California State University, Northridge

Profitability Prediction of Movie Projects

Group 6:
Oguzhan Akan
Darius Hooks
Jaymish Raju Patel
Jason Sabal
Bibinur Zhursinbek

Fall 2020

COMP 541: Data Mining

16 December 2020

Table of Contents

1. Objective **5**

2. Background Information **5**

3. Design Principles **5**

4. Data Exploration and Data Preprocessing **6**

4.1. Data Exploration 6

4.1.1. Preparing the Data 6

4.1.1.1. Data Collection & Description 6

4.1.1.2. Types of Data in the Datasets 7

4.1.1.3. Combining Source Tables & Transforming Features to their Correct Format 7

4.1.2. Data Exploration 8

4.1.2.1. Exploring Continuous Features 8

4.1.2.2. Exploring Categorical Features 9

4.1.2.3. Categorical Feature Dependency Using Chi-Square Test 10

4.1.2.4. Frequent Pattern Analysis 11

4.1.3. Data Exploration Results 12

4.2. Preprocessing 13

4.2.1. Data Cleaning 13

4.2.2. Data Transformation 14

4.2.3. Data Cleaning & Transformation Results 15

4.2.4. Feature Selection 15

5. Modeling and Implementation **18**

5.1. Selected Modeling Algorithms & Performance Metrics 18

5.2. Modeling Strategy 18

5.3. Model Parameters & Results 19

5.3.1. Classification Models 19

5.3.1.1. KNN Classification 19

5.3.1.2. Random Forest 20

5.3.1.3. Gradient Boosting 21

5.3.2. Regression Models 22

5.3.2.1. KNN Regression 22

5.3.2.2. Random Forest 23

5.3.2.2. Gradient Boosting 24

6. Analysis **26**

6.1. Model Comparisons 26

6.1.1. Comparing Classification Models 26

6.1.2. Comparing Regression Models 26

7. Summary **27**

8. Reference **28**

9. Appendix **29**

A1. Types of instances of the Datasets - Movies 29
A2. Types of instances of the Datasets - Metadata 29
A3. Types of instances of the Datasets - Keywords 30
A4. Exploring Continuous Features - Gross 30
A5. Exploring Continuous Features - Profitability Ratio 31
A6. Exploring Categorical Features - Data Exploration Report 32
A7. Data Cleaning & Transformation Report 32
A7. Feature Selection Report 32
A8. Modeling Report 32

1. Objective

The data mining goal is to accurately and precisely predict the profitability ratio of a proposed movie project.

The data mining success criteria for classification model is to achieve 75% in the accuracy and 40% in the F-1 score. For the regression model, the success of the project will be determined by the Root Mean Square Error (RMSE).

The overall goal of the project is to accurately and precisely predict the profitability of the movie before starting production by using historical data.

2. Background Information

The film industry has grown immensely over the past few decades, generating billions of dollars. Now people can watch movies online through Netflix or other streaming services and offline by going to the cinema. The movie producers and directors who have to produce new movies continuously need to be able to predict the demand in order to reduce the uncertainty in the market and to increase the chance of profitability [1]. Predicting how well a movie will perform at the box office is hard because there are so many factors involved in success, like actors, plot, date of the release, and so on.

3. Design Principles

For the purpose of the dataset, we used a publicly available dataset, Kaggle. We decided to use two datasets. One of them is the movie industry, which includes data of revenue and budget. The second dataset is the movie dataset, which includes information about film's cast, crew, plot, keywords, budget, revenue, posters, release dates, languages, production companies, countries and vote averages.

In this project, we used such libraries as Pandas, Numpy, Seaborn and Scikit--Learn to program the process in Python. Pandas and Numpy were used for data transformation and manipulation, while Seaborn was used for plotting and Sci-kit Learn was used for modeling purposes. All the coding was done on Anaconda platform, which has many tools related to data science and machine learning.

4. Data Exploration and Data Preprocessing

4.1. Data Exploration

4.1.1. Preparing the Data

4.1.1.1. Data Collection & Description

Since the movie sector is highly popular and essentially digital, access to public movie data is as easy as searching through a search engine. Kaggle has a series of movies dataset, so we decided to select our data from there, overriding the need for a web scraping process. We use two main resources for our Data Mining project:

The Movies Dataset (MD): The dataset provides a set of tables with an entity-relationship feature so that we can combine relevant information from different tables. The tables include cast, crew, plot keywords, budget, revenue, posters, release dates, languages, production companies, countries, TMDB vote counts and vote averages.

Location: Kaggle

Problems encountered: The Movies Dataset's revenue information represents the global revenue which may be harder to predict since many factors may affect the movie's popularity - like culture, sense of humour, perception of romance etc.

Resolution: We needed to find another resource that includes only-US-revenue of movies. The Movie Industry Dataset had the necessary information, so we acquired it as our second dataset.

Movie Industry Dataset (MI): The dataset includes just one .CSV file with 6820 movies' budget and revenue information.

Location: Kaggle

Problems encountered: Some of the data points did not have budget or revenue information.

Resolution: We needed to wipe off the data points which did not have budget or revenue information. We ended up with 4638 records after filtering out 2182 movies.

Database	Table Name	Records	Fields
Movies Database (MD)	Metadata	45466	11
Movies Database (MD)	Keywords	46419	2
Movies Industry (MI)	Movies	6820	18

The Movies Database has two tables: "metadata" and "keywords". The first table includes basic information about the movie while the keywords table consists of related keywords for each movie. They can be linked together using the “id" column.
The Movie Industry dataset has one table: "movies". This table includes all the relevant information about the movies, including our target, profitability.
Type of the Data: Our data used for the project is **a mix of structured and semi-structured data.** While MI dataset is completely structured, MD dataset includes data where the values are either structured or semi-structured, particularly in JSON format. We convert the semi-structured data into structured data using Python’s *json* library.
Type of Datasets: MI dataset is **a single-file tabular dataset**, while the MD dataset is **a relational database** data. We converted MI such that it is a part of the MD dataset’s schema. We used movies’ titles for joining since MI data did not have movie IDs as MD data did.

4.1.1.2. Types of Data in the Datasets

Please refer to the tables in Appendix 1, 2, and 3, for a detailed description of the datasets (namely, *Movies, Metadata, and Keywords)* used in the project.

4.1.1.3. Combining Source Tables & Transforming Features to their Correct Format

After the initial analysis of our datasets, we combined the tables using different join criteria in order to obtain a single tabular dataset and then be able to perform data mining on. Following snippet covers how the tables are joined together:

```
> Reading raw data:
metadata; records: 45466, fields: 11
keywords; records: 46419, fields: 2
movies; records: 4638, fields: 18

> Removing duplicate rows from each table:
> After the removal, we end up with the following number of rows for each table:
metadata; records: 39943, fields: 11
keywords; records: 44447, fields: 2
movies; records: 4570, fields: 18

> Combining metadata and keywords tables using "id" column that is common in both tables.
metadata + keywords --> shape: 39089, fields: 12

> Combining movies and previously combined metadata+keywords table using movie titles
df = movies + metadata + keywords --> shape: 3524, fields: 30
```

The last table, "df" is the combined data from three sources and thus will be used throughout the project. We will still manipulate the data in terms of preprocessing and feature engineering.

By observing the data, we see that the following columns need not any initial transformation:

- genre, company, budget, country, director, overview, tagline

And the following features had to change due to:

- released: this is a date column with dd.mm.yyyy format. For the sake of simplicity, we are only interested in the year the movies are released. Therefore we apply a transformation to extract year information from this column and write it to *year_released* column.

Of course, all of the categorical features will be converted to continuous (or, at least discrete) fields before modeling stage, but for now, we are just interested in the integrity of the data.

4.1.2. Data Exploration

After we combine datasets and correct data formats, we can now take a look at each feature. We examine each feature according to their data format, meaning whether it is continuous or categorical data. Then, we perform a frequent pattern analysis on some features.

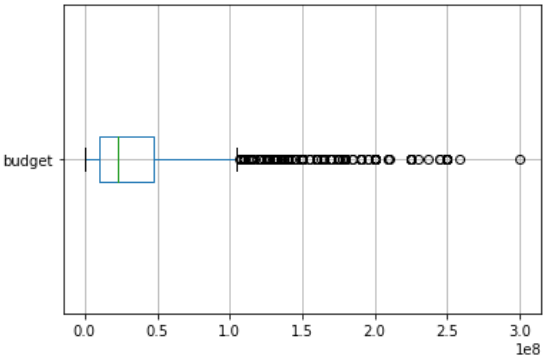
4.1.2.1. Exploring Continuous Features

Our data has the following columns that we can use to conduct the statistical description:

- Budget
- Gross
- Profitability ratio

property	value
count	3524
min	0
max	300000000
mean	36468367
median	23000000
std.dev	40633689
variance	1651096751521254
Q1	10000000
Q3	48000000
IQR	38000000

Budget: Table on the right represents the data summary based on the statistical description of the feature *budget*. There are measures of central tendency (mean, median), dispersion (quartiles, variance, standard deviation, interquartile range), and some other statistical descriptions, like minimum, maximum and count. The count() function returns the number of cells for each row or column. The budget column has 3524 non-empty cells. The minimum value of the budget is 0, while the maximum - 300,000,000.



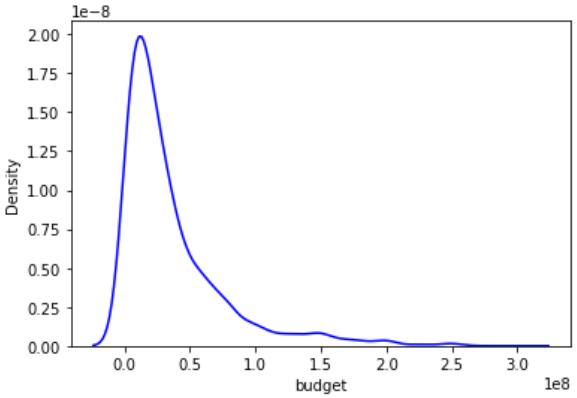


Figure on the left above shows the boxplot of the feature budget. The graph represents five-number summary: “minimum, first quartile, median, third quartile, and ''maximum”. The ends of the box are the quartiles, Q1 and Q3, while the box length is the interquartile range (IQR). While from the graph, we can get an approximate value of Q1 and Q3, it can be seen that the exact value of Q1 is 10,000,000 and Q3 is 48,000,000 from Figure 1. The median is marked as the line within the box, which is equal to 23,000,000. Two lines (called whiskers) outside the box extend to the smallest (Minimum) and largest (Maximum) observations. The circles are the outliers.

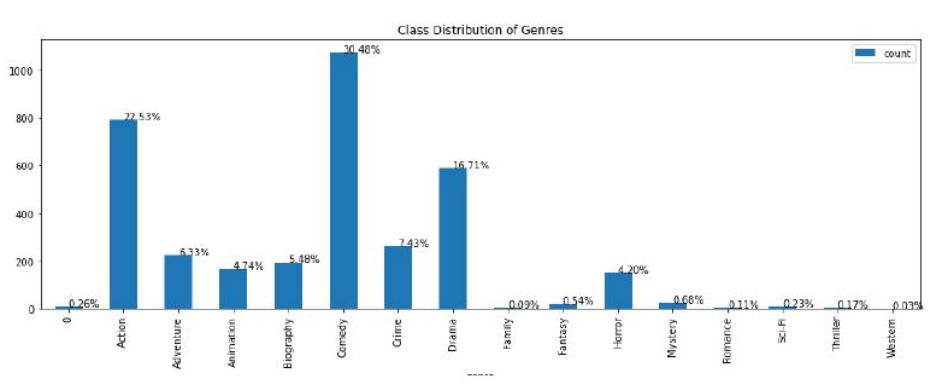
Figure on the right above demonstrates that the budget’s distribution is asymmetrical because the tail is skewed to the right. As the tail is longer towards the right-hand side, the value of skewness is positive. The value of the skewness is 2.253961, which means that the distribution is highly skewed.

For the sake of cleanliness, we share the analysis of the rest of the continuous features in the appendix. Please see Appendix 4 and 5 for analysis of features *gross* and *profitability_ratio.*

4.1.2.2. Exploring Categorical Features

After the initial analysis of continuous features, we move on with analyzing categorical features. We retrieve the balance information of each category in every categorical feature.

Genres: The bar chart below shows the distribution of genre among our dataset, and subsequently, we share the imbalances of each genre.



- Action is Slightly Imbalanced
- Adventure is Severely Imbalanced
- Animation is Severely Imbalanced
- Biography is Severely Imbalanced
- Comedy is balanced
- Crime is Severely Imbalanced
- Drama is Slightly Imbalanced
- Family is Severely Imbalanced
- Fantasy is Severely Imbalanced
- Horror is Severely Imbalanced
- Mystery is Severely Imbalanced
- Romance is Severely Imbalanced
- Sci-Fi is Severely Imbalanced
- Thriller is Severely Imbalanced
- Western is Severely Imbalanced

Again, for the sake of cleanliness, we share only one example of a categorical feature in this section. The rest can be viewed in Appendix X, Y, Z.

4.1.2.3. Categorical Feature Dependency Using Chi-Square Test

Our data has the following columns that we can use to conduct the chi-square test on:

- Genre, Company, Country, Director, Rating, Released, Isprofit

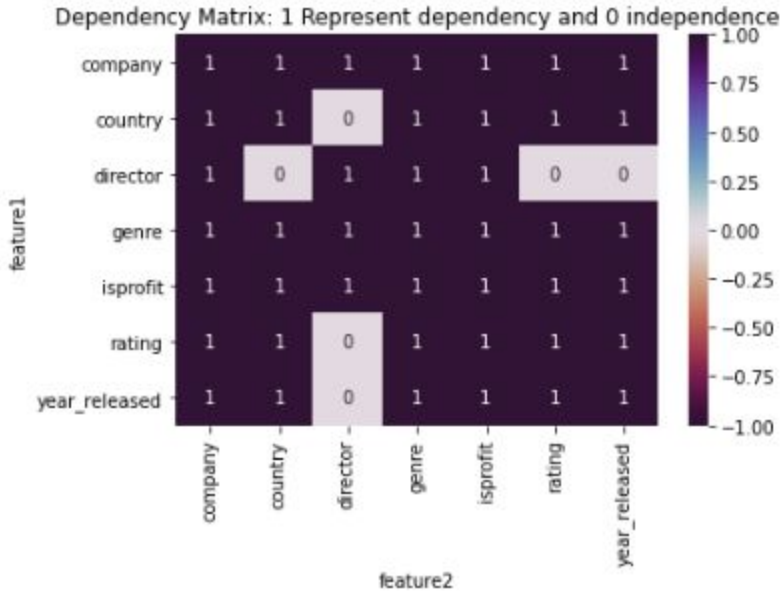
We want to find out whether two categorical attributes are independent or dependent by using the chi-square test. We do this by getting all combinations of length two from the list above and conducting the chi-square test on each of the categorical pairs. After writing some general formulas, we will conduct the test on all 21 combinations.

Using the code on the below, we conduct a chi-square test for each pair. Please refer to the summary at the end of this section, or the HTML report (Appendix 6) for detailed results of the

```
comb = it.combinations(columns, 2)
res=[]
for i in list(comb):
    table = pd.crosstab(df[i[0]], df[i[1]])
    display(table)
    Observed_Values = table.values
    print("Observed Values")
    display(Observed_Values)
    chi2_test_statistic, p, dof, expected = sp.chi2_contingency(table)
    print("chi2_test_statistic, p, dof, expected")
    display(chi2_test_statistic, p, dof, expected)
    prob = 0.95 # significant value = 1 - 0.95 = 0.05
    critical = chi2.ppf(prob, dof)
    print("critical = %.3f, chi2_test_statistic = %.3f" % (critical, chi2_test_statistic))
    if chi2_test_statistic >= critical:
        print("Dependent (reject H0)\n")
    else:
        print("Independent (fail to reject H0)\n")
    alpha = 1.0 - prob
    print("significance = %.3f, p = %.3f" % (alpha, p))
    dependent_p = False
    if p <= alpha:
        print("Dependent (reject H0)")
        dependent_p = True
    else:
        print("Independent (fail to reject H0)")
    res.append(list(i)+[dependent_p])
```

chi-square tests. The summary table contains the information of dependency while the HTML report also contains the test statistics and p-values.

Summary of the Chi-Square test results: In the results of the chi-square test (see figure below), we observe there are quite enough features that are dependent and independent. For example, *company* and *country* are correlated as we initially expected, because most of the movies in our data are made in the US.



4.1.2.4. Frequent Pattern Analysis

Our data has the following columns that we can perform frequent pattern analysis on:

- Genres_edited, Spoken_languages_edited, Keywords_edited, Overview

We want to find out whether there are recurring relationships in the data. For example, which genres are frequently used together to produce movies? Or which keyword combinations are frequent? After performing the analyses, we can decide, for example, which genres to produce in order to make the most profit. After writing some general functions, we perform the analyses one by one.

Genres: First, we show how our *genres* feature look like in the figure below. After applying the apriori algorithm and calculating the association metrics, we generate the following rules:

```
df.genres_edited.value_counts()
```

```
Drama                                  251
Comedy                                 242
Comedy, Drama                          133
Drama, Romance                         124
Comedy, Romance                        110
                                      ...
Drama, Fantasy, Horror, Romance          1
Action, Adventure, Family, Fantasy       1
Adventure, Thriller                      1
Horror, Comedy, Music                    1
Drama, Comedy, Fantasy                   1
Name: genres_edited, Length: 907, dtype: int64
```

```
rules[
    (rules.antecedents_len >= 2) &
    (rules.confidence      >= .3) &
    (rules.lift            >= 1) &
    (rules.leverage        >= 0.01) &
    (rules.conviction      >= 1)
]
```

	antecedents	consequents	antecedent support	consequer
30	(Drama, Comedy)	(Romance)	0.140182	
39	(Crime, Drama)	(Thriller)	0.088820	
40	(Thriller, Drama)	(Crime)	0.117764	

For the sake of cleanliness, we present the rest of our frequent pattern analyses in the report shared in Appendix 6.

4.1.3. Data Exploration Results

Continuous Features: As we analyze three continuous features in Section 4.1.2.1., namely budget, gross (revenue), and profitability_ratio, the data seems clean enough to build our model on. Even though all the three features are positively skewed, we do not observe so many outliers that will disturb training the model.

Our initial hypothesis was that our three continuous features were meant to be correlated, because intiutively, the budget and gross and profitability of a movie sound correlated. That is, however, not the case, at least for profitability ratio and the others.

Nonetheless, we can conclude that budget and gross are correlated. Although this information does not help us predicting the profitability ratio, we understand how independent the profitability ratio is and take it into account when we build our Machine Learning models.

Categorical Features: After analyzing our categorical features in Section 4.1.2.2., we see a somewhat imbalance in our six features. However, we have to take the imbalance problem while modeling, the one feature, *isprofit*, turns out to be very balanced. Since the *isprofit* feature is our target feature, we will choose a metric that objectively evaluates model performance in our classification problem.

Frequent Pattern Analysis: Our data included a set of features that we could perform frequent pattern analysis on, which we used for building association rules in Section 4.1.2.3. Since our dataset is not a kind of transactional data, these rules are not meant to be used for modeling. However, we still wanted to conduct this analysis in order to turn the learnings from our lectures into practice. For feature *genres*, we built the following association rules, after setting thresholds for confidence, leverage, conviction, and lift:

◦ (Family) => (Comedy)

◦ (Romance) => (Drama)

◦ (Mystery) => (Thriller)
◦ (Crime, Drama) => (Thriller)
When we think about those rules, they actually make sense! For example, most of the family movies have comedy inside as well.

4.2. Preprocessing

4.2.1. Data Cleaning

Our data has the following columns that needed cleaning:

- overview
- tagline
- spoken_languages_edited
- production_countries _edited
- Keywords_edited

```
df.isnull().sum()
```

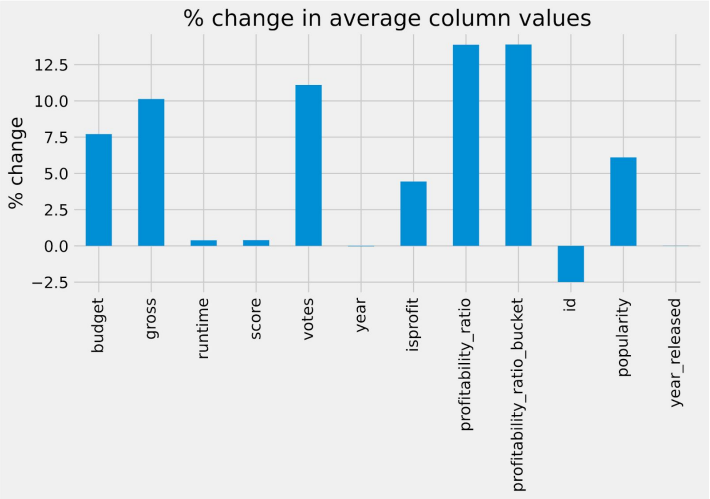
budget	0
company	0
country	0
director	0
genre	0
gross	0
name	0
rating	0
released	0
runtime	0
score	0
star	0
votes	0
writer	0
year	0
isprofit	0
profitability_ratio	0
profitability_ratio_bucket	0
adult	0
id	0
imdb_id	0
original_title	0
overview	1
popularity	0
tagline	371
title	0
genres_edited	0
spoken_languages_edited	7
production_countries_edited	31
keywords_edited	167
year_released	0

dtype: int64

After identifying the columns that need cleaning, we use the code shown below to replace all values of zero with the value None.

```
df['overview'] = df['overview'].map(lambda x:x if x != 0 else None)
df['tagline'] = df['tagline'].map(lambda x:x if x != 0 else None)
df['spoken_languages_edited'] = df['spoken_languages_edited'].map(lambda x:x if x != 0 else None)
df['production_countries_edited'] = df['production_countries_edited'].map(lambda x:x if x != 0 else None)
df['keywords_edited'] = df['keywords_edited'].map(lambda x:x if x != 0 else None)
```

After dropping rows that had the value None, we can see that a little more than half of the features on the figure to the right had a 5% change in mean.



However, after finding the outliers in our dataset, we end up with a reduction of data of almost 50% which was concerning since using less data in the modeling later on might cause skewed or inaccurate results.

```
data_dropped.shape
```

```
(3031, 31)
```

```
data_dropped_outlier_IQR = data_dropped[~((data_dropped < (Q1 - 1.5 * IQR)) | (data_dropped > (Q3 + 1.5 * IQR))).any(axis=1)]
```

```
data_dropped_outlier_IQR.shape
```

```
(1993, 31)
```

4.2.2. Data Transformation

Our data has the following columns that we found important to transform:

- budget
- runtime
- score
- votes
- Popularity

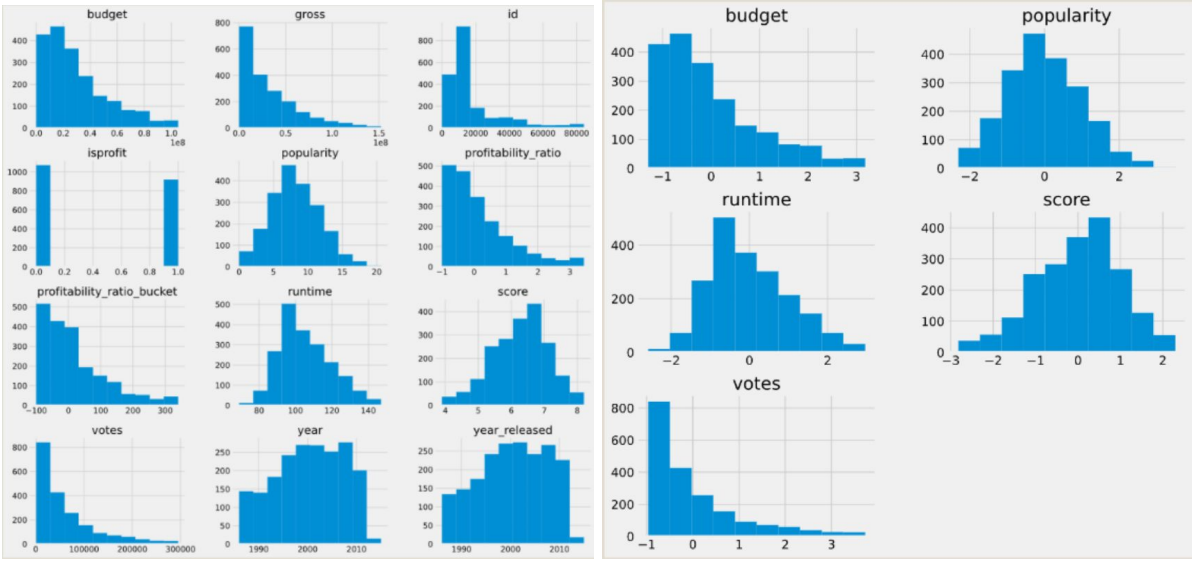
Using the code below, we iterate over all continuous features and use binning to discretise the data. At the end, we summarize the results by indicating the four intervals that our data was partitioned into. Refer to assignment 4 HTML to see the results of the code below.

```
for i in continuous_features:
    display(data_dropped_outlier_IQR[i].describe())

    display(pd.qcut(data_dropped_outlier_IQR[i], q=4))

    display(pd.qcut(data_dropped_outlier_IQR[i], q=4).value_counts())
```

The figure on the left contains histograms of all the features with numerical values while the figure on the right show histograms of the five specific features that we found important to use in our model.



4.2.3. Data Cleaning & Transformation Results

Deciding to drop or impute missing values was an issue since the five mentioned features are categorical, which means imputing with the mean value was not possible. Imputing with the most frequent was an option but risked imputing values that may not mix well with the rest of the data. In the end, we decided to drop the rows with missing values just to be safe.

Transformation of the five mentioned features was fairly simple and allowed us to further categorize our data which was useful for modeling so we wouldn't have to work with such large numbers. After normalization, we can see that the ranges for our feature set have become smaller.

4.2.4. Feature Selection

Feature selection helps us to reduce the number of features needed to be included in modeling without sacrificing predictive power. The features that are irrelevant to modeling may impact adversely, so it is necessary to remove them. It helps the model to concentrate more on the relevant features of the data by eliminating extraneous data, and not get caught up on features that don't matter. Another advantage of eliminating redundant information is that it increases the precision of the model's predictions. It also decreases the processing time needed to get the model.

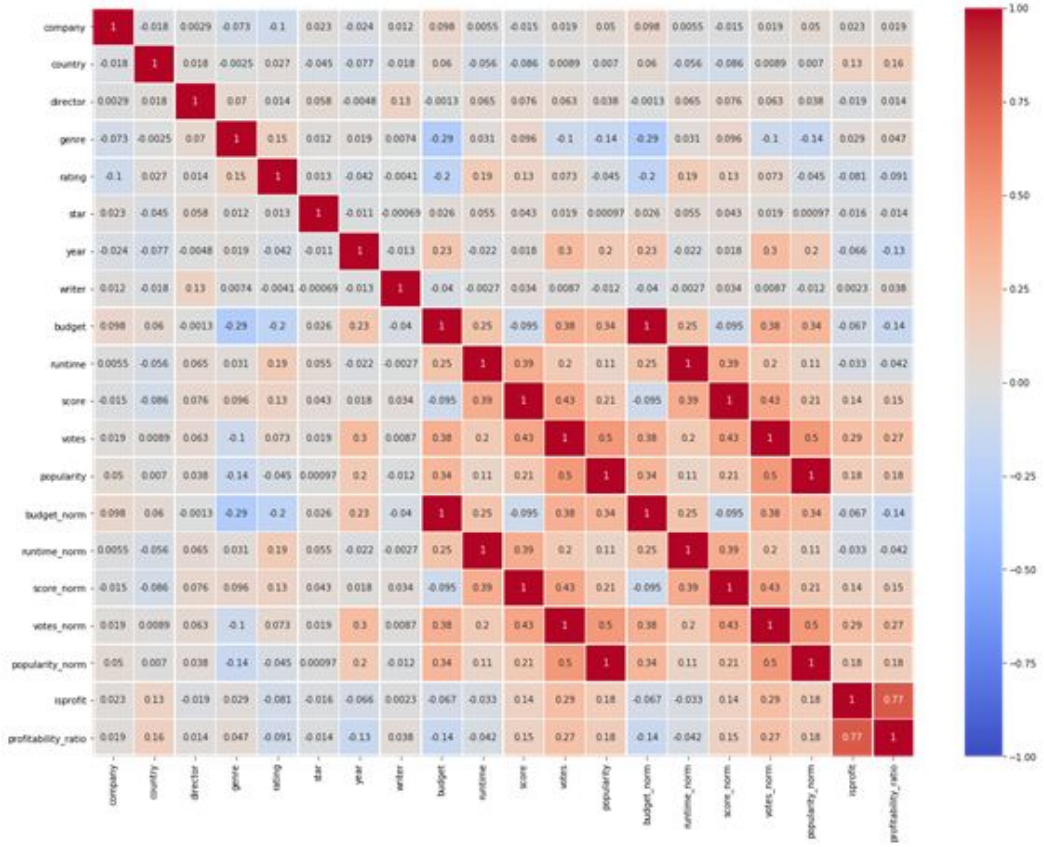
Here for feature selection, we decided to perform a correlation matrix of the available features, so we can determine what are the relevant features to get a higher profitability ratio.

Here, all features from the dataset are divided into 4 groups: categorical_features, continuous_features, normalized_features, and target_features groups. To perform the correlation matrix each feature must be in numerical so we need to convert categorical_features into numerical data. In order to convert the required categorical_features into numerical data, we performed the following

```
df['company']=df['company'].astype('category').cat.codes
df['country']=df['country'].astype('category').cat.codes
df['director']=df['director'].astype('category').cat.codes
df['genre']=df['genre'].astype('category').cat.codes
df['rating']=df['rating'].astype('category').cat.codes
df['star']=df['star'].astype('category').cat.codes
df['writer']=df['writer'].astype('category').cat.codes
```

By running this code the first company name is replaced by 0 and all its occurrence is also converted to 0. Similarly, the second company name is replaced by 1 and all its occurrence is also replaced and so on.

Now as we have all the feature data into numeric data we can perform the correlation between the features. Correlation is performed using .corr() function. A 20x20 matrix shows the correlations between the features. A heat-map is created from the matrix obtained. From the heat-map, we can visualize how much a feature is correlated to other features.



From the correlation matrix obtained it is necessary to determine the features that will affect the accuracy of the data mining results. Here our aim is to find the maximum profitability ratio. So here it is necessary to obtain features that are highly correlated to the profitability ratio. The features that have correlation higher than 0.04 or less than -0.04 with profit and profitability ratio are selected.

```
corr_df_features = set()
for i in range(len(corr_df.columns)-2):
    if corr_df.iloc[i,18]<-0.04 or corr_df.iloc[i,18]>0.04 or \
        corr_df.iloc[i,19]<-0.04 or corr_df.iloc[i,19]>0.04:
        corr_df_features.add(corr_df.columns[i])
```

These are the features that have a correlation higher than 0.04 or less than -0.04 with profit and profitability ratio:

- budget
- budget_norm
- country
- genre
- popularity
- popularity_norm
- rating
- runtime
- runtime_norm
- score
- score_norm
- votes
- votes_norm
- year

5. Modeling and Implementation

5.1. Selected Modeling Algorithms & Performance Metrics

Tools/Language:

- Python: As all of our team members had hands-on experience in Python, we decided to move forward with it. We used Pandas and Numpy for data transformation and manipulation, Seaborn for plotting, and Sci-kit Learn for modeling purposes.

Algorithms: We decided to implement more than one modeling algorithms in order to be able to compare their performances and decide which algorithm outperformed others. The following algorithms were used in the modeling stage:

- K-Nearest Neighbors: Fast to implement and interpret
- Random Forest: Ensemble decision trees that decide according to a voting algorithm
- Gradient Boosting: Ensemble decision trees that reduce its error in every other iteration

Performance Metrics: In order to evaluate the performance of our algorithms, we selected some metrics that objectively represent the performance of those algorithms.

- Classification: Accuracy, Sensitivity, Precision - F1 Score
- Regression: MedianAE, MeanAE, RMSE

5.2. Modeling Strategy

Train-Test Data Split: We decided to split our training and test data by 80%-20% as we wanted to have enough data for objective evaluation on both train and test sets. The X_train and X_test is the same for both classification and regression models. We achieved this by setting a random state for each split. Splitting made by using Sklearn's train_test_split function.

```
X_train, X_test, y1_train, y1_test = train_test_split(X, y1, test_size=0.2, random_state=17)
X_train, X_test, y2_train, y2_test = train_test_split(X, y2, test_size=0.2, random_state=17)
```

```
# This is where we create our folds
kf = KFold(n_splits=3, random_state=17, shuffle=True)
kf.get_n_splits(X_train)
fold_indexes = {}
i = 1
for train_index, valid_index in kf.split(X_train):
    fold_indexes[i]={}
    fold_indexes[i]['train']=train_index
    fold_indexes[i]['valid']=valid_index
    i+=1
```

K-Fold Cross Validation: We applied a 3-fold stratified split to our training dataset in order to overcome possible overfitting issues.

5.3. Model Parameters & Results

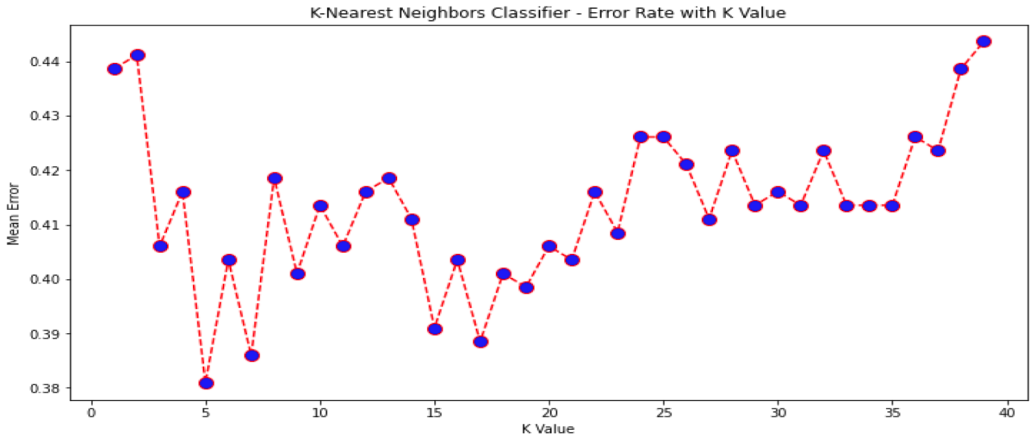
5.3.1. Classification Models

5.3.1.1. KNN Classification

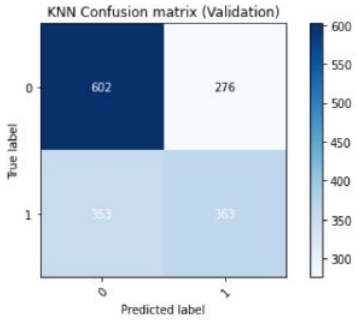
Parameter setting:

- n_neighbors (k) = 2, ..., 20
- weights = 'uniform'

Best value of k turned out to be 5. Note that both very-low and very-high numbers of neighbours result in high error rates. Very-low neighbours cause underfitting and very-high neighbours cause overfitting. The figure below shows how error rate changes when we increase the number of neighbours.



Once we decided that n_neighbors = 5, we ran the 3-fold CV and built confusion matrices for validation and test data separately. As can be seen from the figures below, the KNN model reached ~0.6 accuracy and ~0.5 F- 1 score in validation and test data.

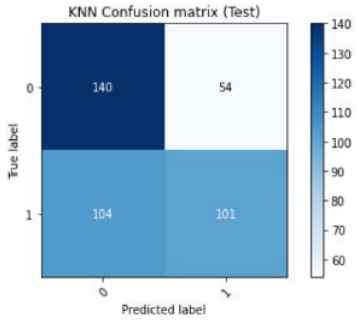


Accuracy: **0.605**

Sensitivity : **0.507**

Precision : **0.568**

F-1 Score : **0.536**



Accuracy: **0.604**

Sensitivity : **0.493**

Precision : **0.652**

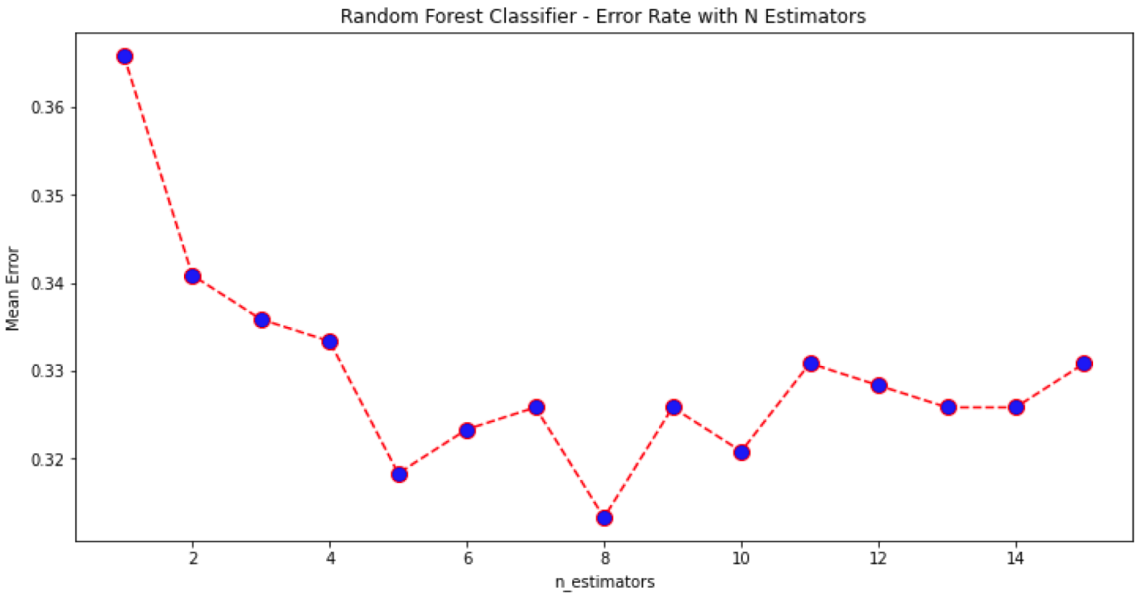
F-1 Score : **0.561**

5.3.1.2. Random Forest

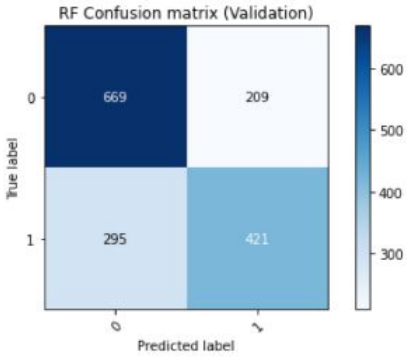
Parameter setting:

- n_estimators = 1, ..., 15
- min_samples_split = 60
- min_samples_leaf = 20
- max_depth = 7
- max_leaf_nodes = 14

Best value of n_estimators turned out to be 8. Again, using a lot of trees may increase the error rate by causing overfitting. The figure below shows how error rate changes when we increase the number of trees in our Random Forest model.



Once we decided that n_estimators = 8, we ran the 3-fold CV and built confusion matrices for validation and test data separately. As can be seen from the figures below, the Random Forest model reached ~0.68 accuracy and ~0.65 F-1 score in validation and test data.

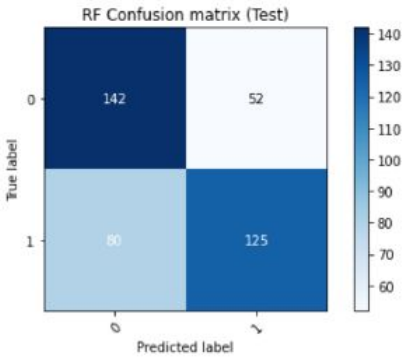


Accuracy: **0.684**

Sensitivity : **0.588**

Precision : **0.668**

F-1 Score : **0.626**



Accuracy: **0.669**

Sensitivity : **0.610**

Precision : **0.706**

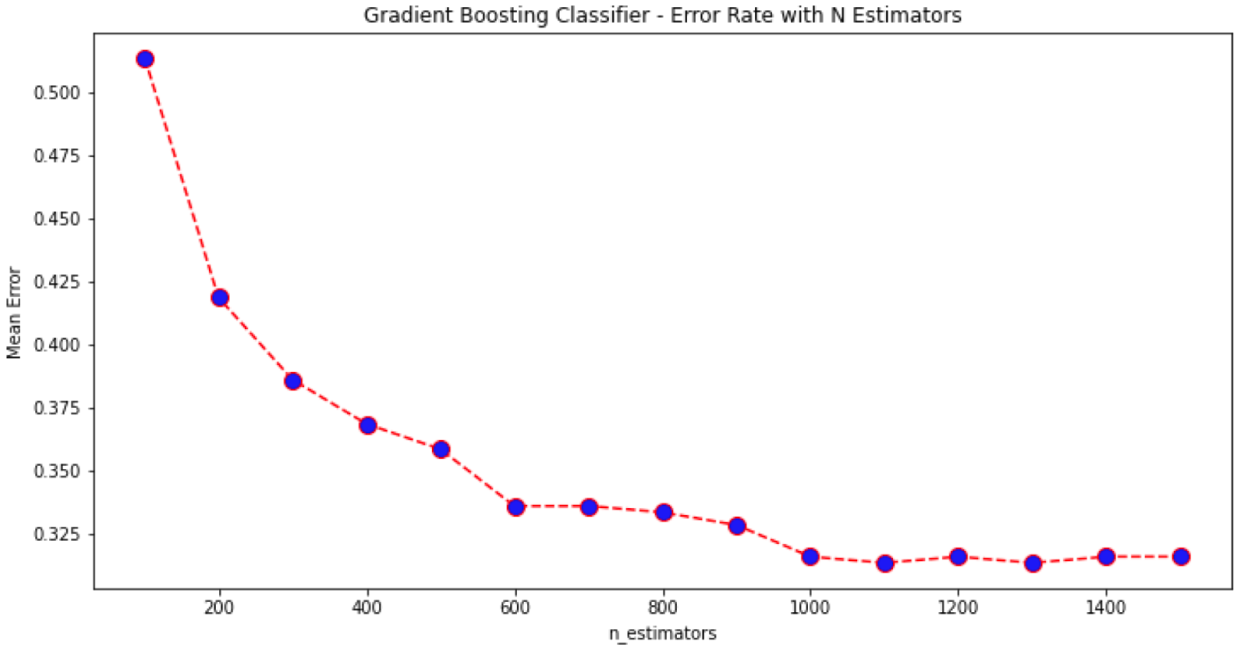
F-1 Score : **0.654**

5.3.1.3. Gradient Boosting

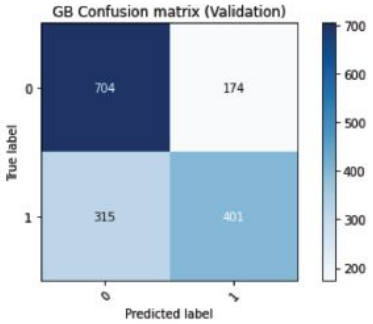
Parameter setting:

- n_estimators=100, 200, ..., 1500
- learning_rate=0.001
- criterion='friedman_mse'
- min_samples_split=60
- min_samples_leaf=20
- max_depth=7
- max_leaf_nodes=14

Best value of n_estimators turned out to be 1000. Note that when n_estimators >= 1000, the improvement is negligible. The overfitting problem observed in KNN and Random Forest is non-existent for Gradient Boosting algorithm because Gradient Boosting works in a way that improves its errors. Thus, we do not expect the error rate to increase as we iterate over n_estimators. The following figure presents how the error rate decreases when we increase the number of estimators.



Once we decided that n_estimators = 1000, we ran the 3-fold CV and built confusion matrices for validation and test data separately. As can be seen from the figures above, the Gradient Boosting model reached ~0.69 accuracy and ~0.65 F-1 score in validation and test data.

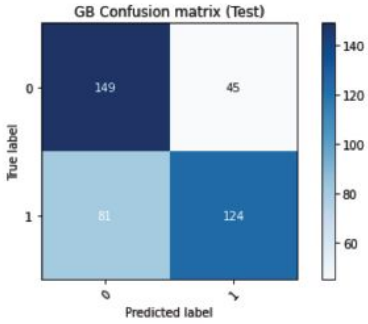


Accuracy: **0.693**

Sensitivity : **0.560**

Precision : **0.697**

F-1 Score : **0.621**



Accuracy: **0.684**

Sensitivity : **0.605**

Precision : **0.734**

F-1 Score : **0.663**

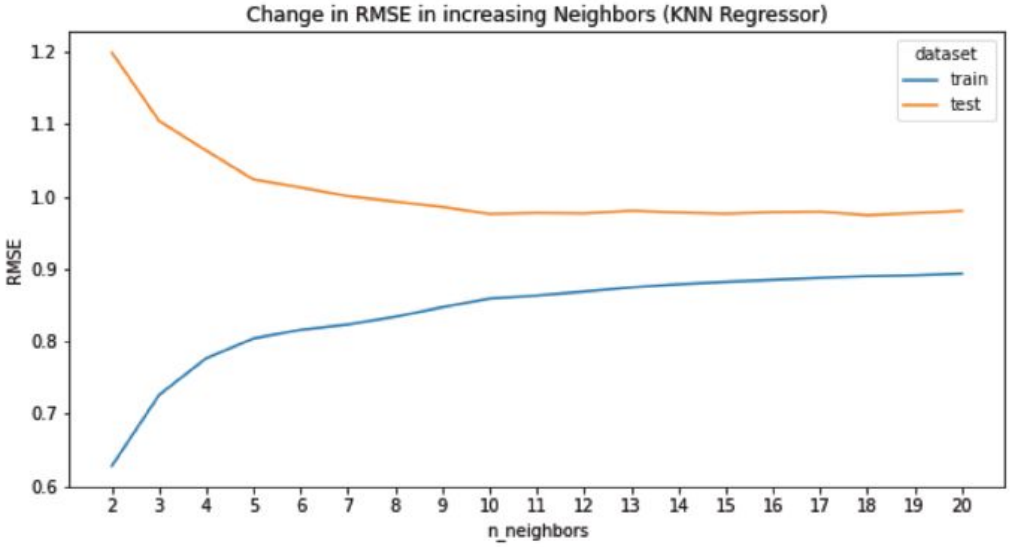
5.3.2. Regression Models

5.3.2.1. KNN Regression

Parameter setting:

- n_neighbors (k) = 2, ..., 20

Best value of k turned out to be 18. This regression model can be considered as a low-bias and low-variance model.



At k = 18;

[TRAIN] >> median_abs_err = 0.546, mean_abs_err = 0.678, **RMSE = 0.890**

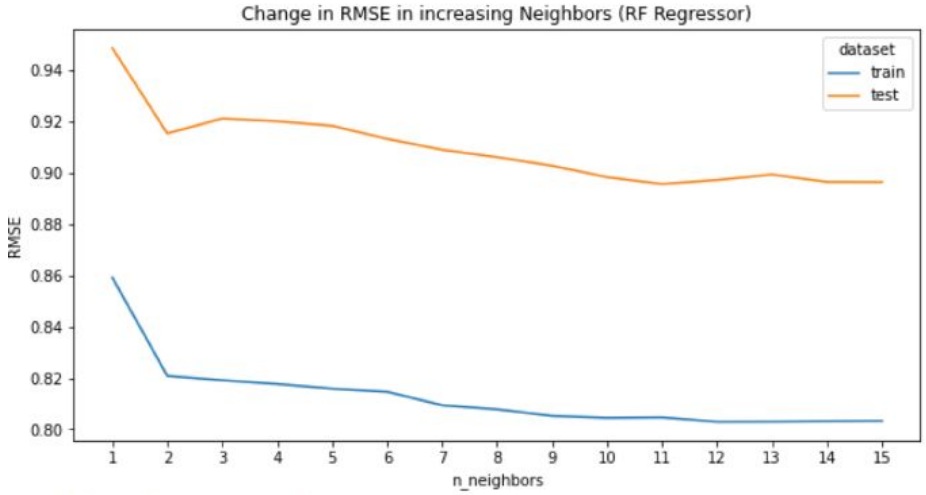
[TEST] >> median_abs_err = 0.597, mean_abs_err = 0.742, **RMSE = 0.974**

5.3.2.2. Random Forest

Parameter setting:

- n_estimators = 1, ..., 15
- min_samples_split = 60
- min_samples_leaf = 20
- max_depth = 7
- max_leaf_nodes = 14

Best value of n_estimators turned out to be 15. The following figure shows how train and test errors get lower as we increase the number of trees in the model.



At n_estimators = 15;

[TRAIN] >> median_abs_err = 0.474, mean_abs_err = 0.605, **RMSE = 0.803**

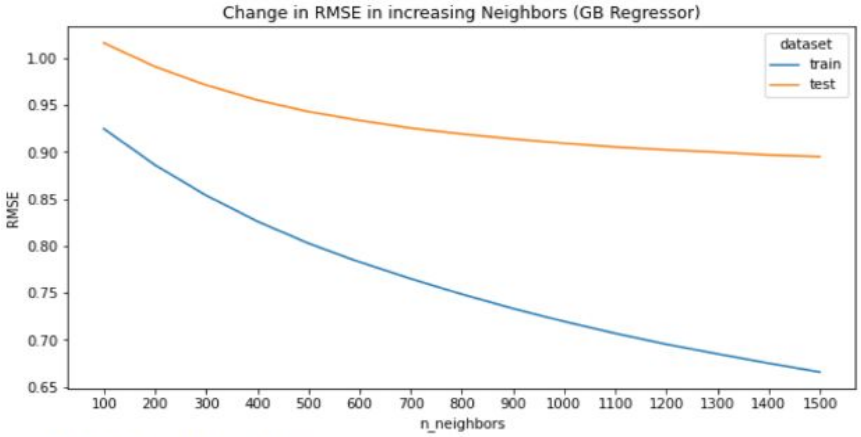
[TEST] >> median_abs_err = 0.528, mean_abs_err = 0.667, **RMSE = 0.896**

5.3.2.2. Gradient Boosting

Parameter setting:

- n_estimators=100, 200, ..., 1500
- learning_rate=0.001
- criterion='friedman_mse'
- min_samples_split=60
- min_samples_leaf=20
- max_depth=7
- max_leaf_nodes=14

Best value of n_estimators turned out to be 1500. The following figure presents how smoothly errors go down when we increase the number of iterations. Also, we can observe that the train error decreases faster than it does for test error, which might cause overfitting in high number of iterations.



At n_estimators = 1500;

[TRAIN] >> median_abs_err = 0.406, mean_abs_err = 0.508, **RMSE = 0.666**

[TEST] >> median_abs_err = 0.488, mean_abs_err = 0.652, **RMSE = 0.895**

6. Analysis

6.1. Model Comparisons

6.1.1. Comparing Classification Models

Model Ranking according to F-1 Score: Gradient Boosting > Random Forest > KNN. The table below shows exact numbers for each model-metric pair. We can see that while Gradient Boosting performed slightly better than Random Forest, they both outperformed the KNN model significantly.

	Validation				Test			
	Accuracy	Sensitivity	Precision	**F-1 Score**	Accuracy	Sensitivity	Precision	**F-1 Score**
KNN	0.605	0.507	0.568	0.536	0.604	0.493	0.652	0.561
Random Forest	0.684	0.588	0.668	0.626	0.669	0.610	0.706	0.654
Gradient Boosting	0.693	0.560	0.697	0.621	0.684	0.605	0.734	0.663

6.1.2. Comparing Regression Models

Model Ranking according to RMSE: Gradient Boosting ~= Random Forest > KNN

The difference between Gradient Boosting and Random Forest is negligible, while they both performed better than the KNN model. Still, KNN Regression was closer to the ensemble model than it was in the Classification problem.

	Validation			Test		
	Median Abs. Err.	Mean Abs. Err.	**RMSE**	Median Abs. Err.	Mean Abs. Err.	**RMSE**
KNN	0.546	0.678	0.890	0.597	0.742	0.974
Random Forest	0.474	0.605	0.803	0.528	0.667	0.896
Gradient Boosting	0.406	0.508	0.666	0.488	0.652	0.895

7. Summary

In conclusion, a movie's success depends on a lot of features that are related to the movies, the situation in the country, and so on. The movie profitability prediction will help film production companies to invest in future movie projects. Our models presented very promising results. All of our classification models reached our initial F-1 score target, which was 0.4. Random Forest and Gradient Boosting models reached 0.65, which is significantly better. In addition, the difference between Validation and Test data metrics is very low, indicating a non-overfitting conclusion. Our regression models also showed really promising results, as they all had Root Mean Square Error less than 1.

For the future work, we could include data from the social media, as the comments in the social media can influence the success of the movie because people might be encouraged or discouraged to go for a particular movie due to the comments of people. In addition, we could use the dataset from the recent films to have up-to-date information, as the dataset that we used includes movies before 2016.

For a better analysis, we can have more features, like sequels, political conditions and economic stability. For example, COVID-19 had affected the movie industry, as films were either postponed or cancelled, due to the cinema closure. In addition, during an economic crisis, a small number of people will go to the cinemas to enjoy movies. Also, a country's GDP rate can be used as one of the features that determine the movie's success, as it represents the financial stability during the period when a movie is released [2].

While doing this project, we faced several challenges and learned lessons from them. For example, we followed the data mining project cycle, CRISP-DM and learned about its phases and their deliverables. Also, we learned new algorithms and worked on a new platform, Anaconda. Communication is key in the group project. While doing this project, we learned that it is important to communicate different ideas and to voice any concerns.

8. Reference

1. T. G. Rhee and F. Zulkernine, "Predicting Movie Box Office Profitability: A Neural Network Approach," *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, Anaheim, CA, 2016, pp. 665-670, doi: 10.1109/ICMLA.2016.0117.
2. N. Quader, M. O. Gani, D. Chaki and M. H. Ali, "A machine learning approach to predict movie box-office success," *2017 20th International Conference of Computer and Information Technology (ICCIT)*, Dhaka, 2017, pp. 1-7, doi: 10.1109/ICCITECHN.2017.8281839.

9. Appendix

A1. Types of instances of the Datasets - Movies

field	data_format	data_attribute	example
budget	int64	continuous	8000000
company	object	categorical -> nominal	Columbia Pictures Corporation
country	object	categorical -> nominal	USA
director	object	categorical -> nominal	Rob Reiner
genre	object	categorical -> nominal	Adventure
gross	int64	continuous	52287414
name	object	categorical -> nominal	Stand by Me
rating	object	categorical -> nominal	R
released	object	continuous -> date	1986-08-22 00:00:00
runtime	int64	continuous	89
score	float64	continuous	8.1
star	object	categorical -> nominal	Wil Wheaton
votes	int64	discrete	299174
writer	object	categorical -> nominal	Stephen King
year	int64	categorical -> ordinal	1986
isprofit	int64	categorical -> nominal -> binary	1
profitability_ratio	float64	continuous -> ratio-scaled	5.53592675

A2. Types of instances of the Datasets - Metadata

field	data_format	data_attribute
adult	object	categorical -> nominal -> binary
id	object	categorical -> nominal
imdb_id	object	categorical -> nominal
original_title	object	categorical -> nominal
overview	object	categorical -> nominal
popularity	float64	continuous
tagline	object	categorical -> nominal
title	object	categorical -> nominal
genres_edited	object	categorical -> nominal
spoken_languages_edited	object	categorical -> nominal
production_countries_edited	object	categorical -> nominal

A3. Types of instances of the Datasets - Keywords

field	data_format	data_attribute
id	int64	categorical -> nominal
keywords_edited	object	categorical -> nominal

A4. Exploring Continuous Features - Gross

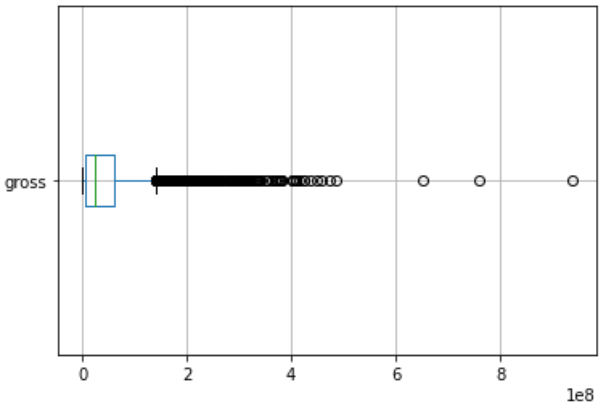
Figure on the right represents the data summary based on the statistical description of the feature gross. There are measures of central tendency (mean, median), dispersion (quartiles, variance, standard deviation, interquartile range), and some other statistical descriptions, like minimum, maximum and count. The gross column has 3524 non-empty cells. The minimum value of the budget is 0, while the maximum - 93,662,225.

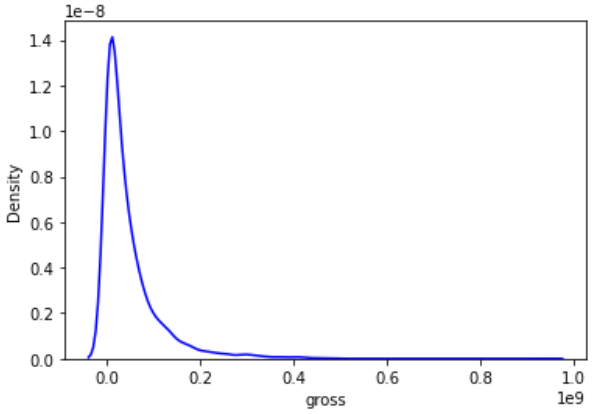
property	value
count	3524
min	0
max	936662225
mean	47725202
median	25111099
std.dev	67071214
variance	4498547873591042
Q1	6955428
Q3	60366724
IQR	53411296

Figure on the left side below shows the boxplot of the feature gross. The graph represents five-number summary: “minimum,

first quartile, median, third quartile, and ''maximum". The ends of the box are the quartiles, Q1 and Q3, while the box length is the interquartile range (IQR). While from the graph, we can get an approximate value of Q1 and Q3, it can be seen that the exact value of Q1 is 6,955,428 and Q3 is 60,366,724 from Figure 4. The median is marked as the line within the box, which is equal to 25,111,099. Two lines (called whiskers) outside the box extend to the smallest (Minimum) and largest (Maximum) observations. The circles are the outliers.

Figure on the right side below demonstrates that the gross's distribution is asymmetrical, because the tail is skewed to the right. As the tail is longer towards the right-hand side, the value of skewness is positive. The value of the skewness is 3.49795, which means that the distribution is highly skewed.





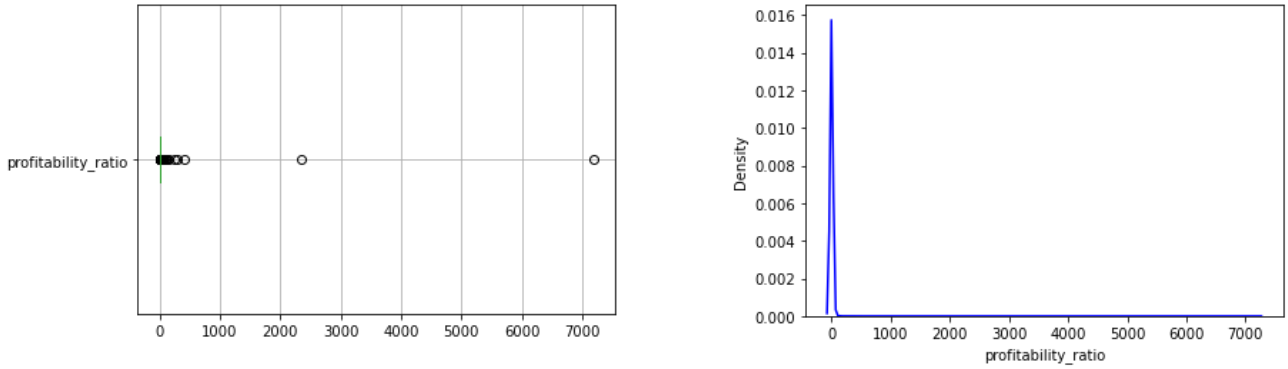
A5. Exploring Continuous Features - Profitability Ratio

Figure 7 represents the data summary based on the statistical description of the feature profitability ratio. There are measures of central tendency (mean, median), dispersion (quartiles, variance, standard deviation, interquartile range), and some other statistical descriptions, like minimum, maximum and count. The profitability ratio column has 3524 non-empty cells. The minimum value of the budget is -0.999979, while the maximum - 7,193.587333.

property	value
count	3524.000000
min	-0.999979
max	7193.587333
mean	3.996876
median	0.020240
std.dev	127.868810
variance	16350.432647
Q1	-0.552571
Q3	1.029155
IQR	1.581726

Figure 8 shows the boxplot of the feature profitability ratio. The graph represents five-number summary: "minimum, first quartile, median, third quartile, and ''maximum". The ends of the box are the quartiles, Q1 and Q3, while the box length is the interquartile range (IQR). While from the graph, we can get an approximate value of Q1 and Q3, it can be seen that the exact value of Q1 is -0.552571 and Q3 is 1.029155 from Figure 7. The median is marked as the line within

the box, which is equal to 0.02024. Two lines (called whiskers) outside the box extend to the smallest (Minimum) and largest (Maximum) observations. The circles are the outliers.
Figure 9 demonstrates that the profitability ratio's distribution is asymmetrical, because the tail is skewed to the right. As the tail is longer towards the right-hand side, the value of skewness is positive. The value of the skewness is 52.233907, which means that the distribution is highly skewed.



A6. Exploring Categorical Features - Data Exploration Report

Please refer to the report in the following link for detailed information about our categorical features:
https://github.com/oguzhanakan0/comp541/blob/master/reports/assignment-3-data-exploration.html

A7. Data Cleaning & Transformation Report

Please refer to the report in the following link for detailed information about our data cleaning and transformation process:
https://github.com/oguzhanakan0/comp541/blob/master/reports/assignment-4-data-cleaning.html

A7. Feature Selection Report

Please refer to the report in the following link for detailed information about our feature selection process:
https://github.com/oguzhanakan0/comp541/blob/master/reports/assignment-5-feature-selection.html

A8. Modeling Report

Please refer to the report in the following link for detailed information about our modeling process:
https://github.com/oguzhanakan0/comp541/blob/master/reports/assignment-6-modeling.html