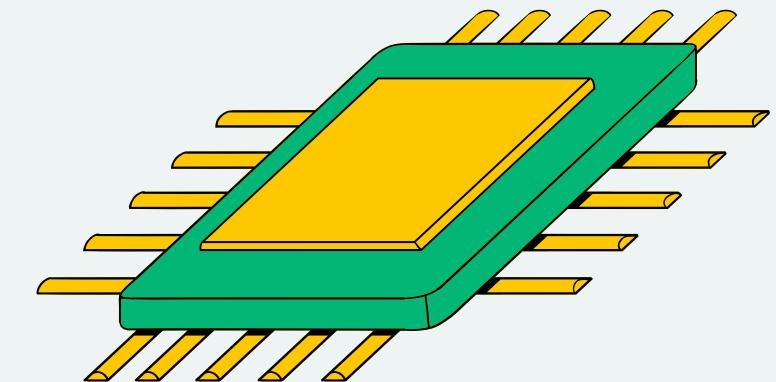
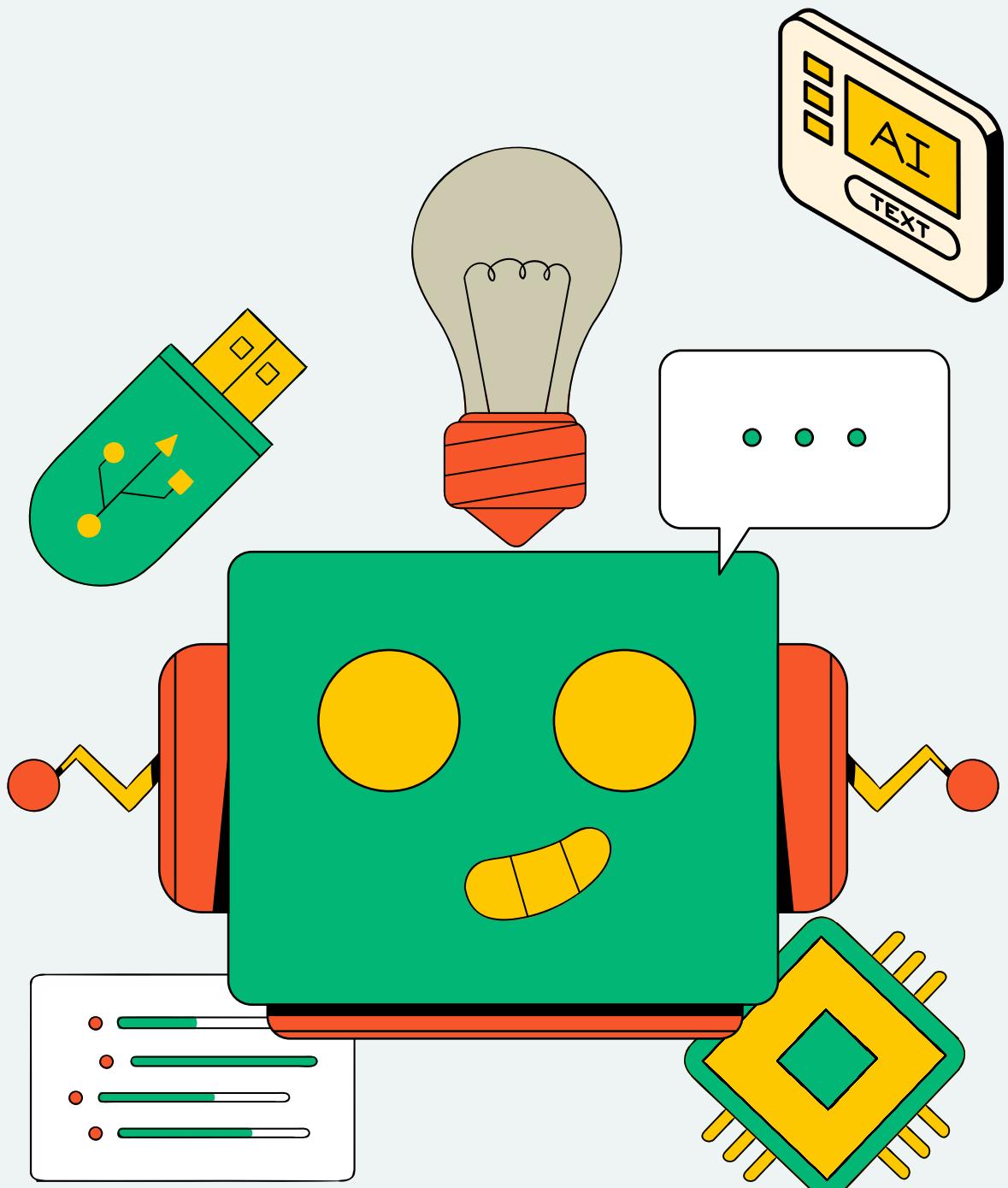


POPULAR TV SHOWS DATASET (TMDB)

PRESENTAZIONE PROGETTO

PRESENTATO DA:

ALESSANDRO CARELLA



INDICE

- Esplorazione Dataset
 - Analisi iniziale, statistiche, correlazioni
- Preprocessing
 - Feature engineering, cleaning, scaling, bilanciamento
- Training
 - Model selection, fine tuning
(classificazione + regressione)
- Metriche
 - Performance test set, residui, validation
- Extra – Predizione Testuale
- Extra – XAI (LIME + SHAP)
 - Interpretabilità, feature importance



INTRODUZIONE

Il progetto si pone l'obiettivo di sviluppare una soluzione per la predizione del voto medio di serie TV popolari utilizzando tecniche di Machine Learning.

- Dataset: [10k Popular TV Shows](#)
- Target: **vote_average**

L'obiettivo dato dalla traccia era predire la colonna target **vote_average**.

Essendo i valori all'interno della colonna numeri reali compresi fra 0 e 10, ho deciso di affrontare la task sia come un problema di classificazione che regressione.



PRIMA ANALISI DEL DATASET

Il dataset contiene informazioni su 10000 serie TV popolari con 16 features:

'adult', 'backdrop_path', 'genre_ids', 'id', 'origin_country',
'original_language', 'original_name', 'overview', 'popularity',
'poster_path', 'first_air_date', 'name', 'vote_average',
'vote_count'

Durante l'analisi e il preprocessing del dataset ho effettuato diverse modifiche fino ad ottenere le seguenti features:

popularity, vote_count, year, genre, continent,
lang_macroarea, vote_average



MODIFICHE DATASET

COLONNE RIMOSSE

- adult: ha solo valori False e nessun True
- backdrop_path: "File path to the show's backdrop image on TMDb" (kaggle)
- poster_path: "File path to the show's poster image on TMDb" (kaggle)
- original_name, id: non utili al fine predittivo

COLONNE TRASFORMATE

- first_air_date → estratto anno di uscita → year
- genre_ids → da 35 generi distinti a 6 cluster
- original_language → lingue mappate 5 macroaree di origine
- origin_country → paesi mappati ai 6 continenti

DATI RIMOSSI E VARIE

- Rimozione duplicati
- One hot encoding colonne categoriche
- Split dataset in:
 - Train (70%)
 - Validation (10%)
 - Test (20%)
- Rimozione Outliers
- Normalizzazione
- Bilanciamento classi (classificazione)



CLUSTERING GENERI

Recupero mapping dei generi utilizzando l'API di TMDB (solo alcuni sono presenti nel dataset)

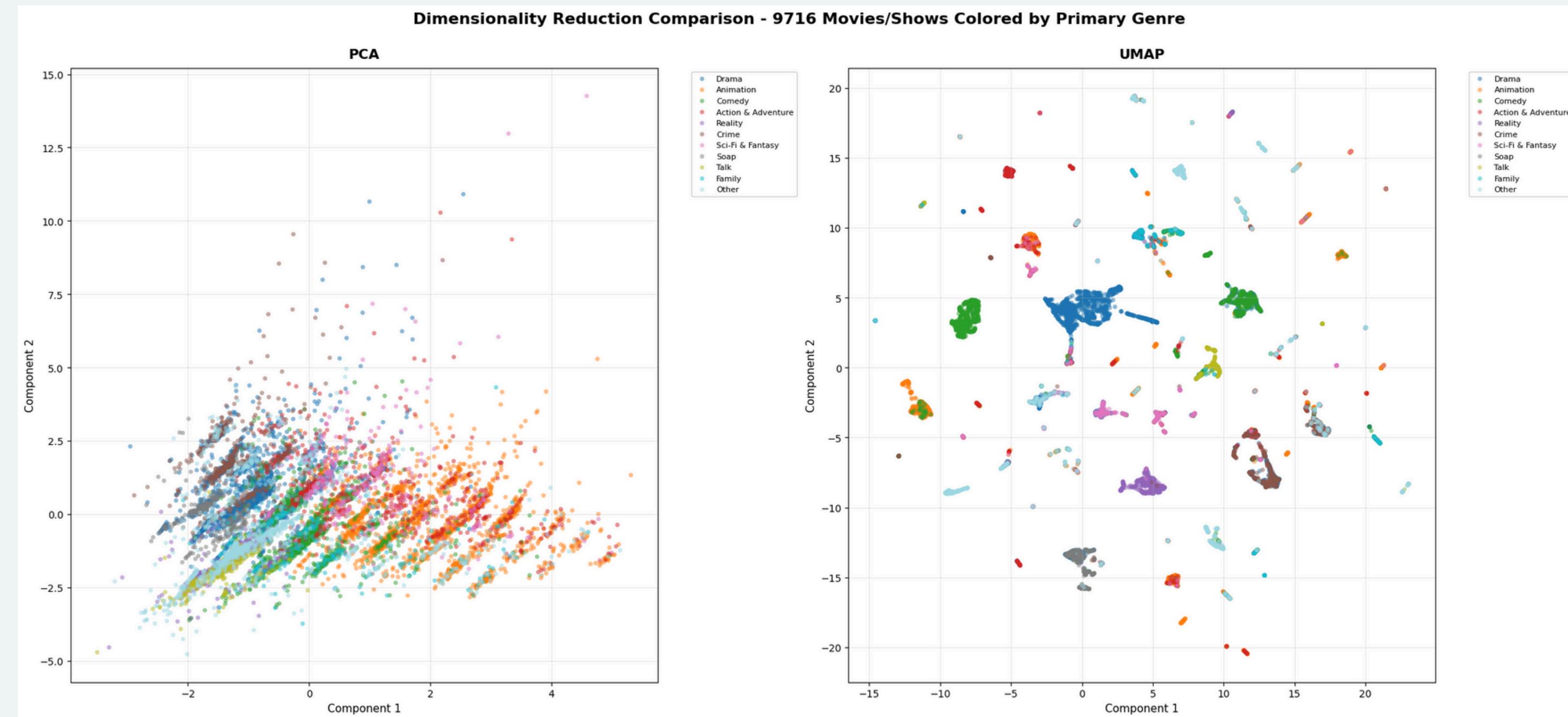
```
1 try:
2     endpoints = ["movie", "tv"]
3
4     headers = {
5         "accept": "application/json",
6         "Authorization": f"Bearer {API_KEY_ACCESS_TOKEN}"
7     }
8
9     genreMapping = []
10    for endpoint in endpoints:
11        url = f'https://api.themoviedb.org/3/genre/{endpoint}/list'
12
13        response = requests.get(url, headers=headers)
14        genreMapping.extend(response.json()["genres"])
15
16    # rimuove id duplicati
17    genreMapping = [dict(t) for t in {tuple(d.items()) for d in genreMapping}]
18    print (len(genreMapping), genreMapping)
19 except Exception as e:
20     print ("L'API non ha funzionato per il seguente motivo:", str(e))
21
```

```
1 genreMapping = [
2     {'id': 10764, 'name': 'Reality'},
3     {'id': 10768, 'name': 'War & Politics'},
4     {'id': 10765, 'name': 'Sci-Fi & Fantasy'},
5     {'id': 14, 'name': 'Fantasy'},
6     {'id': 35, 'name': 'Comedy'},
7     {'id': 18, 'name': 'Drama'},
8     {'id': 16, 'name': 'Animation'},
9     {'id': 10766, 'name': 'Soap'},
10    {'id': 28, 'name': 'Action'},
11    {'id': 80, 'name': 'Crime'},
12    {'id': 27, 'name': 'Horror'},
13    {'id': 53, 'name': 'Thriller'},
14    {'id': 37, 'name': 'Western'},
15    {'id': 10763, 'name': 'News'},
16    {'id': 10751, 'name': 'Family'},
17    {'id': 10762, 'name': 'Kids'},
18    {'id': 12, 'name': 'Adventure'},
19    {'id': 99, 'name': 'Documentary'},
20    {'id': 9648, 'name': 'Mystery'},
21    {'id': 10402, 'name': 'Music'},
22    {'id': 10749, 'name': 'Romance'},
23    {'id': 10752, 'name': 'War'},
24    {'id': 10767, 'name': 'Talk'},
25    {'id': 10759, 'name': 'Action & Adventure'},
26    {'id': 10770, 'name': 'TV Movie'},
27    {'id': 36, 'name': 'History'},
28    {'id': 878, 'name': 'Science Fiction'}
29]
```



CLUSTERING GENERI

Proiezione dei dati in PCA e UMAP con punti colorati per genere



CLUSTERING GENERI

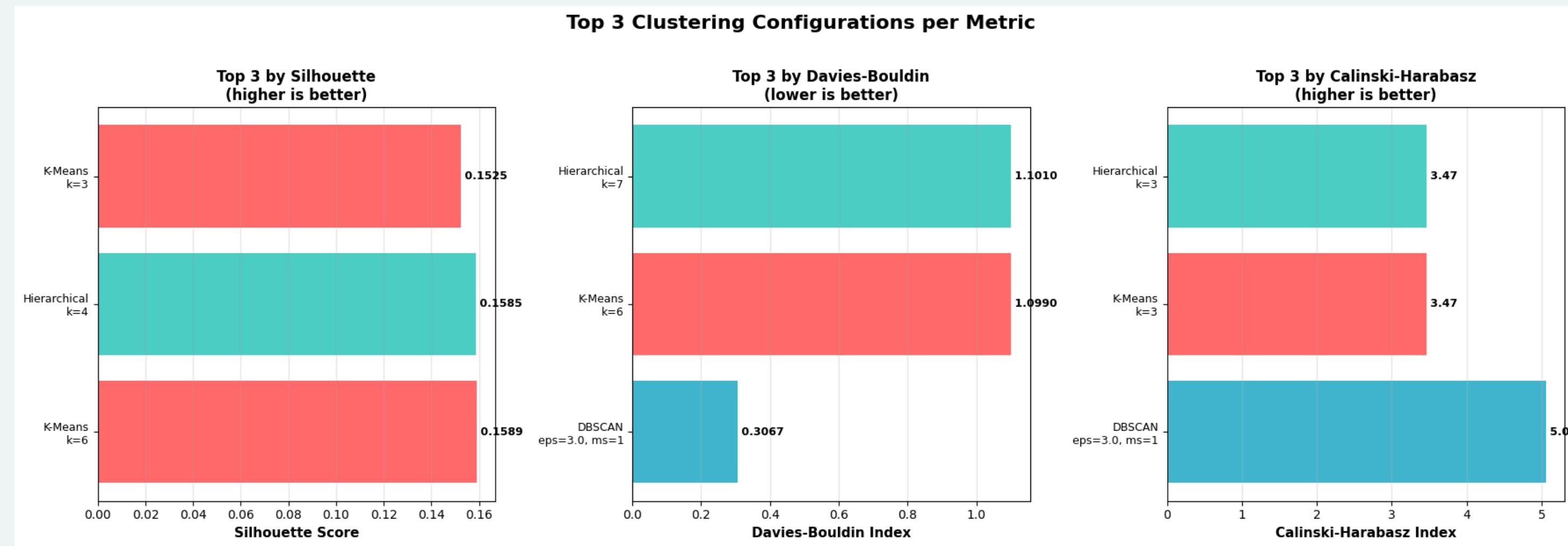
Ho poi provato diversi algoritmi di clustering con diversi parametri, mantenendo il numero di cluster basso visto che l'obiettivo era ridurre il numero di colonne (successivamente è stato applicato one-hot encoding alla colonna genere ottenuta)

```
1 # K-Means
2 for n_clusters in [3, 4, 5, 6, 7]:
3     kmeans = KMeans(n_clusters=n_clusters, random_state=42, n_init=10)
4     labels = kmeans.fit_predict(X_scaled)
5
6 # Hierarchical (Agglomerative)
7 for n_clusters in [3, 4, 5, 6, 7]:
8     hierarchical = AgglomerativeClustering(n_clusters=n_clusters, Linkage='ward')
9     labels = hierarchical.fit_predict(X_scaled)
10
11 # DBSCAN
12 eps_values = [0.5, 1.0, 1.5, 2.0, 2.5, 3.0]
13 min_samples_values = [1, 2, 3]
14 for eps in eps_values:
15     for min_samples in min_samples_values:
16         dbscan = DBSCAN(eps=eps, min_samples=min_samples)
17         labels = dbscan.fit_predict(X_scaled)
18
19     # esclude il rumore, etichetta come -1
20     n_clusters = len(set(labels)) - (1 if -1 in labels else 0)
21     n_noise = list(labels).count(-1)
```



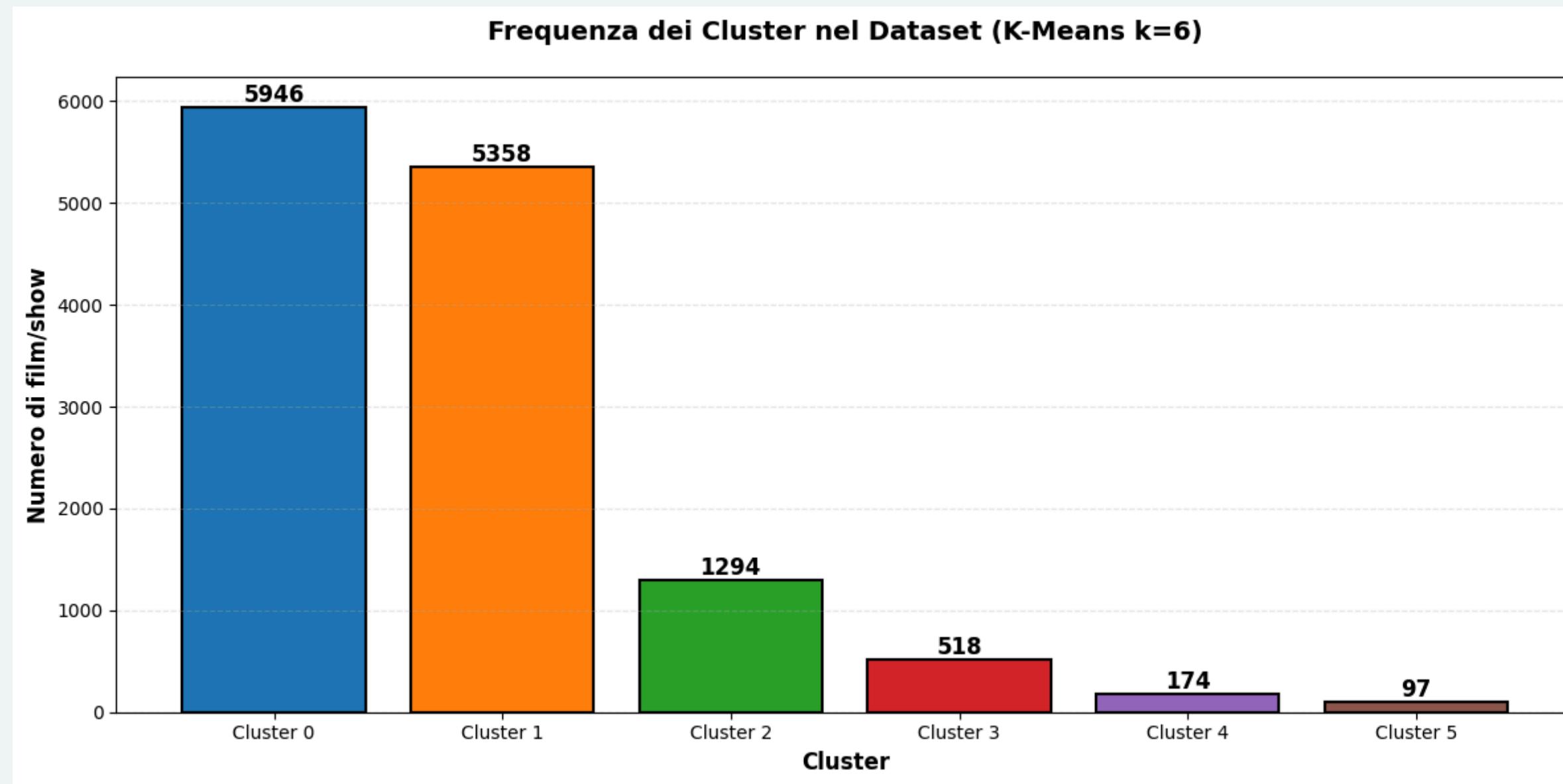
CLUSTERING GENERI

Ho poi provato diversi algoritmi di clustering con diversi parametri, mantenendo il numero di cluster basso visto che l'obiettivo era ridurre il numero di colonne (successivamente è stato applicato one-hot encoding alla colonna genere ottenuta)



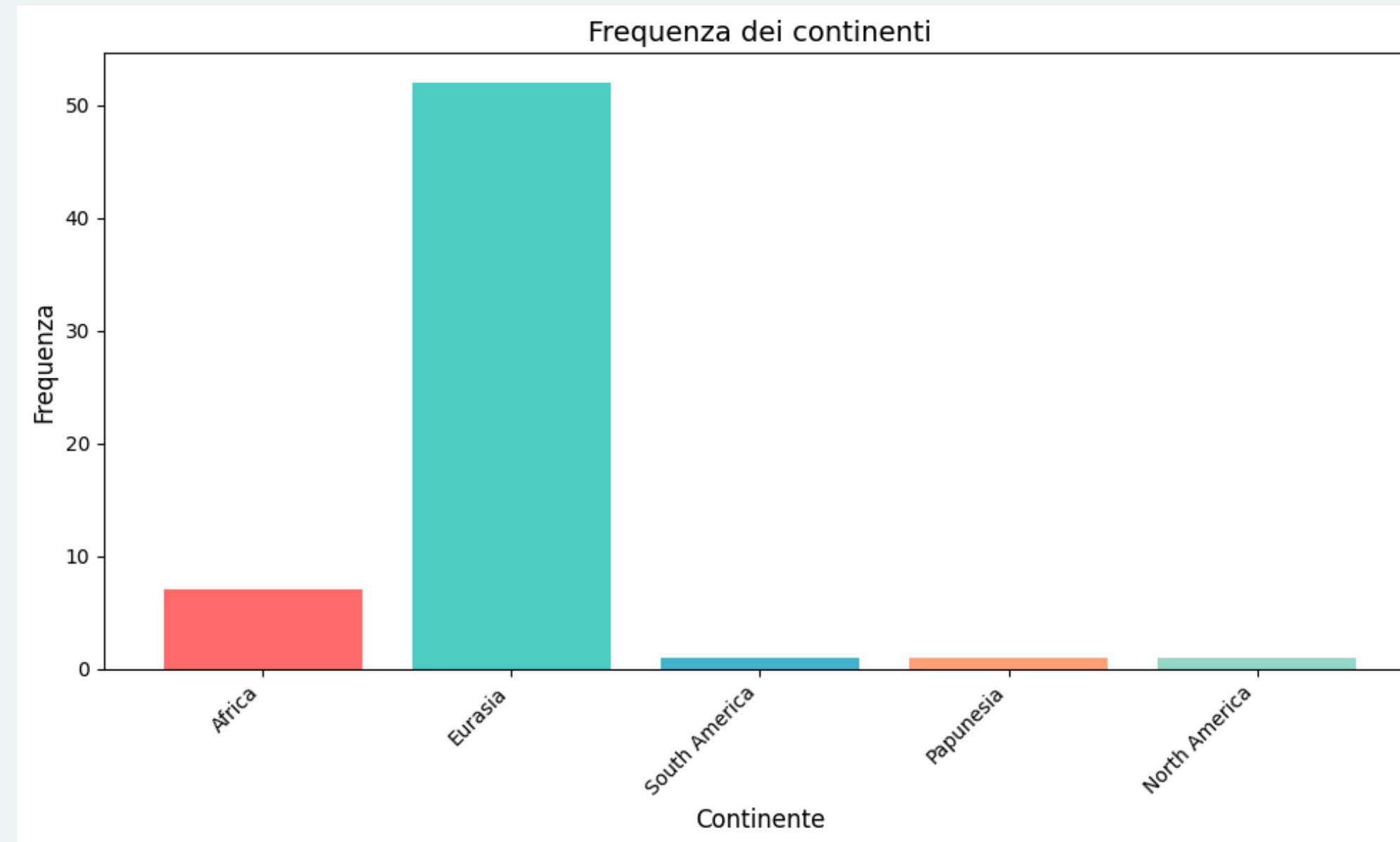
CLUSTERING GENERI

Ho poi provato diversi algoritmi di clustering con diversi parametri, mantenendo il numero di cluster basso visto che l'obiettivo era ridurre il numero di colonne (successivamente è stato applicato one-hot encoding alla colonna genere ottenuta)



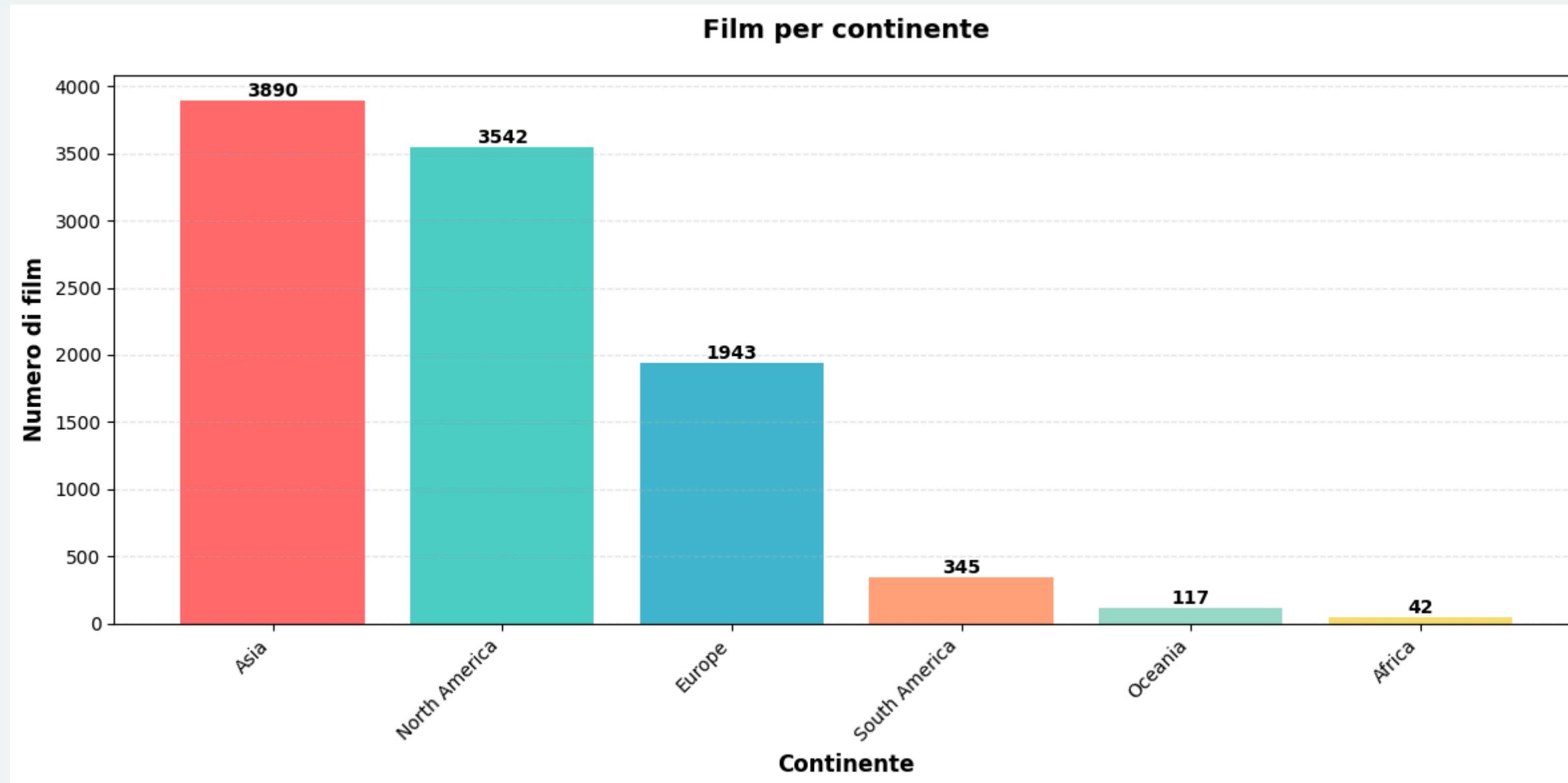
MAPPINGS

original_language



MAPPINGS

origin_country



RIMOZIONE OUTLIERS

Per la rimozione degli outliers ho computato il risultato di diversi metodi

Nome metodo	Numero outliers	Percentuale Dataset	Nome metodo	Numero outliers	Percentuale Dataset
Z-Score (3σ)	285	4.19%	IsoForest (10%)	680	10.00%
Z-Score (2.5σ)	455	6.69%	LOF (n=20)	340	5.00%
Modified Z-Score	2187	32.16%	LOF (n=50)	340	5.00%
IQR ($k=1.5$)	1597	23.49%	KNN ($k=5, 95\%$)	340	5.00%
IQR ($k=2.0$)	1256	18.47%	KNN ($k=10, 95\%$)	340	5.00%
IsoForest (1%)	68	1.00%	DBSCAN ($\text{eps}=0.5$)	6271	92.22%
IsoForest (5%)	340	5.00%	DBSCAN ($\text{eps}=1.0$)	5892	86.65%



RIMOZIONE OUTLIERS

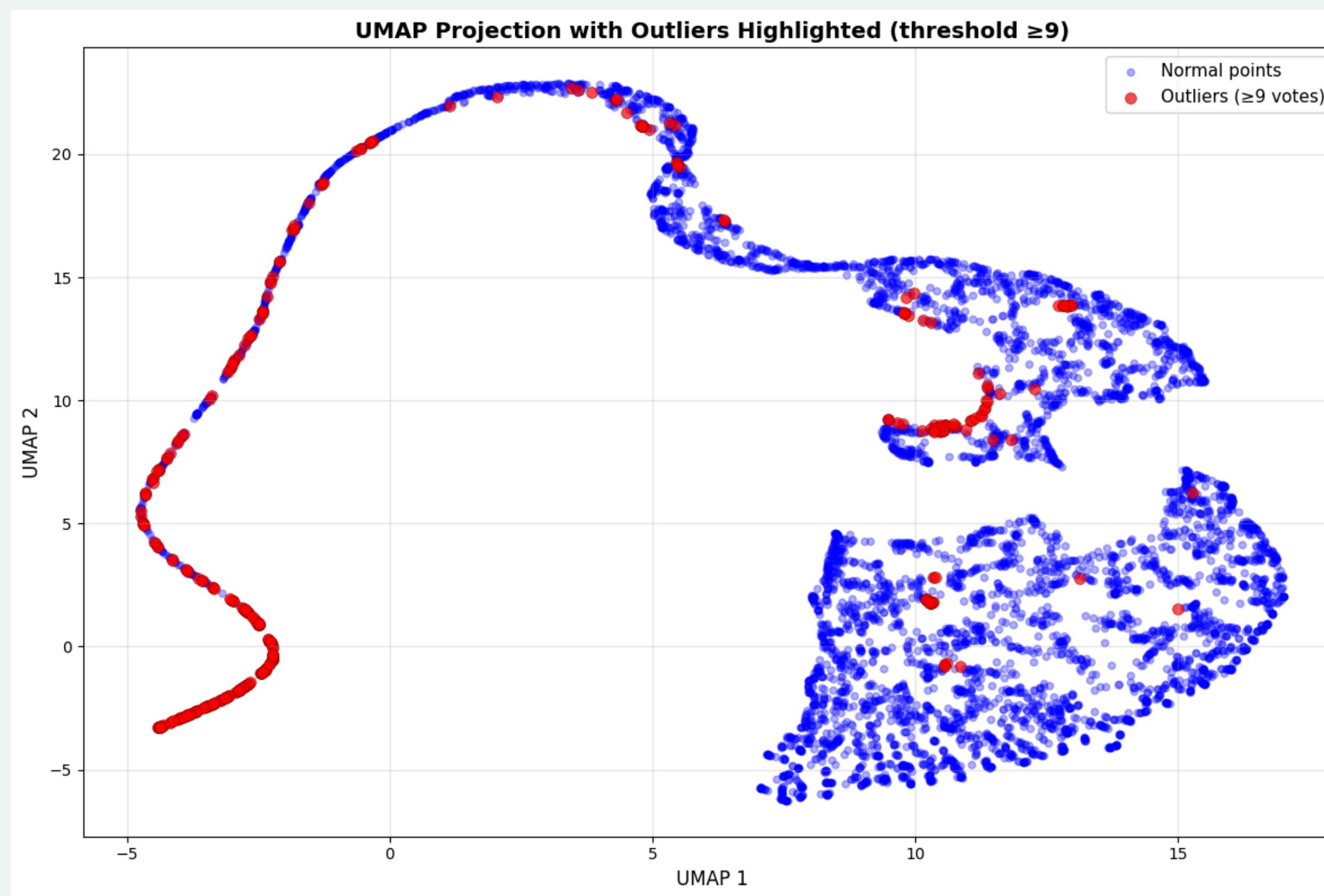
E poi applicato una tecnica di ensamble andando a vedere quanti metodi diversi categorizzavano gli stessi samples come outliers

Numero di metodi	Numero outliers	Percentuale Dataset	Numero di metodi	Numero outliers	Percentuale Dataset
≥2	5895	86.69%	≥8	501	7.37%
≥3	2277	33.49%	≥9	327	4.81%
≥4	1638	24.09%	≥10	243	3.57%
≥5	1423	20.93%	≥11	190	2.79%
≥6	805	11.84%	≥12	104	1.53%
≥7	633	9.31%	≥13	61	0.90%



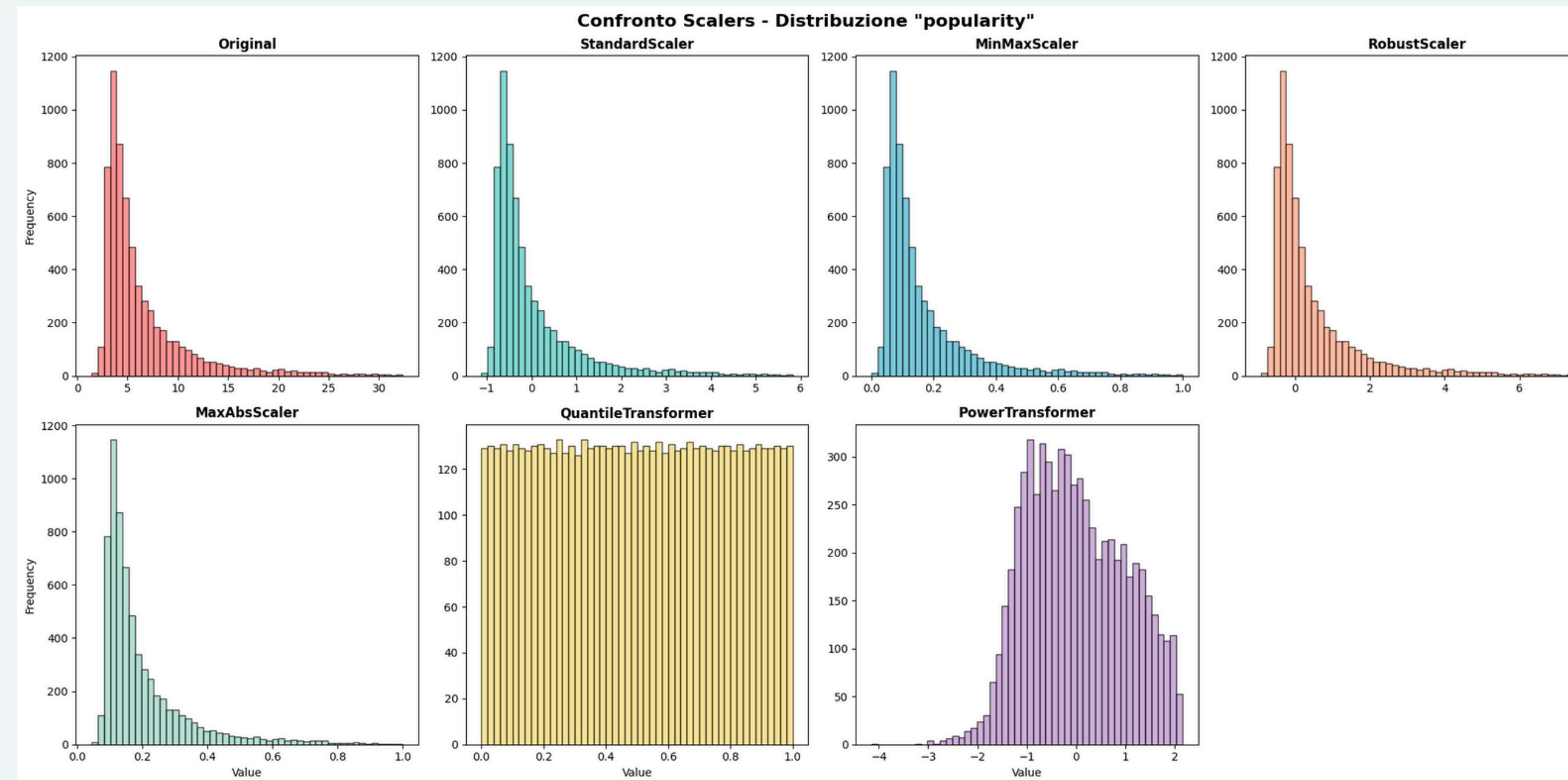
RIMOZIONE OUTLIERS

Ho deciso di rimuovere i sample utilizzando un cutoff a 9 metodi che riconoscevano gli stessi samples come outliers, qui la rappresentazione in UMAP



NORMALIZZAZIONE

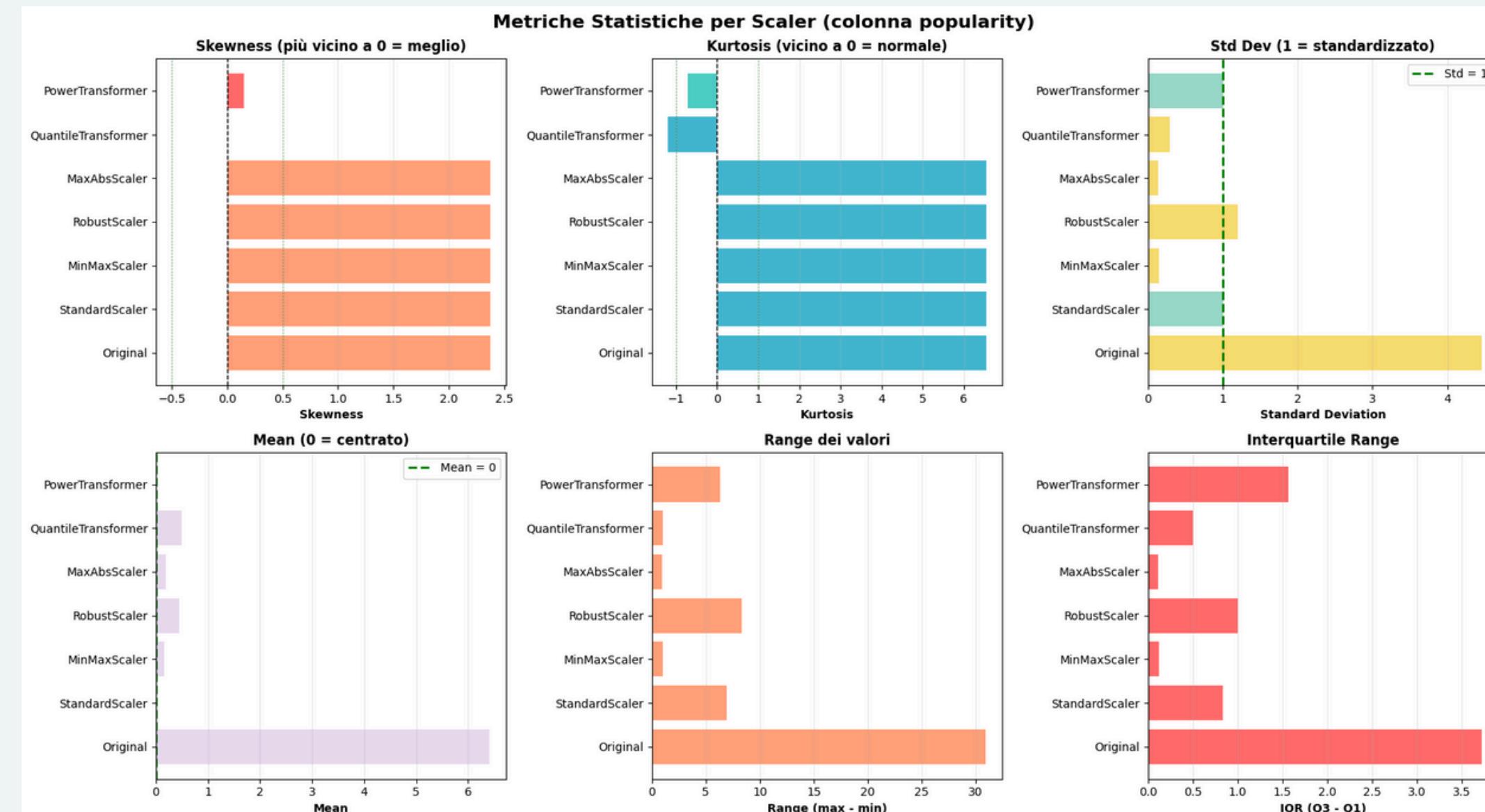
Successivamente ho testato diversi algoritmi di normalizzazione per trovare quello che effettuasse la trasformazione migliore sul dataset di train



NORMALIZZAZIONE

Dai risultati ottenuti lo scaler migliore è lo Standard Scaler

- I Power e Quantile Transformer riescono a ridurre sia la Skewness che il Kurtosis ma allo stesso tempo perdono la distribuzione originale della colonna
- Allo stesso tempo gli altri scaler hanno risultati simili negli altri test ma lo **standard scaler** ottiene sia una deviazione standard pari ad 1, una media a 0 e un range di valori accettabile



BILANCIAMENTO CLASSI

Per quanto riguarda il bilanciamento delle classi ho testato diverse tecniche di undersampling:

- Random Undersampling
- Condensed Nearest Neighbour
- Tomek Links
- Edited Nearest Neighbours

E due tecniche di oversampling

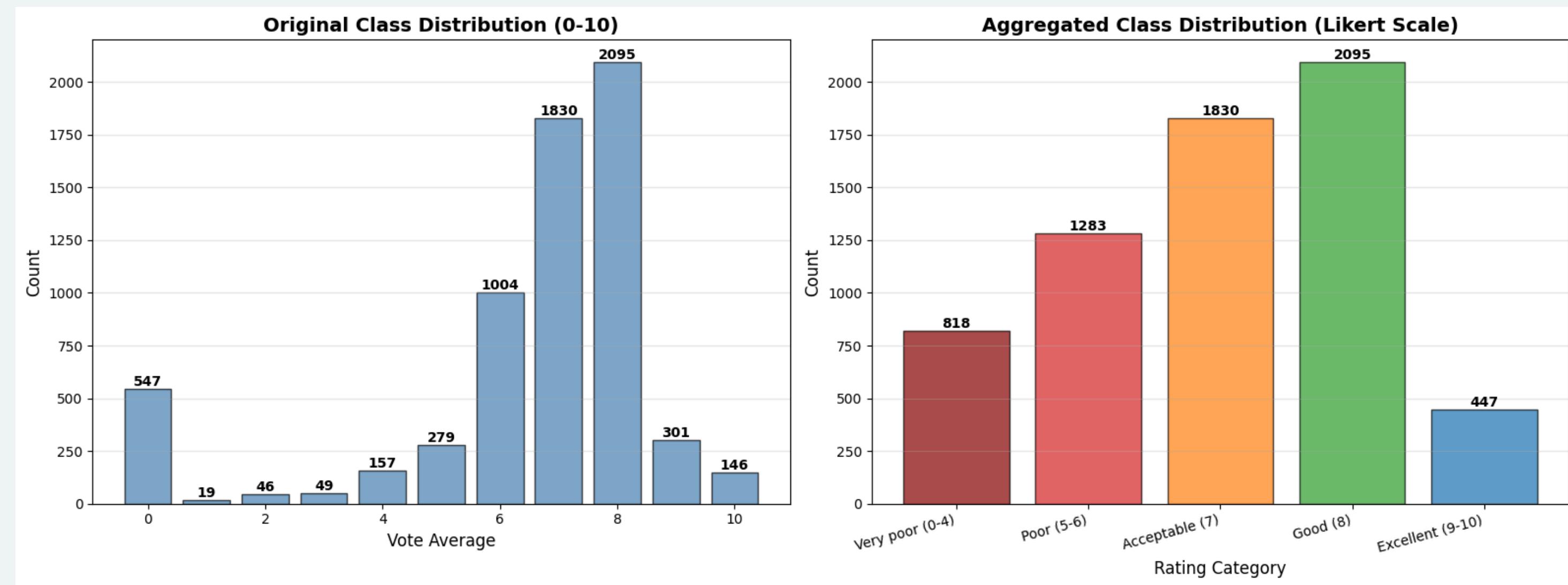
- Random Oversampling
- SMOTE

Nessuna di queste tecniche, da sola, data la distribuzione iniziale della classe target, risultava in un bilanciamento accettabile.



BILANCIAMENTO CLASSI

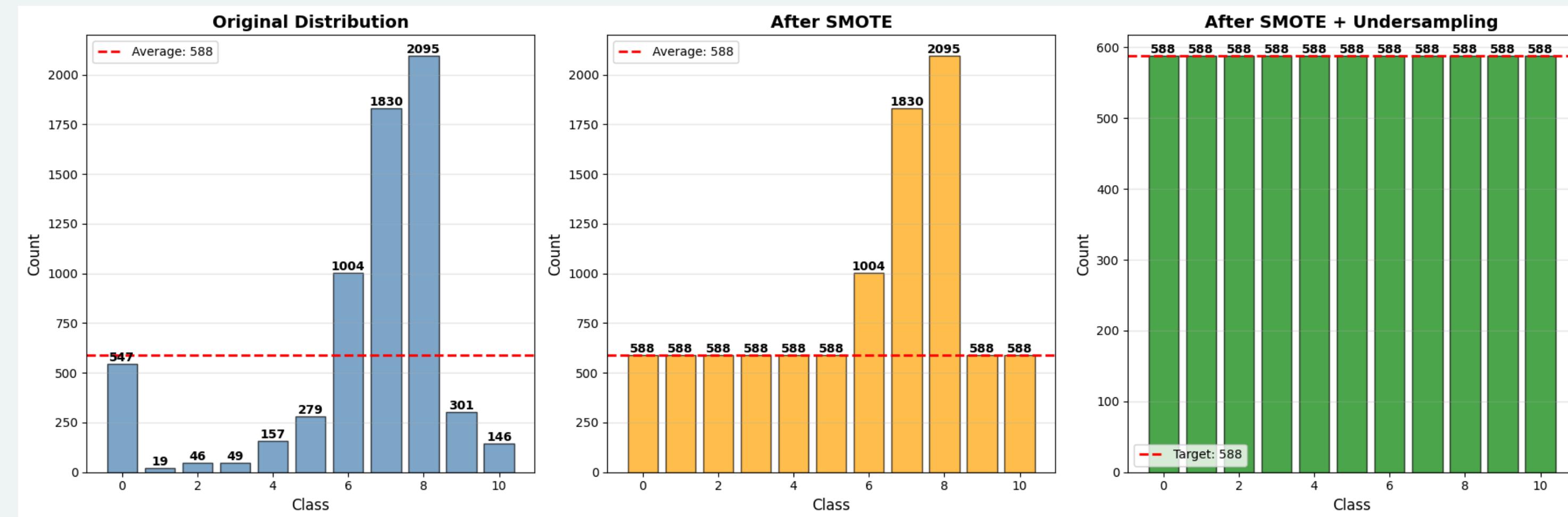
Una possibilità che ho considerato è stata quella di effettuare un binning delle classi target



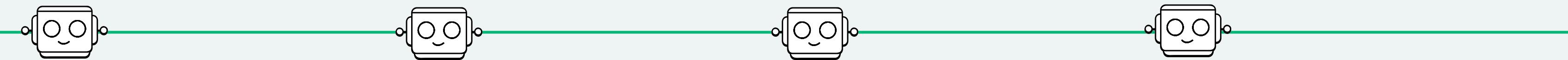
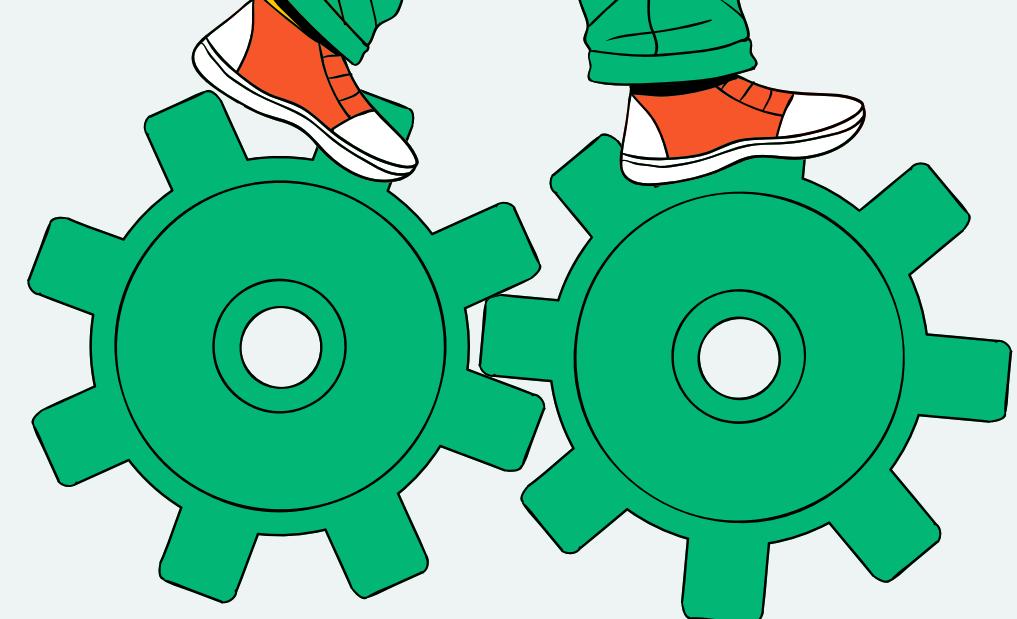
BILANCIAMENTO CLASSI

Ho pensato che questo potesse andare al di fuori della task data dalla traccia e ho quindi proseguito unendo sia una tecnica di undersampling che una di oversampling:

- SMOTE per aumentare le classi minoritarie fino al valore medio di frequenza delle classi
- Random undersampling per le classi con frequenza superiore a quella soglia



TRAINING CLASSIFICATORI E REGRESSORI



SCELTA MODELLI

Sia per la task di classificazione che di predizione ho selezionato diversi modelli da mettere a confronto fra di loro

DEFINIZIONE PARAMETRI

Per ogni modello ho definito una lista di parametri (fino a 9/10 combinazioni per ogni modello)

TRAINING

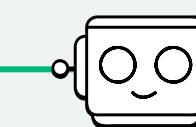
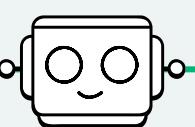
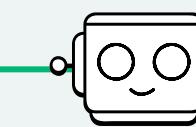
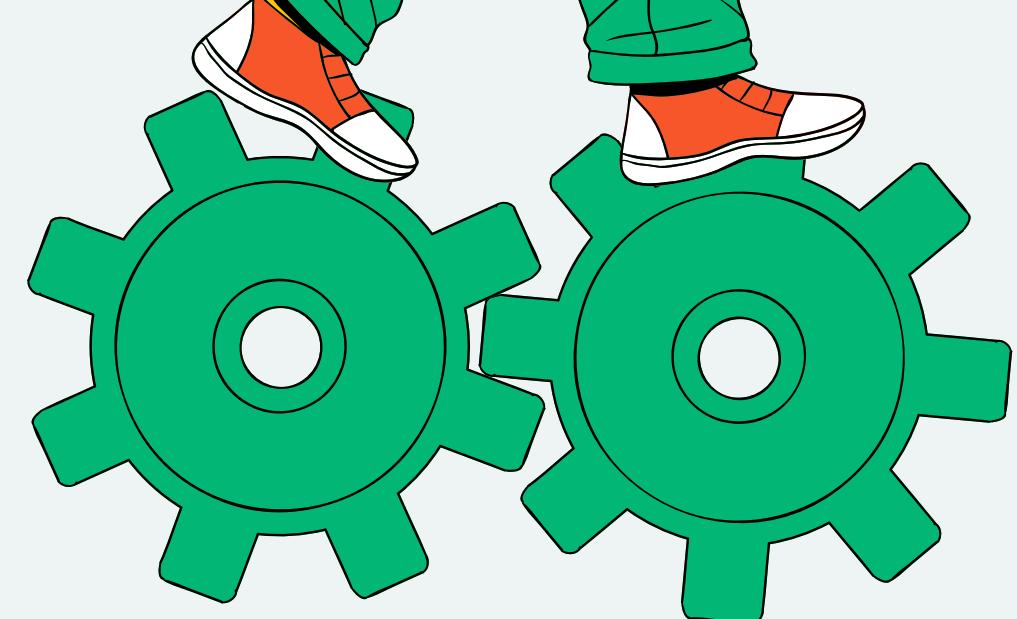
Ho effettuato una prima grid search con **Cross Validation (k=5)**

METRICHE E CONFRONTO

Ho poi calcolato le metriche per ognuno dei modelli sul set di validazione



TRAINING CLASSIFICATORI E REGRESSORI



FINE TUNING

Una volta trovata la combinazione di parametri migliori all'interno della parameter grid iniziale ho cercato di migliorare i parametri cercando attorno a quelli che erano risultati meglio performanti

METRICHE SU TEST SET

Ottenuto un modello accettabile ho effettuato il calcolo delle metriche sul test set

EXTRA

Una volta ottenuto il classificatore finale ho applicato due algoritmi di eXplainable Artificial Intelligence (LIME e SHAP) per ottenere una spiegazione sul valore mediano di ogni classe del test set per la classificazione



CLASSIFICAZIONE

Classificatori e relative parameters grids

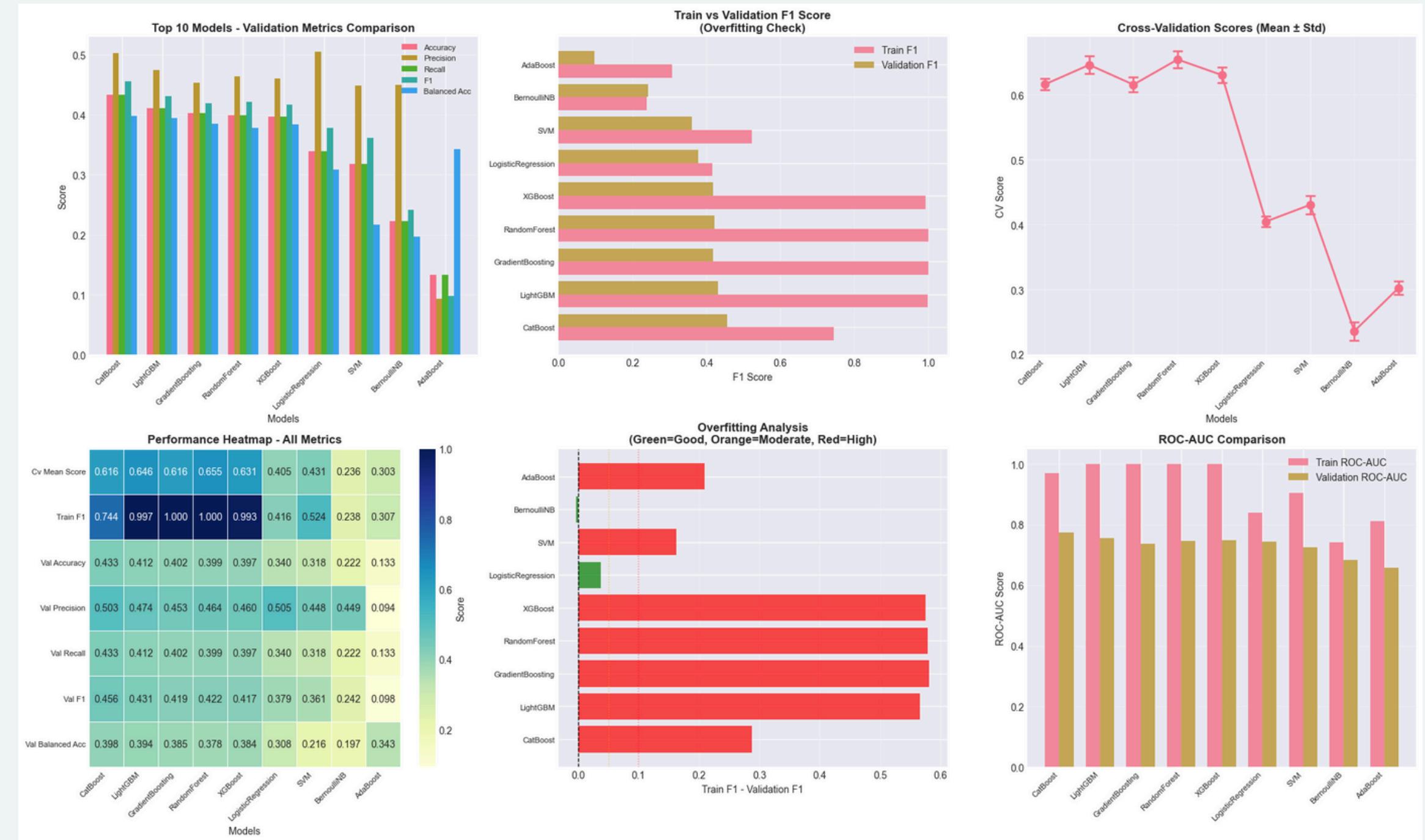
```
1  'RandomForest': {
2      'n_estimators': [100, 200, 300],
3      'max_depth': [10, 20, None],
4      'min_samples_split': [2]
5  },
6
7  'SVM': {
8      'C': [0.1, 1, 10],
9      'kernel': ['rbf', 'linear'],
10     'gamma': ['scale', 'auto']
11 },
12
13 'AdaBoost': {
14     'n_estimators': [50, 100, 200],
15     'learning_rate': [0.01, 0.1, 1.0]
16 },
17
18 'GradientBoosting': {
19     'n_estimators': [100, 200],
20     'learning_rate': [0.01, 0.1, 0.2],
21     'max_depth': [3, 5]
22 },
23
24 'LogisticRegression': {
25     'C': [0.01, 0.1, 1, 10],
26     'penalty': ['l1', 'l2'],
27     'solver': ['liblinear']
28 },
29
```

```
1  'LightGBM': {
2      'n_estimators': [100, 200],
3      'learning_rate': [0.01, 0.1],
4      'num_leaves': [31, 50],
5      'max_depth': [-1]
6  },
7
8  'XGBoost': {
9      'n_estimators': [100, 200],
10     'learning_rate': [0.01, 0.1, 0.2],
11     'max_depth': [3, 6]
12 },
13
14 'CatBoost': {
15     'iterations': [100, 200, 300],
16     'learning_rate': [0.01, 0.1],
17     'depth': [4, 6]
18 },
19
20 'BernoulliNB': {
21     'alpha': [0.1, 0.5, 1.0, 2.0, 5.0],
22     'binarize': [0.0, 0.5]
23 }
24 }
```



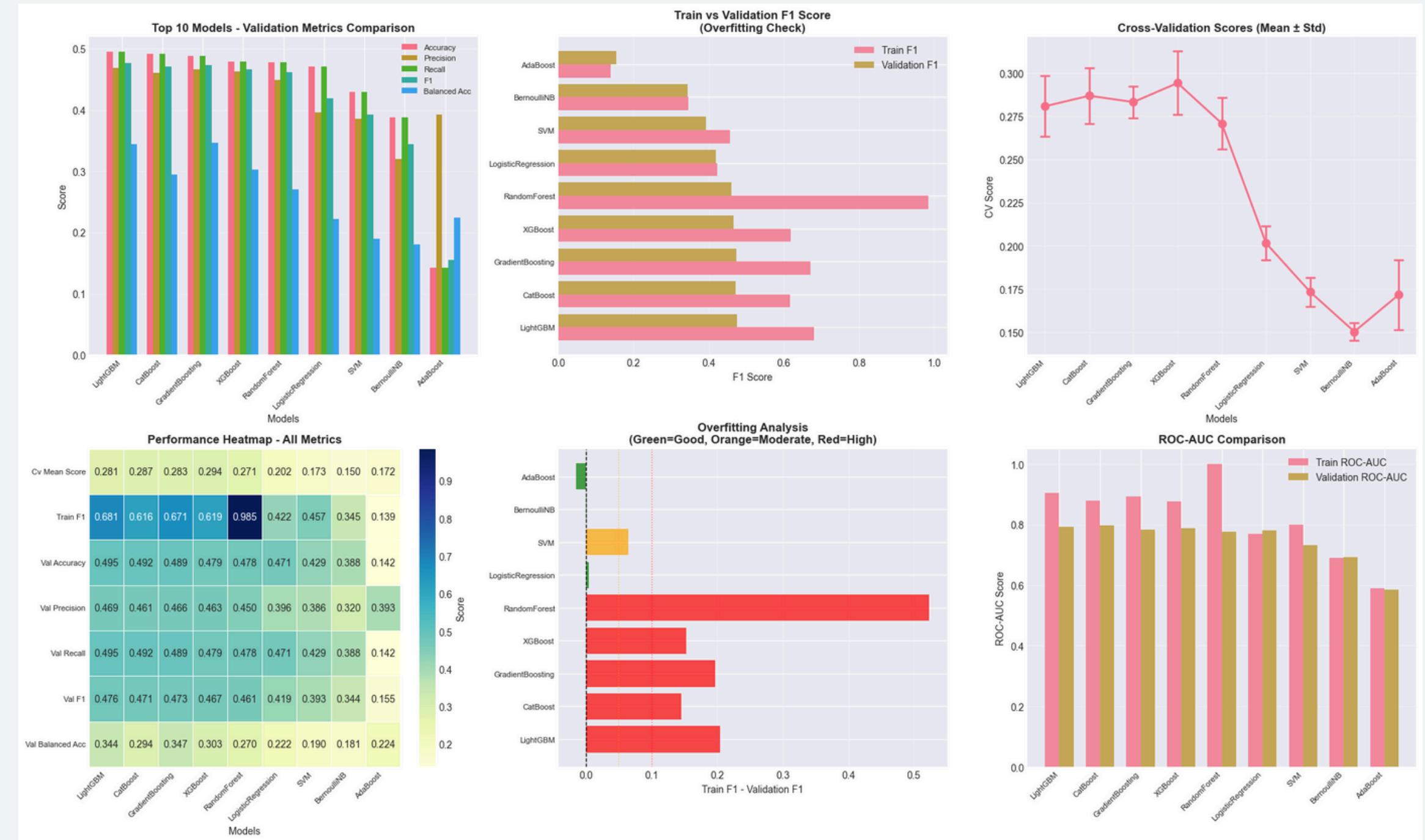
CLASSIFICAZIONE

Risultati training set bilanciato sul validation test



CLASSIFICAZIONE

Risultati training set non bilanciato sul validation test



CLASSIFICAZIONE

Com'è possibile notare dai risultati precedenti, il training sul dataset non bilanciato ottiene performance migliori.

Ho quindi deciso di continuare il fine tuning sul modello allenato sul dataset di training non bilanciato (LightGBM).

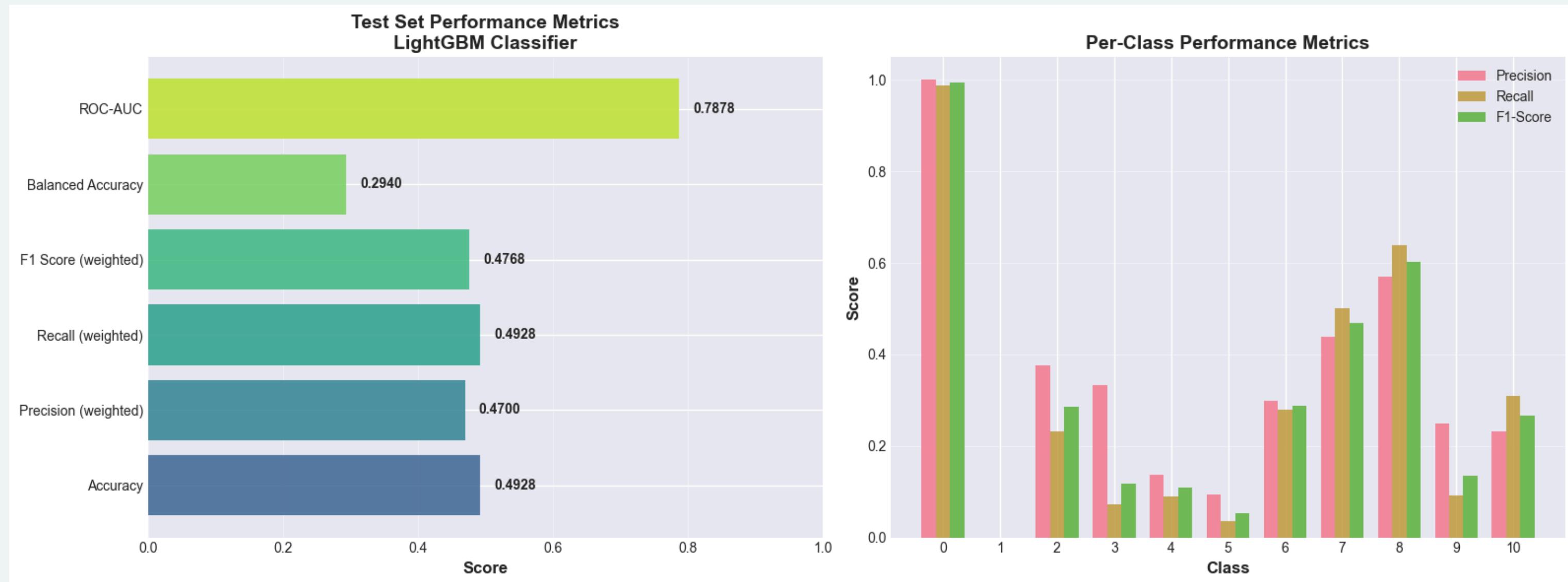
La seguente è la parameter grid estesa per il modello sulla quale è stata effettuata una cross validation a k=10

```
1 # https://lightgbm.readthedocs.io/en/stable/Parameters.html
2 fine_tune_param_grid_light = {
3     'n_estimators': [
4         200, 150, 250
5     ],
6     'learning_rate': [
7         0.01, 0.005, 0.015
8     ],
9     'num_leaves': [
10        50, 30, 70
11    ],
12     'max_depth': [-1, 10, 30],
13     # ho deciso di rimuovere gli altri parametri in quanto ci avrebbe messo troppo tempo
14     # 'min_child_samples': [20, 10, 30],
15     # 'subsample': [1.0, 0.8],
16     # 'colsample_bytree': [1.0, 0.8],
17     # 'reg_alpha': [0, 0.1],
18     # 'reg_lambda': [0, 0.1]
19 }
```



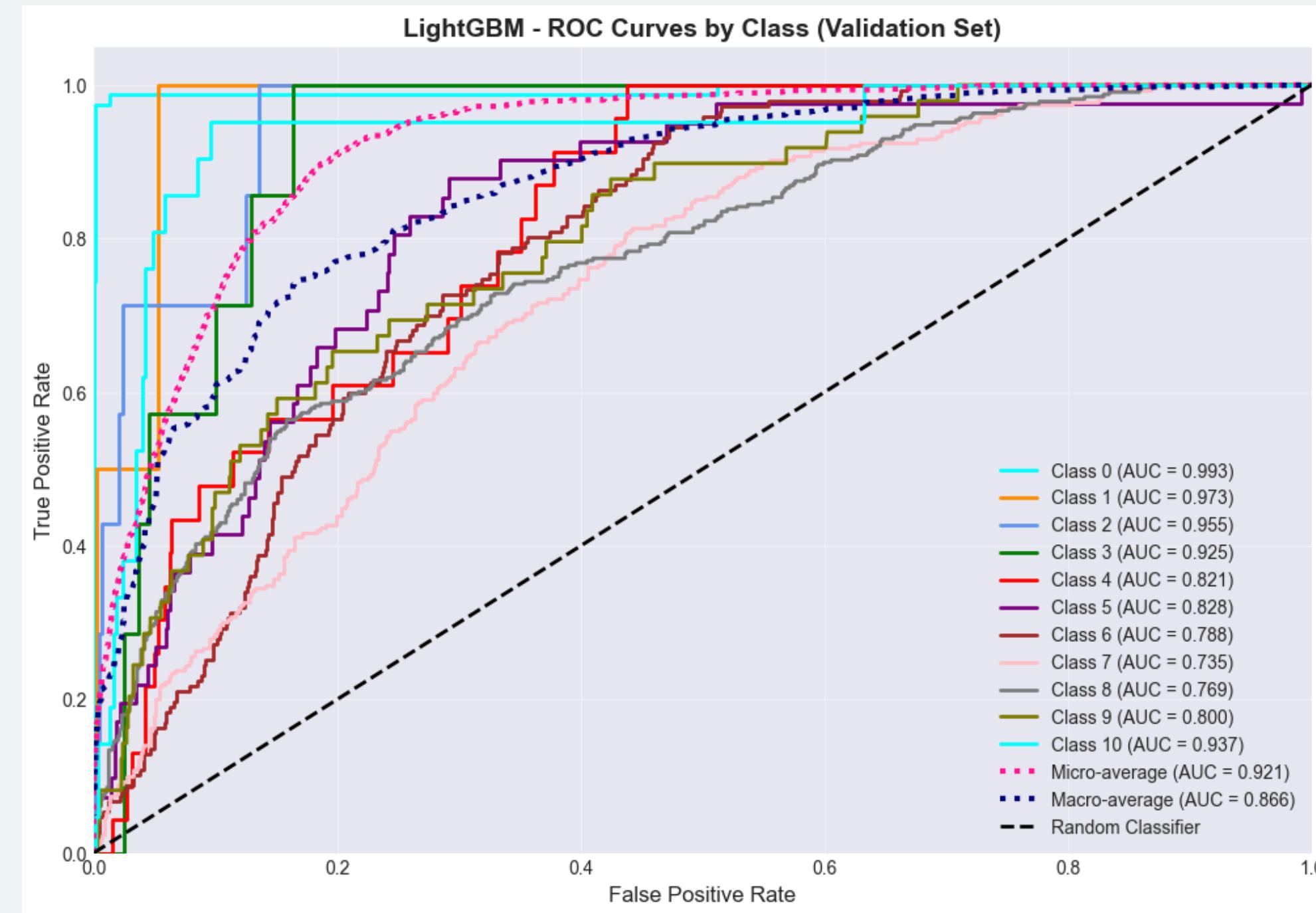
CLASSIFICAZIONE

Risultati training set non bilanciato con fine tuning sul validation test



CLASSIFICAZIONE

Risultati training set non bilanciato con fine tuning sul validation test



CLASSIFICAZIONE

Modello finale:

- Model: LightGBM Classifier
 - Parameters:
 - learning_rate: 0.015
 - max_depth: 10
 - n_estimators: 250
 - num_leaves: 70
 - random_state: 42
 - Test Set Performance:
 - Accuracy: 0.4928
 - Precision (weighted): 0.4700
 - Recall (weighted): 0.4928
 - F1 Score (weighted): 0.4768
 - Balanced Accuracy: 0.2940
 - ROC-AUC: 0.7878



REGRESSIONE

Regressori e relative parameters grids

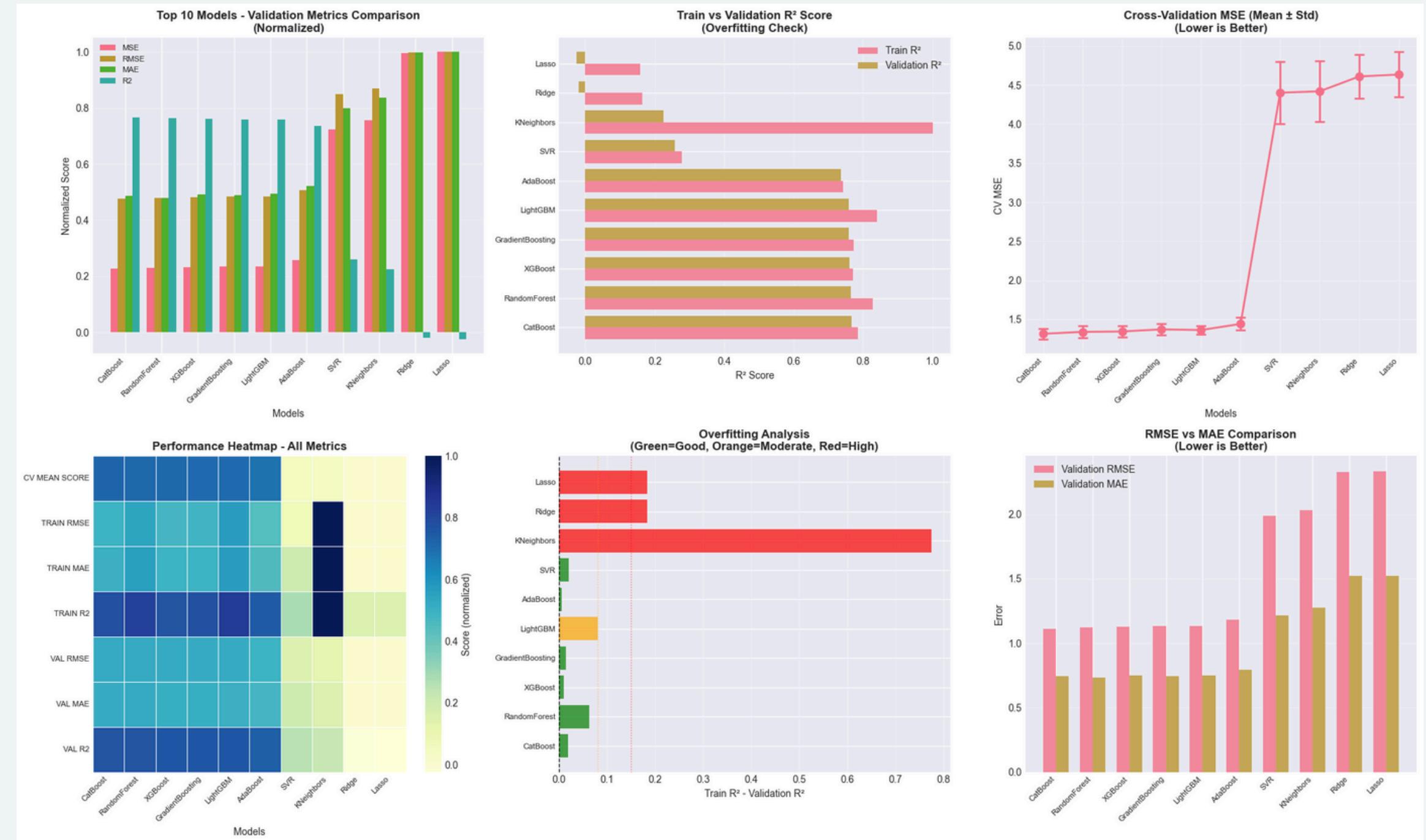
```
1  'RandomForest': {
2      'n_estimators': [100, 200, 300],
3      'max_depth': [10, 20, None],
4      'min_samples_split': [2]
5  },
6
7  'SVR': {
8      'C': [0.1, 1, 10],
9      'kernel': ['rbf', 'linear'],
10     'gamma': ['scale', 'auto']
11 },
12
13 'AdaBoost': {
14     'n_estimators': [50, 100, 200],
15     'learning_rate': [0.01, 0.1, 1.0]
16 },
17
18 'GradientBoosting': {
19     'n_estimators': [100, 200],
20     'learning_rate': [0.01, 0.1, 0.2],
21     'max_depth': [3, 5]
22 },
23
24 'Ridge': {
25     'alpha': [0.01, 0.1, 1, 10, 100],
26     'solver': ['auto', 'svd']
27 },
28
29 'Lasso': {
30     'alpha': [0.01, 0.1, 1, 10, 100]
31 }
```

```
1  'ElasticNet': {
2      'alpha': [0.01, 0.1, 1, 10],
3      'l1_ratio': [0.2, 0.5, 0.8]
4  },
5
6  'KNeighbors': {
7      'n_neighbors': [3, 5, 7, 10],
8      'weights': ['uniform', 'distance'],
9      'p': [1, 2]
10 },
11
12 'LightGBM': {
13     'n_estimators': [100, 200],
14     'learning_rate': [0.01, 0.1],
15     'num_leaves': [31, 50],
16     'max_depth': [-1]
17 },
18
19 'XGBoost': {
20     'n_estimators': [100, 200],
21     'learning_rate': [0.01, 0.1, 0.2],
22     'max_depth': [3, 6]
23 },
24
25 'CatBoost': {
26     'iterations': [100, 200, 300],
27     'learning_rate': [0.01, 0.1],
28     'depth': [4, 6]
29 }
```



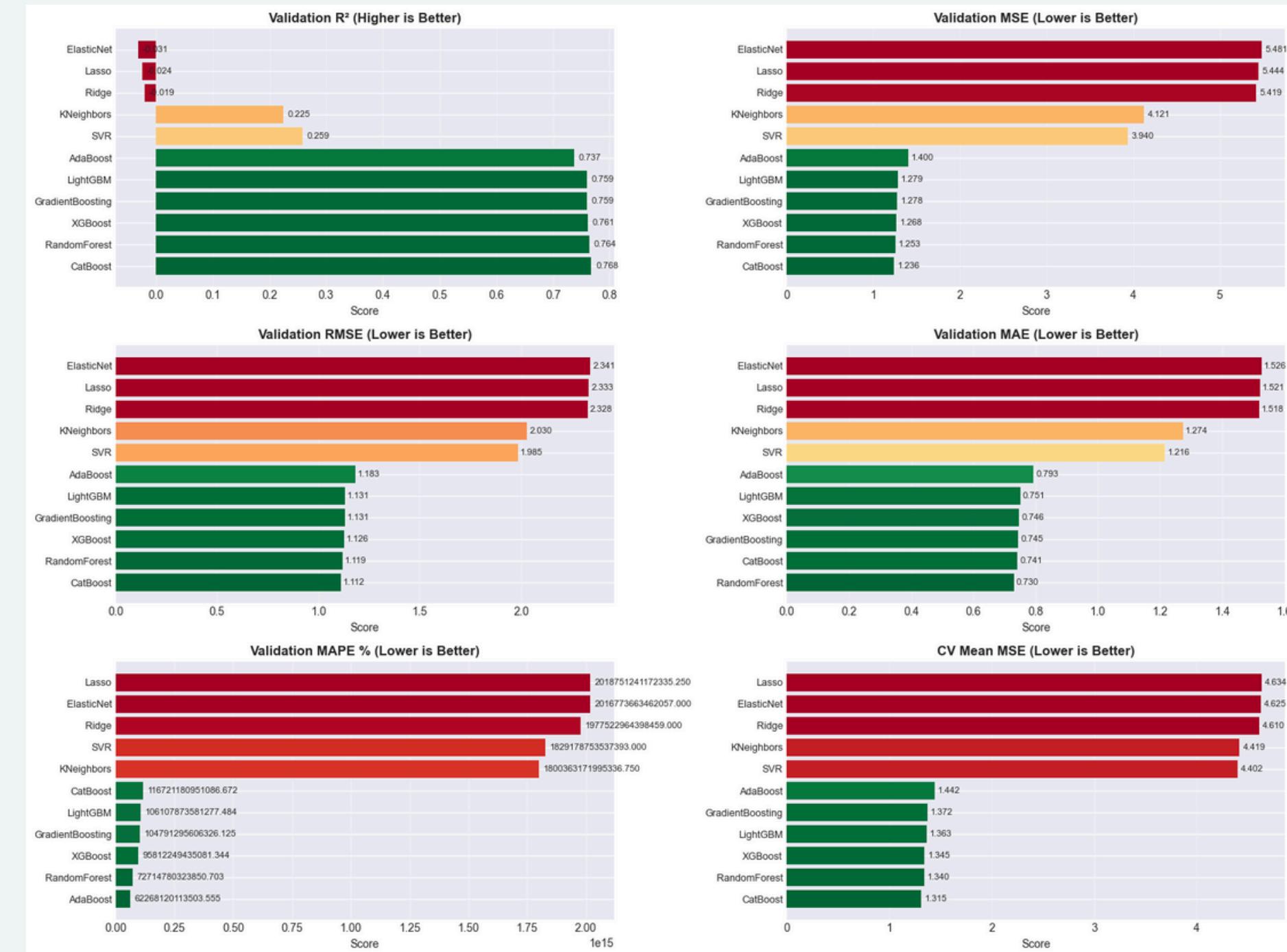
REGRESSIONE

Risultati training set sul validation test



REGRESSIONE

Confronto metriche dei modelli sul validation set



REGRESSIONE

Com'è possibile vedere dai risultati precedenti CatBoost regressor è il modello con le migliori performance, per quanto altri regressori si avvicinino molto. Ho deciso quindi di reiterare l'approccio precedente e migliorare la parameter grid iniziale, con la cross validation k=10, qui i nuovi parametri:

```
1 # https://catboost.ai/docs/en/concepts/python-reference_catboostregressor
2 # catboost ha davvero molti parametri settabili, qui mi limiterò o modificare quelli che avevo
3 # precedentemente impostato ed aggiungerò solo una nuova loss_function
4 fine_tune_param_grid_light = {
5     'iterations': [100, 70, 130],
6     'learning_rate': [0.1, 0.05, 0.2],
7     'depth': [6, 10, 14],
8     'loss_function': ['RMSE', 'MAPE'] # MAPE: https://en.wikipedia.org/wiki/Mean_absolute_percentage_error
9 }
```



REGRESSIONE

Una volta allenata la nuova parameter grid ho trovato dei risultati contrastanti.

I risultati fra la run precedente e questa non sono cambiati molto.

La differenza importante è che il risultato qui riportato indica che, per quanto i parametri risultati migliori nella prima grid search fossero presenti anche nella seconda run, il risultato migliore della seconda grid search sono parametri leggermente diversi, che però ottengono risultati peggiori.

Questo è dovuto al fatto che il valore di cross validation precedente fosse impostato a 5, mentre nella seconda grid search, con i nuovi parametri a 10.

Ciò significa che gli stessi parametri di prima hanno raggiunto un risultato **medio** peggiore una volta che il numero di test è aumentato.

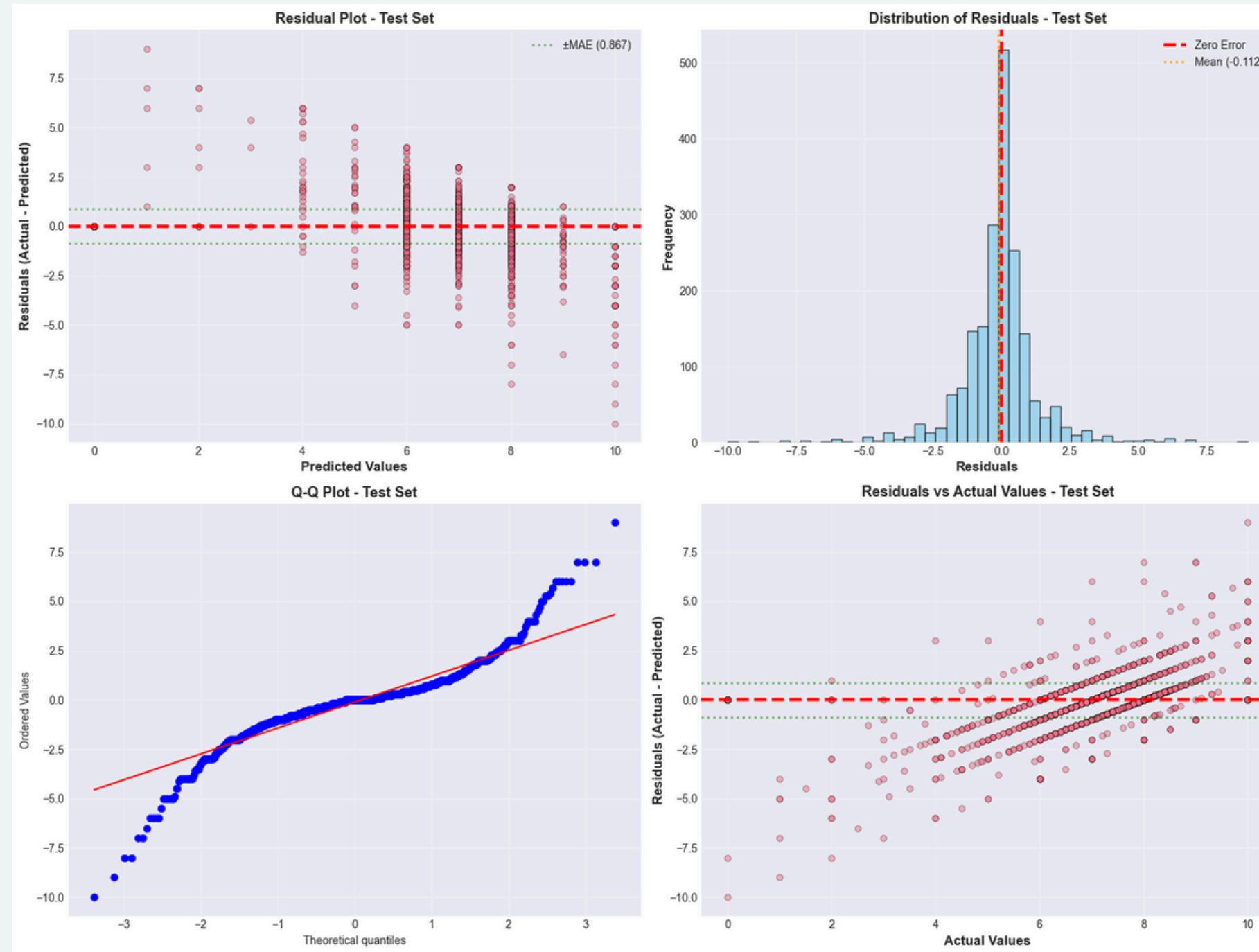
Per quanto i valori siano peggiorati non lo sono di molto e, in entrambi i casi, sono accettabili

MODEL COMPARISON						
Model	CV Score	Val MSE	Val RMSE	Val MAE	Val R2	
Initial	-1.315321	1.235672	1.111608	0.741313	0.767512	
Light Fine-Tuned	-1.297481	1.249010	1.117591	0.744986	0.765002	



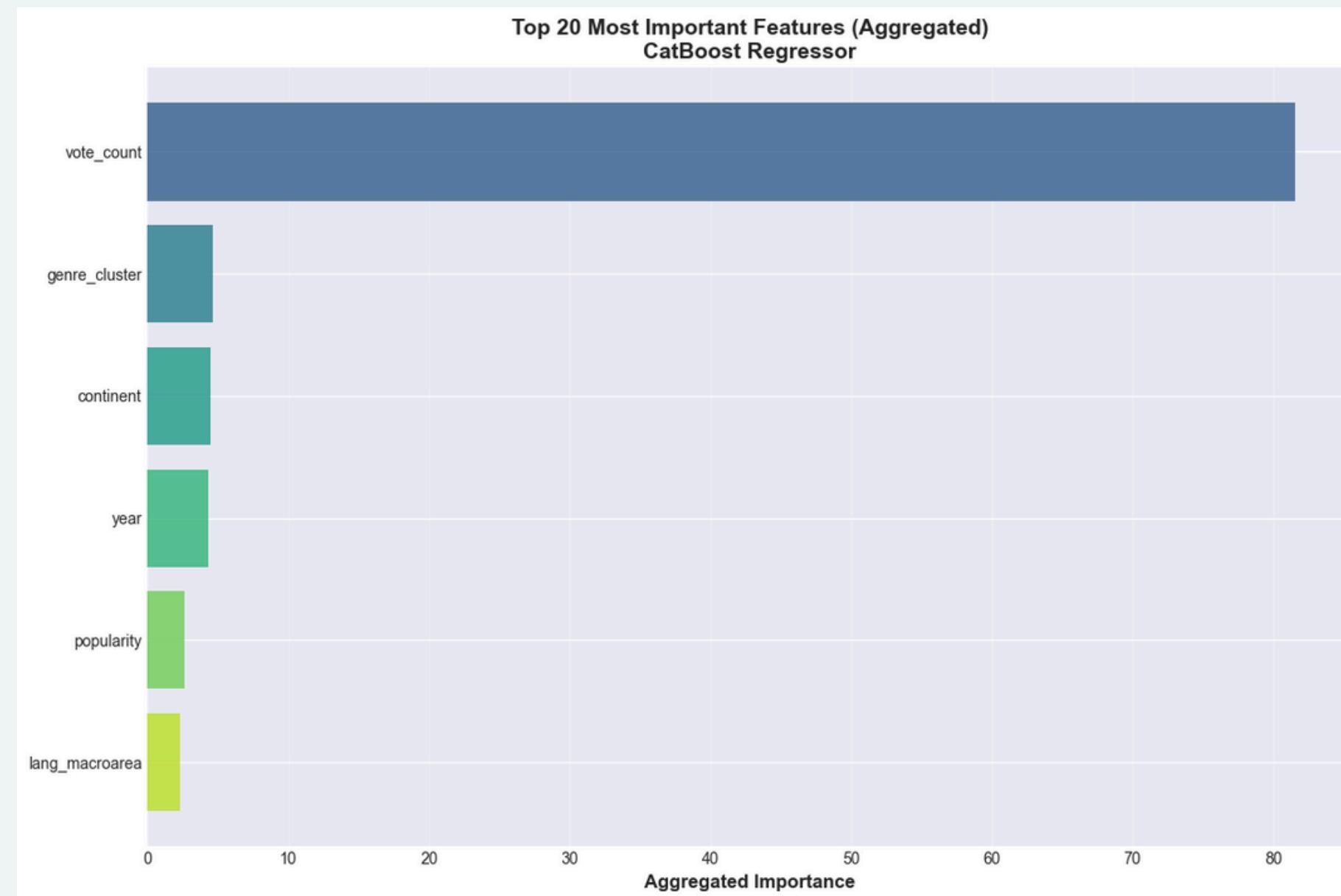
REGRESSIONE

Questi sono i dati relativi ai valori residui (residuals) calcolati sul test set



REGRESSIONE

È inoltre interessante notare la distribuzione delle feature importance per il regressore, in quanto la feature `vote_count` risulta avere un'importanza **molto** maggiore rispetto alle altre per quanto riguarda la regressione. Questo ha senso logico in quanto i film/show più polarizzanti avranno più recensioni



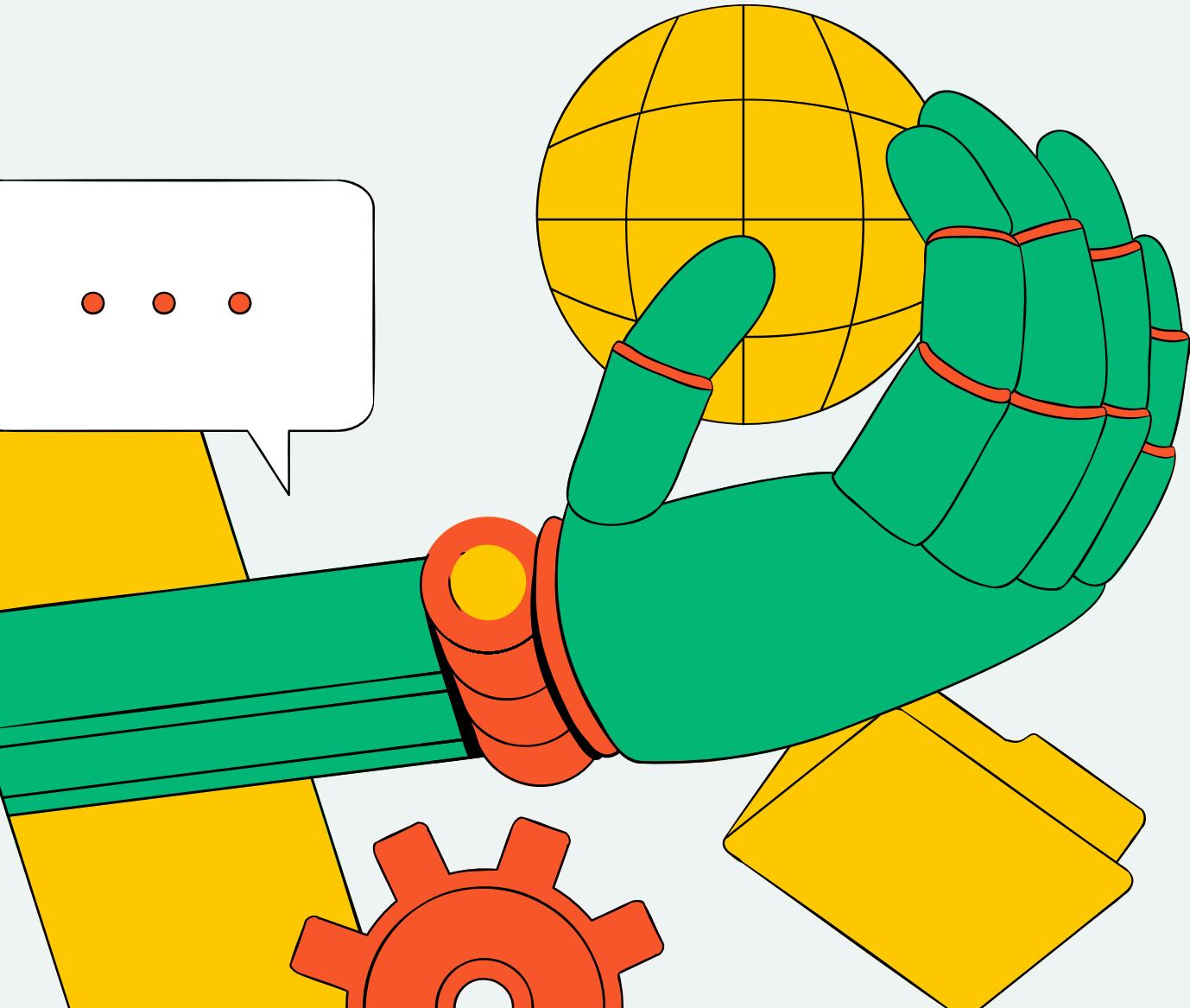
REGRESSIONE

Modello finale:

- CatBoost Regressor
 - Parameters:
 - iterations: 70
 - learning_rate: 0.1
 - depth: 10
 - loss_function: RMSE
 - random_state: 42
 - Test Set Performance:
 - Mean Squared Error (MSE): 1.9811
 - Root Mean Squared Error (RMSE): 1.4075
 - Mean Absolute Error (MAE): 0.8673
 - R-squared (R^2): 0.6301
 - Mean Absolute Percentage Error (MAPE %): 16.71%



EXTRA



Ho effettuato due task extra:

- Classificazione attraverso dati testuali (colonne name e overview del dataset originale)
- Spiegazione delle predizioni utilizzando le librerie LIME e SHAP

CLASSIFICAZIONE CON DATI DI TESTO



Per quanto riguarda questa nuova task, la pipeline che ho implementato è molto simile a quella precedente di classificazione. Cambiano i modelli e, per ragioni di tempo, non ho effettuato l'inizio del fine-tuning come per i modelli precedenti.



CLASSIFICAZIONE

Classificatori e relative parameters grids

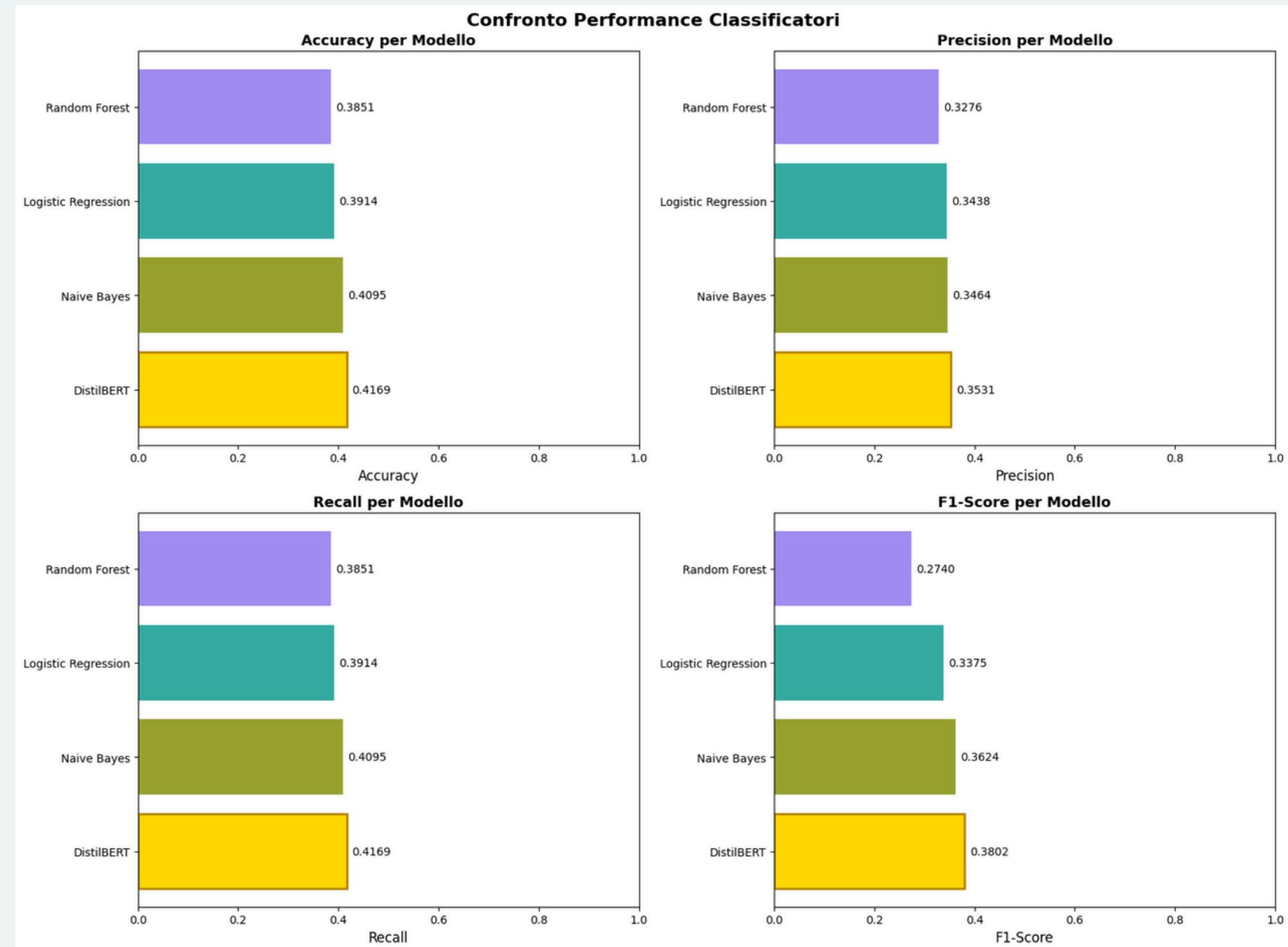
```
1  'Logistic Regression': {  
2      'tfidf_max_features': [5000, 10000],  
3      'tfidf_ngram_range': [(1, 1), (1, 2)],  
4      'classifier_C': [0.1, 1.0, 10.0],  
5      'classifier_solver': ['lbfgs']  
6  },  
7  
8  'Naive Bayes': {  
9      'tfidf_max_features': [5000, 10000],  
10     'tfidf_ngram_range': [(1, 1), (1, 2)],  
11     'classifier_alpha': [0.1, 1.0]  
12  },  
13  
14 'Random Forest': {  
15     'tfidf_max_features': [5000],  
16     'tfidf_ngram_range': [(1, 2)],  
17     'classifier_n_estimators': [100, 200],  
18     'classifier_max_depth': [10, 20]  
19  }
```

Ed in aggiunta ho effettuato
il training di DISTILBERT,
nello specifico
DistilBertForSequenceClassi-
fication



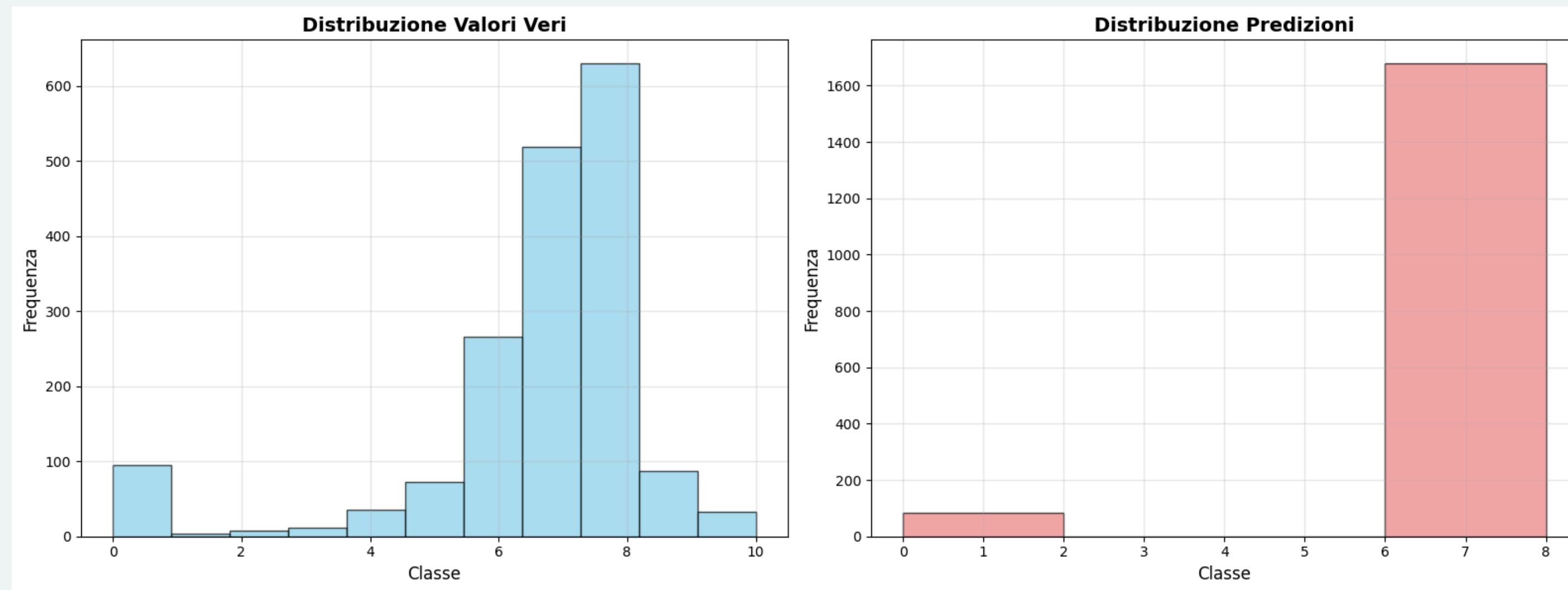
CLASSIFICAZIONE

Risultati su 4 metriche di performance, calcolate sul test set



CLASSIFICAZIONE

Sfortunatamente, per quanto le metriche ottenute da DISTILBERT siano promettenti andando a vedere la distribuzione delle classi predette correttamente notiamo che si concentrano solo dove il dataset presenta una grande quantità di dati

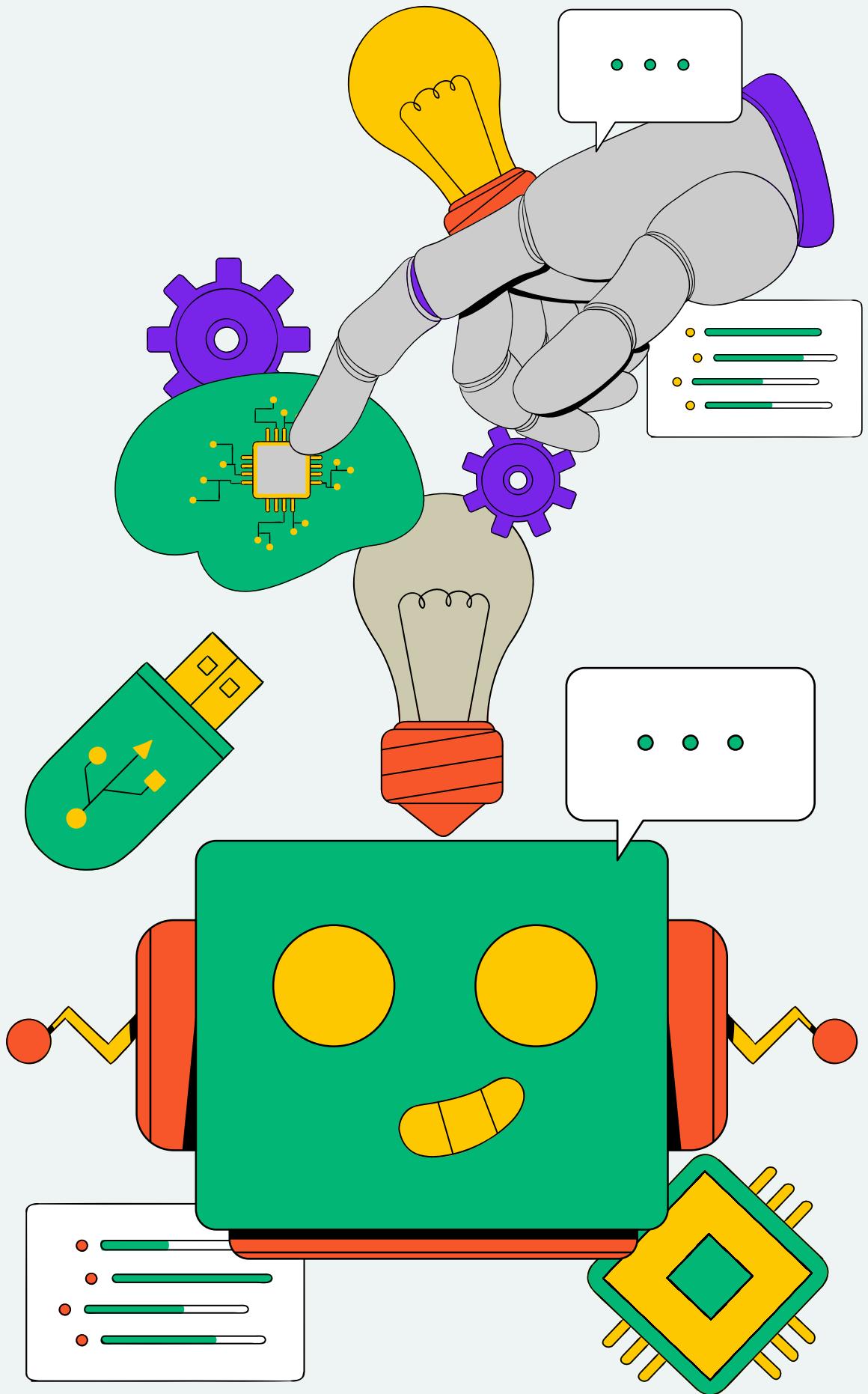


EXPLAINABLE ARTIFICIAL INTELLIGENCE

Infine ho implementato una pipeline XAI per analizzare come viene effettuata la predizione dal miglior classificatore precedentemente trovato (dati tabulari).

Per ogni classe target, presente nel dataset di test, ho estratto il valore centrale e dato in pasto ad entrambi i metodi XAI più popolari (LIME e SHAP) il classificatore, per cercare di capire cosa succede effettivamente.

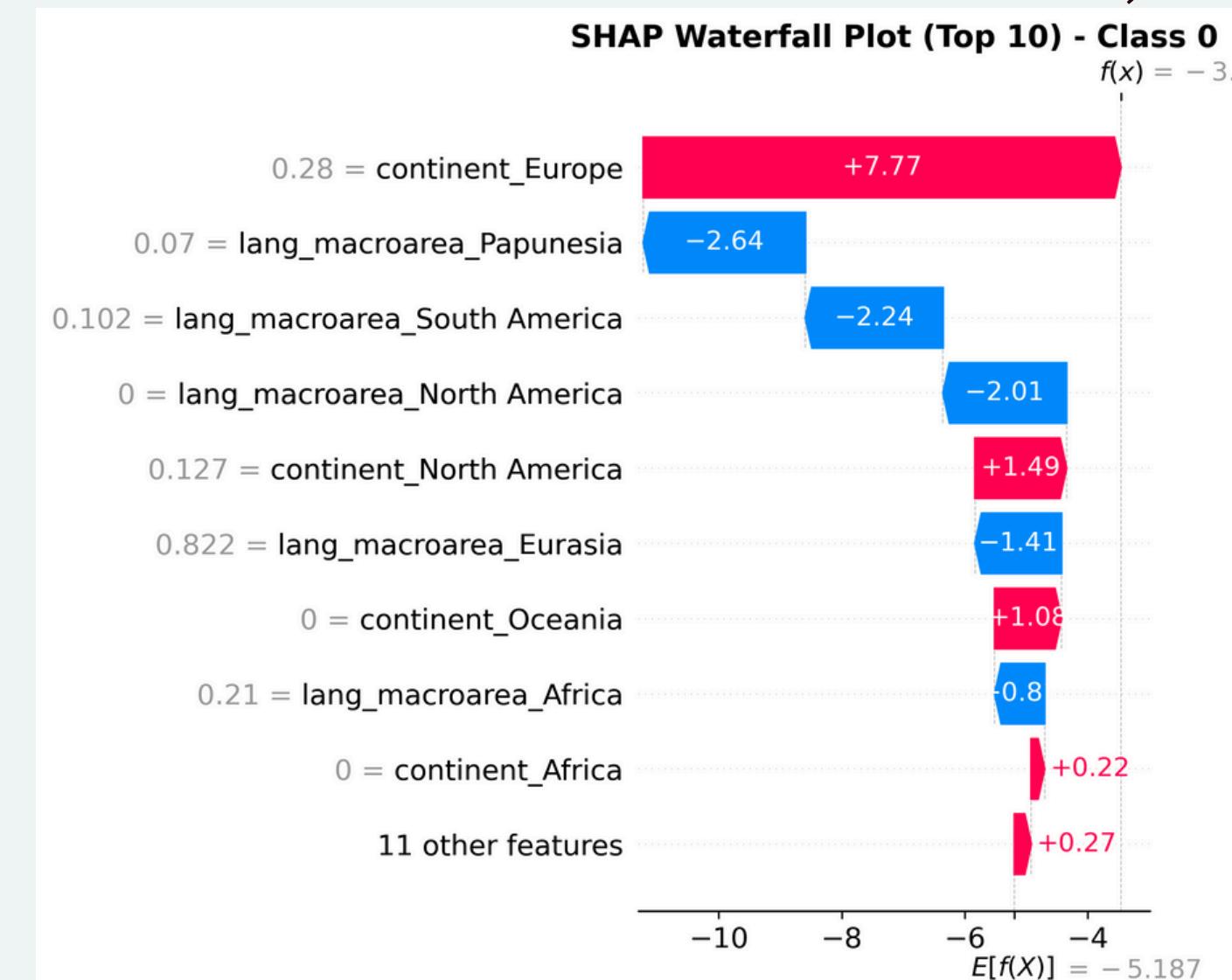
Nelle prossime slide alcuni dei risultati più rilevanti.



VOTE_AVERAGE = 0

Questa classe risulta interessante in quanto:

- la Predicted Class Probability risulta essere molto alta (0.9940)
- la predizione è molto influenzata positivamente dal fatto che il film sia di origine Europea
- è presente una influenza negativa dalle regioni non europee per la feature linguistica (Papunesia -2.64, South America -2.24)



VOTE_AVERAGE = 10

Questa classe risulta interessante in quanto:

- A parte la feature vote_count che abbiamo visto essere molto rilevante sia nella predizione che nella regressione,
- Le colonne relative ai continenti d'origine e alla macro area linguistica risultano essere molto influenti
- Probabilmente il film che risulta essere centroide della classe 10 (voto più alto) è un film internazionale con diverse influenze da diversi paesi

