12/8/2020

Profitability Prediction of Movie Projects

Final Presentation

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

California State University, Northridge
COMP 541 - Data Mining - F2020
Final Presentation

Table of Contents

Introduction

Background

- The film industry has grown immensely over the past few decades.
- The movie producers and directors need to be able to predict the demand in order to reduce the uncertainty in the market and to increase the chance of profitability

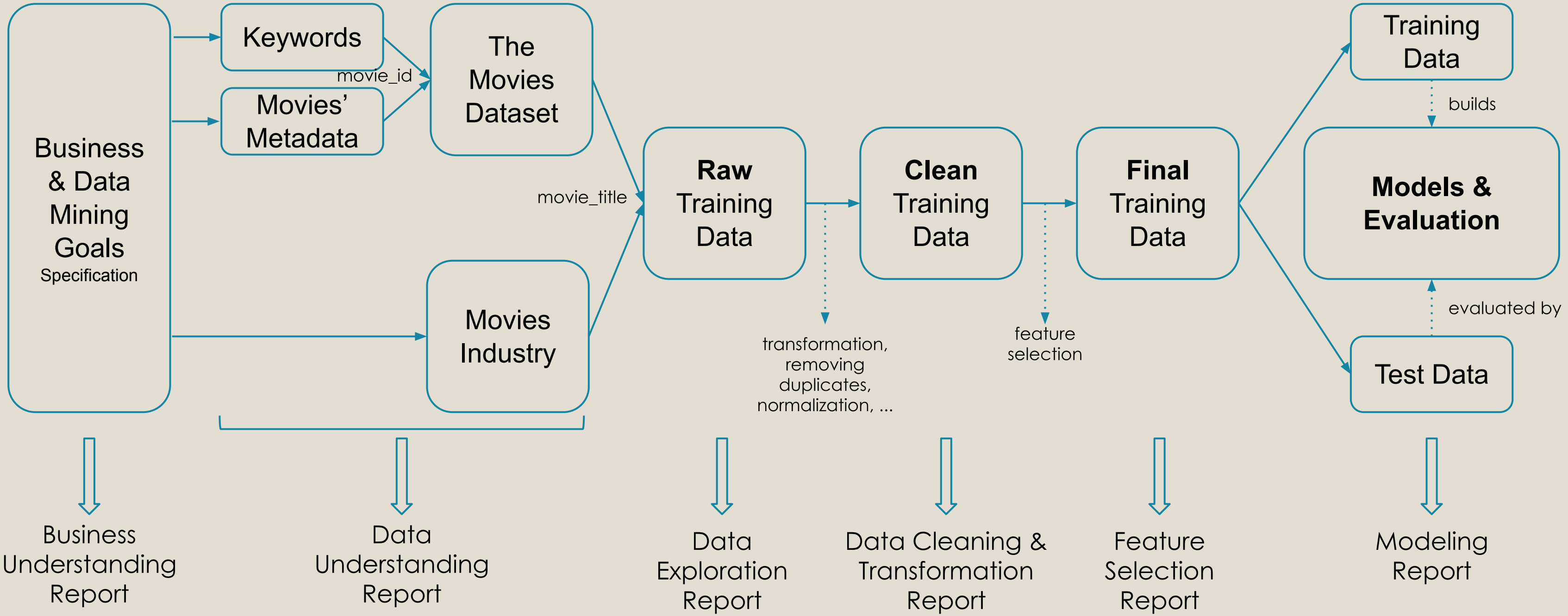
Objective

- The goal is to accurately and precisely predict the profitability of the movie project before starting a production.

Dataset

- Kaggle
- The Movies Dataset (MD): cast, crew, plot, keywords, budget, revenue, posters, release dates, languages, production companies, countries, vote averages.
- Movie Industry (MI): movies' budget and revenue information from 1986-2016.

Introduction - **Project Flow & Structure**



Methodology

Tools/Language

- Python

Algorithms

- K-Nearest Neighbors, Random Forest, Gradient Boosting

Performance Metrics

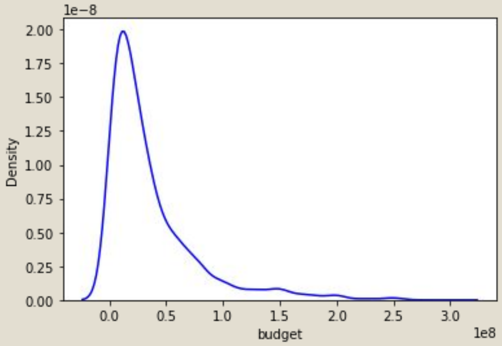
- Classification: Accuracy, Sensitivity, Precision - F1 Score
- Regression - MedianAE, MeanAE, RMSE

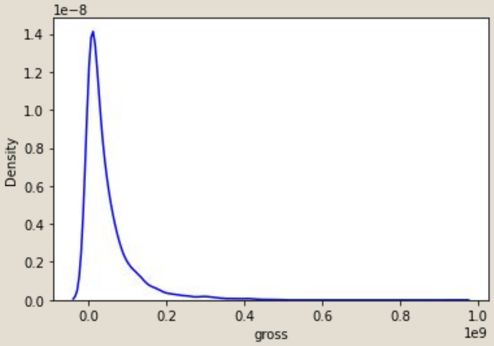
Data Exploration - **Exploring features and their dependency**

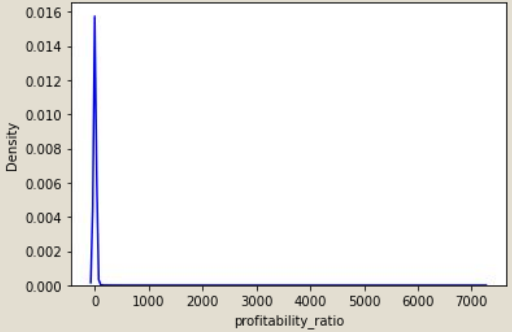
- In our data-set we have 3 continuous features such as budget, gross, and profitability ratio.
- By exploring them we can determine minimum, maximum, mean, median, variance, and standard-deviation of the features.
- For the categorical features in our data, a class-distribution of the categorical features among our dataset can be done and we can obtain the imbalances.
- To identify whether two categorical attributes are independent or dependent, we used chi-square test.
- From the results obtained by performing chi-square test on all the possible pairs of the categorical feature, we can observe that what features are dependent on what feature.

Data Exploration - **Exploring features and their dependency**

- All of our continuous features had positive skew:
  - Budget: 2.25 → improved after excluding outliers (new skew is 1.01)
  - Gross: 3.50 → not used in modeling, skewness calculated just for exploratory purpose
  - Profitability Ratio: 52.23 → improved after excluding outliers (new skew is 1.19)

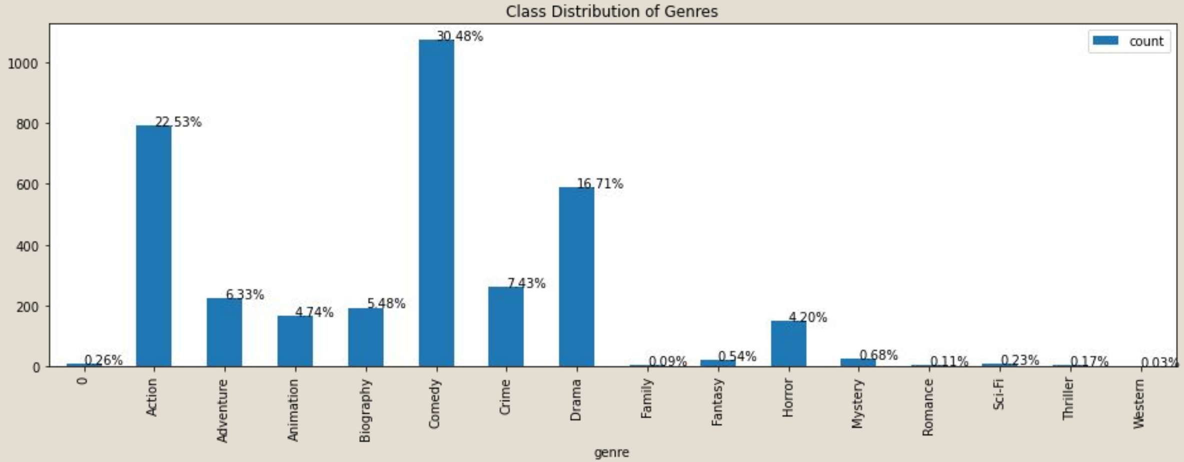






Data Exploration - **Exploring features and their dependency**

- Sample class distribution graph of **genres**:
  - The genre “comedy” is a balanced category
  - Some other features are slightly imbalance and some are severely imbalance



Data Exploration - **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 this section.
- 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 makes sense! For example, most of the family movies have comedy inside as well.

After applying apriori algorithm and calculating the association metrics, we generate the following rules:

```
rules = association_rules(f
rules.sort_values('confiden
```

	antecedents	consequents
16	(Family)	(Comedy)
2	(Romance)	(Drama)
36	(Mystery)	(Thriller)
40	(Crime, Drama)	(Thriller)
8	(Romance)	(Comedy)

Data Cleaning & Transformation

- overview
- tagline
- spoken_languages_edited
- production_countries_edited
- keywords_edited

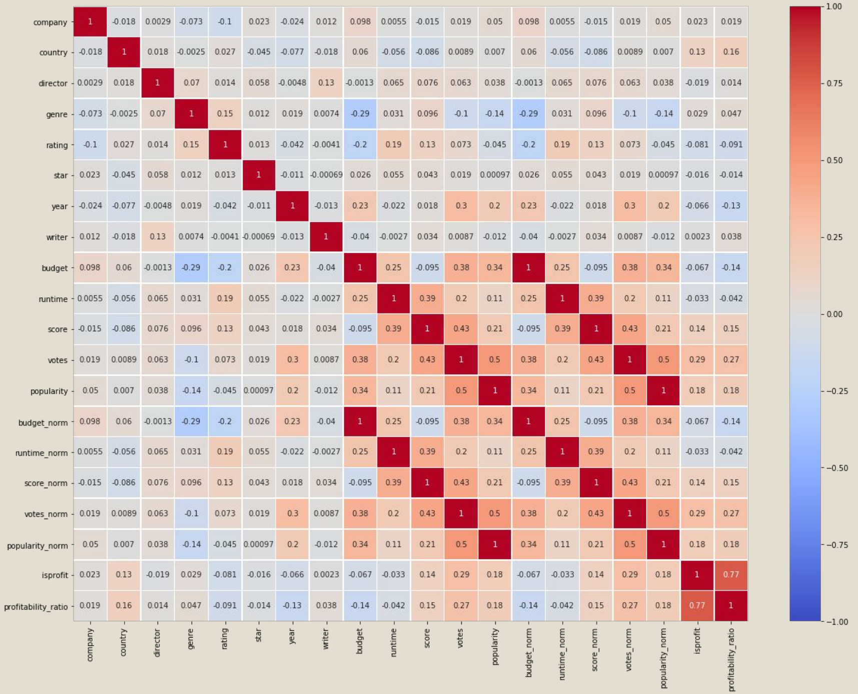
- budget
- runtime
- score
- votes
- popularity

Feature Selection

- All the features are divided into four groups- categorical_features, continuous_features, normalized_features, and target_features.
- The required continuous_features are converted to numerical data.
- Calculating the correlation matrix for all the features we get a 20 x 20 matrix.
- A heat map is created of the matrix obtained so we can visualize that which features are correlated to other features.
- Our aim here is to find the maximum profitability ratio.
- From this matrix we can determine what features are correlated to profitability ratio.
- The features that have correlation higher than 0.04 or less than -0.04 with profit and profitability ratio are selected

Feature Selection - **Heatmap**

- We observed how much two features are correlated and selected according to our predefined threshold.
- The number of features increased for this analysis because new ones are produced during Data Cleaning & Transformation Step



Modeling - **Overview**

Two supervised model types:

1. Classification: Will the movie profit?
2. Regression: How much the movie will profit compared to its budget?

Algorithms used:

1. K-Nearest Neighbors
2. Random Forest
3. Gradient Boosting

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)
```

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

```
# 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
```

Classification Models

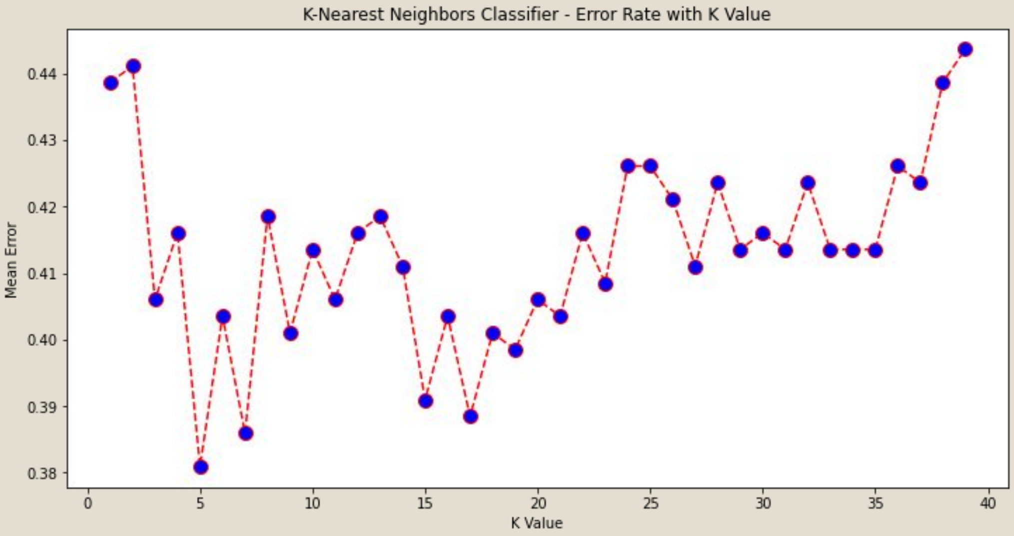
Modeling - Classification Model - **KNN**

Parameter setting:

1. n_neighbors (k) = 2, …, 20
2. weights = ‘uniform’

Best value of k → 5

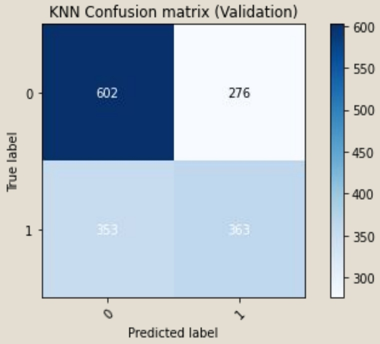
Note that both very-low and very-high number of neighbors result in high error rate. Very-low neighbors cause underfitting and very-high neighbors cause overfitting.



Modeling - Classification Model - **KNN**

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 on the right, KNN model reached ~0.6 accuracy and ~0.5 F-1 score in validation and test data.

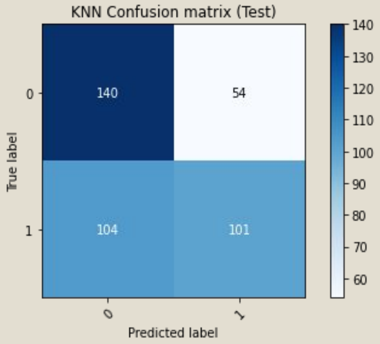


Accuracy: **0.605**

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**

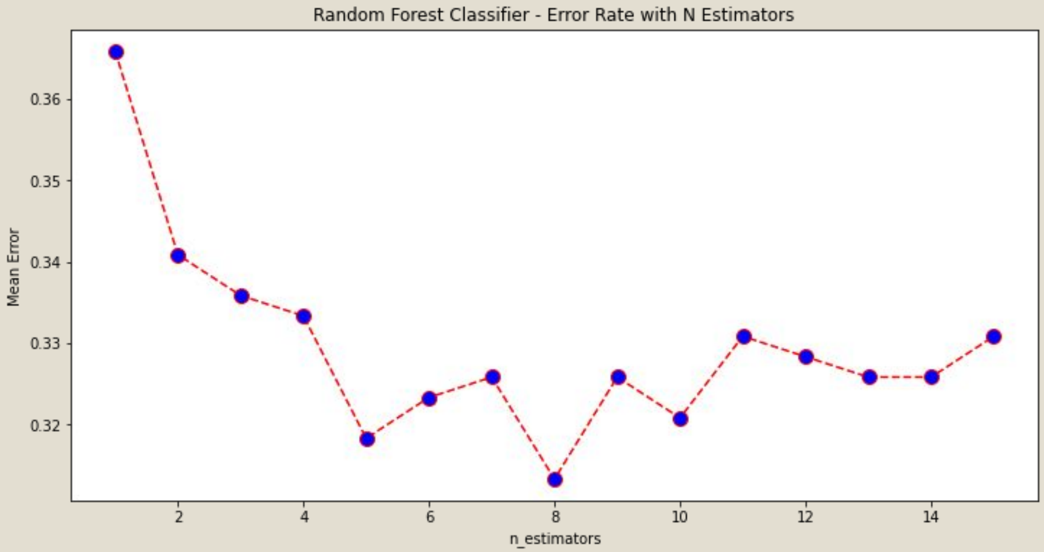
Modeling - Classification Model - **Random Forest**

Parameter setting:

1. n_estimators = 1, …, 15
2. min_samples_split = 60
3. min_samples_leaf = 20
4. max_depth = 7
5. max_leaf_nodes = 14

Best value of n_estimators → 8

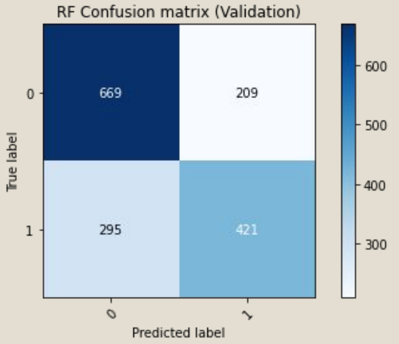
Again, using a lot of trees may increase the error rate by causing overfitting.



Modeling - Classification Model - **Random Forest**

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 on the right, Random Forest model reached ~0.68 accuracy and ~0.65 F-1 score in validation and test data.

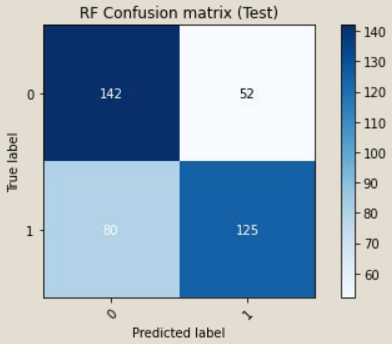


Accuracy: **0.684**

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**

Modeling - Classification Model - **Gradient Boosting**

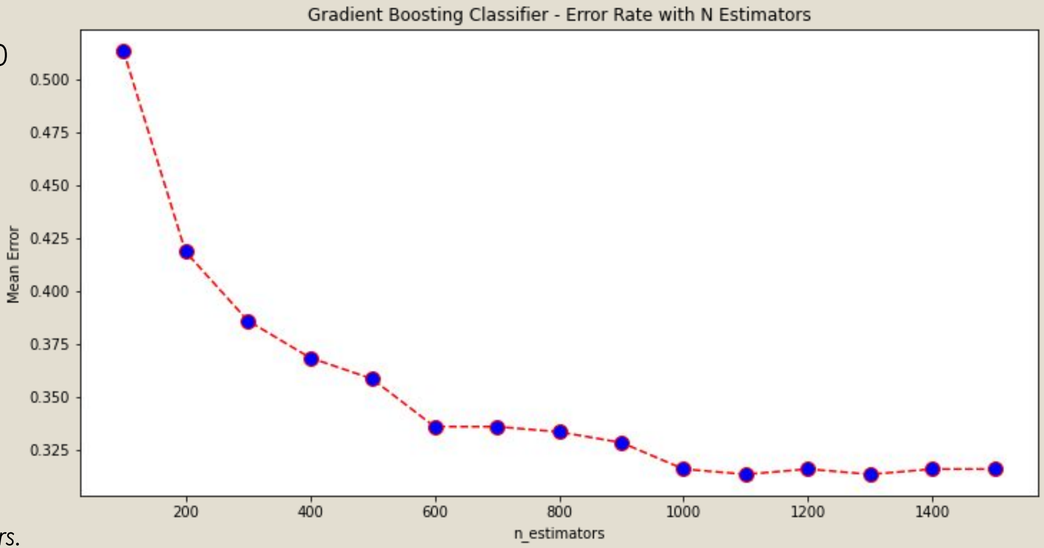
Parameter setting:

1. n_estimators=100, 200, …, 1500
2. learning_rate=0.001
3. criterion='friedman_mse'
4. min_samples_split=60
5. min_samples_leaf=20
6. max_depth=7
7. max_leaf_nodes=14

Best value of n_estimators → 1000

Note that when n_estimators >= 1000, the improvement is negligible.

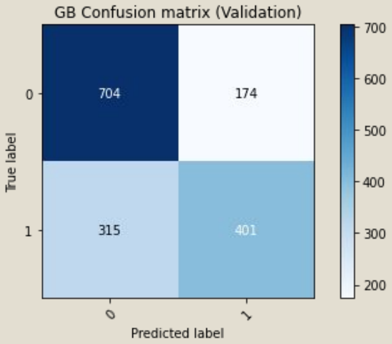
The overfitting problem observed in KNN and Random Forest is non-existent for Gradient Boosting algorithm, because Gradient Boosting works in a way that improves its errors. Thus, we do not expect the error rate to increase as we iterate over n_estimators.



Modeling - Classification Model - **Gradient Boosting**

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 on the right, KNN model reached ~0.69 accuracy and ~0.65 F-1 score in validation and test data.

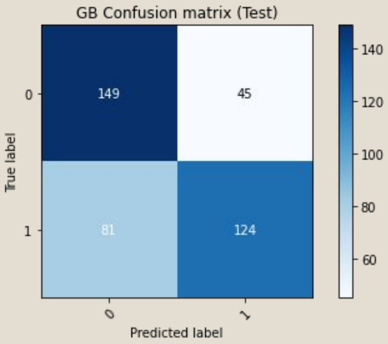


Accuracy: **0.693**

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**

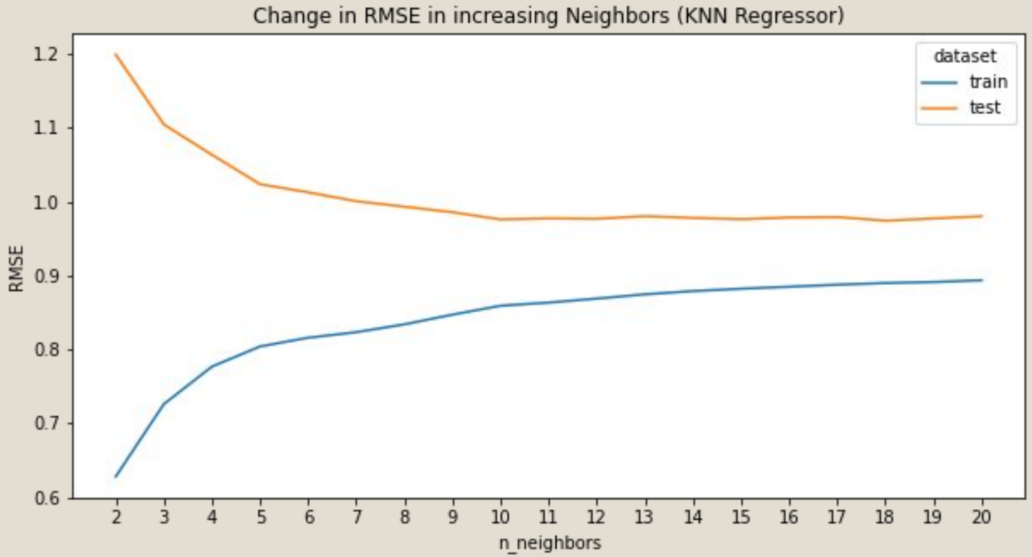
Regression Models

Modeling - Regression Model - **KNN Regression**

Parameter setting:

1. n_neighbors (k) = 2, …, 20

Best value of k → 18



At k = 18;

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

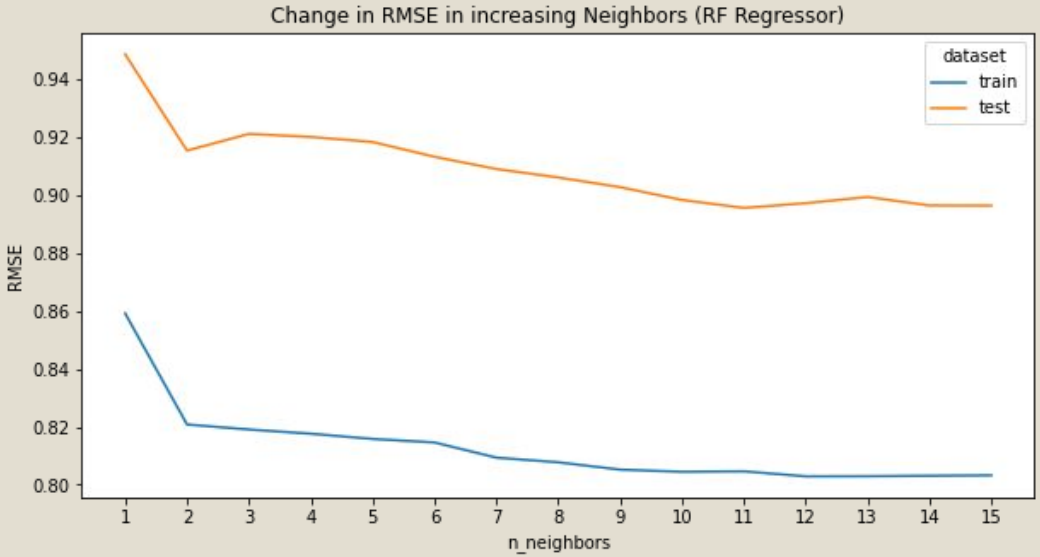
[TEST] >> median_abs_err = 0.597, mean_abs_err = 0.742, **RMSE = 0.974**

Modeling - Regression Model - **Random Forest**

Parameter setting:

1. n_estimators = 1, …, 15
2. min_samples_split = 60
3. min_samples_leaf = 20
4. max_depth = 7
5. max_leaf_nodes = 14

Best value of n_estimators → 15



At n_estimators = 15;

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

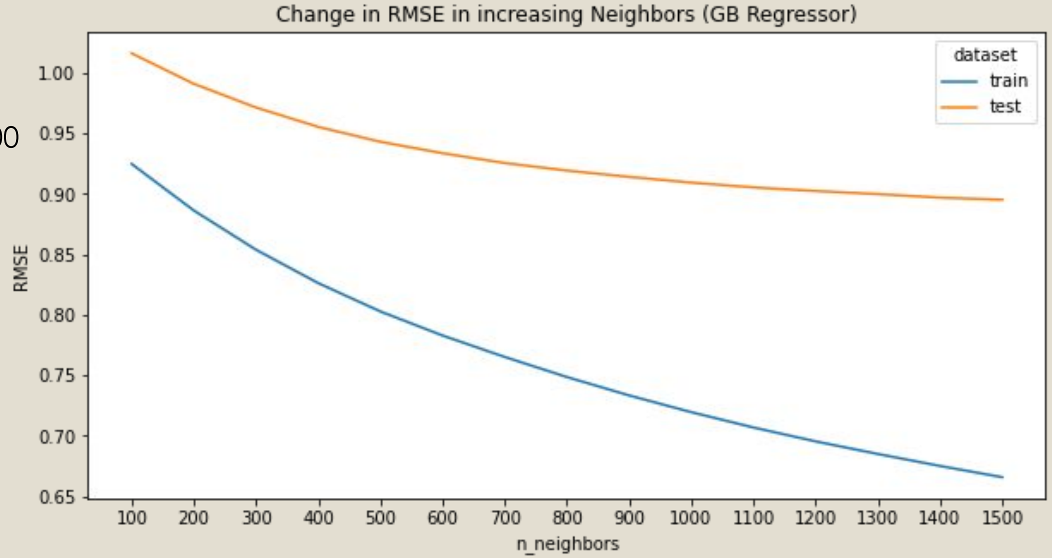
[TEST] >> median_abs_err = 0.528, mean_abs_err = 0.667, **RMSE = 0.896**

Modeling - Regression Model - **Gradient Boosting**

Parameter setting:

1. n_estimators=100, 200, …, 1500
2. learning_rate=0.001
3. criterion='friedman_mse'
4. min_samples_split=60
5. min_samples_leaf=20
6. max_depth=7
7. max_leaf_nodes=14

Best value of n_estimators → 1500



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**

Evaluation - **Classification Models**

	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
	Overall, how often is the classifier correct?	*When it's actually yes, how often does it predict yes?*	*When it predicts yes, how often is it correct?*	*Harmonic mean of sensitivity and precision*	*Overall, how often is the classifier correct?*	*When it's actually yes, how often does it predict yes?*	*When it predicts yes, how often is it correct?*	*Harmonic mean of sensitivity and precision*

Evaluation - **Regression Models**

	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
	Midpoint of the sorted absolute errors	*Average of the absolute errors*	*Root Mean Squared Error*	*Midpoint of the sorted absolute errors*	*Average of the absolute errors*	*Root Mean Squared Error*

Conclusion

- A movie success depends on a lot of features that are related to the movies, situation in the country, and so on.
- The movie profitability prediction helps movie production companies to invest in the movie projects.
- All of our classification models reached to our initial F-1 score target which was 0.4. Actually, the Random Forest and Gradient Boosting models reached to 0.65 which is significantly good. Besides, the difference between Validation and Test data metrics is very low, indicating a non-overfitting conclusion.
- Our Regression models also performed really promising as they all had a RMSE of less than 1.

Future Work/ Challenges & Lessons

Future Work

- Include data from social media
- Include more recent films
- Include more features
- Label films as average, success, failure

Lessons & Challenges

- Data mining project cycle
- Learning new algorithms and working on new platform
- Communication

Q & A

Thank you for your attention.
Project source code and phase reports can be found at Github:
https://github.com/oguzhanakan0/comp541