Universitá Politecnica Delle Marche

Ingegneria Informatica e dell'Automazione

Progetto Data Science 2021-2022

Analisi dei Boardgames tramite l'utilizzo dell'ecosistema Python

Studenti:
Massimo Ciaffoni
Simone Cappanera

Indice

Capitolo 1

Introduzione

Python è un linguaggio ampiamente utilizzato nell'ambito della Data Science, in quanto mette a disposizione una serie di librerie particolarmente utili e complete come *pandas, seaborn, Matplotlib, scikit-learn, statsmodels,* e tante altre. Lo scopo del progetto è quello di utilizzare tali librerie per svolgere task di clustering e classificazione. In particolare le librerie maggiormente utilizzate per questi task sono:

- **Pandas**: è una libreria software per la manipolazione e l'analisi dei dati. In particolare, offre strutture dati (DataFrames e Series) e operazioni per manipolare tabelle numeriche e serie temporali.
- **Scikit-learn**: è una libreria utilizzata principalmente per analisi di tipo predittivo; in particolare è possibile effettuare task di classificazione, regressione e clustering.
- **Seaborn**: è una libreria basata su *Matplotlib* che permette di creare diverse visualizzazioni utili. E' molto utilizzata per la creazione di grafici a barre e correlazioni.

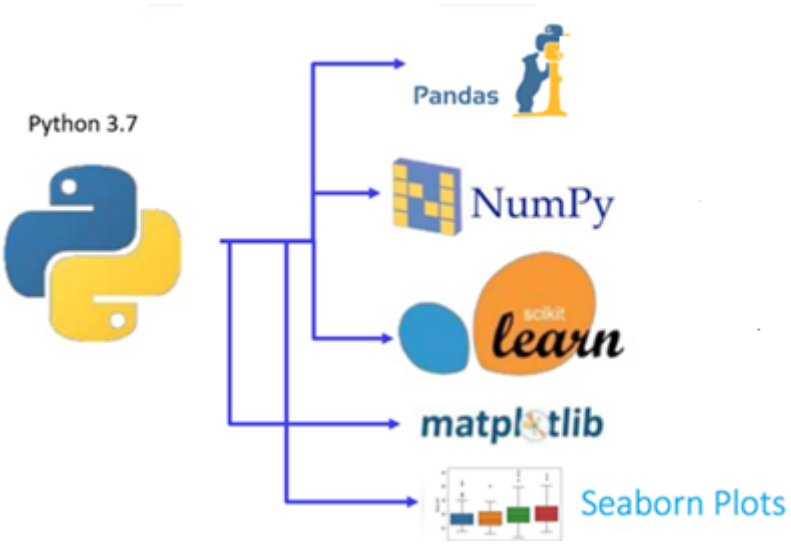


Figura 1.1: Librerie di Python per la Data Science

1.1 Dataset e Preprocessing

Il dataset utilizzato è quello relativo ai boardgames del sito *boardgamegeek.com*, già descritto e impiegato nell'analisi in Power BI. Le trasformazioni effettuate sono le medesime già descritte nella precedente relazione, alle quali sono state aggiunte queste ulteriori modifiche:

- Poiché a ciascun boardgame era associata più di una categoria, per effettuare la classificazione è stato necessario filtrare tutti i giochi da tavolo andando a selezionare solo una categoria (la prima) per ognuno di essi. In questo modo il numero di righe è notevolmente diminuito, facendo tornare il dataset alla dimensione originale.
- I campi ***Name, YearPublished, SortIndex, BoardgameDesigner, BoardgameArtist, BoardgamePublisher, BoardgameMechanic, Description*** sono stati eliminati in modo da alleggerire il processamento dei dati.
- I valori nulli nei campi ***Min_community, Max_community*** e ***PlayerAge*** sono stati sostituiti con il valor medio (nel caso di min_community e max_community si è anche effettuato un controllo semantico sul valore, in modo che, in ogni caso, il valore di min_community non fosse mai maggiore di max_community).
- I record con valori di ***BoardgameCategory*** nulli sono stati eliminati, in quanto non era possibile per questi effettuare la classificazione.

Capitolo 2

Visualizzazioni

Dopo il caricamento e il preprocessing, si sono realizzate delle visualizzazioni sui relativi dati per ottenere delle informazioni di rilievo dal punto di vista analitico. Questi grafici sono stati realizzati tramite le librerie *seaborn* e *Matplotlib*.

2.1 Distribuzioni

La prime due distribuzioni mostrano, rispettivamente, il voto medio e il numero di recensioni dei boardgames. I due grafici sono stati realizzati utilizzando il metodo *distplot* della libreria *seaborn*. Questo metodo crea grafici di distribuzione ad istogramma, dove i valori degli attributi in analisi sono aggregati per intervallo, a cui viene sovrapposta la funzione di distribuzione generata mediante algoritmo *Kernel Density Estimation*. È possibile notare come, nel caso dei voti totali, il numero massimo di recensioni per boardgame si trova solitamente nel primo intervallo, che identifica un range che va all'incirca da 0 a 50 recensioni. Per quanto riguarda, invece, il voto medio delle recensioni, la distribuzione è maggiormente disomogenea, con un valore medio che si attesta al 6,7.

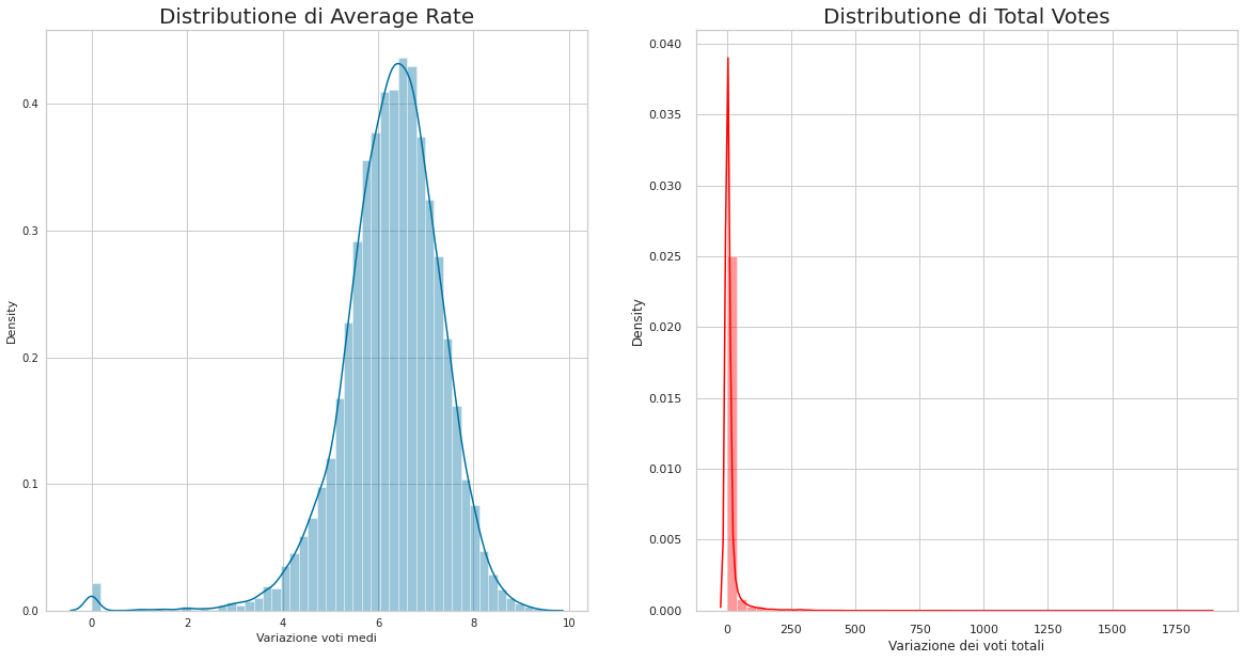


Figura 2.1: Distribuzione delle recensioni

I due grafici in figura 2.2 mostrano l'età minima consigliata per giocare ai vari boardgames, considerando sia quella consigliata dal publisher che dai player stessi. In entrambi i casi notiamo delle distribuzioni con più picchi. In figura 2.3 è presente un confronto tra le due distribuzioni, dove si può notare un maggior accumulo nell'attributo *PlayerAge*, che ha un picco più elevato al valore 10; mentre il campo *MinAge* presenta tre picchi abbastanza omogenei nei valori 8, 10 e 12.

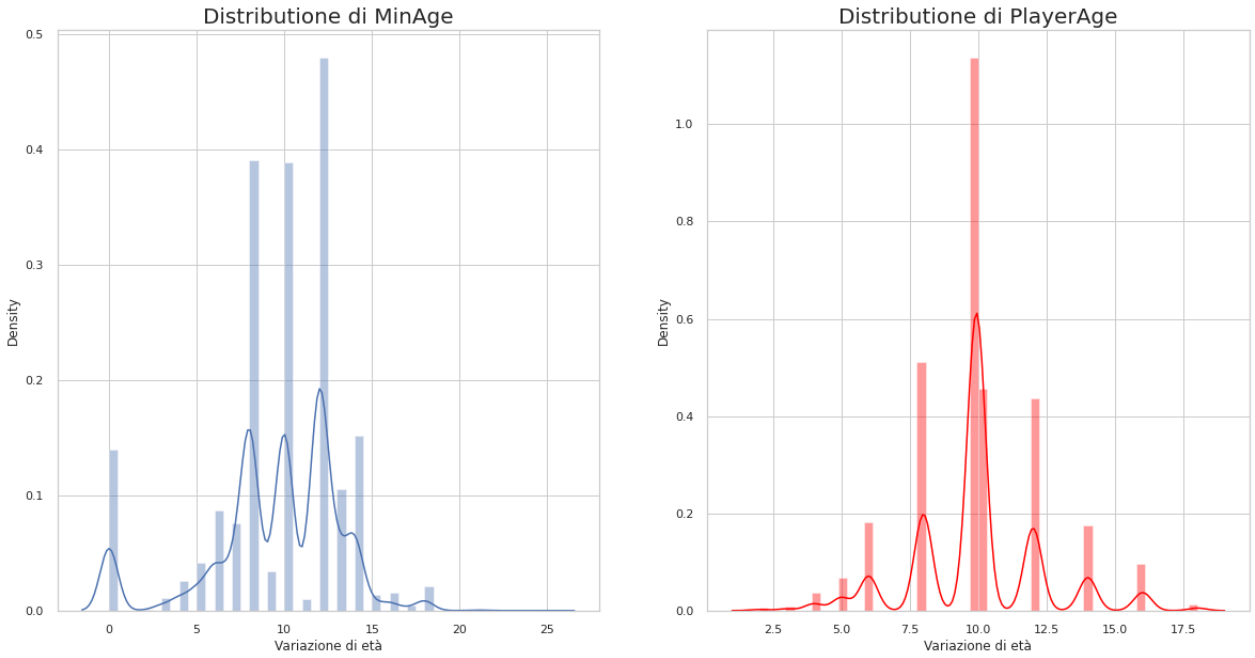


Figura 2.2: Distribuzione dell'età dei giocatori

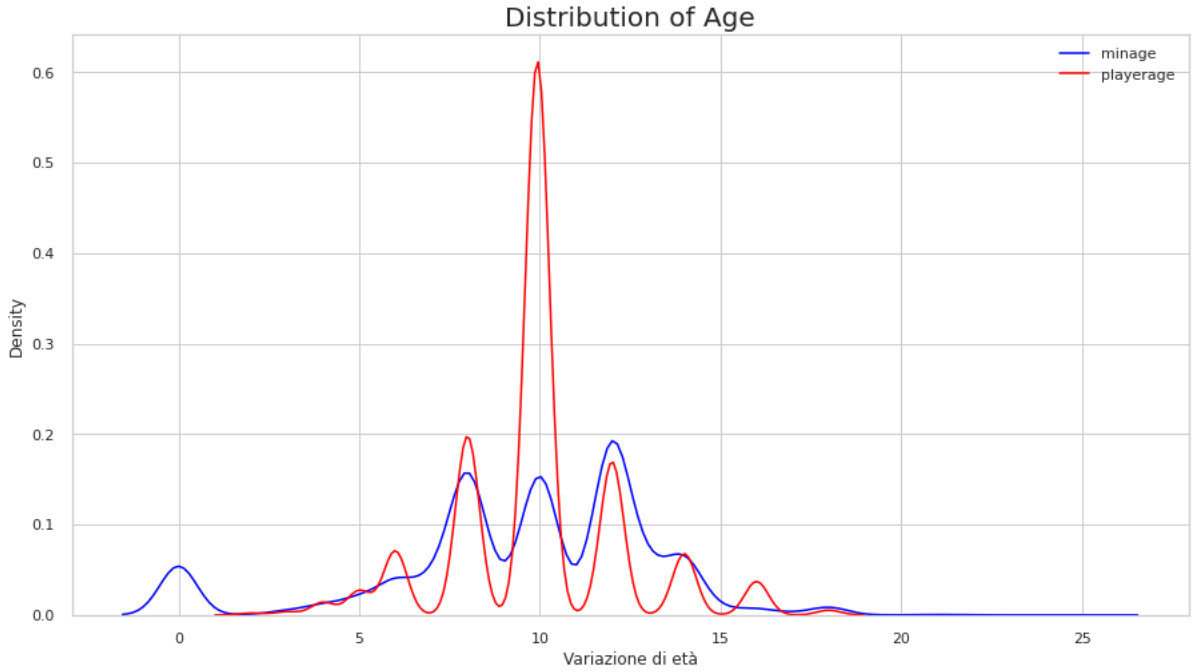


Figura 2.3: Confronto tra MinAge e PlayerAge

2.2 Correlazioni

Per terminare l'analisi descrittiva del dataset, si è andati ad analizzare la correlazione presente tra gli attributi, vale a dire quanto le colonne del dataset siano dipendenti tra di loro. Per prima cosa si è andati ad utilizzare il metodo *pairplot* della libreria grafica *seaborn* per visualizzare, a due a due, tutte le coppie di valori tra gli attributi in un piano cartesiano, in modo che possa risaltare subito all'occhio se ci sono evidenti relazioni tra questi. In particolare, sono stati realizzati quattro pairplot diversi, riguardanti quattro sottogruppi del dataset che sembravano avere una maggiore correlazione tra di loro:

- La figura 2.4 mostra la correlazione tra diversi attributi relativi alle statistiche degli utenti del sito *boardgamegeek.com*, da cui è stato generato il dataset. In particolare, notiamo un'elevata correlazione tra molti degli attributi presi in esame, come, ad esempio, la coppia di attributi *UserRated* e *NumComments*.

- La figura 2.5 mostra la correlazione tra gli attributi relativi alla ricezione mediatica dei vari boardgames. Anche in questo caso sembra esserci una buona correlazione tra alcuni degli attributi.

- La figura 2.6 mostra la correlazione tra i diversi attributi relativi alle caratteristiche dei vari boardgames, come età, tempo e numero di giocatori. In questo caso non è presente un'elevata correlazione, tranne per la coppia di attributi *Min_ community* e *Max_ community*

- La figura 2.7 mostra la correlazione tra alcuni attributi relativi ai conteggi delle diverse liste di artisti, designer e publisher, per ogni gioco da tavolo. Anche in questo caso non è presente un'elevata correlazione tra gli attributi presi in esame.

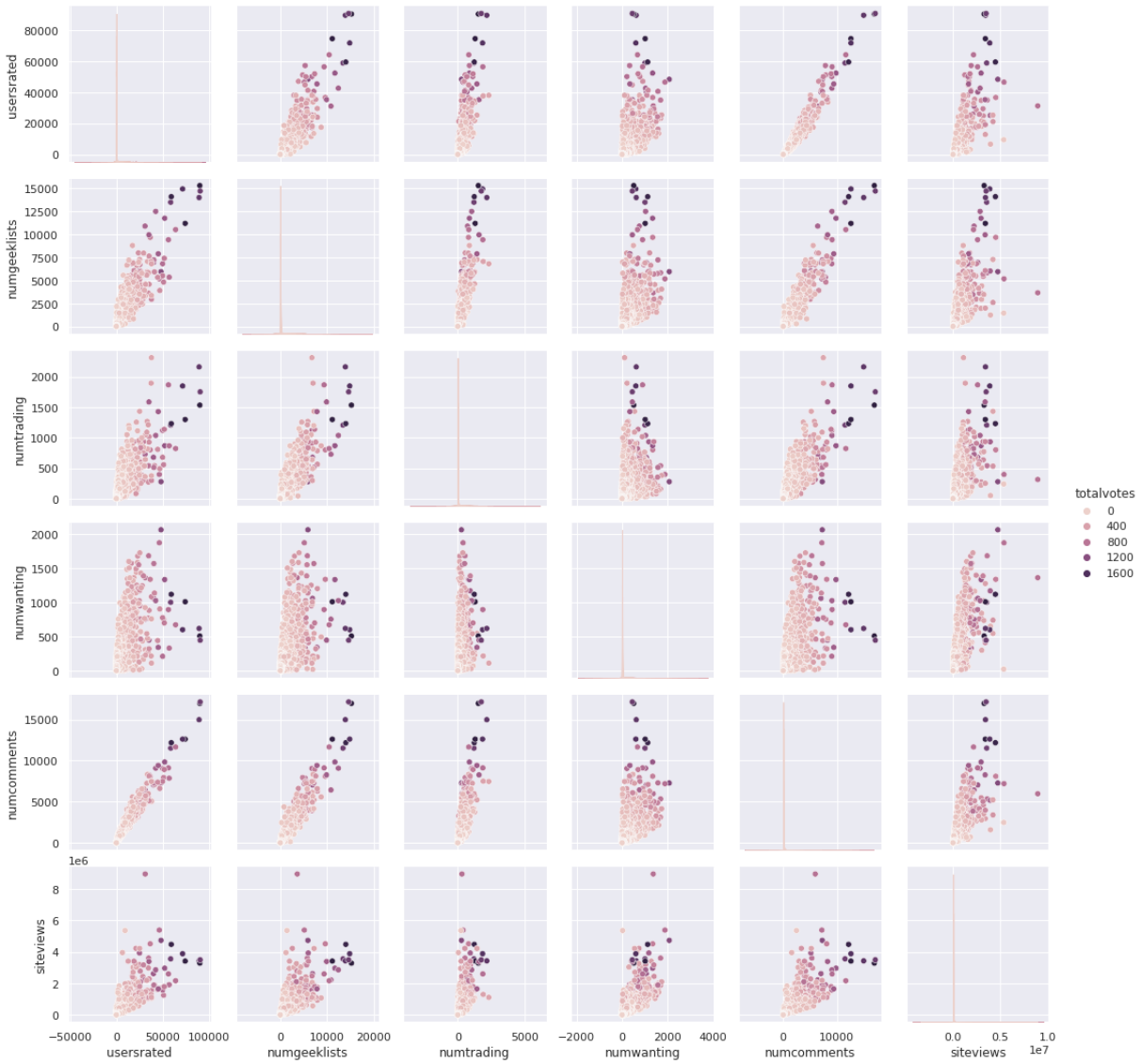


Figura 2.4: Pairplot sulle statistiche degli utenti

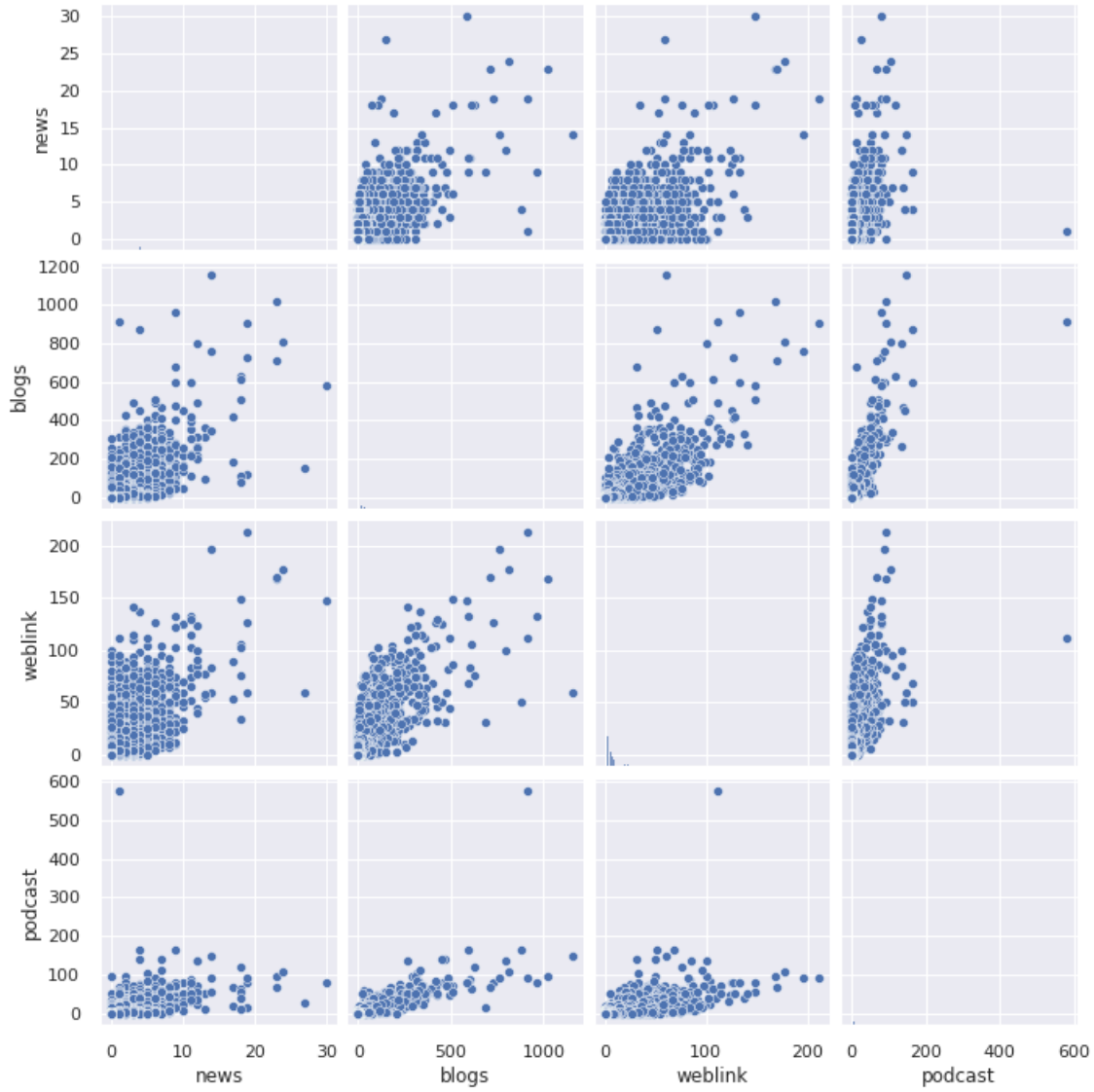


Figura 2.5: Pairplot sulla ricezione mediatica dei boardgames

Figura 2.6: Pairplot relativo alle caratteristiche dei boardgames

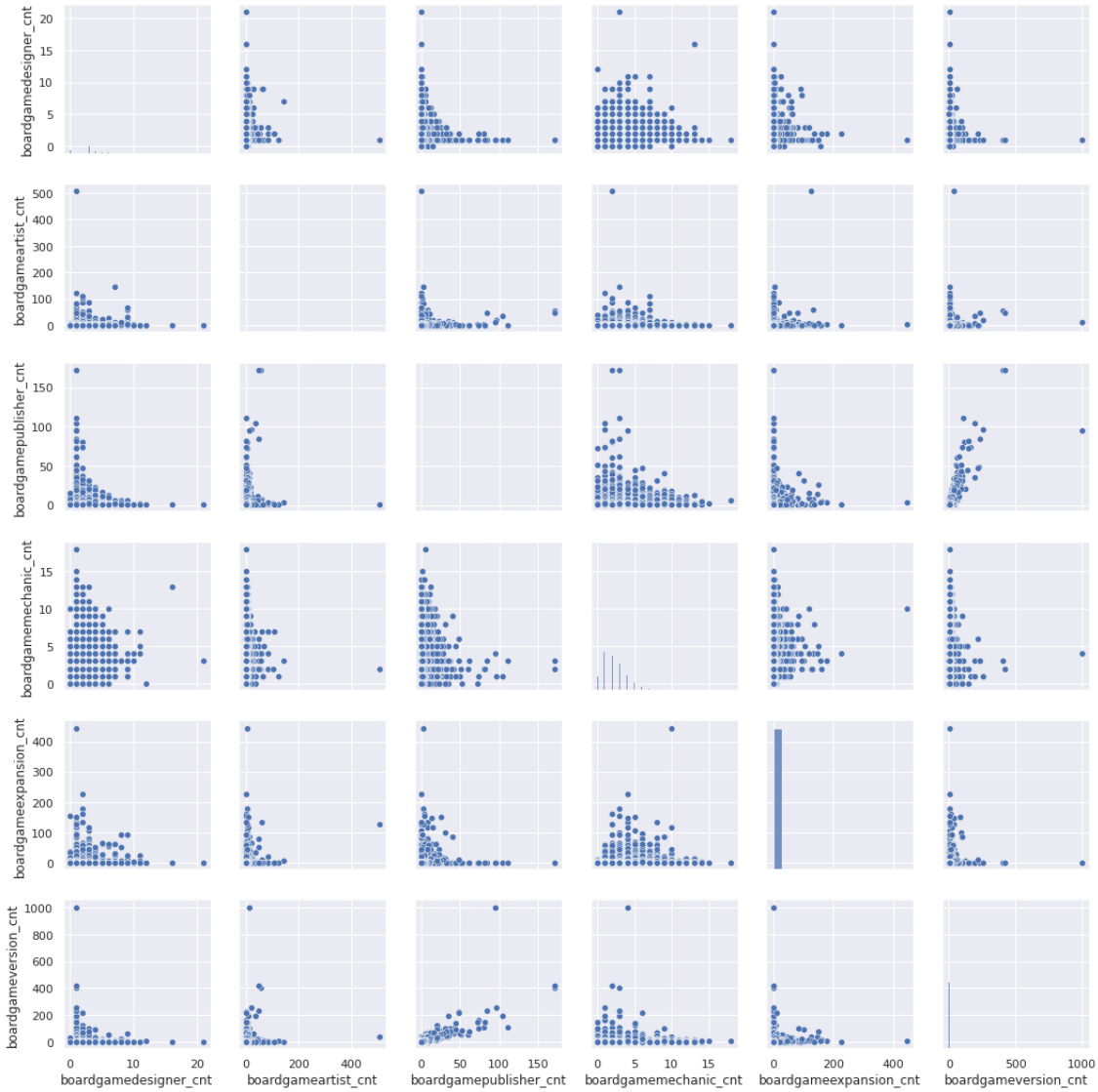


Figura 2.7: Pairplot sugli attributi relativi ai conteggi

Capitolo 3

Clustering

Il clustering è un' insieme di tecniche non supervisionate in grado di separare gli elementi di un dataset in gruppi omogenei il più dissimili tra loro. Tali gruppi sono chiamati cluster e rappresentano un insieme di record del dataset con caratteristiche "simili" gli uni dagli altri. L'obiettivo del primo task, dunque, è quello di utilizzare alcune features del dataset per cercare di separare i dati in diversi gruppi. Inoltre, i dati di input, prima di essere clusterizzati, sono stati normalizzati in modo da garantire un miglior comportamento degli algoritmi di clustering. Le tecniche utilizzate sono molteplici e verranno descritte nei prossimi paragrafi.

3.1 K-means

La prima tecnica utilizzata è il **K-means**. L'algoritmo va a minimizzare la varianza intra-gruppo dei vari cluster, i quali sono identificati da un punto medio chiamato centroide. L'algoritmo, una volta creati in maniera casuale K partizioni ognuna con un suo punto di riferimento, segue una procedura iterativa:

- Associa ogni elemento del dataset alla partizione più vicina formando K cluster.
- Calcola il centroide di ogni cluster che diventa il nuovo punto di riferimento.

La procedura viene iterata finché la posizione dei centroidi non converge.
Per trovare il numero K ottimale si è utilizzato l'**Elbow Method** (metodo del gomito): tale metodo itera il K-means per diversi valori di K ed ogni volta si calcola la somma delle distanze al quadrato tra ogni centroide ed i punti del proprio cluster. Graficando i valori di K e i valori della somma delle distanze al quadrato (distortion score), si ottiene un grafico simile a quello in figura 3.1. Il valore ottimale sarà quello in cui la curva tende a salire in maniera più consistente.

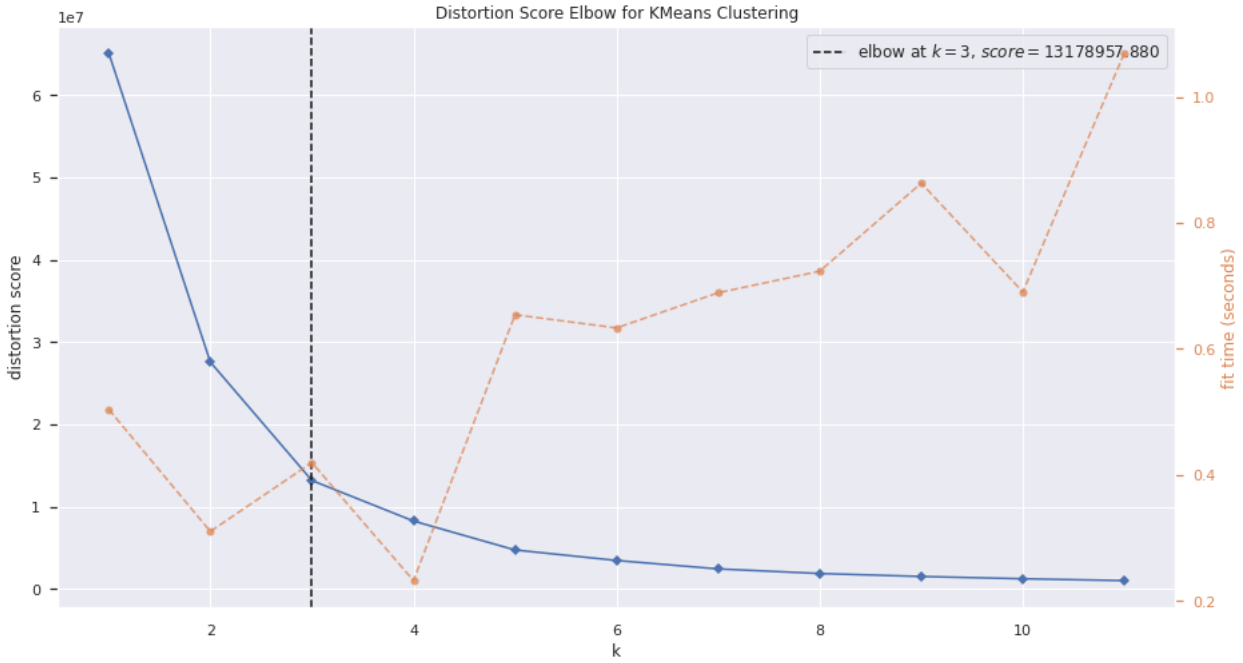


Figura 3.1: Elbow Method

La figura 3.2 mostra il risultato ottenuto dall'algoritmo andando a considerare un numero di cluster pari a 3 (come mostrato nella figura precedente) e utilizzando come features *UserRated* e *TotalVotes*. I tre cluster individuati sono stati divisi principalmente in base al voto medio degli utenti, piuttosto che al numero di recensioni, in quanto la maggior parte degli elementi aveva un numero di recensioni basso. In particolare, si può notare come i giochi con più recensioni da parte degli utenti, abbiano una valutazione più che sufficiente (superiore al 6,5/7).

Figura 3.2: Risultati del K-means

Per misurare la correttezza dei cluster individuati, si è fatto uso del coefficiente di "silhouette", il quale misura quanto un oggetto è simile agli oggetti appartenenti allo stesso cluster, rispetto a quelli contenuti in altri cluster. Per ogni punto di un cluster, il suo coefficiente di silhouette è la differenza tra la distanza media con i punti appartenenti allo stesso gruppo (coesione) e la distanza media con i punti di altri gruppi vicini (separazione). Se questa differenza è negativa, il punto è mediamente più vicino a uno dei gruppi vicini, piuttosto che al proprio: è quindi scarsamente classificato. Viceversa, se questa differenza è positiva, il punto è mediamente più vicino al suo gruppo rispetto a uno dei gruppi vicini: è quindi ben classificato. La figura 3.3 mostra come la silhouette media sia pari a 0.5, e che molti degli elementi appartenenti ai vari cluster superano tale media; dunque gli elementi vengono ben rappresentati dal cluster di appartenenza. Nel secondo cluster, in verde, che rappresenta il gruppo con recensioni insufficienti, ci sono alcuni punti con silhouette negativa. Questi rappresentano punti borderline, che verrebbero meglio rappresentati da altri cluster.

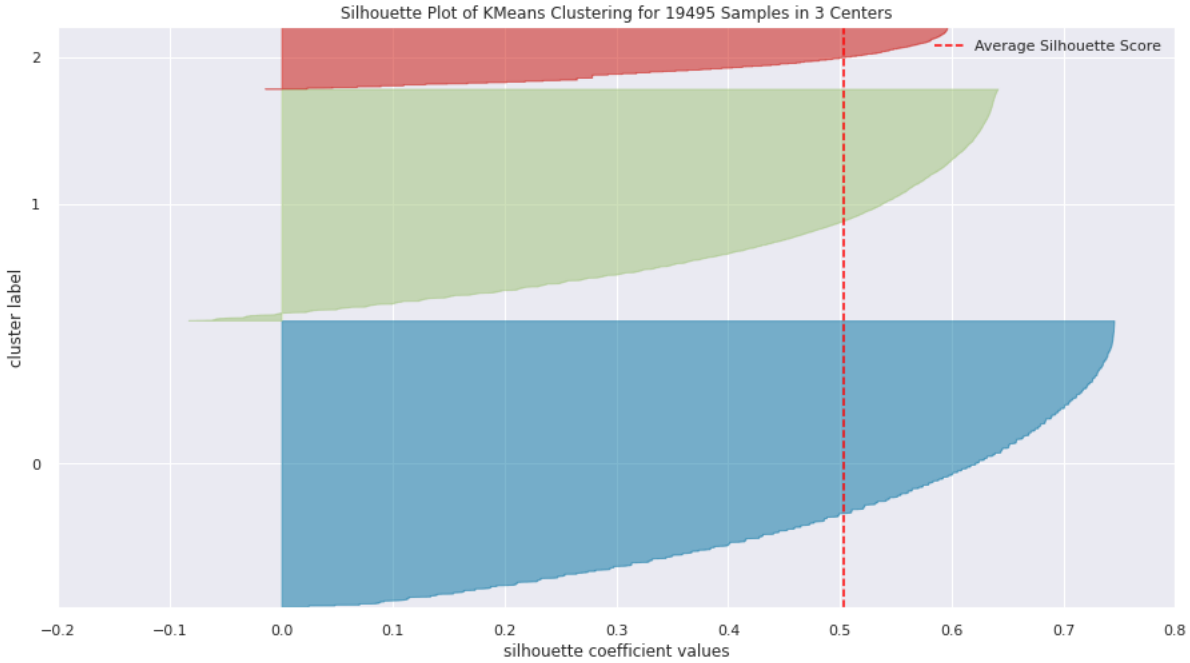


Figura 3.3: Silhouette Score

3.2 PCA Clustering

Il metodo **PCA (Principal Component Analysis)** permette di effettuare un clustering n-dimensionale andando a ridurre il numero di variabili misurate, mantenendo, però, il più possibile intatta la conoscenza iniziale. Nel nostro caso il numero di variabili sono state ridotte da 4 a 2. Le features utilizzate sono quelle relative alla ricezione mediatica dei boardgames (*news, blogs, weblink, podcast*). L'algoritmo utilizzato è sempre il K-means, di cui si è calcolato il K ottimale tramite l'Elbow Method.

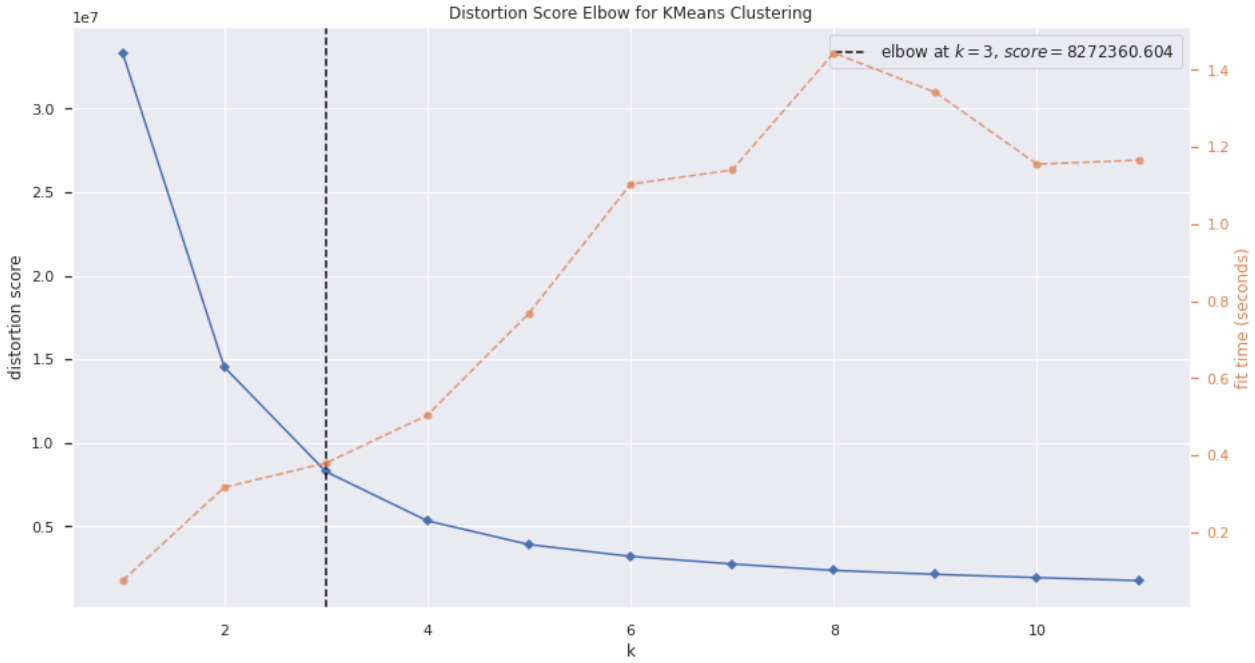


Figura 3.4: Elbow Method

I risultati dell'algoritmo di clustering, in questo caso, sono poco espressivi dal punto di vista semantico, in quanto abbiamo ridotto le dimensioni del problema da 4 a 2. Per questo motivo, per valutare la correttezza dei risultati ottenuti, si è fatto nuovamente utilizzo del coefficiente di silhouette: in questo caso solo uno dei cluster individuati supera il valor medio pari a 0.68, mentre i restanti due mostrano molti punti con silhouette negativa. Infatti, la figura 3.5 mostra i tre cluster molto vicini tra loro e dunque non c'è una separazione "netta" fra i vari gruppi individuati.

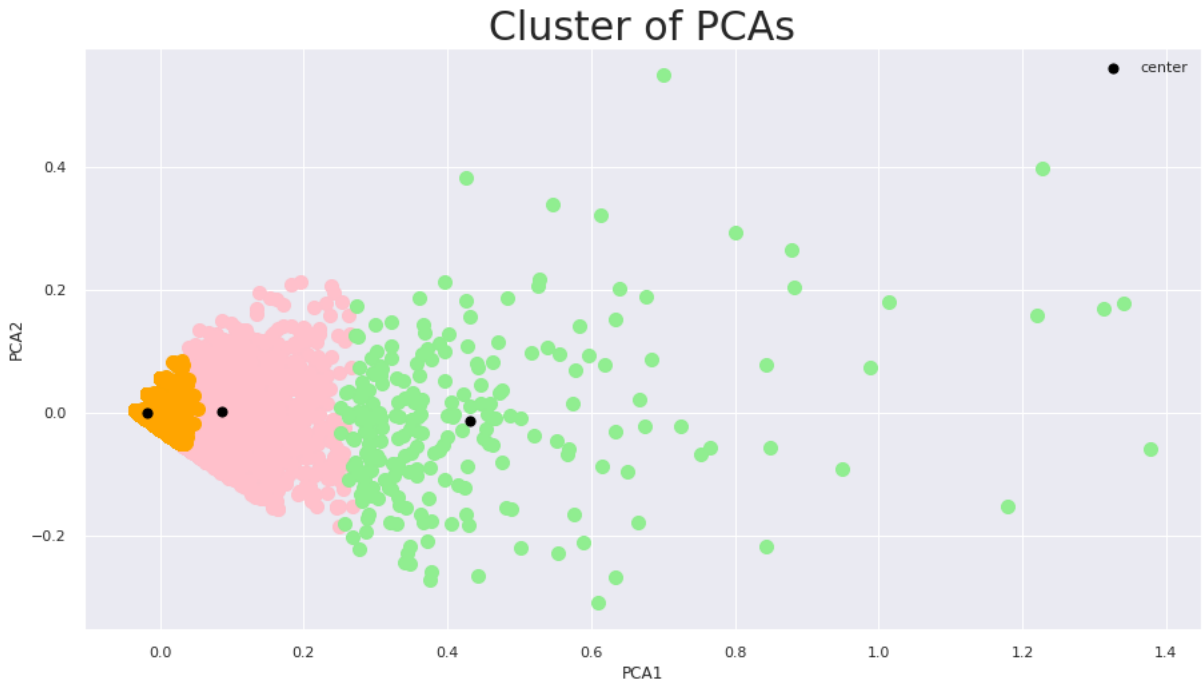


Figura 3.5: Risultati PCA

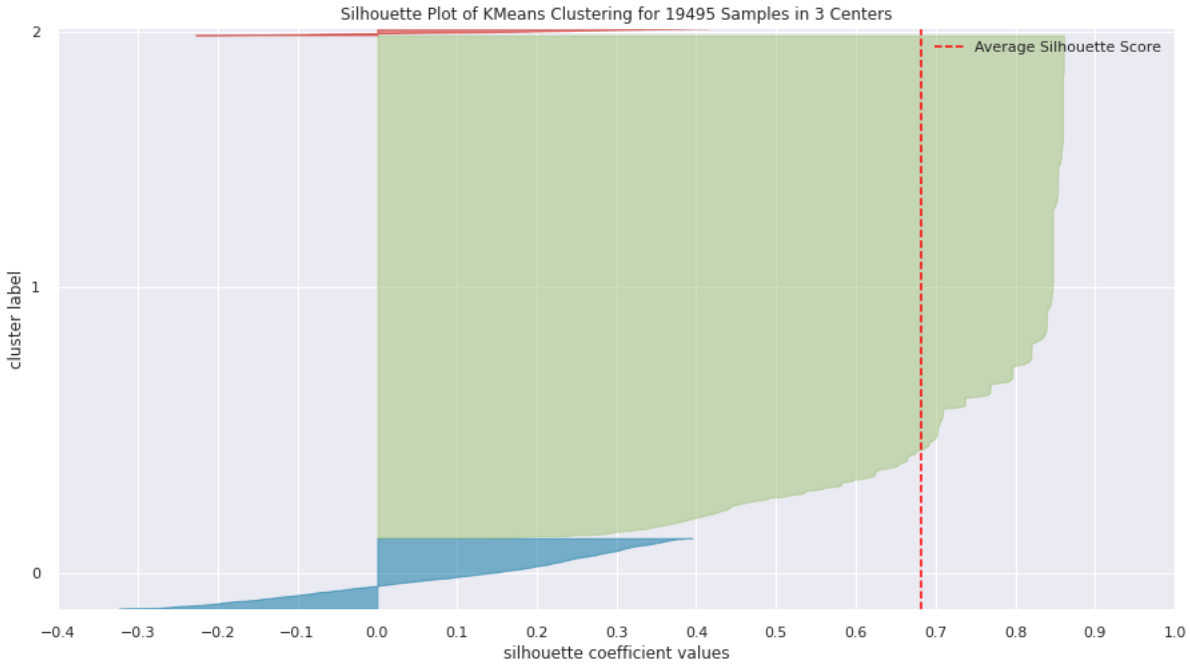


Figura 3.6: Silhouette Score

3.3 Hierarchical Clustering

Il clustering gerarchico è un approccio che mira a costruire una gerarchia di cluster. Le strategie per il clustering gerarchico sono tipicamente di due tipi:

- *Agglomerativo*: si tratta di un approccio "bottom-up" (dal basso verso l'alto), in cui si parte dall'inserimento di ciascun elemento in un cluster differente e si procede, quindi, all'accorpamento graduale di cluster a due a due.
- *Divisivo*: si tratta di un approccio "top-down" (dall'alto verso il basso), in cui tutti gli elementi si trovano inizialmente in un singolo cluster, che viene via via suddiviso ricorsivamente in sotto-cluster.

Per decidere quali cluster devono essere combinati (approccio agglomerativo) o quale cluster deve essere suddiviso (approccio divisivo), è necessario definire una misura di dissimilarità tra cluster. Nella maggior parte dei metodi di clustering gerarchico si fa uso di metriche specifiche, che quantificano la distanza tra coppie di elementi, e di un criterio di collegamento, che specifica la dissimilarità di due insiemi di elementi (cluster) come funzione della distanza, a coppie, tra elementi nei due insiemi. Nel nostro caso si è utilizzato l'approccio agglomerativo. La metrica scelta è la *distanza euclidea*, mentre, come criterio di collegamento, si è utilizzato il *ward*, che minimizza la varianza dei cluster uniti.

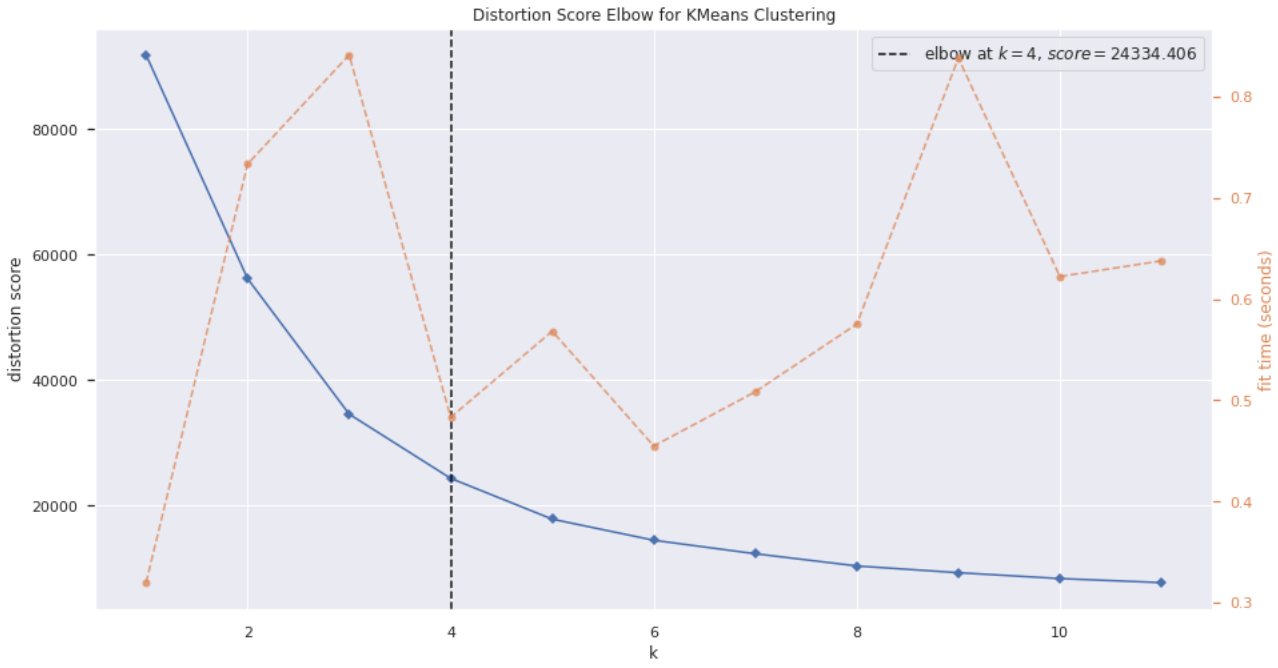


Figura 3.7: Elbow Method

Le features utilizzate in questo task sono il numero minimo e massimo di giocatori consigliati dalla community (*Min_ community* e *Max_ community*). Anche in questo caso si è utilizzato l'Elbow Method per definire il numero ottimale di cluster, che è risultato essere pari a 4. La figura 3.8 mostra i quattro gruppi individuati dall'algoritmo, i quali sono stati etichettati in base al numero di giocatori previsti. Il calcolo è stato effettuato tenendo in considerazione che le features non normalizzate avevano entrambe un valore compreso tra 0 e 32.

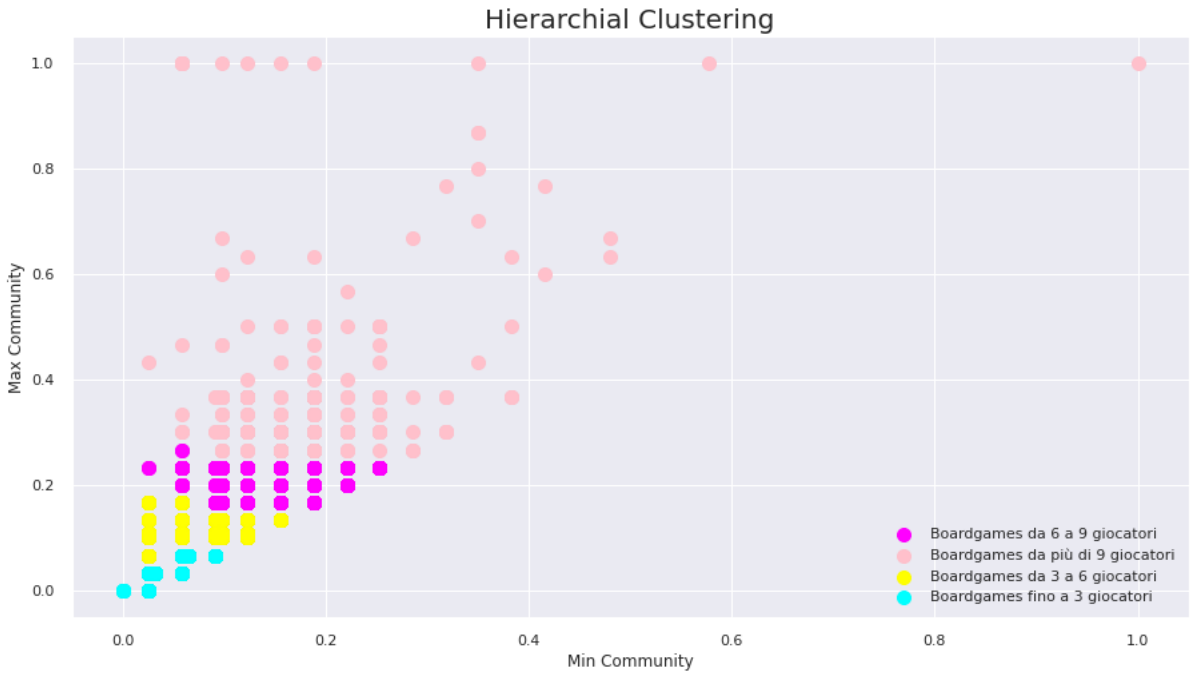


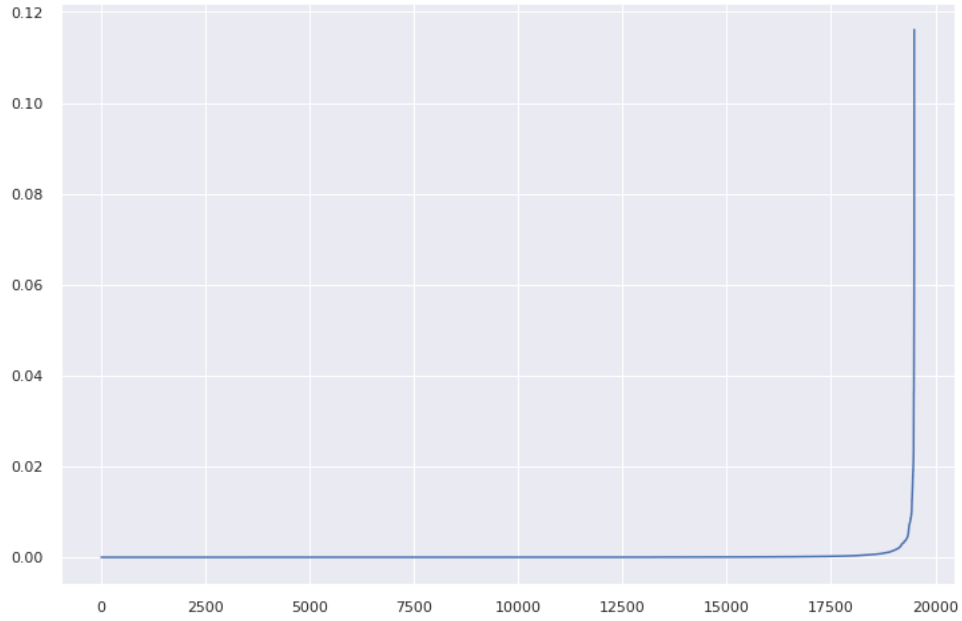
Figura 3.8: Risultati del Hierarchical Clustering

3.4 DBSCAN

Il DBSCAN è un algoritmo che usa una definizione di cluster basata sulla nozione di densità. A tale scopo, utilizza due parametri: ϵ e n. Infatti, ogni punto del dataset è classificato in:

- Punti interni: che hanno almeno n punti di distanza inferiore a ϵ;
- Punti di confine: che non sono interni, ma hanno un punto interno a distanza inferiore a ϵ;
- Rumore: punti né interni né di confine.

Una volta terminata la classificazione dei punti del dataset, i punti interni adiacenti sono considerati appartenenti allo stesso cluster; i punti di confine vengono associati ai cluster dei punti interni vicini; mentre il rumore viene ignorato. Per trovare il valore ottimale di ϵ, si definisce un grafico, in cui sulle ascisse vengono rappresentati tutti i punti del dataset, mentre sulle ordinate la distanza che separa ciascuno dall'ennesimo elemento più vicino. Il grafico viene quindi ordinato e reso monotono crescente. Il valore di ϵ, analogamente a quanto visto con l'Elbow Method, viene determinato dal punto in cui si presenta una netta variazione di pendenza. Nel nostro caso si è cercato di clusterizzare il dataset usando come features *BoardgameExpansion_cnt* e *BoardgameVersion_cnt*. La figura seguente mostra il grafico sopra descritto ricavato dal dataset preso in esame, il quale, tuttavia, presenta una funzione anomala. Questa, infatti, è caratterizzata inizialmente da un profilo costante per poi terminare con un picco improvviso. Questa forma particolare determina un valore di ϵ estremamente basso, che porta inevitabilmente a classificare la quasi totalità dei punti come rumore, rendendo quindi inutile la procedura.



N.B Il DBSCAN è stato utilizzato per cercare di clusterizzare anche altre features, ma con scarsi risultati.

Capitolo 4

Classificazione

La classificazione è una tecnica di apprendimento supervisionato che cerca di associare le diverse entità di un dataset ad una classe. In pratica, si definisce un profilo descrittivo per ogni classe, in modo tale da assegnare oggetti di classe ignota alla classe appropriata. Il task di classificazione si divide in due fasi:

- Nella prima fase si divide il dataset in due gruppi chiamati training e test set. Il primo gruppo sarà utilizzato nella fase di addestramento, la quale specificherà un modello.
- Nella seconda fase il modello sarà poi utilizzato per prevedere le classi del test set, in modo da verificare il suo corretto funzionamento.

La libreria *scikit-learn* mette a disposizione diversi algoritmi di classificazione, insieme ad altre funzioni. I modelli utilizzati nel nostro progetto sono i seguenti:

- **Regressione Logistica**
- **Albero di Decisione**
- **SVC**
- **Random Forest**
- **Ada Boost**
- **Gradient Boost**

Prima di iniziare il task di classificazione, si è generata un'heatmap che mettesse in risalto le varie correlazioni tra le diverse variabili utilizzate. Inoltre, tutte le features sono state normalizzate ed alcune di esse addirittura eliminate.

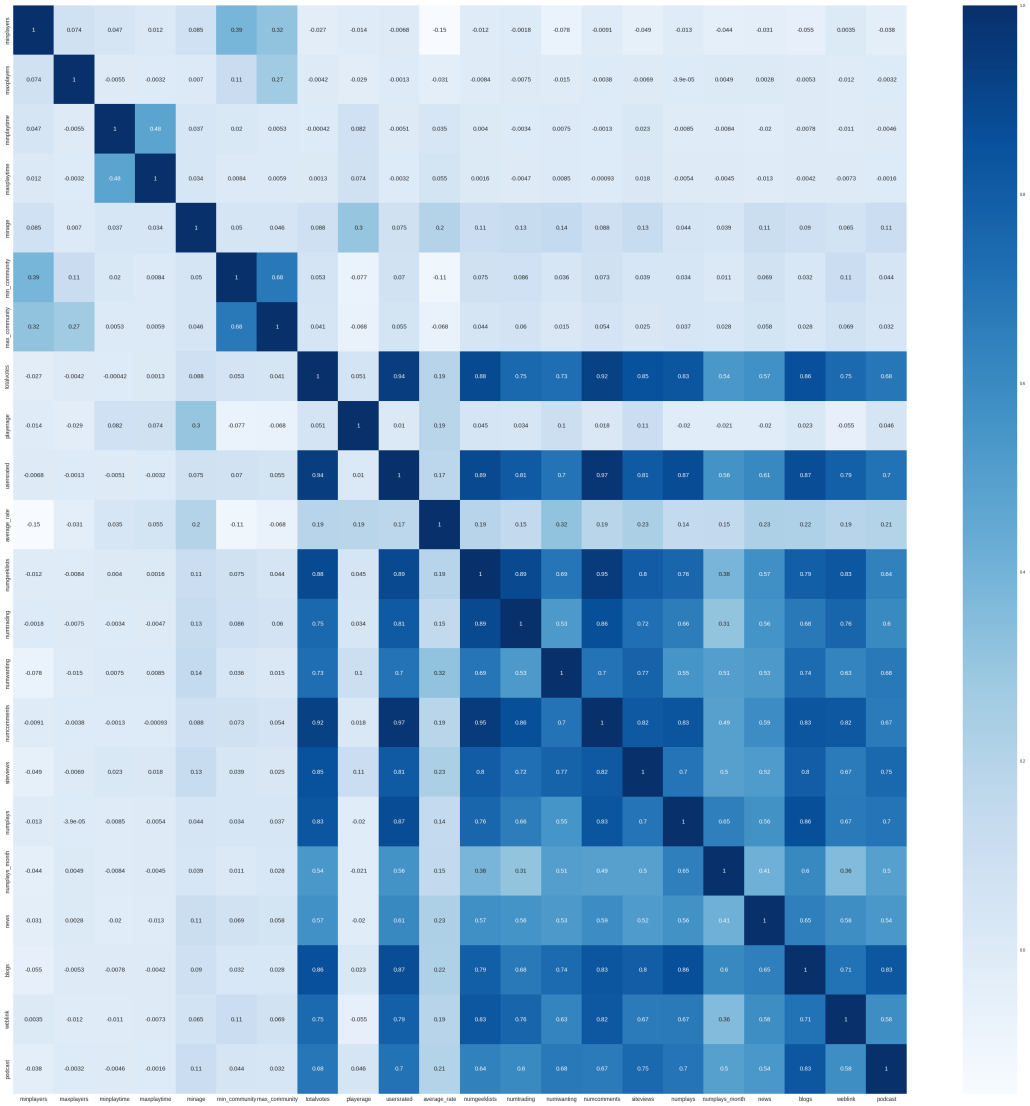


Figura 4.1: Heatmap delle features

4.1 Classificazione per Categoria

Nel primo task si è cercato di classificare i vari boardgames in base alla categoria, rappresentata dall'attributo di tipo stringa *BoardgameCategory*. In particolare, sono state individuate 83 classi, le quali, come è possibile vedere dall'immagine 4.2, risultano altamente sbilanciate. Tale sbilanciamento è dovuto principalmente al fatto che, in realtà, i boardgames possono essere classificati in una lista di categorie, che però sono state ridotte ad una sola (la prima della lista) durante la fase di ETL. Per migliorare i risultati della classificazione, sono state eliminate le classi con minore numerosità. In particolare, si sono eliminate tutte quelle classi rappresentate da un numero di record minore di 500. In questo modo le classi sono state diminuite a 12.

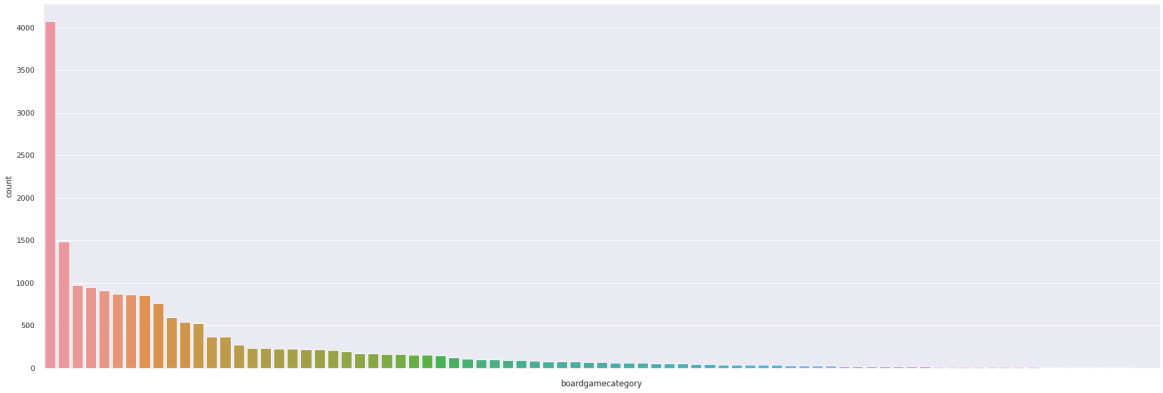


Figura 4.2: Distribuzione delle categorie

Il problema delle classi sbilanciate ha portato a dei risultati poco convincenti come mostrato nella figura 4.3.

```
Accuracy: 0.36   ---> LogisticRegression
Accuracy: 0.32   ---> DecisionTreeClassifier
Accuracy: 0.36   ---> SVC
Accuracy: 0.4    ---> RandomForestClassifier
Accuracy: 0.39   ---> AdaBoostClassifier
Accuracy: 0.47   ---> GradientBoostingClassifier
```

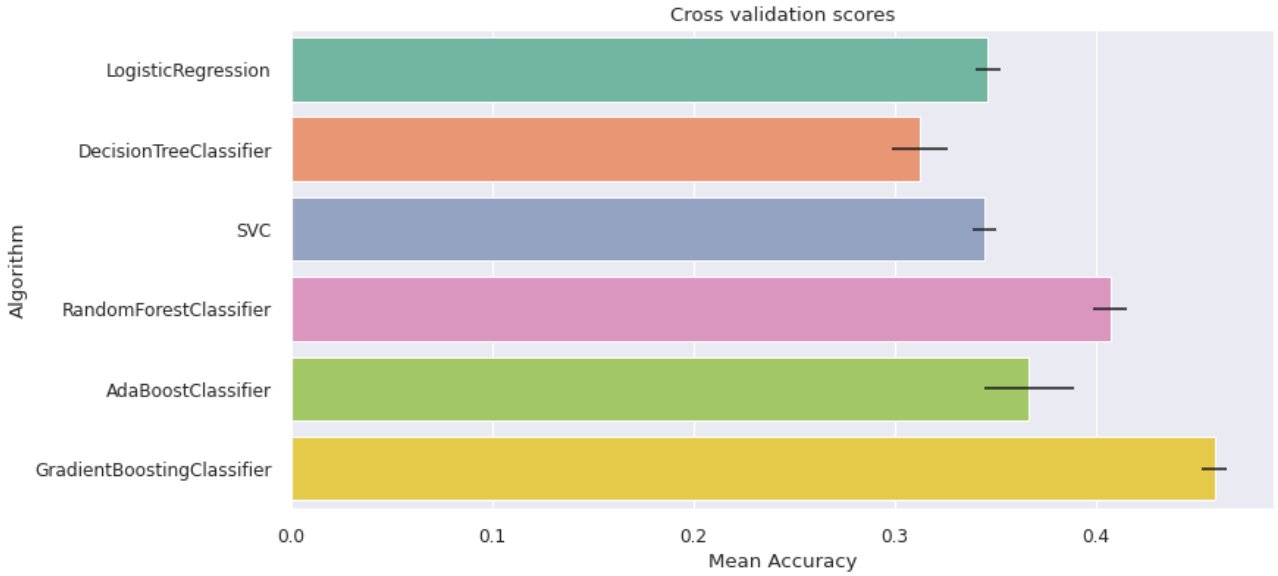


Figura 4.3: Risultati ottenuti prendendo in considerazione classi con almeno 500 record

Questi risultati sono dovuti principalmente al gran numero di classi presenti nel dataset e dal loro elevato sbilanciamento. Infatti, nonostante si è ridotto il numero di classi a 12, queste risultano ancora molto sbilanciate, come mostrato nella figura sottostante.

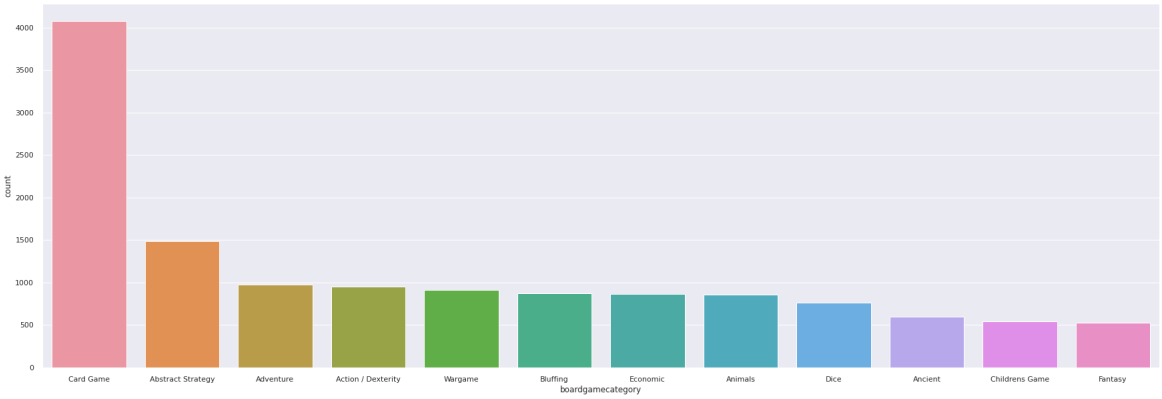


Figura 4.4: Distribuzione delle categorie dopo il filtraggio

Le matrici di confusione mostrano come la classe *Card Game* venga, per la maggior parte dei casi, classificata correttamente, in quanto è la più numerosa; mentre le restanti classi vengono classificate erroneamente sempre in *Card Game*.

Figura 4.5: Matrici di confusione per BoardgameCategory

4.2 Classificazione per Language Dependance

Nel secondo task si sono classificati i boardgames in base all'attributo numerico *LanguageDependance*, il quale è costituito da 6 classi numeriche crescenti che indicano la quantità di testo da leggere nel gioco e, quindi, la necessità di conoscere la lingua in cui il boardgame è pubblicato. Anche in questo caso le classi risultano sbilanciate, come è chiaramente visibile dalla figura 4.6, e quindi i risultati ottenuti sono appena sufficienti, con un'accuratezza dei modelli che di media supera di poco lo 0.5.

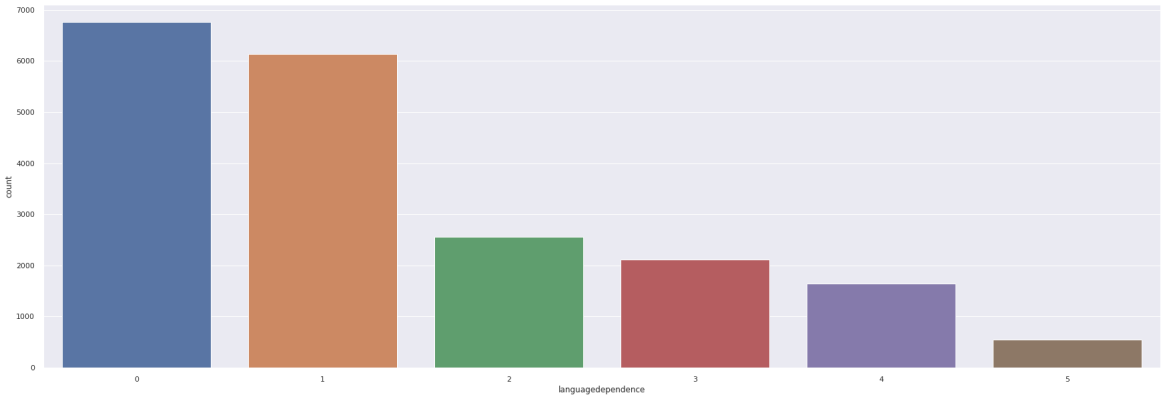


Figura 4.6: Distribuzione di LanguageDependance

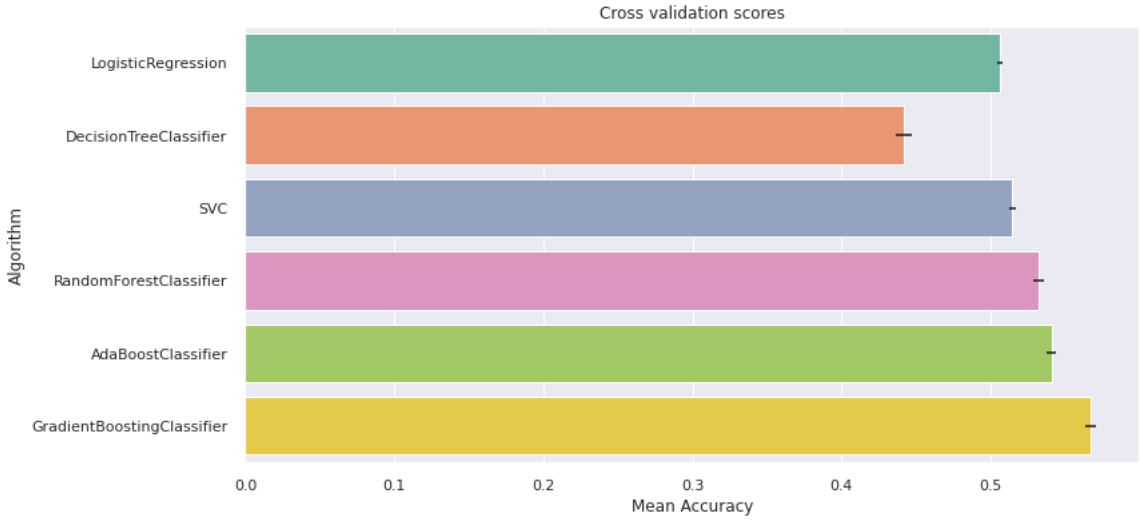


Figura 4.7: Risultati della classificazione per LanguageDependance

Le matrici di confusione, infatti, mostrano come le due classi più numerose siano tendenzialmente classificate correttamente, mentre le restanti quattro, con un numero di record molto minore, siano erroneamente classificate nelle prime due.

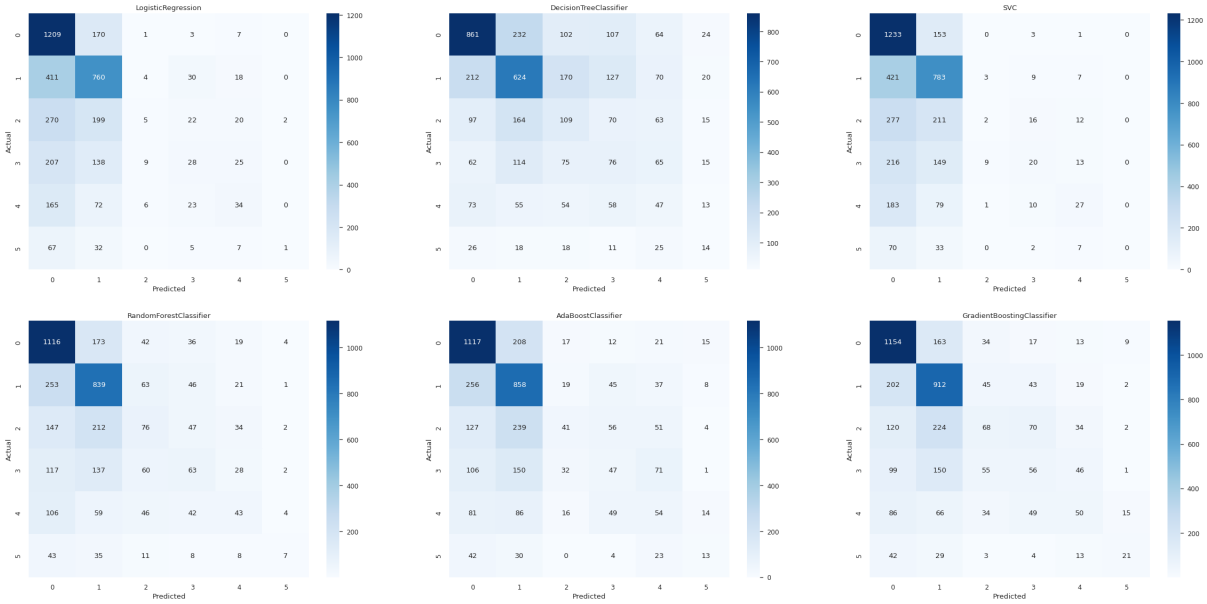


Figura 4.8: Matrici di confusione per LanguageDependance

La valutazione dei vari classificatori ci permette di definire un'heatmap, nella quale possiamo mostrare la correlazione tra i modelli. Questa va, quindi, a mostrare i

classificatori che ci restituiscono predizioni simili tra loro. Il grafico mostra come ci sia una buona correlazione tra l'*SVC* e la *Logistic Regression* che forniscono predizioni simili.

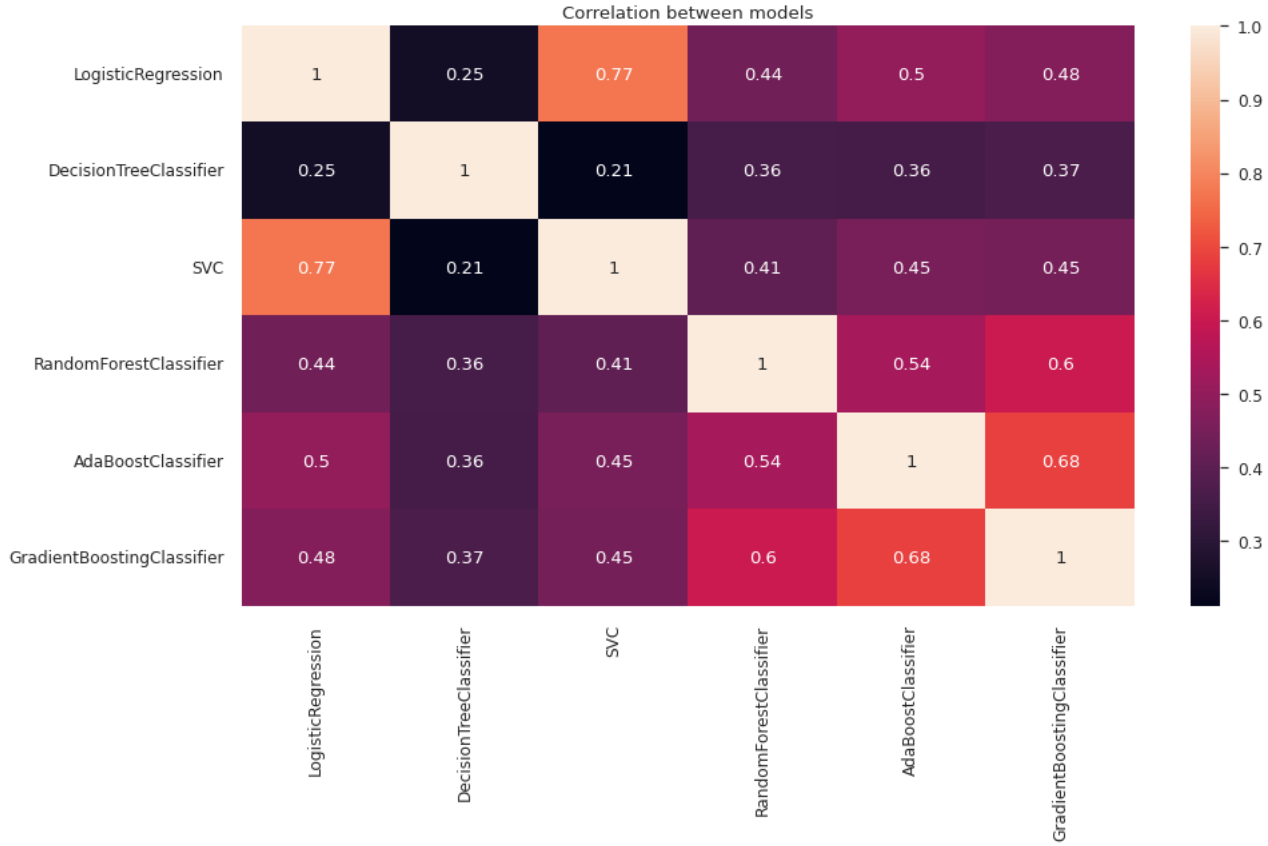


Figura 4.9: Correlazione tra i modelli di classificazione

Per migliorare i risultati della classificazione, si è deciso, in questo caso, di bilanciare le classi, in modo tale che risultassero equiprobabili, cioè rappresentate dallo stesso numero di record. Come si può notare, i modelli ottenuti in questo modo sono migliori rispetto a quelli ottenuti effettuando un normale training. In particolare, gli algoritmi di *Decision Tree* e *Random Forest* sono quelli più convincenti con valori di accuratezza superiori allo 0.8.

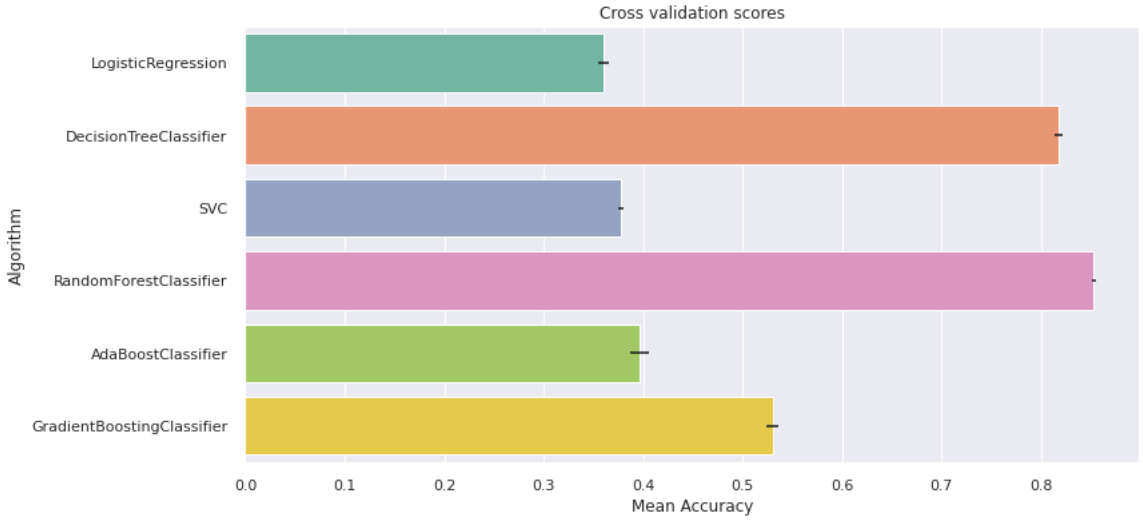


Figura 4.10: Risultati della classificazione per LanguageDependance dopo il bilanciamento

Le matrici di confusione mostrano, in questo caso, una buona classificazione per quanto riguarda gli algoritmi di *Decision Tree* e *Random Forest*; mentre, negli altri casi, c'è un leggero miglioramento della classificazione delle altre classi, rispetto a quella effettuata precedentemente senza bilanciamento.

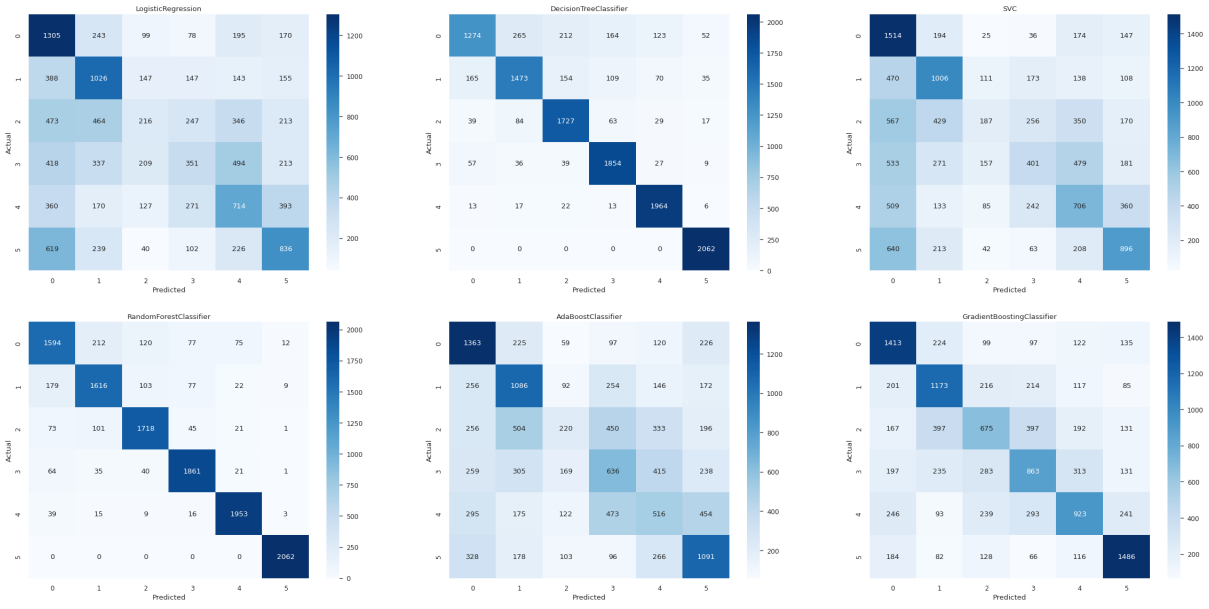


Figura 4.11: Matrici di confusione per LanguageDependance dopo il bilanciamento

Anche in questo caso si è analizzata la correlazione tra i vari modelli. Si può notare nuovamente una buona correlazione tra l'*SVC* e la *Logistic Regression*; mentre,

rispetto alla classificazione precedente, è presente anche un'elevata correlazione tra il *Decision Tree* e il *Random Forest*.

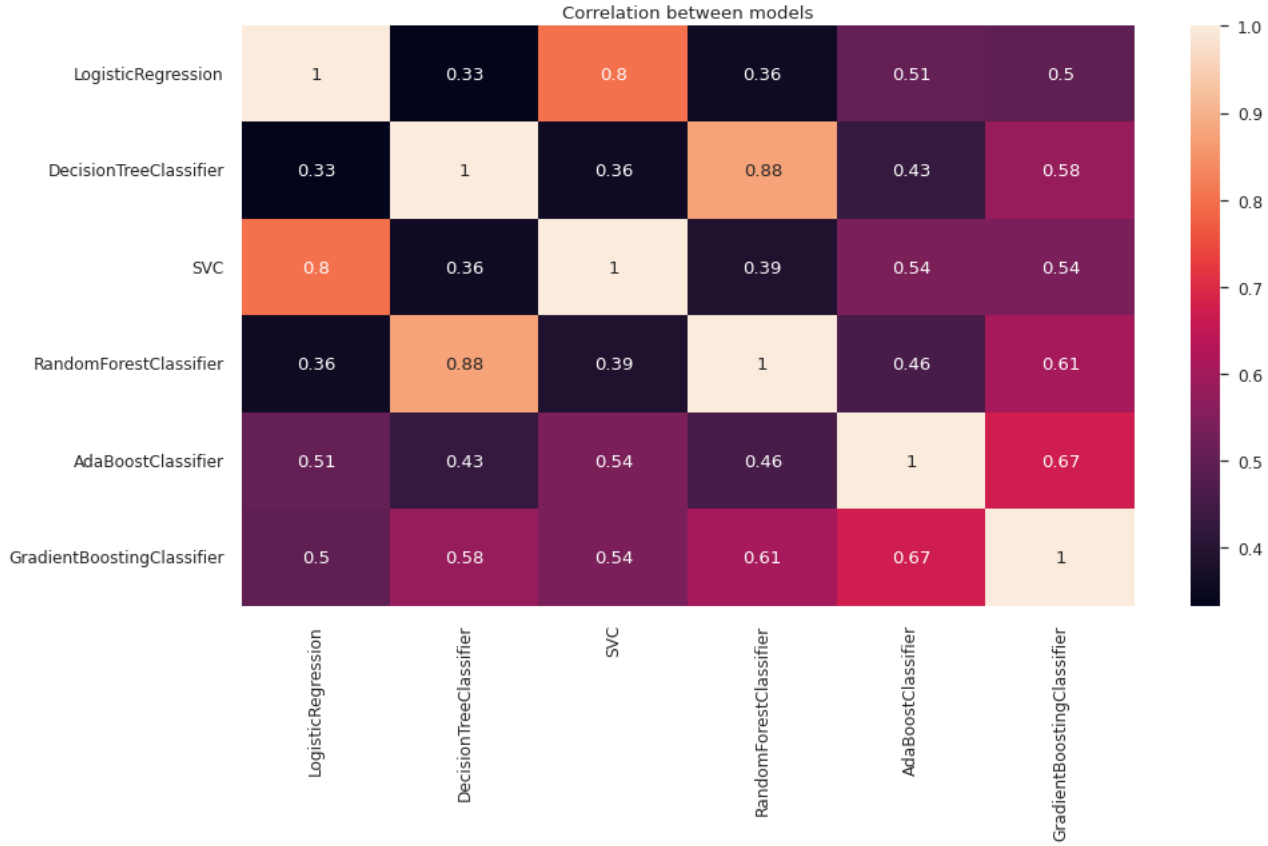


Figura 4.12: Correlazione tra i modelli di classificazione dopo il bilanciamento