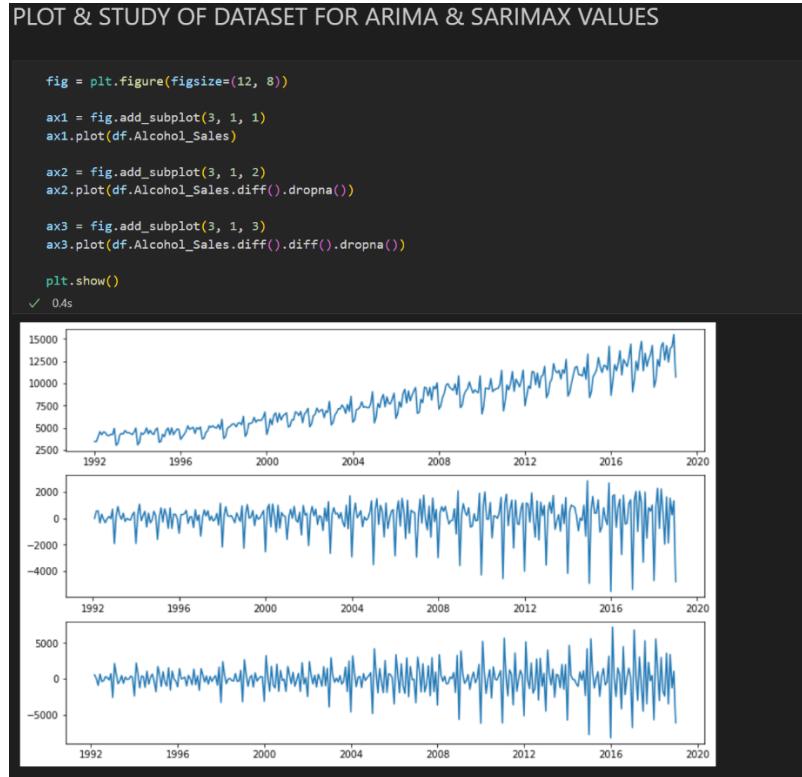
CAUSE IDENTIFICATION FOR HIGH P-VALUE (TREND)

dfarima = df['Alcohol_Sales'] - df['Alcohol_Sales'].rolling(12).mean()
✓ 0.6s

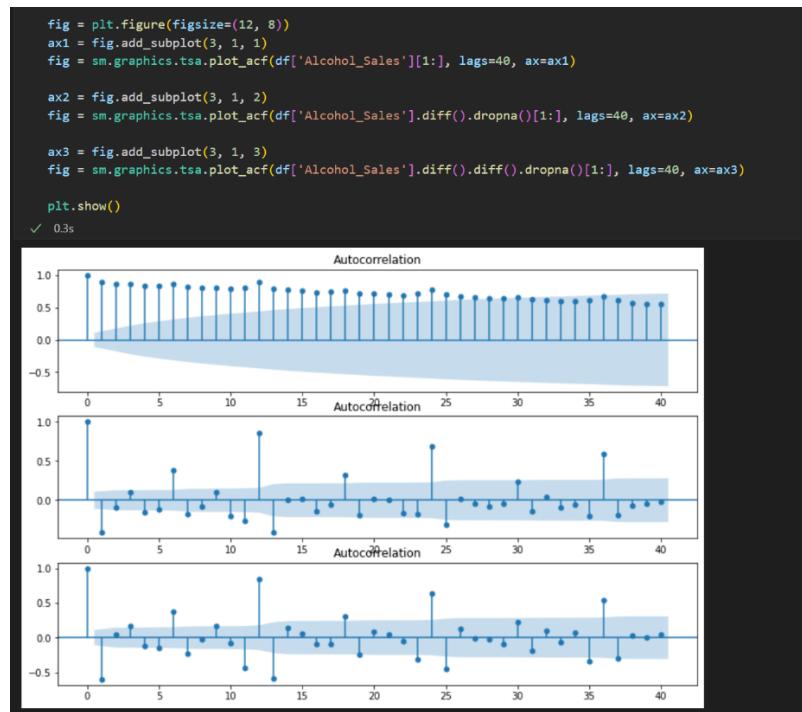
result = adfuller(dfarima.dropna())
print('ADF Statistics: %f' % result[0])
print('p-value: %f' % result[1])
✓ 0.5s
ADF Statistics: -3.968880
p-value: 0.001583
```

Per la verifica della stazionarietà facciamo uso del p-value ed otteniamo un valore si 0.998720, decisamente negativo. Vista la concreta possibilità di avere un trend, e vista l'influenza che ha quest'ultimo sulla stazionarietà della serie, decidiamo di indagare sottraendo alla serie stessa la media calcolata su finestre di 12 valori. Calcoliamo nuovamente il p-value ed otteniamo 0.001583. Questa importante differenza è da ricercare nella definizione di serie stazionaria, cioè nell'indipendenza di media e varianza dal tempo. Dunque, la media diventa dipendente dal tempo qualora sia presente un trend.

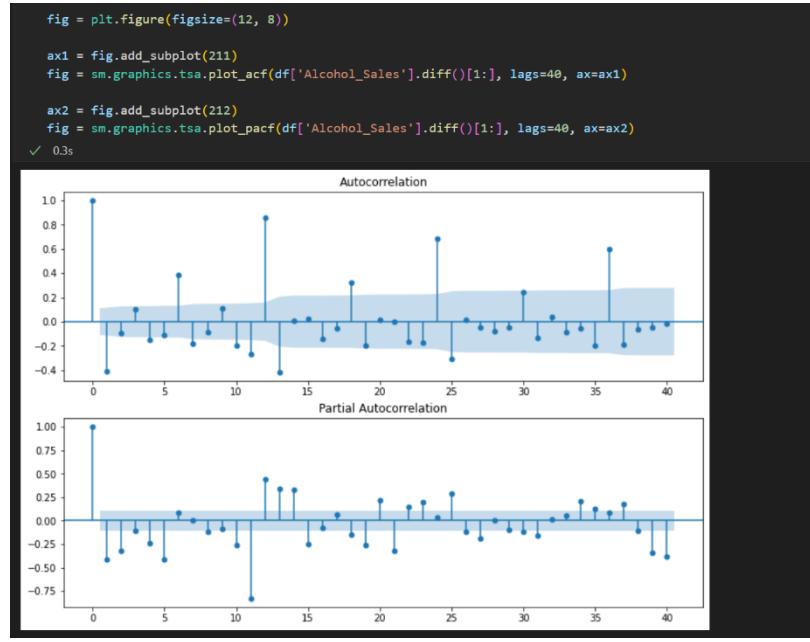
ARIMA



Per verificare in che “ordine delle differenze” la serie diventa stazionaria, e decidere l’ordine di ARIMA, calcoliamo appunto il primo ed il secondo ordine di differenza.

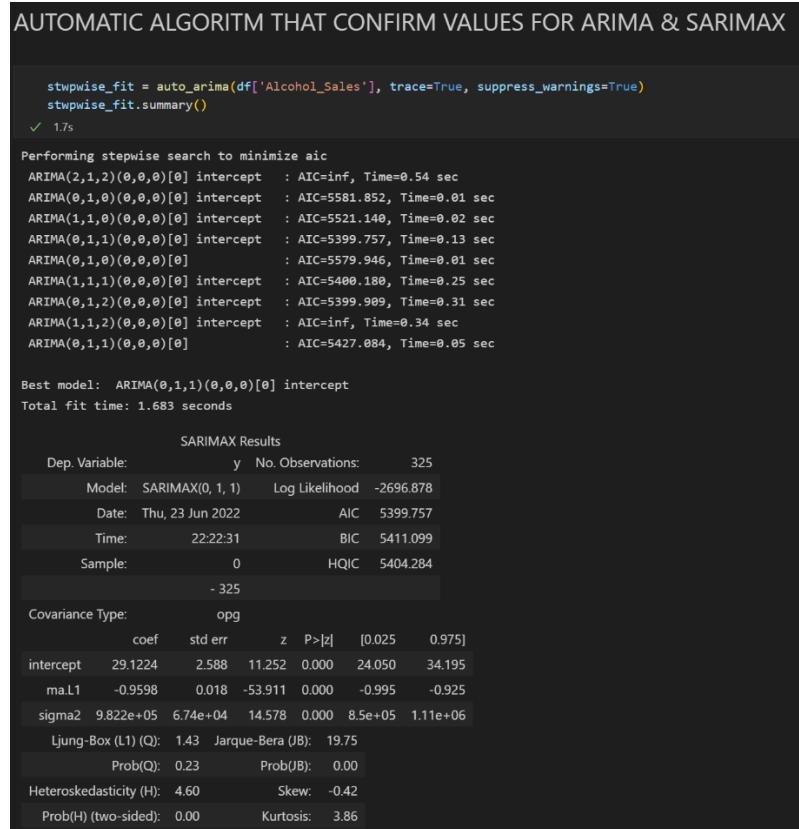


Vengono quindi graficati i relativi valori delle funzioni di autocorrelazione e viene scelto $d = 1$ in quanto la prima autocorrelazione indica come sia da escludere $d = 0$, mentre la terza non porta a vantaggi rispetto alla seconda.



Per calcolare il valore di p per ARIMA viene graficata l'autocorrelazione parziale ed, escluso il primo punto, nessun altro immediatamente successivo ad esso mantiene un valore positivo e superiore al livello di significatività. Dunque, viene scelto p = 0.

Stesso discorso vale per il grafico sull'autocorrelazione e settiamo provvisoriamente q = 0.



Lanciando l'algoritmo della figura precedente vengono testati diversi valori di p, d, q e vengono selezionati coloro che portano ad un AIC più piccolo (da non considerare i valori in cui compare la scritta Intercept). L'AIC fornisce una misura della qualità della stima di un modello statistico tenendo conto sia della bontà di adattamento che della complessità del modello. Anche se in contrasto con quanto detto precedentemente, per il valore di q, ci fidiamo di questo algoritmo ed otteniamo ARIMA(0,1,1).

```

ARIMA MODEL FIT

model = ARIMA(df.Alcohol_Sales, order=(0, 1, 1))
model_fit = model.fit(disp=0)
print(model_fit.summary())

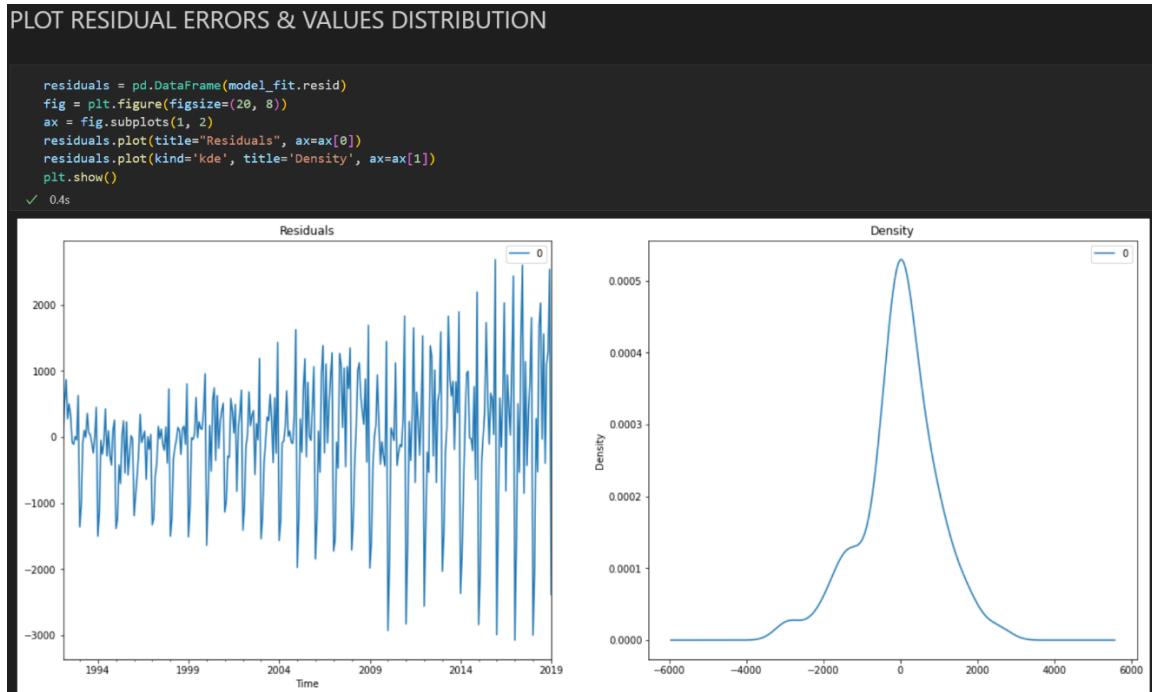
```

0.1s

```

ARIMA Model Results
=====
Dep. Variable: D.Alcohol_Sales No. Observations: 324
Model: ARIMA(0, 1, 1) Log Likelihood: -2695.820
Method: css-mle S.D. of innovations: 989.970
Date: Thu, 23 Jun 2022 AIC: 5397.639
Time: 22:22:31 BIC: 5488.982
Sample: 02-01-1992 HQIC: 5402.167
- 01-01-2019
=====
coef std err z P>|z| [0.025 0.975]
-----
const 28.8602 2.549 11.322 0.000 23.864 33.856
ma.L1.D.Alcohol_Sales -0.9570 0.015 -62.817 0.000 -0.987 -0.927
Roots
=====
Real Imaginary Modulus Frequency
-----
MA.1 1.0450 +0.0000j 1.0450 0.0000
-----
```

A questo punto viene creato il modello ARIMA sulla base del dataframe a nostra disposizione e sui valori di p, d, q precedentemente calcolati. Nel riassunto si può notare come $p > |z|$ abbia entrambi i valori pari a 0, elemento decisamente positivo in quanto solitamente si desiderano valori inferiori allo 0.05.



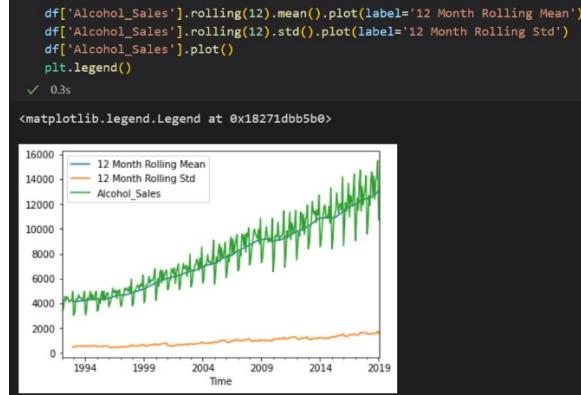
Dal calcolo e dalla visualizzazione dei residui non si notano pattern particolari tranne che una leggera stagionalità ed un leggero incremento dei valori dei residui nel tempo. Possiamo dire che il tutto viene giustificato dalla struttura della serie di partenza caratterizzata da una forte stagionalità e da oscillazioni sempre più ampie. Dunque, non siamo all'ottimo ma abbiamo comunque una distribuzione a campana e centrata nello zero.

SARIMA

Quanto svolto fino ad ora serviva principalmente per individuare i valori di p, d, q e per eseguire analisi di vario tipo. Sappiamo però che ARIMA è un modello che non tiene conto della stagionalità, per questo sarà necessario creare un modello SARIMA. Prima di creare il modello appena citato si eseguono analisi di vario tipo e si separano trend, stagionalità, livello di base e residui.

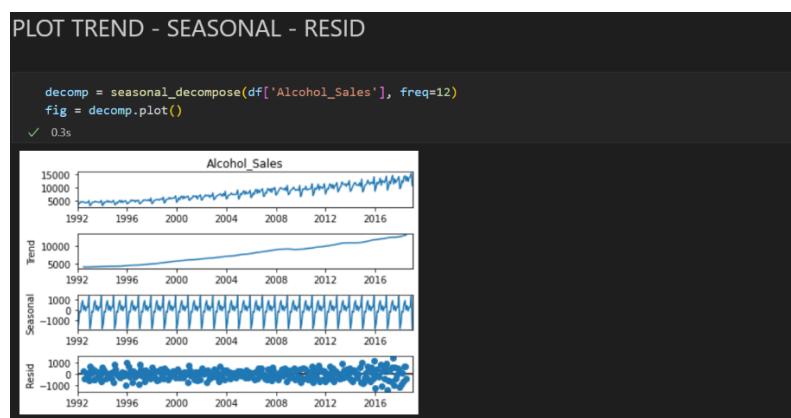


SEPARATION OF MEAN & STD



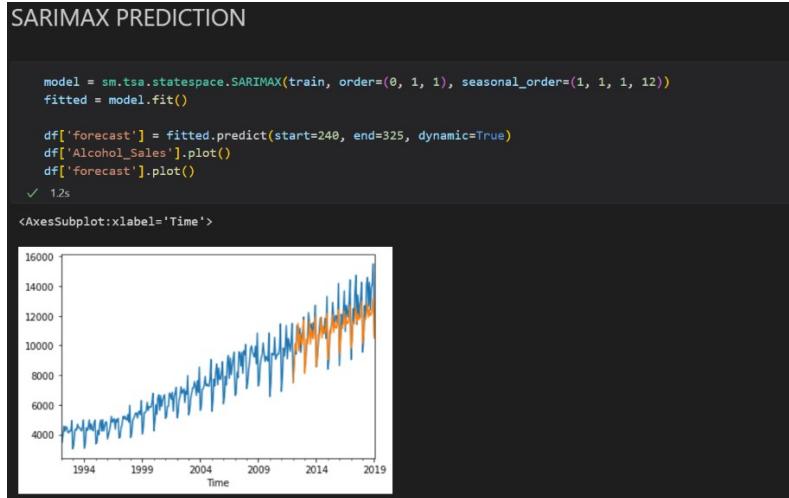
Troviamo una conferma di quanto appena detto dal primo grafico che confronta la predizione di ARIMA con la serie temporale originale. ARIMA riesce a predire tutto nel migliore dei modi ma ovviamente non tiene conto della stagionalità.

Nel secondo grafico, oltre alla serie originale, vengono visualizzati media e deviazione standard calcolati su una finestra temporale di dodici mesi. Abbiamo che il grafico prodotto da ARIMA si avvicina molto a quest'ultimo (media), cosa piuttosto prevedibile.



Ora si vuole tirare in ballo anche la stagionalità e vengono separate le componenti principali della serie.
Considerazioni:

- Il trend è crescente e piuttosto lineare
- La stagionalità forma un grafico molto pulito
- I residui sono piuttosto randomici (elemento decisamente positivo)



In quest'ultima fase viene addestrato un modello SARIMA con $p = 0$, $d = 1$, $q = 1$ (gli stessi precedentemente calcolati in ARIMA) e stagionalità di 12 (mesi).

Viene plottata la serie originale e la serie predetta a partire da 240 valori su 325 totali.

SARIMA riesce a predire bene sia il trend che stagionalità riuscendo a produrre una stima piuttosto affidabile dell'evoluzione futura delle vendite di alcool. L'unico piccolo neo riguarda la magnitudine delle escursioni che viene sottostimata.

Una ultima analisi riguarda il fenomeno studiato ovvero il consumo di alcool nel tempo. È chiaro come le vendite siano più che triplicate in 27 anni ma non è chiaro se questo sia in qualche misura correlato alla crescita demografica mondiale. Riassumiamo questi semplici dati per trarre le conclusioni:

- 1992 : Consumo di alcool 3459 (gennaio) / Popolazione mondiale: 5,7 miliardi circa
- 2019 : Consumo di alcool 10718 (gennaio) / Popolazione mondiale: 7,7 miliardi circa

Si può dedurre che le vendite di alcool crescono molto più rapidamente della popolazione mondiale, dunque, ciascuna persona tende a consumare mediamente sempre più alcoolici.

Social Network Analysis

Introduciamo brevemente il concetto di Social Network Analysis definendola come lo studio delle relazioni umane per mezzo della teoria dei grafi. Nel caso in questione si parte da un dataset, scaricato dal sito snap.stanford.edu come soc-redditHyperlinks-title.tsv, di collegamenti ipertestuali che rappresenta le connessioni dirette tra due subreddit.

Il file di partenza ha la seguente struttura:

SOURCE_SUBREDDIT	TARGET_SUBREDDIT	POST_ID	TIMESTAMP	LINK_SENTIMENT	PROPERTIES
rddtgaming	rrdtrust	1u4pzzs	2013-12-31 16:39:18 1	25,0,23,0,0,76,0,0,0,44,0.....	
....

La libreria core di questo progetto sarà networkx che, come il nome lascia intuire, è stata creata per la costruzione, l'elaborazione e l'analisi dei grafi. Altre librerie di rilievo che verranno usate sono matplotlib e seaborn.

IMPORT LIBRARIES

```
import networkx as nx
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors
import random
import seaborn as sns
import csv
```

✓ 13.9s Python

Caricamento ed esplorazione dati

IMPORT GRAPH

```
g = nx.Graph()
csvfile = csv.reader(open("./soc-redditHyperlinks-title.tsv", "r"), delimiter="\t")
for line in csvfile:
    x, y = line[:2]
    g.add_edge(x, y)
```

✓ 10.6s Python

SUBSAMPLE OF G

```
k = 10000 #Numero di nodi del sample

sample_nodes = random.sample(g.nodes, k)
g_small = g.subgraph(sample_nodes)
```

✓ 0.1s C:\Users\crist\AppData\Local\Temp\ipykernel_17256\830814946.py:3: DeprecationWarning: Sampling from a set deprecated since Python 3.9 and will be removed in a subsequent version.
sample_nodes = random.sample(g.nodes, k)

print(nx.info(g_small))

✓ 0.3s Graph with 10000 nodes and 6385 edges Python

Come prima operazione importiamo il file tsv e creiamo il grafo g.

La parte di subsample randomico dei nodi (e di conseguenza degli archi) è stata necessaria per le operazioni che seguiranno in quanto con pc domestici sarebbe stato necessario attendere moltissime ore per l'esecuzione di alcuni pezzi di codice. Con un subsample da 55863 a 10000 nodi si riescono ad ottenere risultati in diversi minuti, quindi tempistiche più che accettabili.

³ Dataset disponibile a [questo link](#).

GRAPH EXPLORATION

PRINT SOME NODES

```
list(g_small.nodes)[:10]
✓ 0.1s
['fullmetalalchemist',
 'yourselfyou',
 'broforcegame',
 'trendingsubreddits',
 'camarillo',
 'coldplay',
 'papertowns',
 'de_it',
 'fallout_shelter',
 'amazonlumberyard']
```

Python

NETWORK DIAMETER

```
components = nx.connected_components(g_small)
largest_component = max(components, key=len)
subgraph = g_small.subgraph(largest_component)
diameter = nx.diameter(subgraph)
print("Network diameter: ", diameter)
✓ 2m 34.9s
Network diameter: 12
```

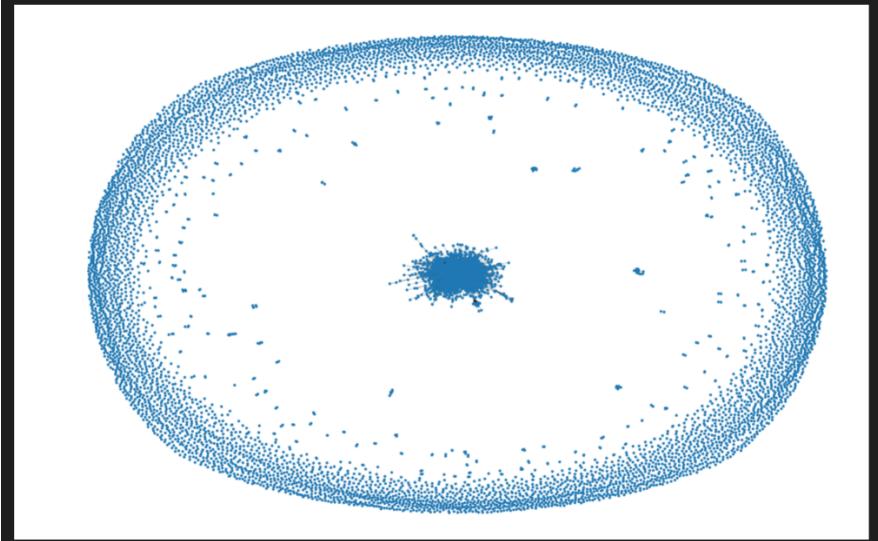
Python

Con le parti di codice precedenti si esplorano alcuni nodi, per poter comprendere i dati si stanno trattando, e si calcola il diametro massimo del grafo che risulta essere pari a 12.

VISUALIZATION WHOLE SAMPLE

```
import matplotlib.pyplot as plt
pos = nx.spring_layout(g_small)
plt.figure(figsize=(80,50))
nx.draw(g_small, pos)
plt.show()
✓ 6m 37.6s
```

Python



Dalla visualizzazione precedente è evidente come moltissimi nodi siano isolati o costituiscono gruppi di grafi connessi di piccole dimensioni.

Selezione del componente connesso più grande

BIGGEST CONNECTED COMPONENT EXTRACTION & VERIFICATION

```

g_connected = g_small.subgraph(max(nx.connected_components(g_small), key=len))
✓ 0.2s Python

print("Graph is connected: " + str(nx.is_connected(g_connected)))
print("Number of different connected components: " + str(nx.number_connected_components(g_connected)))
print("Number of nodes to be removed (to be disconnected): " + str(nx.node_connectivity(g_connected)))
print("Number of edges to be removed (to be disconnected): " + str(nx.edge_connectivity(g_connected)))
✓ 1m 30.9s Python

Graph is connected: True
Number of different connected components: 1

```

Dalle precedenti considerazioni emerge la necessità di trattare il componente connesso di dimensioni maggiori e dunque eliminare tutti i nodi che non appartengono ad esso.

SOME PROPERTIES OF CONNECTED GRAPH

```

print("Radius (minimum eccentricity): " + str(nx.radius(g_connected)))
print("Center: ", list(nx.center(g_connected)))
triadic_closure = nx.transitivity(g_connected)
print("Triadic closure:", triadic_closure)

print("Periphery: ", list(nx.periphery(g_connected)))
] ⓘ 5m 34.8s Python

```

Qui vengono stampate alcune caratteristiche di questa componente principale come il raggio, gli elementi centrali, percentuale di triadi chiuse e la periferia. Curioso come il raggio pari a 6 rispecchi il criterio secondo cui nel mondo fra una persona e qualsiasi altra ci sia generalmente un grado di separazione che difficilmente supera questo valore. Interessante è anche la presenza di Italy tra gli elementi che costituiscono il centro del grafo. Invece, una percentuale di chiusura delle triadi di quasi il 6% non è sicuramente un valore altissimo.

TOP 10 NODES BY CENTRALITY

```

degCent = nx.degree_centrality(g_connected)
degCent_sorted=dict(sorted(degCent.items(), key=lambda item: item[1],reverse=True))

print("Top 10 centrality nodes: ")
print(list(degCent_sorted.items())[:10])
] ⓘ Top 10 centrality nodes:
[('subredditdrama', 0.16446357208730403), ('news', 0.08023363049492775), ('science', 0.0562557639102367),
('adviceanimals', 0.04918536735321242), ('technology', 0.048263141715339686), ('theydidthemath', 0.04242237934214571),
('soccer', 0.03811865969873962), ('dataisbeautiful', 0.03504457423916384), ('atheism', 0.03473716569320627),
('television', 0.032277897325545646)]
Python

SHORTEST PATH BETWEEN TOP 3 NODES BY CENTRALITY
```

```

print("SHORTEST PATH BETWEEN TOP 3 NODES BY CENTRALITY:")
listTop3 = list(degCent_sorted.keys())[3]
for s in listTop3:
    for t in listTop3:
        if s!=t:
            fell_whitehad_path = nx.shortest_path(g, source=s, target=t)
            print("Path"+ str(s) + " - "+ str(t) + ": ", fell_whitehad_path)
] ⓘ SHORTEST PATH BETWEEN TOP 3 NODES BY CENTRALITY:
Paths subredditdrama - news: ['subredditdrama', 'news']
Paths subredditdrama - science: ['subredditdrama', 'science']
Path news - subredditdrama: ['news', 'subredditdrama']
Path news - science: ['news', 'science']
Path science - subredditdrama: ['science', 'subredditdrama']
Path science - news: ['science', 'news']
Python

```

Nella prima parte di codice si vogliono visualizzare i primi 10 nodi per centralità ed il rispettivo grado di centralità. In questo caso spiccano subredditdrama, news e science.

Dalla stampa dei percorsi minimi che mettono in relazione questi nodi è evidente come siano tutti molto vicini tra loro e strettamente connessi.

AVERAGE CLUSTERING COEFFICIENT

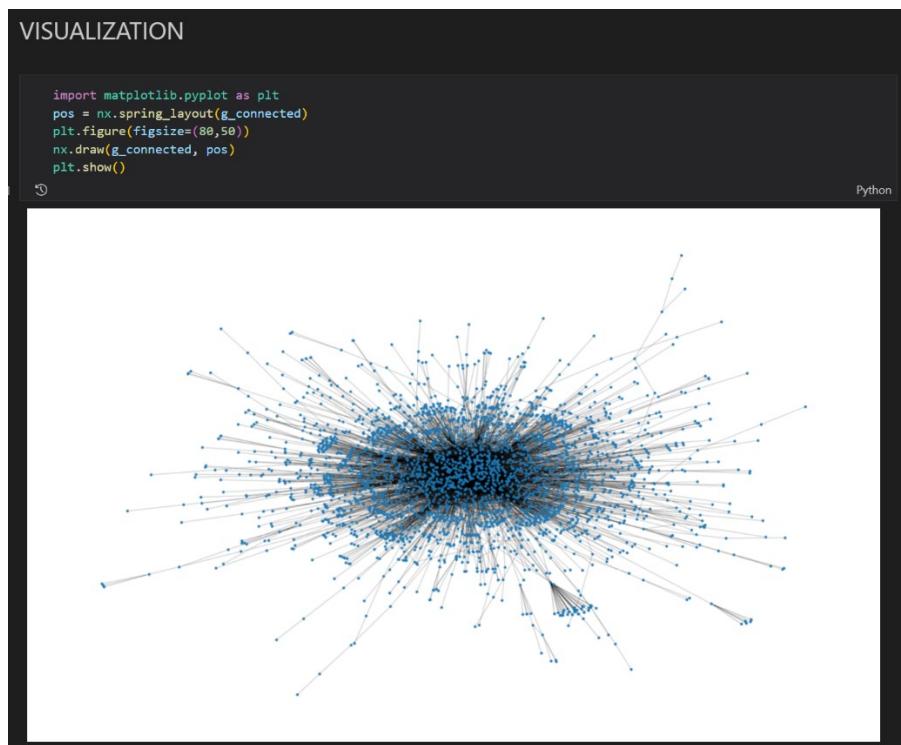
```
print("Avarage clustering coefficient: ")
print(nx.average_clustering(g_connected))
```

Avarage clustering coefficient:
0.11191033073046762

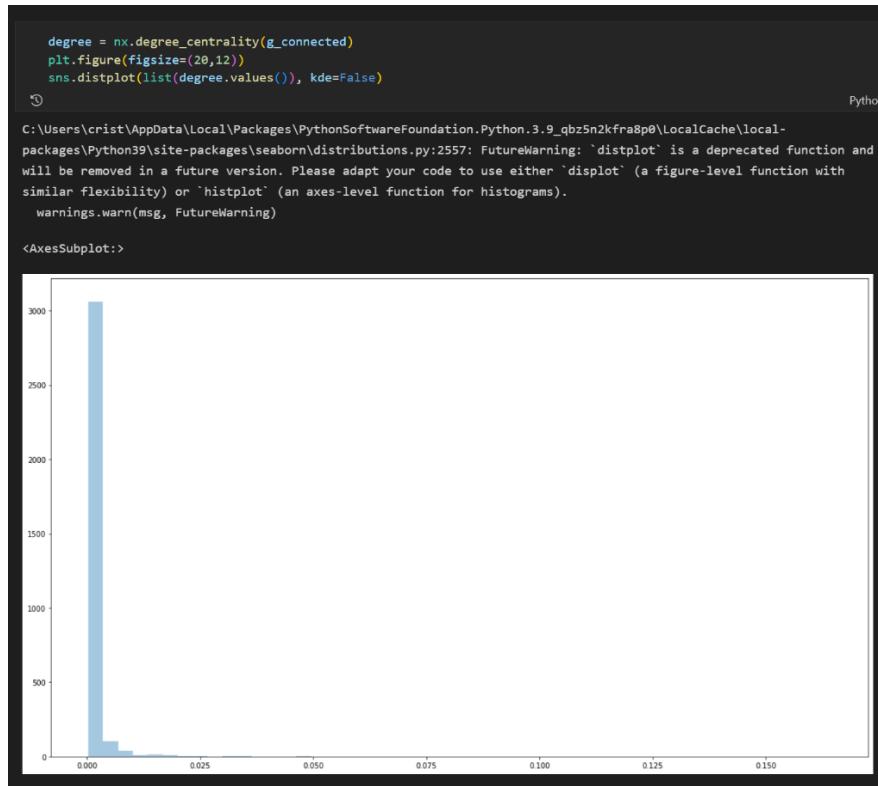
Python

Qui viene stampato semplicemente il valore del clustering coefficient che risulta pari a 0.11 circa.

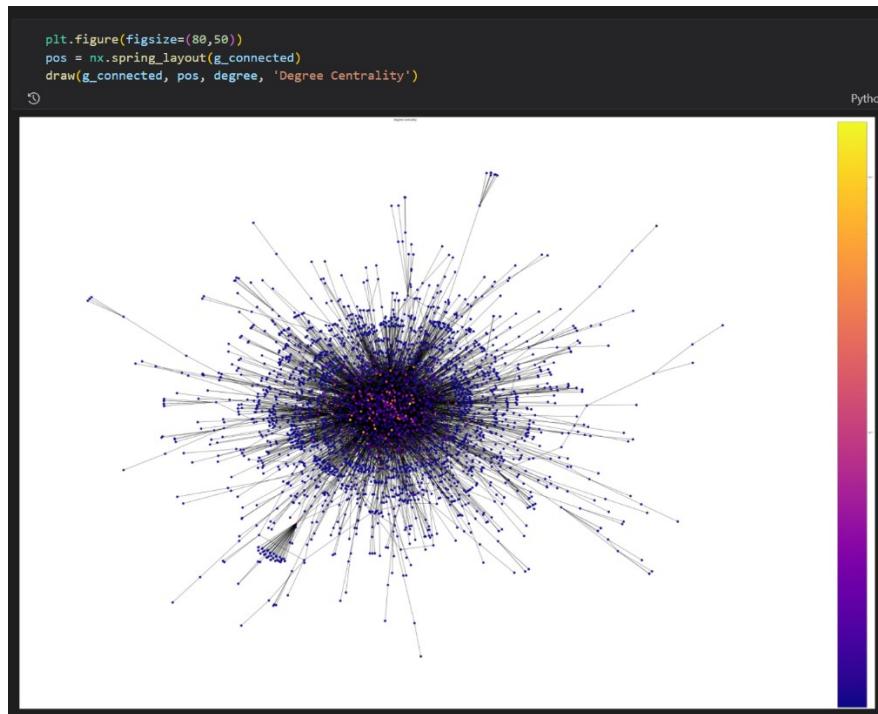
Visualizzazioni



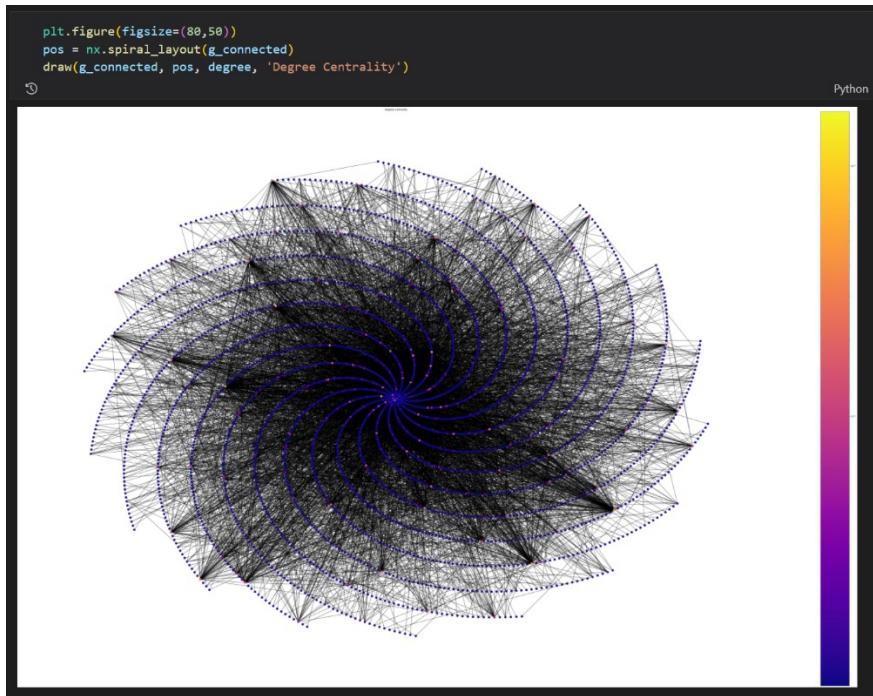
In questa sezione viene visualizzato il grafo connesso di dimensioni maggiori.



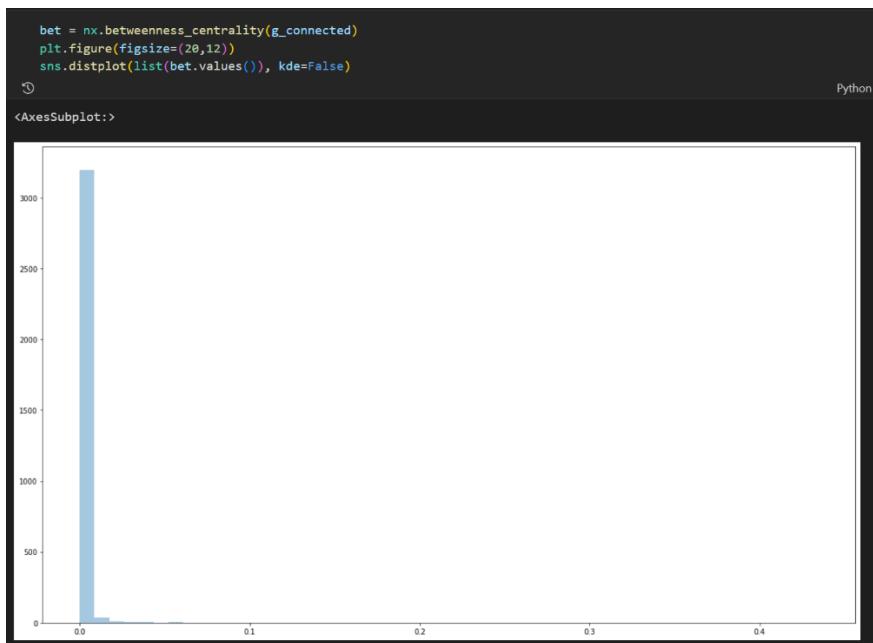
Da qui si osserva come pochi grafi hanno un grado di centralità alto e molti hanno un grado di centralità molto basso. Ciò è piuttosto comune nelle reti di questo tipo.



Quanto detto viene confermato da questa figura. Infatti, la maggior parte dei nodi assume una colorazione blu, indice di una bassa centralità, mentre solo al centro appaiono dei nodi di colore tendente al giallo, indice di un' alta centralità.



Questa è una rappresentazione alternativa di quanto visto precedentemente.



Infine, si può osservare come moltissime persone hanno una bassa influenza nel flusso delle informazioni e solo alcuni ricoprono questo compito in modo importante.