ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΜΑΤΙΚΗΣ

Τελική Αναφορά: Εξόρυξης Δεδομένων

Ομάδα 13:

Χρήστος Καζάκος, it22033

Κωνσταντίνος Κατσάρας, it22045

Μανούσος Λιναρδάκης, it22064

Utility Functions:

Για την υλοποίηση της εργασίας, φτιάξαμε δύο βοηθητικά functions, τα οποία βοηθούν στο tokenization των στηλών Genre, Distributor και Script Type.
Το πρώτο function (get_dummies_genre) δημιουργεί one hot encodings μίας ζητούμενης στήλης (όχι απαραίτητα της genre) λαμβάνοντας υπόψη και τις “επαναλήψεις” των τιμών της στήλης οι οποίες δεν μπορούν να χωριστούν με την get_dummies καθώς είναι strings με κόμματα. Για παράδειγμα, ένα value της στήλης genre μπορεί να είναι ‘horror, horror’ ή ‘drama, comedy’, οπότε η συνάρτηση δημιουργεί one hot encoding της στήλης horror ή (one hot encodings) των δύο στηλών ‘drama’ και ‘comedy’ αντίστοιχα. Η συνάρτηση λαμβάνει επίσης υπόψην και ορθογραφικά.
Λόγω της πρώτης συνάρτησης, και επειδή η στήλη primary genre έχει λίγες και παρόμοιες τιμές με τη genre, αποφασίσαμε να φτιάξουμε και μία συνάρτηση combine_primary_genre, η οποία πάει τα string values της στήλης primary_genre στη genre (χωρίζοντας τα με κόμμα). Έπειτα μπορούμε να χρησιμοποιήσουμε τη get_dummies_genre για να φτιάξουμε τα one hot encodings.

Προετοιμασία Δεδομένων:

Ξεκινόντας την υλοποίηση της εργασίας βλέπουμε το είδος των γνωρισμάτων (αριθμητικα, κατηγορικά) έτσι ώστε να έχουμε μια πρώτη εικόνα.

```
# Summary statistics for numerical variables
print(df.describe().T)

print("========================================================================")

# Summary statistics for categorical variables
print(df.describe(include='object').T)
```

Στη συνέχεια, βλέπουμε τα ποσοστά των missing values ανά στήλη. Παρατηρούμε ότι οι στήλες IMDb Rating, IMDB vs RT disparity και Distributor είναι κενές για αυτό τις πετάμε (προς το παρόν).

```
missing_data = df.isnull().sum()
missing_percentage = (missing_data[missing_data > 0] / df.shape[0]) * 100
missing_percentage.sort_values(ascending=True, inplace=True)
print(missing_percentage)
df.drop(columns=['IMDb Rating'], inplace=True)
df.drop(columns=['IMDB vs RT disparity'], inplace=True)
df.drop(columns=['Distributor'], inplace=True)
```

Έπειτα, παίρνουμε δεδομένα από csv του imdb (πηγή εδώ) και ξαναπροσθέτουμε τη στήλη IMDb Rating γεμίζοντάς την με όσα δεδομένα έχουμε από τα csv του imdb. Για να γεμίσουμε τη στήλη αυτή, κάνουμε join το movies.xlsx με τα csv του imdb στο όνομα ταινίας και έτος.

Μετά ελέγχουμε αν υπάρχουν διπλότυπες πλειάδες και αν υπάρχουν τις διαγράφουμε. Ελέγχουμε επιπλέον αν υπάρχουν διπλότυπα με βάση το όνομα των ταινιών καθώς μετά την ένωση με τα csv του imdb είναι πιθανό μία ταινία να έχει το ίδιο όνομα και χρονολογία με μία άλλη, με αποτέλεσμα στο join να θεωρούνται “ίδιες”. Εμείς κρατάμε την πρώτη εκ των δύο.

```
df = df.drop_duplicates()
df = df.drop_duplicates(subset='Film', keep='first')
```

Έπειτα πετάμε όσες στήλες δεν έχουν τουλάχιστον ένα από τα υπόλοιπα:

```
df.dropna(subset=['Genre', 'Rotten Tomatoes  critics', 'Metacritic  critics',
'Metacritic Audience ', 'Rotten Tomatoes Audience ', 'runtimeMinutes',
'directors'], inplace=True)
```

Επειδή τα csv δεν έχουν όλες τις ταινίες του movies.xlsx χρησιμοποιούμε επιπλέον τη βιβλιοθήκη PyMovieDb (είναι ένα wrapper που αντιπροσωπεύει το IMDB API. Τις πληροφορίες για ταινίες/τηλεοπτικές σειρές από το IMDB τις παίρνει με scraping). Έτσι, μπορούμε να πάρουμε τα ratings σχεδόν για όλες τις ταινίες (μένει μόνο το 1.5% των ταινιών χωρίς IMDb Rating). Για να γεμίσουμε και όσες εγγραφές δεν έχουν IMDb Rating, παίρνουμε τη μέση διαφορά μεταξύ Average Audience και IMDb Rating, και την αφαιρούμε από το Average Audience της γραμμής. Η τιμή που προκύπτει είναι η εκτίμηση του IMDb Rating. Ο λόγος που επιλέξαμε την στήλη Average Audience είναι επειδή είχε την μικρότερη μέση διαφορά με τη IMDb Rating.

΄Υστερα, κάνουμε webscrape χρησιμοποιώντας το beautifulsoup για να μαζέψουμε τους distributors των ταινιών από το rotten tomatoes. Για τις ταινίες που δεν υπάρχουν distributors, βάζουμε την τιμή -1, ενώ για αυτές που δεν μπορούμε να κάνουμε webscrape βάζουμε nan. Καθώς οι ταινίες που δεν μπορούμε να κάνουμε webscrape είναι μόνο το 6.9% των δεδομένων, θεωρούμε ότι αυτές οι ταινίες δεν έχουν distributor (γεμίζουμε τα nans της στήλης distributor με -1).

Έπειτα σβήνουμε κάποιες στήλες που δεν τις θεωρούμε τόσο σημαντικές όσο κάποιες άλλες, καθώς η πληροφορία τους υπάρχουν ήδη στις άλλες στήλες. Συγκεκριμένα, πετάμε τις Worldwide Gross ($million), Release Date (US), Opening Weekend ($million), Domestic Gross ($million), Foreign Gross ($million). Γενικά για τα “χρήματα” σβήσαμε όσες τιμές είναι στα million. Ακόμα και τη στήλη Budget ($million) τη σβήνουμε και δημιουργούμε τη στήλη Budget η οποία προκύπτει από το Bugdet ($million) * 1000000.

Επίσης, χρησιμοποιώντας το utility function “combine_primary_genre” μπορούμε να πετάξουμε και τη στήλη primary genre (και το κάνουμε). Μία ακόμη στήλη που πετάξαμε είναι η oscar details, καθώς δεν υπάρχει στο ανώνυμο dataset.

Στη συνέχεια κατηγοριοποιούμε τις ταινίες με βάση αν έχουν κερδίσει όσκαρ (1) και σε εκείνες που δεν έχουν κερδίσει (0). Στις γραμμές όπου δεν υπάρχει τιμή (δηλαδή δεν έχουν κερδίσει) βάζουμε τον αριθμό 0 για να είναι πιο εύκολα διαχειρίσιμες.

```
df['Oscar Winners'] = df['Oscar Winners'].map({'Oscar winner': 1, 'Oscar
Winner':1})
df['Oscar Winners'].fillna(0, inplace=True)
```

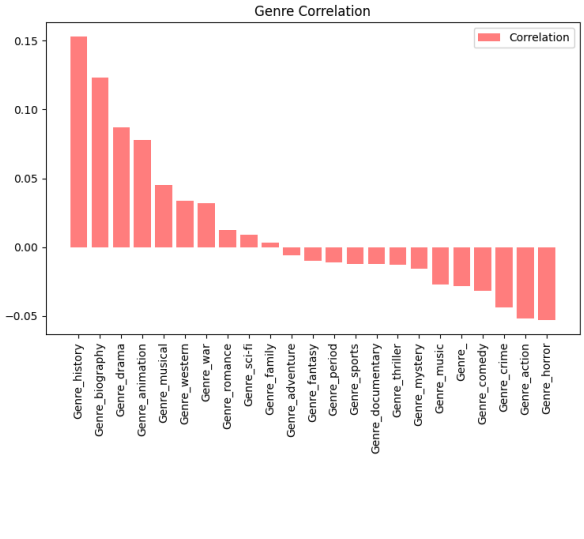
Τέλος, μετατρέπουμε σε numeric τις εναπομείναντες στήλες:

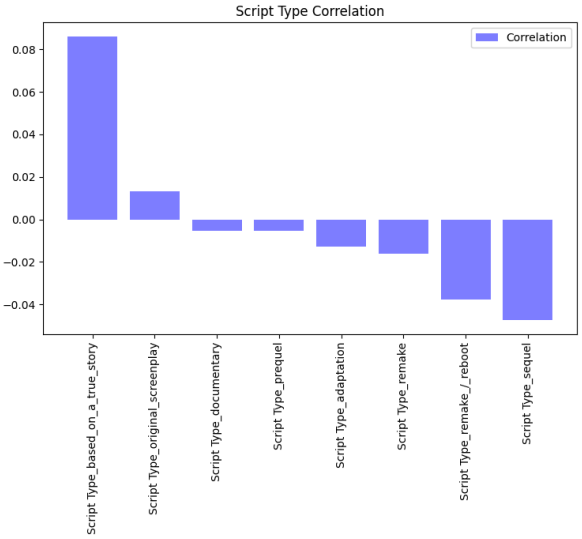
```
df['Rotten Tomatoes  critics'] = pd.to_numeric(df['Rotten Tomatoes  critics'])
df['Metacritic  critics'] = pd.to_numeric(df['Metacritic  critics'])
df['Metacritic Audience '] = pd.to_numeric(df['Metacritic Audience '])

df['Opening Weekend'] = pd.to_numeric(df["Opening Weekend"].replace(',', '',
regex=True))
df['Domestic Gross'] = pd.to_numeric(df["Domestic Gross"].replace(',', '',
regex=True))
df['Foreign Gross'] = pd.to_numeric(df["Foreign Gross"].replace(',', '',
regex=True))

df['Rotten Tomatoes vs Metacritic  deviance'] = pd.to_numeric(df['Rotten
Tomatoes vs Metacritic  deviance'])
df['Audience vs Critics deviance '] = pd.to_numeric(df['Audience vs Critics
deviance '])
df['Average critics '] = pd.to_numeric(df['Average critics '])
df['Average audience '] = pd.to_numeric(df['Average audience '])
df['Worldwide Gross'] = pd.to_numeric(df['Worldwide Gross'].replace(',', '',
regex=True))
df['Worldwide Gross'] = pd.to_numeric(df['Worldwide Gross'])

df[' of Gross earned abroad'] = pd.to_numeric(df[" of Gross earned
abroad"].replace('%', '', regex=True))
df[' Budget recovered'] = pd.to_numeric(df[" Budget recovered"].replace('%',
'', regex=True))
df[' Budget recovered opening weekend'] = pd.to_numeric(df[" Budget recovered
opening weekend"].replace('%', '', regex=True))
```

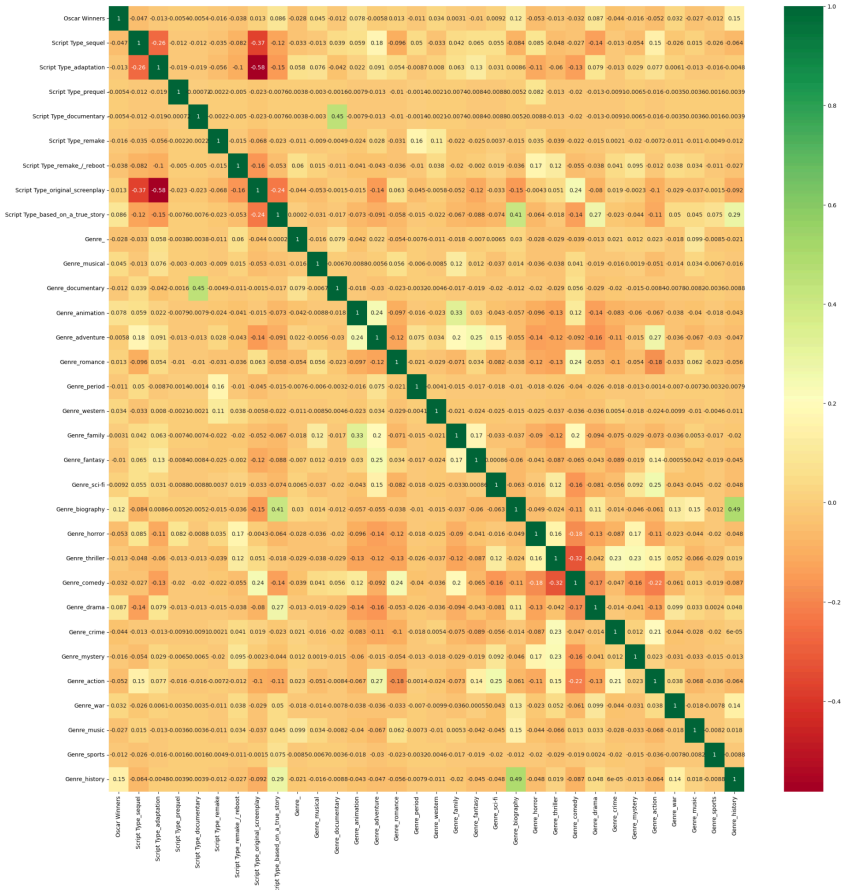
Έχοντας κάνει τα παραπάνω καταλήξαμε σε ένα dataset (1384x25) με κανένα missing value. Αποθηκεύουμε το dataset αυτό σε ένα xlsx για την εύκολη δοκιμή πειραμάτων στην συνέχεια.

Επιλογή Γνωρισμάτων:

Για την επιλογή των γνωρισμάτων που θα χρησιμοποιήσουμε τελικά, λάβαμε υπόψη διάφορες μετρικές όπως το feature importance και το correlation με τη στήλη Oscar Winners. Αρχικά αποφασίσαμε να δούμε το correlation των one hot encoding στηλών που κρατήσαμε, δηλαδή των genre, script type και distributor. Ακολουθούν τα σχετικά διαγράμματα:



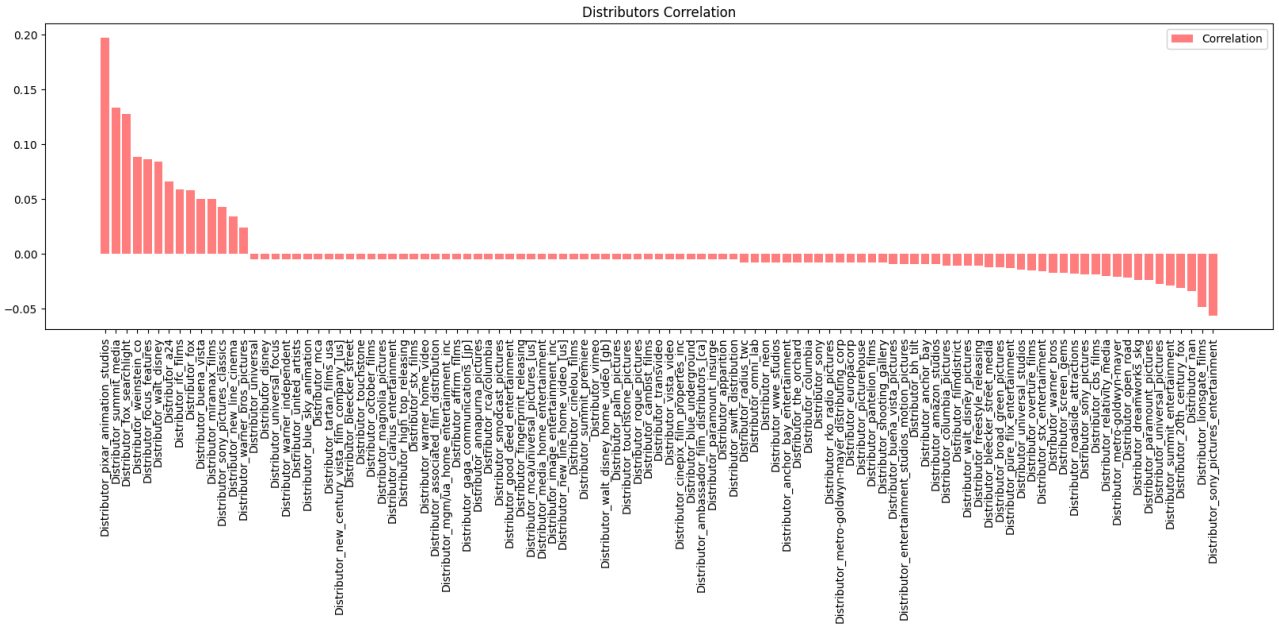




Από τα διαγράμματα αυτά μπορούμε εύκολα να συμπεράνουμε ότι:

- Για την επιλογή των genre, παίρνουμε τις στήλες ['Genre_history', 'Genre_biography', 'Genre_drama', 'Genre_animation'], καθώς αυτές έχουν το μεγαλύτερο (κατά απόλυτη τιμή) correlation.
- Για την επιλογή του script type, επιλέγουμε τη στήλη ['Script Type_based on a true story'], καθώς αυτή έχει το μεγαλύτερο (κατά απόλυτη τιμή) correlation.

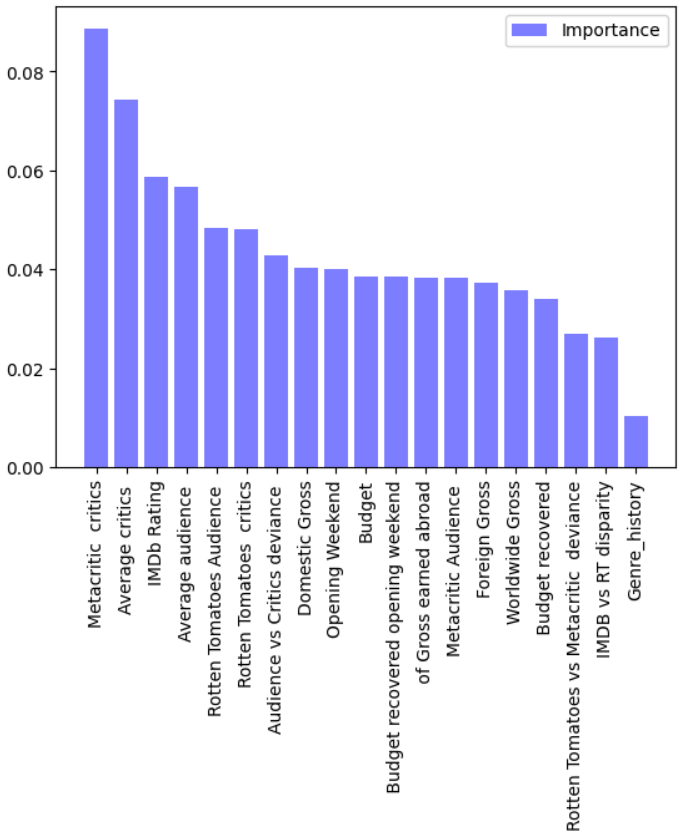
Ακολουθεί το correlation γράφημα για τους distributors:



Από το παραπάνω διάγραμμα βλέπουμε ότι οι στήλες ['Distributor_pixar_animation_studios', 'Distributor_summit_media', 'Distributor_fox_searchlight'] έχουν υψηλό (κατά απόλυτη τιμή) correlation με το oscar winners, οπότε της επιλέγουμε.

Έπειτα, για την επιλογή των numeric columns, υπολογίσαμε το feature importance των στηλών με βάση το μοντέλο RandomForest, μέσως της συνάρτησης calc_importance, η οποία επιστρέφει ένα dictionary με κλειδιά τα ονόματα των features και values τα αντίστοιχα feature importance scores.

Εκτελώντας την συνάρτηση αυτή, καταλήξαμε στο εξής γράφημα των feature importances:

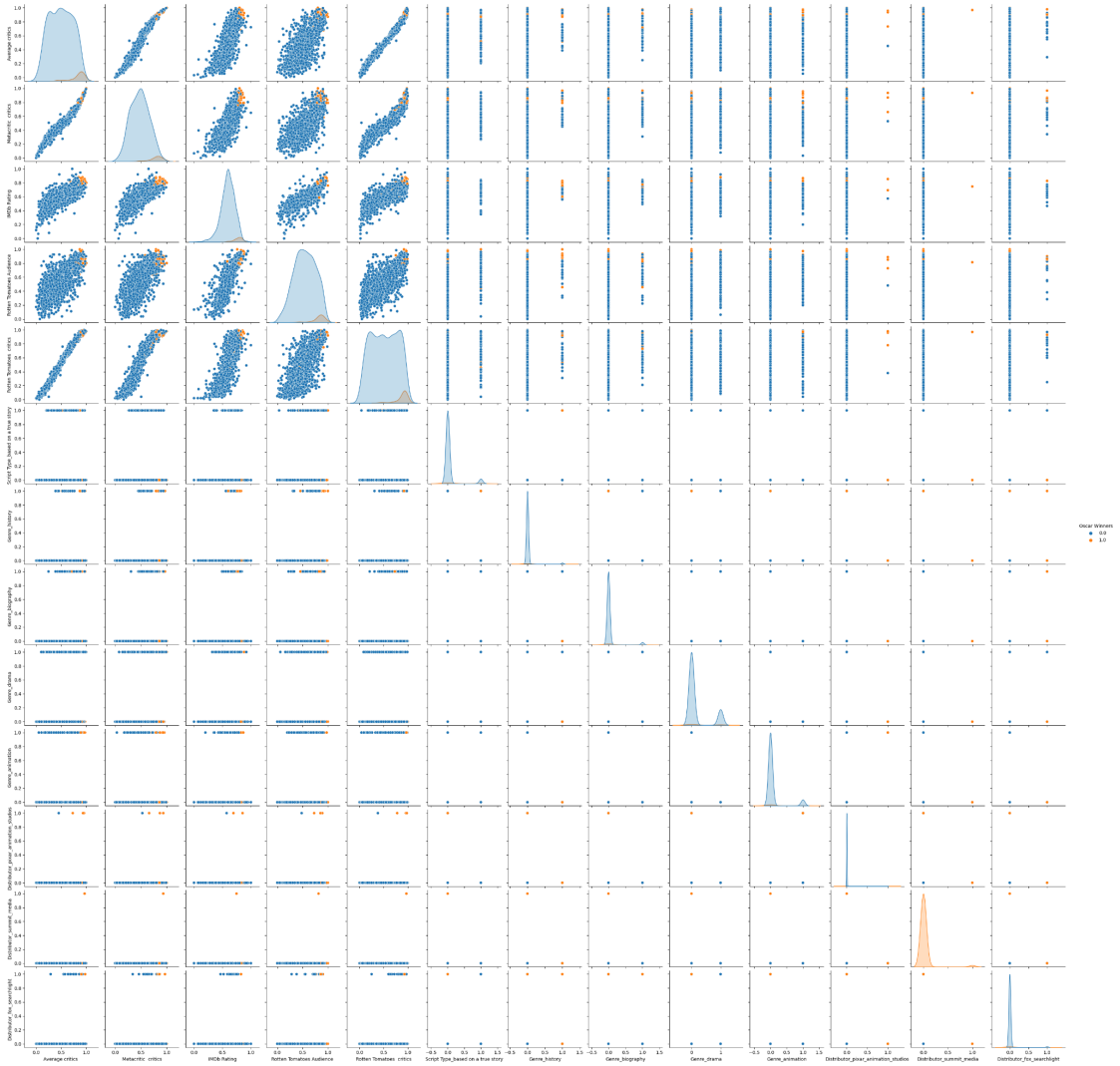


Παρατηρούμε ότι οι στήλες ['Average critics ', 'Metacritic critics', 'IMDb Rating', 'Rotten Tomatoes Audience ', 'Rotten Tomatoes critics', 'Average audience '] είναι υψηλής σημαντικότητας. Μετά από πειραματισμό όμως, τις επιλέξαμε όλες εκτός από την 'Average audience ', καθώς (οι υπόλοιπες) κάνουν καλύτερες προβλέψεις στο train / test set.

Συνεπώς οι στήλες που επιλέγουμε είναι: ['Average critics ','Metacritic critics','IMDb Rating','Rotten Tomatoes Audience ', 'Oscar Winners', 'Rotten Tomatoes critics', 'Script Type_based on a true story', 'Genre_history', 'Genre_biography', 'Genre_drama', 'Genre_animation', 'Distributor_pixar_animation_studios', 'Distributor_summit_media', 'Distributor_fox_searchlight']. Οπότε τις επιλέγουμε και έπειτα κανονικοποιούμε τα values με τον MinMaxScaler.

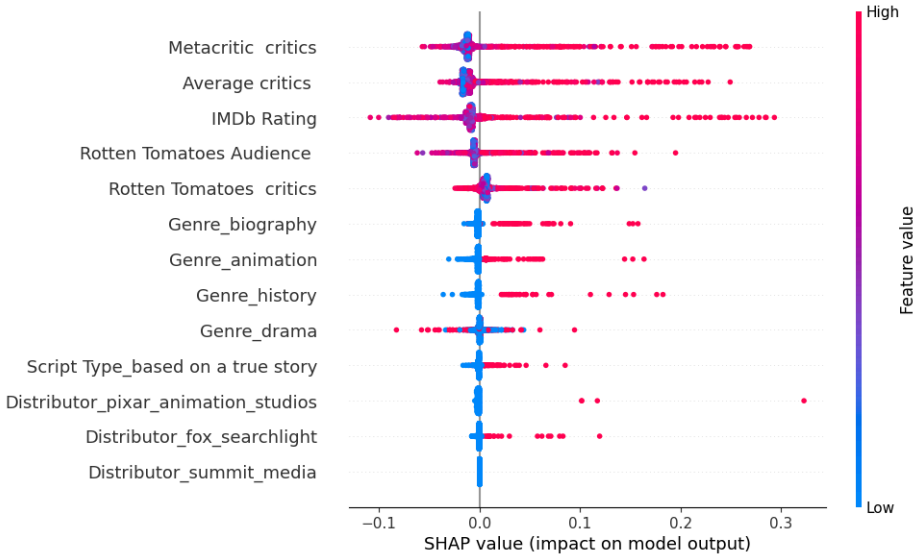
Για να οπτικοποιήσουμε τα επιλεγμένα features, δημιουργήσαμε τα ακόλουθα γραφήματα:

Pairplot:



Παρατηρούμε ότι τα επιλεγμένα features δίνουν έναν σχετικά καλό διαχωρισμό στις ταινίες που έχουν πάρει oscar και σε αυτές που δεν έχουν πάρει oscar (μπορούμε να δούμε ότι στα περισσότερα, οι ταινίες με oscar βρίσκονται πάνω δεξιά). Χρειάζεται όμως να αναφερθεί ότι ο διαχωρισμός αυτός δεν είναι τόσο “καθαρός” καθώς οι ταινίες “μπλέκονται” μεταξύ τους.

SHAP summary plot των επιδράσεων των γνωρισμάτων για όλο το dataset:



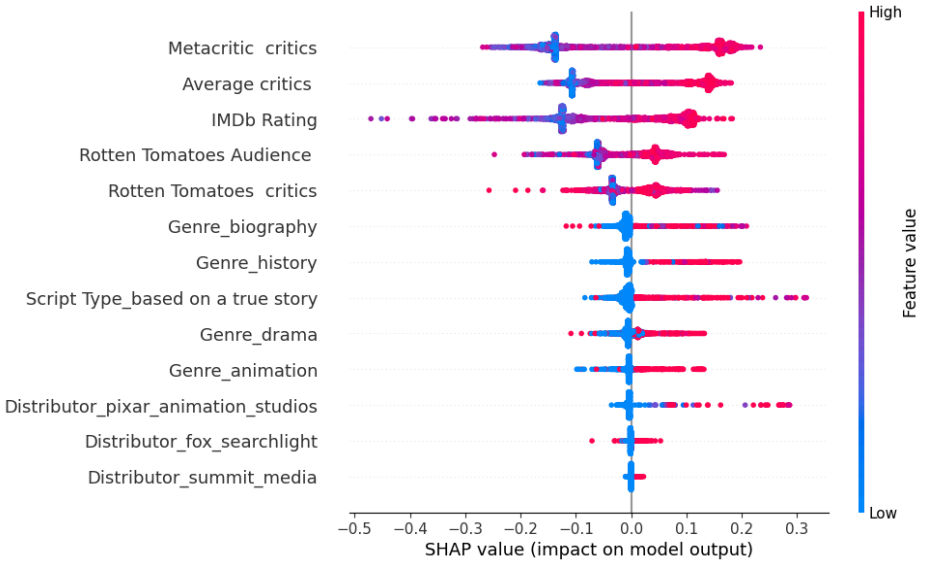
Τα πιο σημαντικά features (σύμφωνα με το SHAP διάγραμμα αυτό) είναι αυτά που δεν είναι one-hot encodings (πχ Metacritic critics, Average critics κ.α). Βλέπουμε ότι όσο είναι "μεγάλα" αυτά τα values αυξάνουν (κυρίως) το αποτέλεσμα SHAP, δηλαδή τη πιθανότητα η ταινία να έχει πάρει oscar. Όταν όμως έχουν μεσαίες τιμές ή χαμηλώνουν μειώνεται (κυρίως) και η πιθανότητα να πάρουν oscar (στη πρόβλεψη).

Παρατηρούμε επίσης ότι για features με one-hot encodings (πχ Genre_biography, Genre_history κ.α) όταν είναι 1 (ψηλό value) αυξάνουν τη πιθανότητα να προβλεφθεί oscar (αντίστροφα όταν μειώνουμε το value των one-hot encodings). Είναι σημαντικό να σημειωθεί ότι η "ίδια" συμπεριφορά αναδεικνύεται και στα correlation plots, καθώς έχουν θετικό correlation με το oscar winners column και άρα μία "ομόρροπη" σχέση.

Αφού επιλέξαμε τα features, βάλαμε στόχο να ισορροπησουμε το dataset, έτσι ώστε το μοντέλο να μπορέσει να “δει” περισσότερα παραδείγματα oscar ταινιών (αφού στο αρχικό, οι ταινίες με oscar ήταν περίπου το 4% όλων των ταινιών του dataset). Έτσι, το μοντέλο θα μπορούσε να προβλέψει καλύτερα τις ταινίες με oscar.

Για αυτό χρησιμοποιήσαμε την τεχνική SMOTE (Synthetic Minority Oversampling Technique) η οποία δημιουργεί συνθετικά δείγματα από την κλάση μειοψηφίας (ταινίες με oscars) για να εξισορροπήσει τα δεδομένα. Η τεχνική SMOTE φτιάχνει τα συνθετικά δεδομένα αυτά παίρνοντας τις ευκλείδειες αποστάσεις με τους κοντινότερους γείτονες, για αυτό η κανονικοποίηση των δεδομένων πριν την εφαρμογή της είναι απαραίτητη. Μετά από το SMOTE, το dataset αυξήθηκε σε μέγεθος (2660, 14) και τα μισά από αυτά τα rows είναι ταινίες με oscar.

SHAP summary plot των επιδράσεων (μετά το SMOTE):



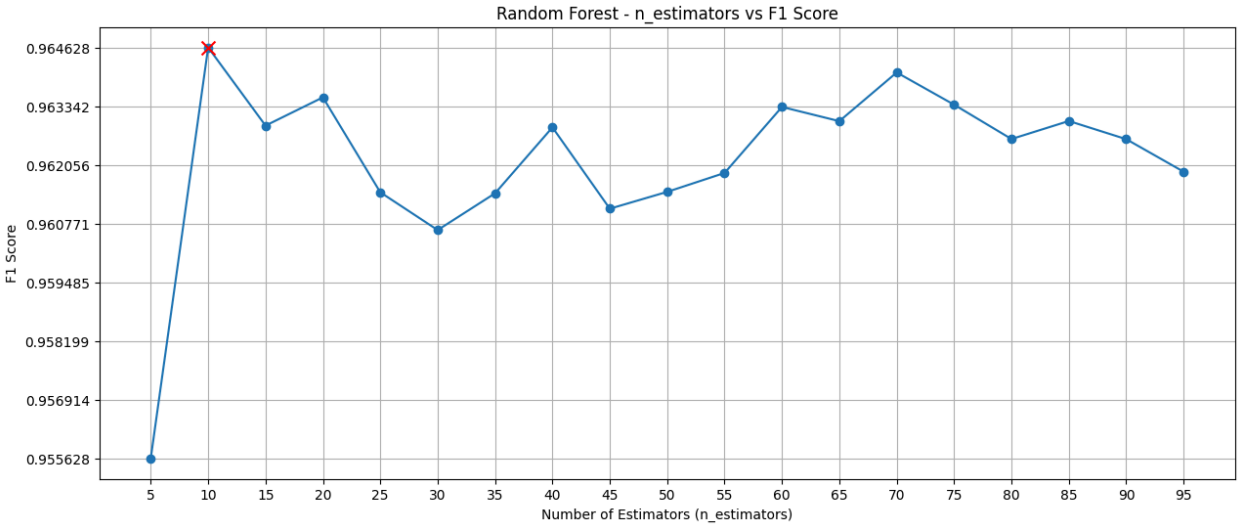
Κάνοντας το SHAP γράφημα ακόμη μία φορά (μετά από το SMOTE) παρατηρούμε ότι τα αποτελέσματα είναι κοντά στο αρχικό SHAP (έχοντας όμως παραπάνω "δείγματα" από ταινίες με oscars). Τέλος αποθηκεύουμε το dataset (με το SMOTE) στο moviesNew3.xlsx για να το χρησιμοποιήσουμε αργότερα στο train/test.

Προβλέψεις & Αξιολόγηση Μοντέλων:

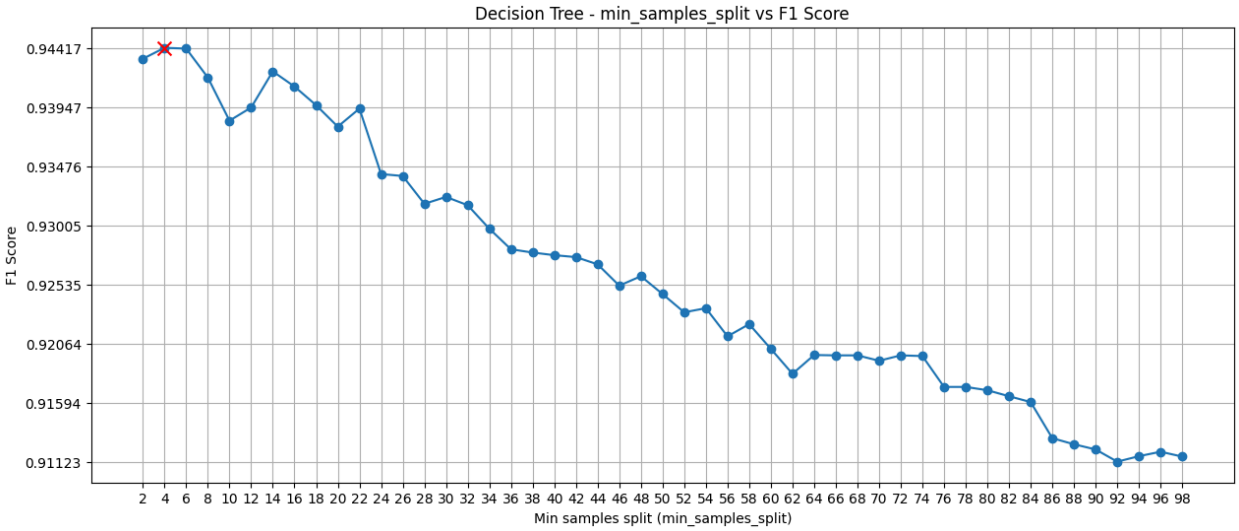
Για τις προβλέψεις συγκρίναμε 3 μοντέλα: το random forest, logistic regression και decision tree.

Αρχικά, διαβάζουμε το από το moviesNew3.xlsx της προετοιμασίας. Για την αποφυγή overfit και underfit κάνουμε 10-cross validation στις μεθόδους μας. Με τη χρήση επίσης της GridSearchCV μπορούμε να βρούμε τις καλύτερες παραμέτρους για το μοντέλο μας κάνοντας ουσιαστικά πολλά 10-cross validation. Για την αξιολόγηση των παραμέτρων χρησιμοποιούμε το fmeasure. Ακολουθούν τα διαγράμματα επίδοσης των μοντέλων random forest και decision tree για διάφορες τιμές των παραμέτρων n_estimators και min_samples_split αντίστοιχα.

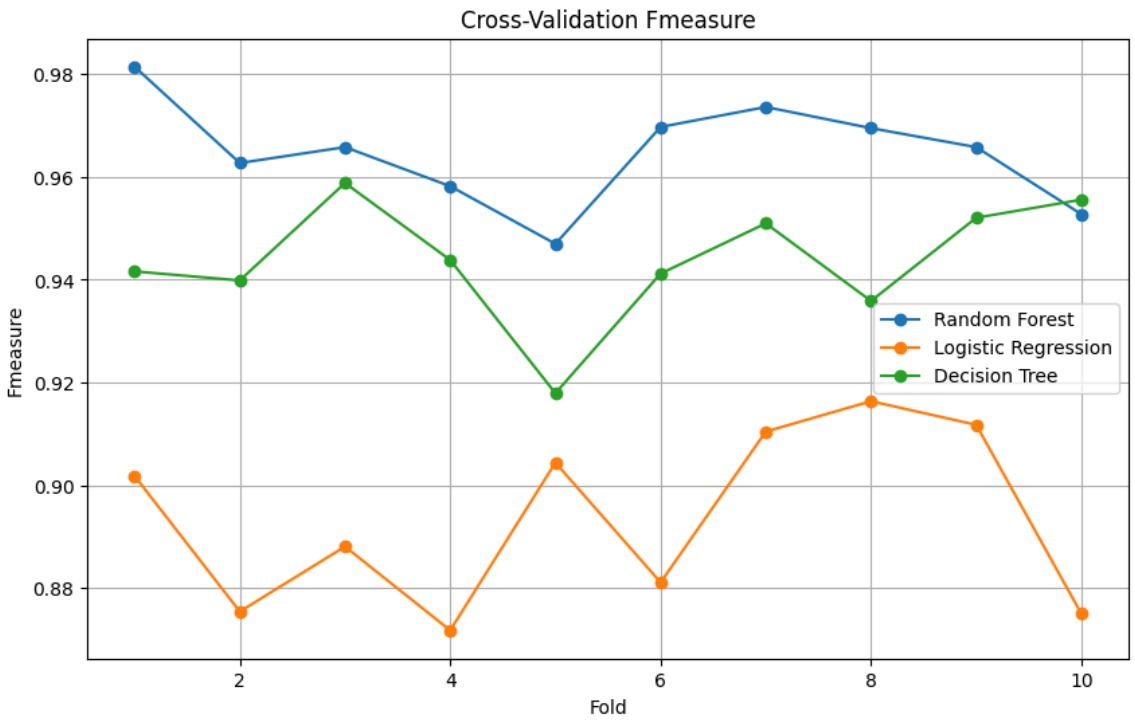
Random Forest:



Decision Tree:



Για seed = 42, οι καλύτεροι παράμετροι του random forest είναι n_estimators = 10 ενώ για το decision tree το καλύτερο min_samples_split είναι 4. Για το logistic regression δεν βάζουμε κάποια παράμετρο. Έπειτα κάνουμε ένα ακόμη 10-cross validation με τις καλύτερες παραμέτρους και βάζουμε όλα τα Fmeasure των μοντέλων στο παρακάτω plot:



Επίσης εκτυπώνουμε τα μέσα Fmeasure του παραπάνω διαγράμματος:

```
Average Random Forest Fmeasure: 0.9646277540782844
Average Logistic Regression Fmeasure: 0.8936251285334764
Average Decision Tree Fmeasure: 0.9437541633353028
```

Παρατηρώντας τα διαγράμματα και παίρνοντας τα μέσα Fmeasure, το ψηλότερο είναι το random forest (με n_estimators=10), οπότε αυτό επιλέγουμε για τις επόμενες προβλέψεις.

Χρησιμοποιώντας το random forest με seed = 42 και n_estimators = 235, εκπαιδεύουμε ένα μοντέλο και βλέπουμε πως πήγαν οι προβλέψεις. Τα αποτελέσματα που προέκυψαν από την εκπαίδευση είναι:

```
Mean Squared Error: 0.03634085213032581
[[377  13]
 [ 16 392]]

              precision    recall  f1-score   support

           0       0.96      0.97      0.96       390
           1       0.97      0.96      0.96       408
    accuracy                           0.96       798
   macro avg       0.96      0.96      0.96       798
```

```
weighted avg       0.96      0.96      0.96       798

Roc auc score 0.9637254901960786
F1 Score: 0.964329643296433
```

Στον παραπάνω πίνακα (confusion matrix) βλέπουμε πως το μοντέλο μας έχει προβλέψει σωστά ως αρνητικά 377. Έχει προβλέψει λανθασμένα ως θετικά 13. Έχει προβλέψει λανθασμένα ως αρνητικά 16 και τέλος έχει προβλέψει σωστά ως θετικά 377.
Επίσης, το 97% των γραμμών που το μοντέλο μας τα προβλέπει ως κλάση 1, πράγματι είναι κλάση 1 και αντίστοιχα το 96% των γραμμών που το μοντέλο μας ταξινομεί ως κλαση 0, είναι πράγματι κλάση 0.

Όσον αφορά το precision:

- Για την κλάση 0: 0.96. Αυτό σημαίνει ότι το 96% των παραδειγμάτων που το μοντέλο τα προβλέπει ως κλάση 0, πράγματι είναι κλάση 0.
- Για την κλάση 1: 0.97. Αυτό σημαίνει ότι το 97% των παραδειγμάτων που το μοντέλο τα προβλέπει ως κλάση 1, πράγματι είναι κλάση 1.

Όσον αφορά το recall:

- Για την κλάση 0: 0.97. Αυτό σημαίνει ότι το 97% των πραγματικών παραδειγμάτων της κλάσης 0, εντοπίζονται σωστά από το μοντέλο.
- Για την κλάση 1: 0.96. Αυτό σημαίνει ότι το 96% των πραγματικών παραδειγμάτων της κλάσης 1, εντοπίζονται σωστά από το μοντέλο.

Για το F1-score (Είναι ένας συνδυασμός του precision και του recall.)

- Για την κλάση 0: 0.96
- Για την κλάση 1: 0.96

Accuracy: Αυτό σημαίνει ότι το 96% των συνολικών παραδειγμάτων ταξινομούνται σωστά από τον ταξινομητή.

Επίσης από το Roc Auc Score και F1 Score, βλέπουμε ότι το μοντέλο κάνει πολύ καλή δουλειά στην ταξινόμηση των δεδομένων (περίπου 96%).

Συνεπώς το μοντέλο ταξινομεί πολύ καλά τις ταινίες σε οσκαρικές ή μη οσκαρικές.

Δοκιμή σε Δείγμα Ανώνυμων Δεδομένων:

Αφού επιλέξουμε το καλύτερο μοντέλο (random forest) προχωράμε στη πρόβλεψη των oscar winners για το δείγμα των ανώνυμων δεδομένων. Διαβάζουμε το ανώνυμο dataset και μετατρέπουμε τις στήλες του έτσι ώστε να μοιάζουν με τα δεδομένα του movies.xlsx που χρησιμοποιήθηκαν για το training. Όσες one hot encoding στήλες υπήρχαν στο trainning αλλά δεν υπάρχουν στα ανώνυμα δεδομένα, τις δημιουργούμε (στο ανώνυμο) και τις γεμίζουμε με 0 (πχ το 'Distributor_pixar_animation_studios' που δεν υπάρχει στο ανώνυμο dataset, το

προσθέτουμε και το γεμίζουμε με 0). Επίσης συγκεκριμένα για τους distributors, είδαμε ότι στα ανώνυμα δεδομένα υπήρχαν οι εταιρείες (όπως το trainnng) αλλά γραμμένες αλλιώς, για αυτό κάναμε τις απαραίτητες ονομασίες για να είναι τα values (και έπειτα τα one hot encodings) ίδια με αυτά στο trainning (για παράδειγμα το train dataset είχε το value summit_media ενώ το ανώνυμο dataset είχε το summit_entertainment, οπότε απλώς μετονομάσαμε το summit_entertainment σε summit_media για να προκύψει ύστερα έτσι το αντίστοιχο one hot encoding).
Έπειτα φροντίσαμε όλα τα δεδομένα του ανώνυμου dataset να είχαν τους ίδιους τύπους με το train dataset (πχ numeric και όχι categorical) και γεμίσαμε τα nans των στηλών είτε με το Μ.Ο από προηγούμενα values της στήλης είτε με πιο συγκεκριμένους τύπους που χρησιμοποιήσαμε και στο train dataset (πχ για να γεμίσουμε τα nans της στήλης 'IMDb Rating' στο ανώνυμο dataset, πήραμε τη μέση διαφορά μεταξύ Average Audience και IMDb Rating, και την αφαιρούμε από το Average Audience της γραμμής – όπως κάναμε και στο train). Τέλος, επιλέγουμε τις στήλες που πήραμε στο train, μετατρέπουμε τις στήλες Distributor, Genre και Script Type σε one hot με τη χρήση της συνάρτησης get_dummies_genre (όπως και στο train) και κάνουμε κανονικοποίηση των δεδομένων με τη χρήση του MinMaxScaler (όπως κάναμε και στο train). Το τελικό dataset είναι ένα dataset που έχει δημιουργηθεί με τις ίδιες διαδικασίες και έχει τα ίδια columns με το train dataset.

Έχοντας κάνει λοιπόν όλη αυτή τη προετοιμασία, συνεχίζουμε στην εκπάιδευση του μοντέλου random forest με n_estimarors = 10 (όπως είχαμε επιλέξει και πριν). Για την εκπαίδευση χρησιμοποιούμε όλο το train dataset από πριν.

Τελικά προβλέψαμε ότι 57 ταινίες πήραν oscar στο anon dataset. Ακολουθούν τα αποτελέσματα της πρόβλεψής μας (όπως είναι στο 1st round results στο eclass):

```
Omada Xriston 13 (it22064@hua_gr).csv
Oscars Predicted: 57
Precision: 0.14035087719298245
Recall: 0.7272727272727273
F-measure: 0.23529411764705882
Coverage: 1.0
Accuracy: 0.9079646017699115
```

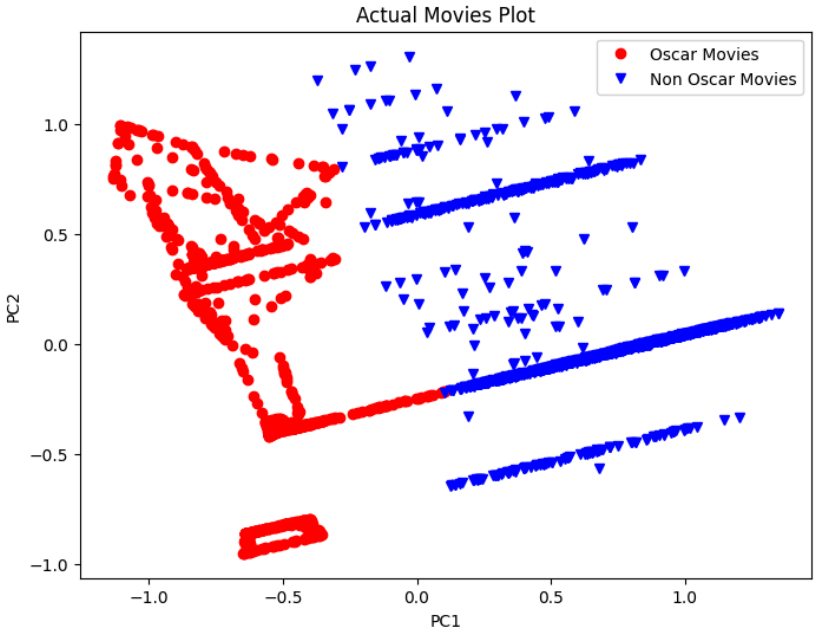
Συσταδοποίηση:

Για την συσταδοποίηση διαβάζουμε το SMOTE dataset που χρησιμοποιήθηκε στην εκπαίδευση και αξιολόγηση. Ύστερα μειώνουμε τις διαστάσεις σε 2 χρησιμοποιώντας το PCA για την καλύτερη εύρεση των clusters.

```
import pandas as pd
df = pd.read_excel('/content/drive/My Drive/DataMining/moviesNew3.xlsx')
X = df
y = df['Oscar Winners']
seed = 42
```

```
# PCA:
from sklearn.decomposition import PCA
pca = PCA(n_components = 2)
X = pca.fit(X).transform(X)
dfpca =pd.DataFrame(X,columns=['PC1','PC2'])
df = pd.concat([dfpca, y],axis=1)
```

Αρχικά, κάνουμε plot τα δεδομένα όπως είναι στο dataset. Με κόκκινο χρώμα βάζουμε τις ταινίες με Oscar και με μπλε τις ταινίες χωρίς Oscar:



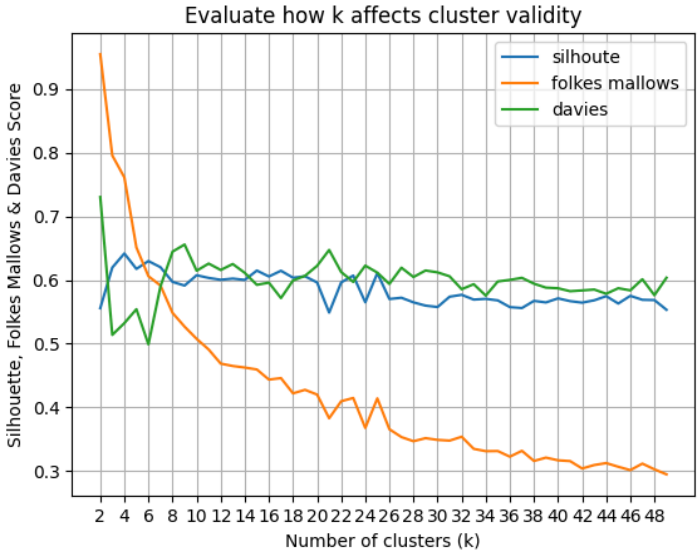
Ήδη παρατηρούμε ότι οι ταινίες είναι κάπως χωρισμένες σε 2 ομάδες. Δεξία και κάτω είναι αυτές με oscar ενώ αριστερά και πάνω είναι αυτές χωρίς (oscar).

Για την επιλογή του αριθμού των cluster παίρνουμε τις μετρικές silhouette, fowlkes mallows και davies bouldin.

Επίσης, φτιάξαμε και την συνάρτηση print_cluster_scores η οποία εκτυπώνει αυτά τα scores μαζί με το Adjusted Rand Index Score του clustering (θα την χρησιμοποιήσουμε αργότερα).

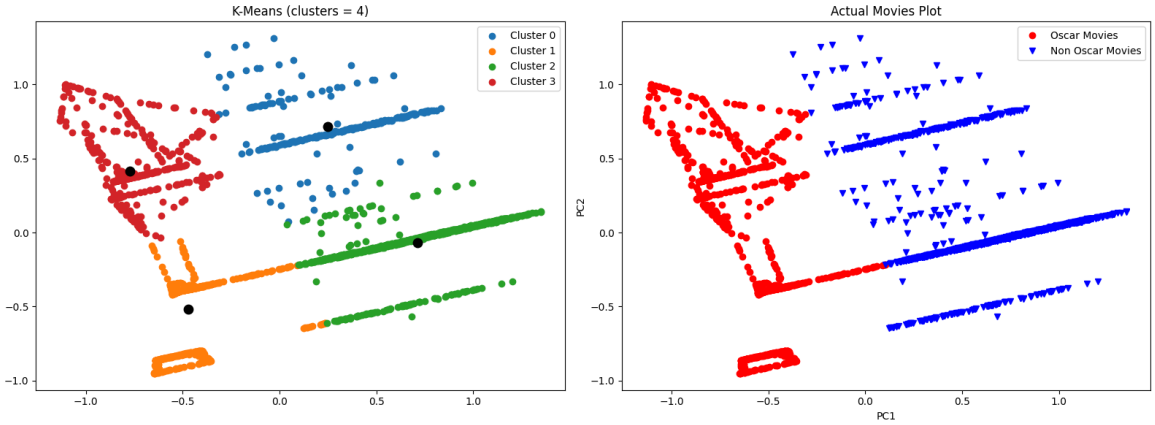
Kmeans:

Για να αποφασίσουμε τον καλύτερο αριθμό cluster για το Kmeans, δημιουργούμε το ακόλουθο διάγραμμα με τα scores που αναφέρθηκαν:



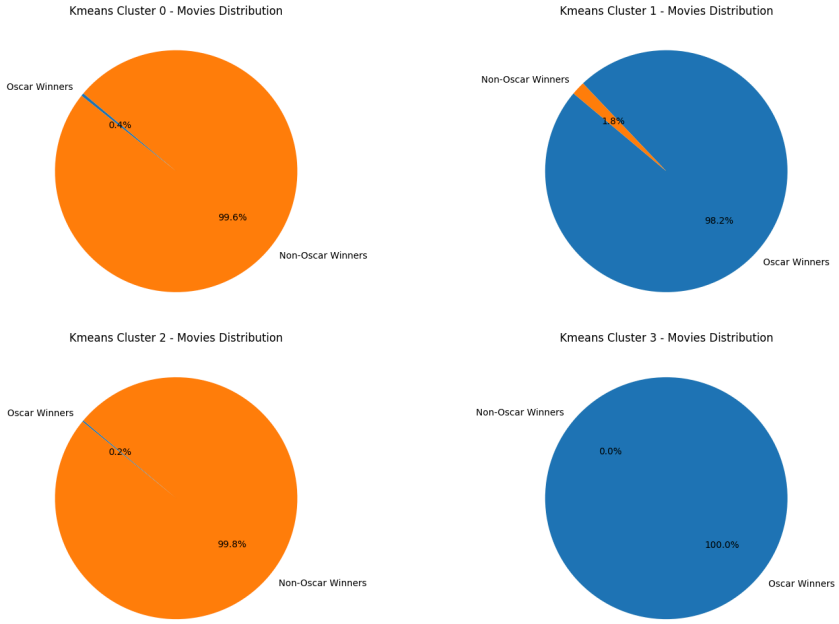
Στην αρχή παρατηρούμε ότι το folkes mallows είναι πολύ υψηλό. Αλλά το davies boulding είναι επίσης το ψηλότερο που υπάρχει, για αυτό και δεν επιλέγουμε k = 2 ή 3. Βλέποντας ολόκληρο το σχήμα, παρατηρούμε ότι για k = 4 το silhouette είναι το ψηλότερο που υπάρχει, το folkes mallows είναι ψηλό και το davies έχει αρκετά χαμηλή τιμή. Για k = 6 παρατηρούμε μία αντίστοιχη συμπεριφορά, όμως το folkes mallows είναι περισσότερο μειωμένο από το k = 4. Στην συνέχεια των k βλέπουμε ότι τα scores folkes mallows και silhouette όλο και μειώνονται, ενώ το davies αυξάνεται κάτι που δεν είναι επιθυμητό. Για αυτό καταλήξαμε ότι η καλύτερη επιλογή είναι k = 4.

Για να δούμε πόσο καλά τα πήγε το clustering, συγκρίνουμε τα clustered data με τα πραγματικά μέσω του παρακάτω plot:



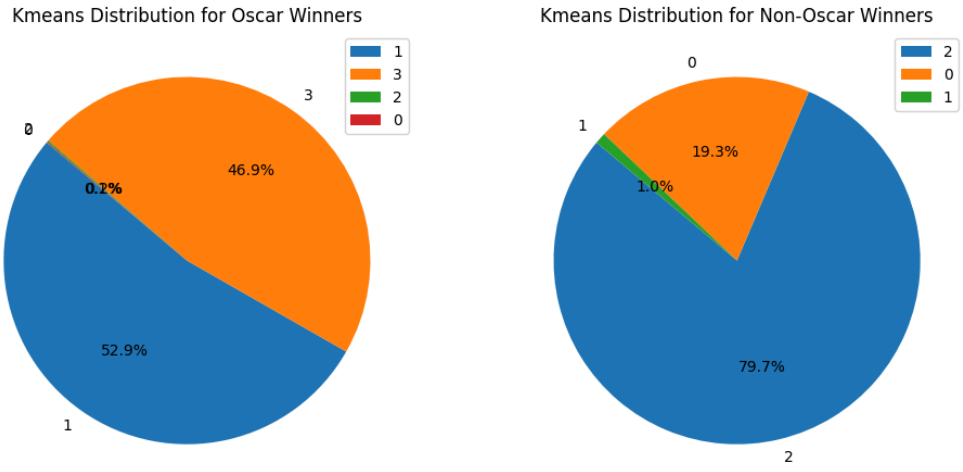
Παρατηρούμε ότι έχει γίνει πολύ καλή ομαδοποίηση των δεδομένων. Συγκεκριμένα, σχεδόν όλες οι ταινιές στο cluster 1 και cluster 3 μπορούν να θεωρηθούν ως οι ταινίες με oscar, ενώ τα clusters 2, 3 είναι οι ταινίες χωρίς oscar.

Η παρατήρηση αυτή επιβεβαιώνεται αν δούμε το piechart:



Όπως παρατηρήσαμε και πριν, βλέπουμε ότι το cluster 3 έχει μόνο ταινίες με oscars και στο cluster 1 το 98% είναι ταινίες με oscar. Στα clusters 0, 2 βρίσκονται κυρίως ταινίες χωρίς oscar.

Ακολουθεί και το piechart των ταινιών (και σε ποια cluster ανήκουν):



Με βάση αυτά τα αποτελέσματα βλέπουμε ότι οι σχεδόν όλες οι ταινίες με oscar (περίπου το 99.8%) βρίσκεται στα clusters 1, 3 ενώ σχεδόν το 99% των ταινιών χωρίς oscar βρίσκεται στα clusters 0, 2. Όπως και πριν φαίνεται ότι στο cluster 3 υπάρχουν μόνο ταινίες με oscar.

Εκτυπώνοντας τα score, βλέπουμε τα εξής:

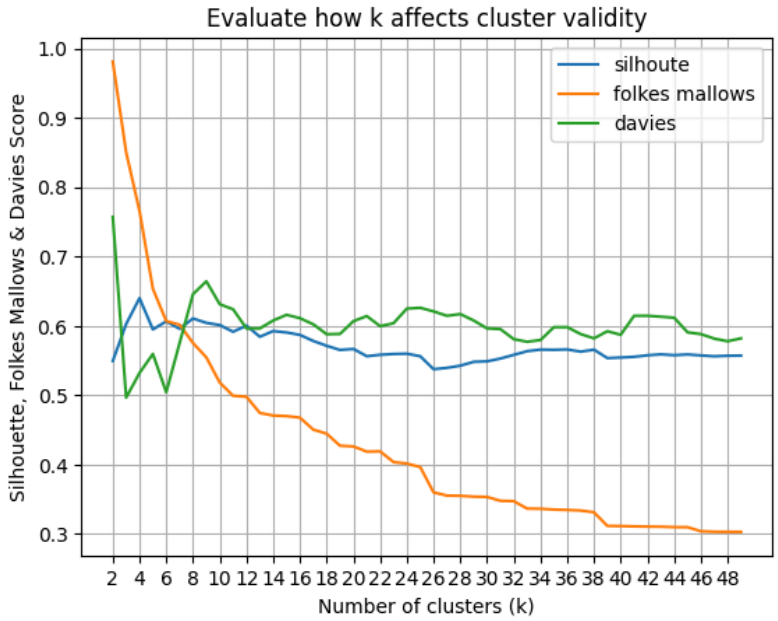
```
Silhouete Score: 0.6416936239627882
Adjusted Rand Index (ARI): 0.5793397831624262
Fowlkes-Mallows Score: 0.7611292608123397
```

Σύμφωνα με το ARI συμπεραίνουμε πως το επίπεδο clustering είναι καλό αφού αν ARI = 1 τότε σημαίνει τέλειο clustering (δηλαδή τέλεια “συμφωνία” μεταξύ των προβλεπόμενων clusters και των πραγματικών τιμών) ενώ αν ARI < 0 σημαίνει τυχαίο (ή χειρότερο) clustering.

Το Silhouette score περιγράφει πόσο πολύ ένα αντικείμενο μίας κλάσης ταιριάζει στο χώρο γύρω του (γειτονιά του). Παίρνει τιμές από -1 ως 1. Όσο πιο κοντά στο 1 είναι η τιμή του, τόσο το καλύτερο για το μοντέλο μας. Συνεπώς η τιμή 0.641 μας δίνει την ασφάλεια πως τα αντικείμενα του εκάστοτε cluster ταιριάζουν ικανοποιητικά μεταξύ τους.

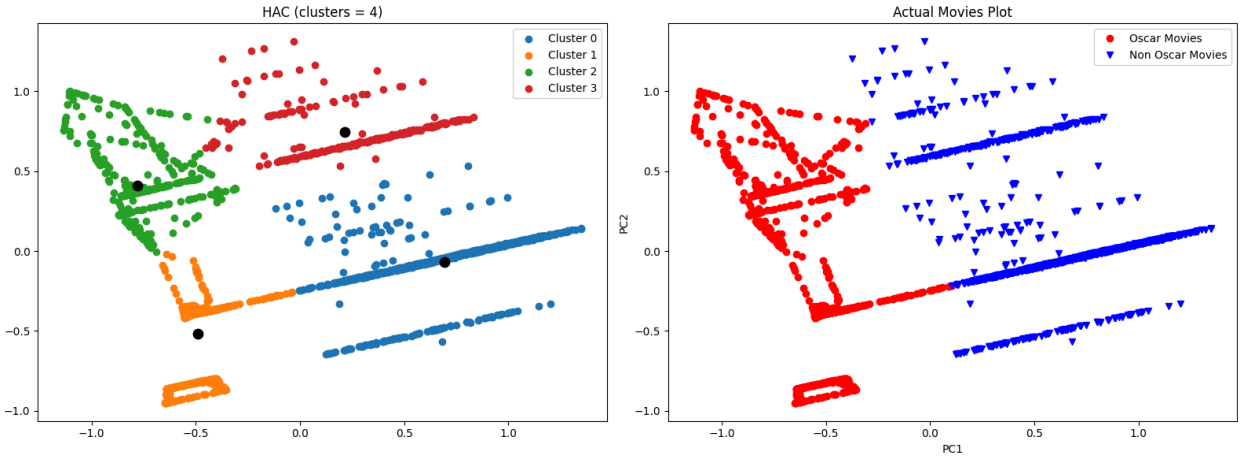
Το Fowlkes-Mallows Score είναι μια μετρική (όπως το ARI) που μετράει κατά πόσο τα predicted clusters είναι κοντά στα πραγματικά data (χρησιμοποιώντας συνδυαστικά το precision και το recall). Όσο μεγαλύτερο είναι, τόσο το καλύτερο για το μοντέλο μας. Συνεπώς το 0.76 είναι ένα νούμερο που μας εξασφαλίζει ότι το clustering είναι καλό.

<u>HAC:</u>
Αντίστοιχα για το HAC, κάνουμε το παρακάτω διάγραμμα:



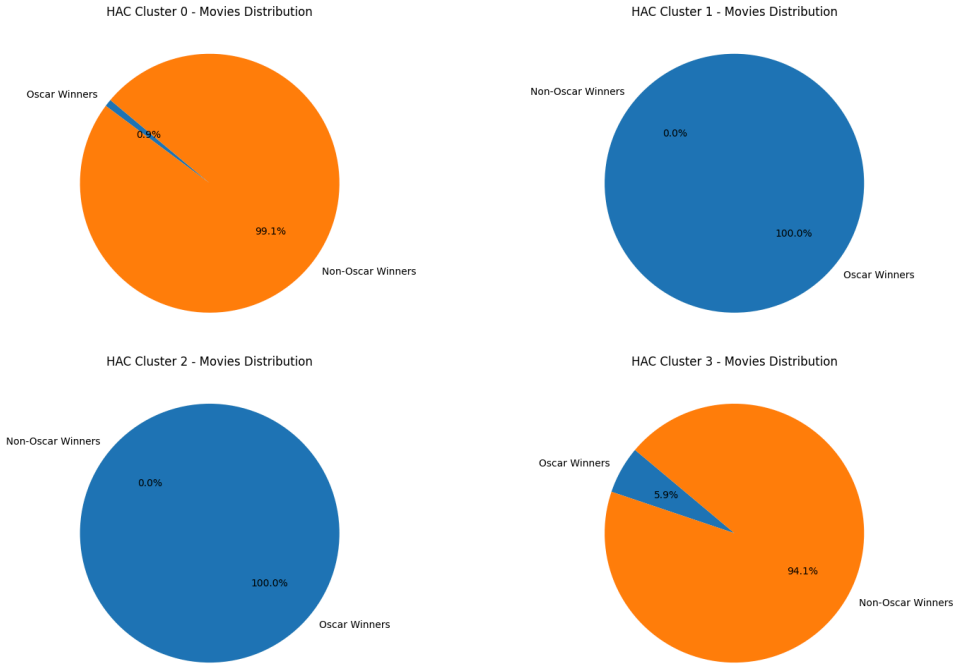
Πάλι παρατηρούμε ότι η ψηλότερη τιμή του silouette score βρίσκεται στο k = 4. Επίσης, για k = 4, το davies είναι σχετικά χαμηλό και το folkes mallows υψηλό, οπότε και επιλέγουμε k = 4.

Για να δούμε πόσο καλά τα πήγε το clustering, συγκρίνουμε τα clustered data με τα πραγματικά μέσω του παρακάτω plot:



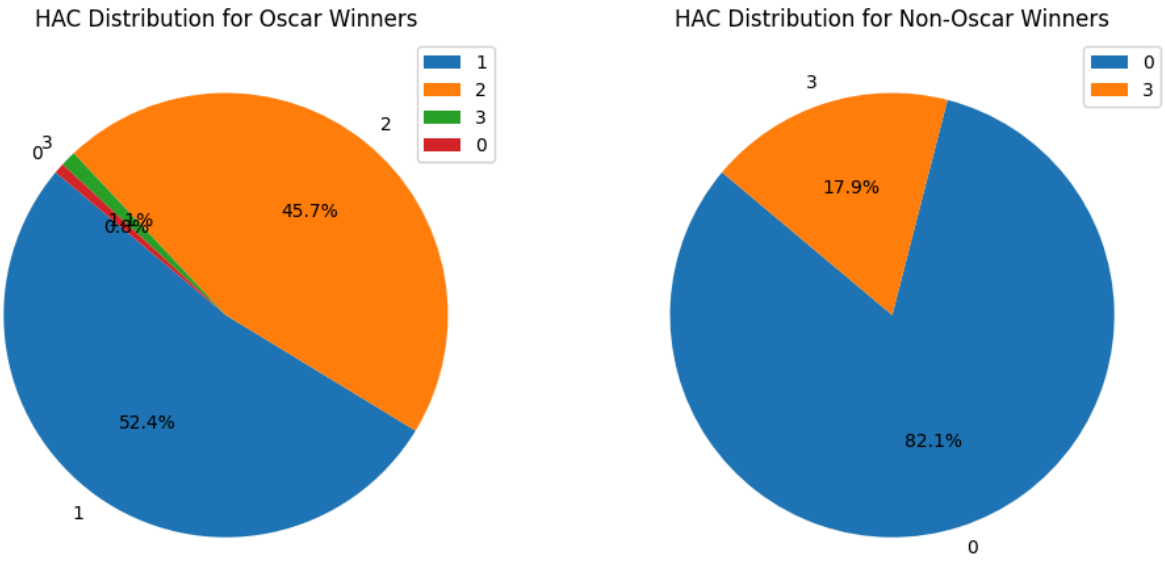
Παρατηρούμε ότι πάλι έχει γίνει πολύ καλή ομαδοποίηση των δεδομένων. Θα μπορούσαμε ξανά να πούμε ότι σχεδόν όλες οι ταινίες με oscar βρίσκονται σε 2 clusters (στο cluster 1, 2) ενώ οι υπόλοιπες βρίσκονται στα clusters 0, 3. Βέβαια, μπορούμε επίσης να δούμε ότι το cluster 0, 3 έχει και κάποιες ταινίες με oscar στα "'όρια" τους αριστερά.

Οι παρατηρήσεις αυτές επιβεβαιώνονται αν δούμε το piechart:



Παρατηρούμε ότι το 100% των clusters 1, 2 αποτελείται από ταινίες με oscar. Τα cluster 0, 3 από την άλλη έχουν κυρίως ταινίες χωρίς oscar, αλλά υπάρχουν και μερικές ταινίες με oscar σε αυτά.

Ακολουθεί και το piechart των ταινιών (και σε ποια cluster ανήκουν):



Με βάση αυτά τα αποτελέσματα βλέπουμε ότι το 100% των ταινιών χωρίς oscar βρίσκονται στα clusters 0, 3 ενώ σχεδόν όλες οι ταινίες με oscar (περίπου το 98.1%) βρίσκονται στα clusters 1, 2 (όπως επισημάνθηκε και πριν).

Εκτυπώνοντας τα score βλέπουμε:

```
Silhouete Score: 0.6400645055001494
Adjusted Rand Index (ARI): 0.5865672567916242
Fowlkes-Mallows Score: 0.7658883670748172
```

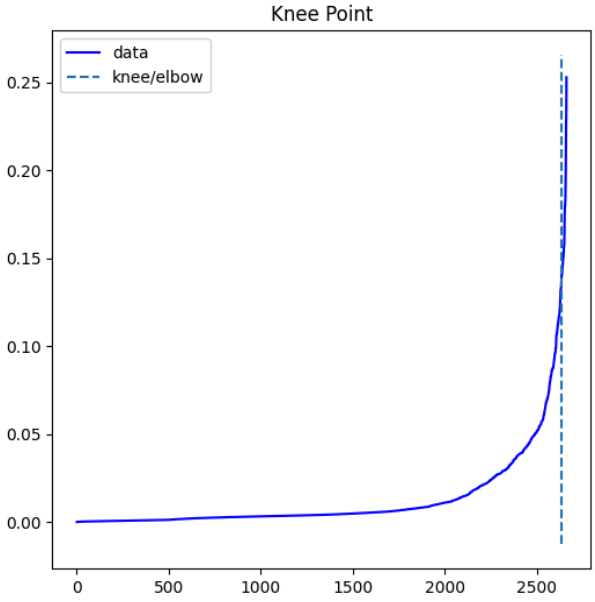
Το ARI score λίγο ψηλότερο από το Kmeans, δείχνοντας ότι το HAC έχει προβλέψει τα clusters καλύτερα και πιο κοντά στα πραγματικά δεδομένα (ομοίως και το fowlkes-mallows). Το silhouette score από την άλλη όμως είναι ελάχιστα πιο χαμηλό από το Kmeans, δείχνοντας ότι τα δεδομένα εντός των HAC clusters δεν είναι τόσο “όμοια” μεταξύ τους (σε αντίθεση με το Kmeans).

DBSCAN:

Για το DBSCAN δεν επιλέγουμε εμείς τον αριθμό των clusters, αλλά πρέπει να επιλέξουμε το epsilon (την ακτίνα γύρω από ένα σημείο δεδομένων εντός του οποίου ο αλγόριθμος αναζητά γειτονικά σημεία δεδομένων). Συγκεκριμένα, εάν η απόσταση μεταξύ δύο σημείων είναι μικρότερη ή ίση με eps, θεωρούνται γείτονες. Επίσης, χρειάζεται να ορίσουμε το min_samples (ο ελάχιστος αριθμός σημείων δεδομένων που απαιτούνται για να σχηματιστεί ένας πυρήνας). Τα min_samples ισούνται συνήθως με το διπλάσιο του αριθμού των features (σύμφωνα με αυτή τη πηγή). Και λόγω του PCA από πριν, τα features είναι 2, ορίζουμε το min_samples = 4.

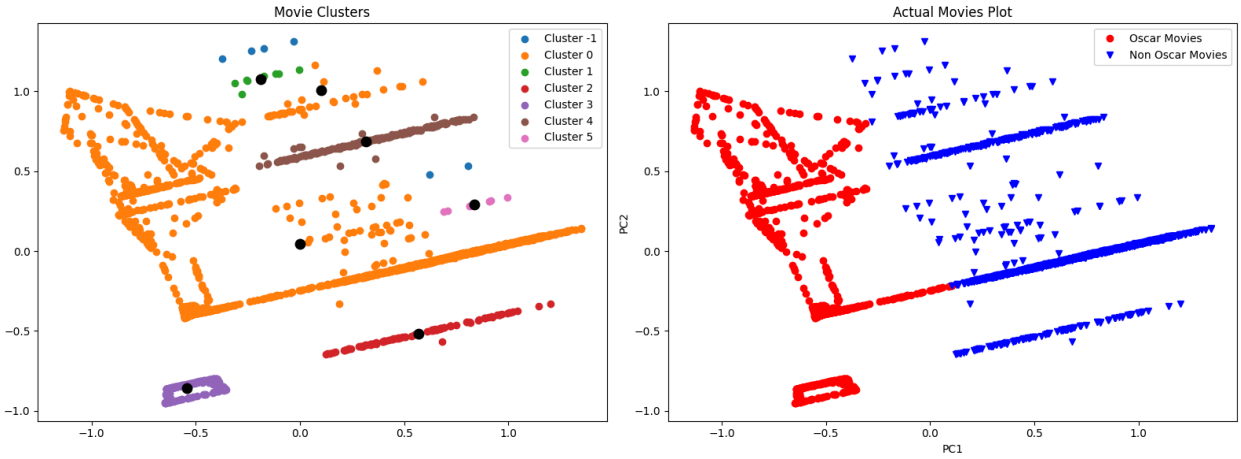
Για να επιλέξουμε το eps χρησιμοποιούμε από την έτοιμη βιβλιοθήκη kneed, τη κλάση KneeLocator, η οποία σχεδιάζει και βρίσκει το knee-point (δηλαδή το eps που θα χρησιμοποιήσουμε).

Συγκεκριμένα, το knee point (eps) το βρίσκει ίσο με 0.14, και παράγει το ακόλουθο σχήμα για να αιτιολογήσει την επιλογή:



Και όντως στο σημείο αυτό παρατηρούμε ότι υπάρχει μία έντονη αλλαγή στη καμπύλη, οπότε θέτουμε eps = 0.14.

Έχοντας ορίσει το eps και το min samples, προχωράμε στην υλοποίηση του DBSCAN. Ορίζοντας τις παραμέτρους αυτές, το DBSCAN έκανε 7 clusters τα οποία φαίνονται ως εξής:



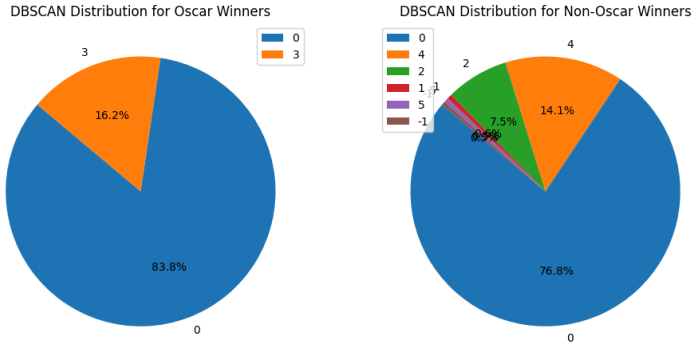
Ήδη από το σχήμα μπορούμε να δούμε ότι το DBSCAN δεν έκανε καλή κατηγοριοποίηση των ταινιών. Μπορούμε να δούμε ότι έβαλε τις περισσότερες ταινίες (oscar και μη) στο cluster 0. Το cluster 3, φαίνεται να έχει προβλέψει σωστά τις ταινίες με oscar. Αντίστοιχα τα υπόλοιπα clusters (εκτός του 0) βλέπουμε ότι περιγράφουν ταινίες χωρίς oscar. Μία σημαντική

παρατήρηση (σύμφωνα με το documentation του sklearn) είναι ότι το cluster -1 έχει τις ταινίες που θεωρούνται ως outliers.

Οι παρατηρήσεις αυτές επιβεβαιώνονται και από τα παρακάτω piechart:

Από τα piecharts αυτά βλέπουμε ότι όλα τα clusters (εκτός του cluster = 0) του DBSCAN έχουν μία κατηγορία ταινιών. Όπως παρατηρήθηκε και πριν, το cluster 3 έχει μόνο ταινίες με oscar, ενώ τα cluster -1, 1, 2, 4, 5, 6, 7 έχουν ταινίες χωρίς oscar. Το cluster 0, έχει σχεδόν ίδιο ποσό ταινιών με oscar (52.2%) και χωρίς oscar (47.8%), με λίγο περισσότερες ταινίες oscar, κάτι που είναι αρκετά λάθος αν λάβουμε υπόψη (όπως αναφέραμε και πριν) ότι οι ταινίες

Το ίδιο αποτέλεσμα μπορούμε να δούμε και από το piechart των ταινιών (και σε ποια cluster ανήκουν):



Ομοίως παρατηρούμε ότι μόνο το cluster 3 έχει ταινίες oscars, ενώ τα -1, 1, 2, 4, 5, 6, 7 έχουν ταινίες χωρίς oscar. Το cluster 0 έχει το μεγαλύτερο αριθμό ταινιών και (δυστυχώς) έχει το μεγαλύτερο ποσοστό και στις ταινίες με oscar και σε αυτές χωρίς (oscar).

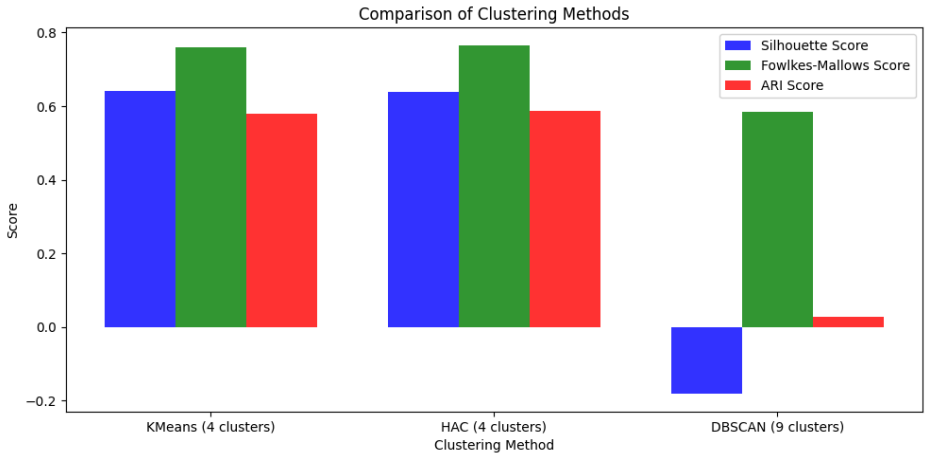
Ακολουθούν τα αποτελέσματα των μετρικών:

```
Silhouete Score: -0.1820774640078261
Adjusted Rand Index (ARI): 0.02824118839207375
Fowlkes-Mallows Score: 0.5854114173823263
```

Παρατηρούμε ότι όλα τα scores είναι τα χαμηλότερα μέχρι στιγμής. Από το χαμηλό ARI (και fowlkes-mallows) βλέπουμε ότι τα predicted clusters δεν είναι κοντά στα αληθινά δεδομένα (και μάλιστα επειδή είναι κοντά στο 0, σημαίνει τυχαίο clustering). Το silhouette score είναι πολύ πιο χαμηλό από τι προηγουμένως, δείχνοντας ότι τα δεδομένα εντός των clusters δεν είναι όμοια μεταξύ τους.

<u>Τελικά Συμπεράσματα clustering:</u>

Δημιουργούμε ένα barchart με όλα τα scores (που σχολιάστηκαν) των δοκιμασμένων μοντέλων:



Παρατηρούμε ότι το KMeans και HAC έχουν πολύ παρόμοια (και σχετικά καλά) scores σε σχέση με το DBSCAN που έχει όλα του τα scores πιο χαμηλά και από το KMeans και από το HAC.

Για την επιλογή λοιπόν του καλύτερου μοντέλου σε σχέση με το διαχωρισμό των οσκαρικών ταινιών από τις υπόλοιπες, μπορούμε να επιλέξουμε είτε το KMeans είτε το HAC, καθώς είδαμε και από πριν ότι κάνουν καλή δουλειά στην κατηγοριοποίηση αυτή.

Μπορείτε να δείτε όλο το κώδικα που χρησιμοποιήσαμε εδώ.