DIC - 4/587
Phase 2 - Report

Jyothismaria Joseph: CSE 487
Pratik Pokharel: CSE 587
Chirun Teja: CSE 587

Movie Market Prediction

Problem Statement : Estimating the popularity, fate and revenue of a film.

a) The global film industry is dynamic and intensely competitive. Success hinges on various factors, from creative elements to strategic marketing and business decisions. Predicting a film's revenue and popularity becomes crucial as it empowers stakeholders—such as production companies, distribution firms, and investors—to navigate this complex landscape with better foresight and precision. It helps them make informed decisions, allocate resources effectively, and stay ahead in an ever-evolving industry. Predicting revenue and popularity is significant because it directly impacts financial outcomes, enables more strategic decision making, reduces the risks of getting losses, and ensures the efficient allocation of resources.

b) The film industry involves substantial financial investments. Production costs, marketing expenses, and distribution budgets are considerable. Accurate revenue prediction is crucial to ensure that these investments yield profitable returns. The scope of this project is to help the production companies, investors across the globe in identifying the target audiences and optimizing advertising campaigns and also to allocate budgets more effectively, reduce the financial risks, and also optimize the allocation of different resources. This can leverage the revenue prediction to enhance their financial decision making and enhance the viewers experience too.

Section 1: Models and Visualizations

1.1 Logistic Regression

In this work, the movie data was preprocessed by filtering columns and **binning popularity(into low, average and high as 0, 1 and 2) and revenue(into hit, average and flop as 0,1 and 2)**. The problem was modeled into a **classification problem for five of the algorithms,** a **regression problem for one algorithm,** and a **clustering problem**. Altogether we worked on 7 algorithms, 4 of them were discussed in class and the remaining three from outside the class.We used **scaling as part of our preprocessing, following the previously done preprocessing works in phase 1**.

Initially, a **sanity check** was performed by feeding the output variable to the input set of features to check if our preprocessing was fine and logistic regression model works. The **sanity check gave 100% accuracy** which allowed us to proceed further. Since logistic regression was used for classification, we used classification metrics like accuracy, precision, recall and f1-scores to evaluate the model. The following scores were achieved:

Accuracy: *0.9080068143100511*
Precision: *0.90372669443329*
Recall: *0.9080068143100511*
F1 Score: *0.9055477951839763*

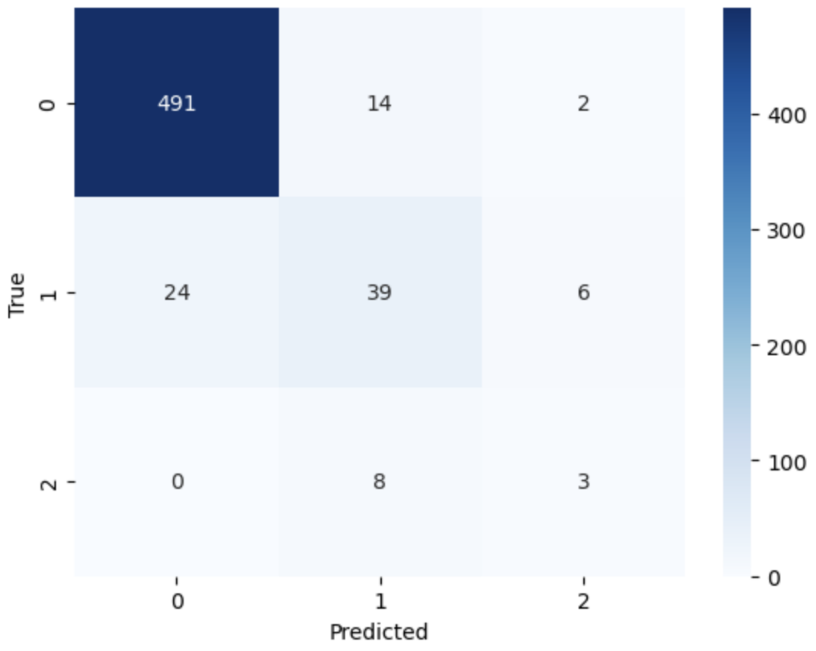

Fig 1.1: Confusion matrix for logistic regression

1.2 XGBoost Classifier

XGBoost, which stands for "eXtreme Gradient Boosting," is a powerful and efficient machine learning algorithm that belongs to the class of ensemble learning methods. Specifically, XGBoost is an implementation of gradient boosted decision trees designed for speed and performance.

XGBoost was used for classification, where we predicted the popularity class. The following scores were achieved:

Accuracy: *0.8926746166950597*
Precision: *0.8954163748228698*
Recall: *0.8926746166950597*
F1 Score: *0.8939800830384165*

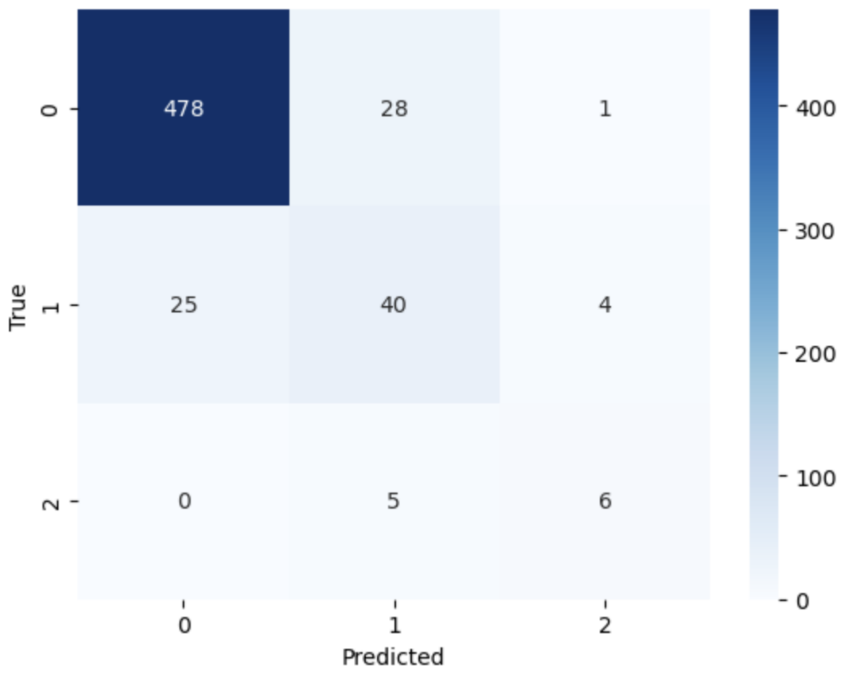

Fig 1.2: Confusion matrix for xgboost

1.3 Decision Tree Classifier

Decision tree is a non parametric supervised learning algorithm. The goal is to create a model that predicts the target by learning rules from the data. It builds a flowchart-like tree structure where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label. We are using this model to classify popularity.

Model Effectiveness:

The Decision Tree was used for a classification problem(classification of popularity), so we used Accuracy, Precision, Recall and F1-Score to evaluate the model. The following scores indicates that our model is working fairly well:

- ***Accuracy:*** *0.9011925042589438*
- ***Precision:*** *0.9037803927071388*
- ***Recall:*** *0.9011925042589438*
- ***F1 Score:*** *0.9021027558283271*

We also used a confusion matrix to see the model's prediction for the bins we created to categorize popularity.

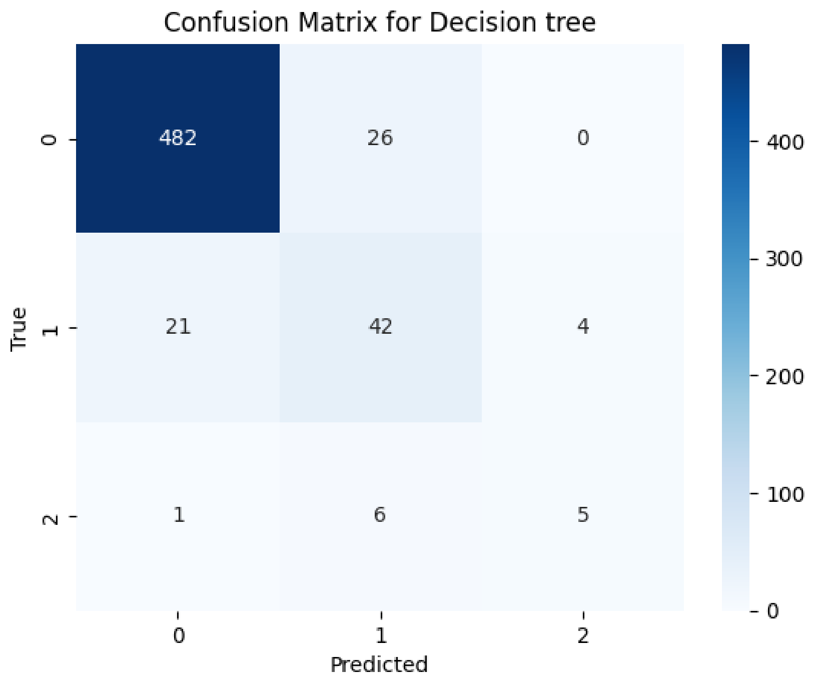

Fig 1.3: Confusion matrix for Decision Tree Classification

1.4 Random Forest

Random Forest combines the output of multiple decision trees to reach a single result. Instead of relying on a single decision tree, Random forest builds a “forest” of trees during its training.The random forest algorithm is an extension of the bagging method as it utilizes both bagging and feature randomness to create an uncorrelated forest of decision trees . We are using a random tree for regression. Random tree is used to predict a rough revenue based on the features.

Model Effectiveness:

The problem was modeled into a regression problem where the revenue was predicted. So, for evaluation, we used Mean Square Error, Root Mean Square Error and R-squared value to evaluate the model. The scores are following:

- ***Mean Squared Error:*** *8476393612946106.0*
- ***Root Mean Squared Error:*** *92067331.953012*
- ***R-squared value:*** *0.7788742037640555*

The mean squared error is high as the revenue had high variability. Revenue had a standard deviation of 187021342.284266. We plotted the R- squared trendline to better visualize the predictions from the model.

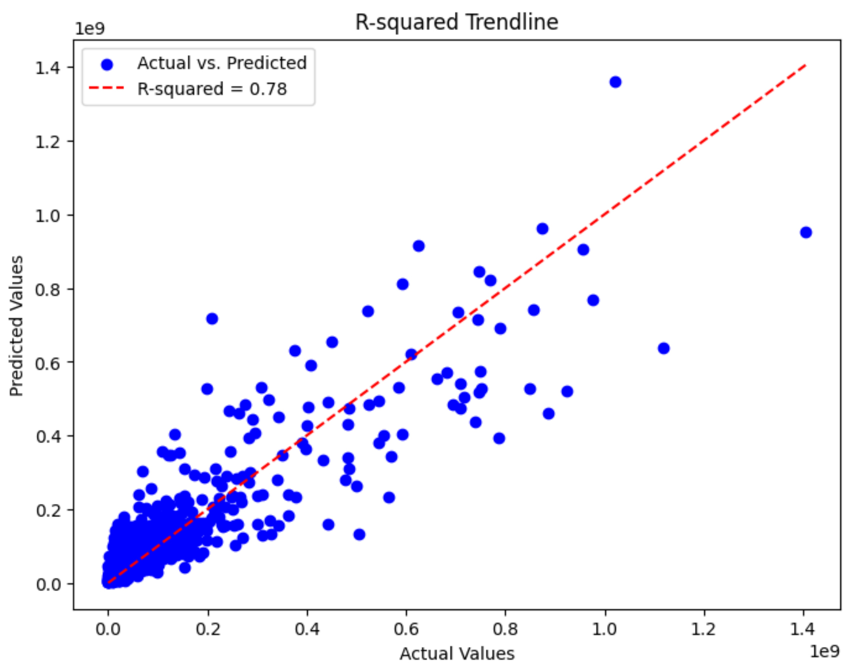

Fig 1.4: R-squared Trendline for Random Forest Regression

1.5 Support Vector Machine(SVM)

Support Vector Machines (SVMs) are supervised learning models used for classification and regression tasks. SVMs construct a hyperplane to maximize the margin between different classes in high-dimensional space. The vectors closest to the hyperplane are support vectors that influence where it is placed. Kernel functions like 'radial basis function' can transform data to make it more separable.

In this work, the prediction was done on the **hit/miss/average** status of a movie. The features were standardized to normalize ranges. An SVM model was initialized and its **hyperparameters- regularization parameter C, kernel type, and class weights were tuned using grid search cross-validation** to maximize F1 score. The optimized model was fit on the training data and then used to make predictions on the test set. The SVM achieved an **accuracy of 71% and F1 score of 72%** in classifying hit, miss or average.

- ***Accuracy:*** *0.7086882453151618*
- ***Precision:*** *0.7360719123652864*
- ***Recall:*** *0.7086882453151618*
- ***F1 Score:*** *0.7169367253594888*

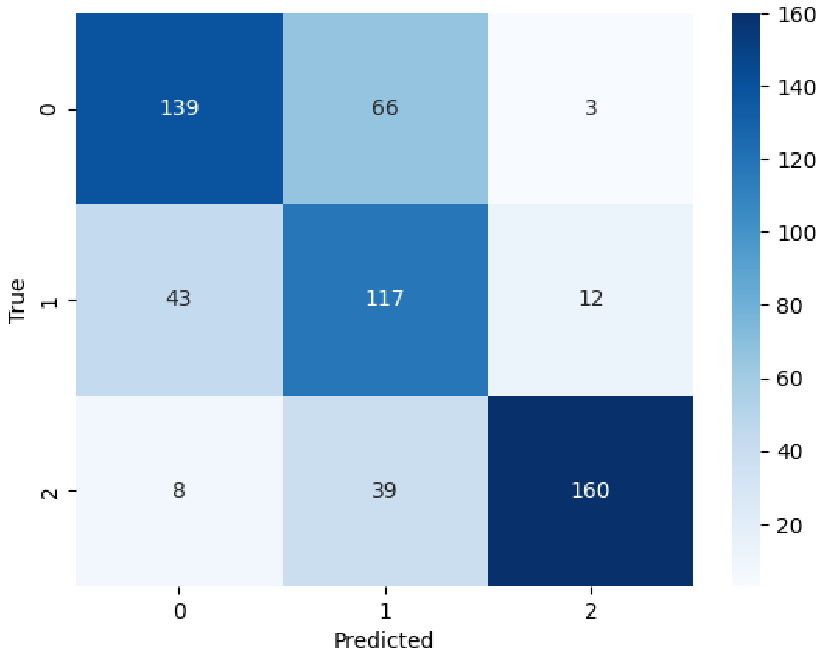

Fig 1.5: Confusion matrix for SVM Classifier

1.6 Naive Bayes Classifier

Naive Bayes classifiers are probabilistic models that make predictions using Bayes' theorem and the assumption that features are independent. They calculate conditional probabilities of classes given feature values. Multinomial Naive Bayes is suitable for classification with discrete features.

For this problem, the movie data was preprocessed in the same way as for SVM, and the prediction was done on the **hit/miss/average** status. A **MultinomialNB** model was initialized and fitted on the training data without much tuning. Predictions were made on the test set and evaluated using metrics like accuracy and F1 score. The Naive Bayes model achieved an accuracy of 63.4%, precision of 61.6%, recall of 63.4%, and F1 score of 62.2% in classifying movie hit/miss/average status.

- ***Accuracy***:*0.6337308347529813*
- ***Precision***:*0.6157813173758591*
- ***Recall***: *0.6337308347529813*
- ***F1 Score***: *0.6223906865478505*

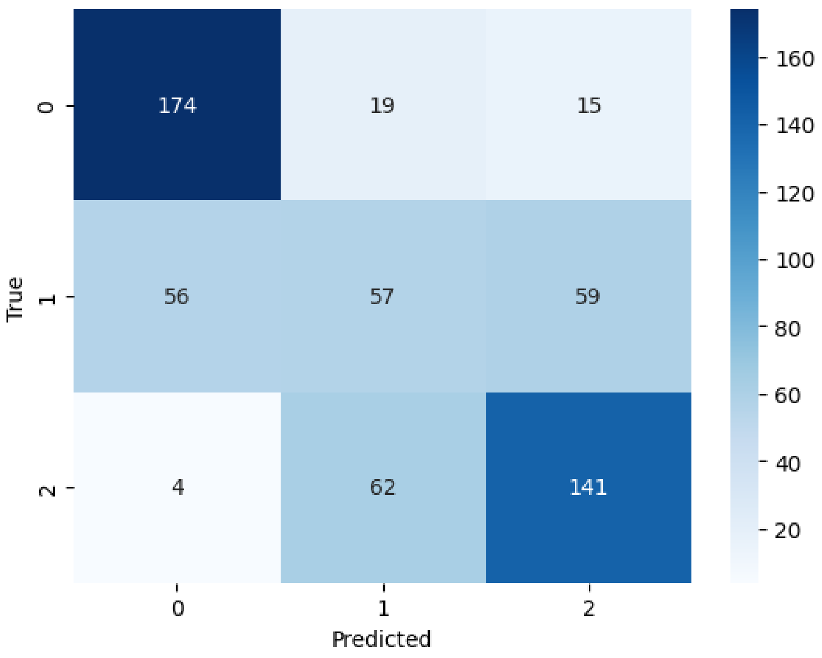

Fig 1.6(a): Confusion matrix for Naive bayes Classification

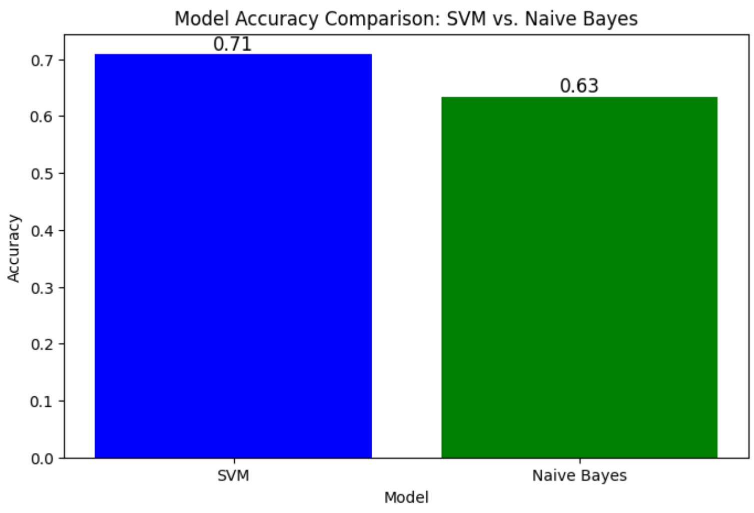

Fig 1.6(b): Comparison between SVM and Naive Bayes

1.7 K-means Clustering

The sklearn-KMeans class was used to create a k-means clustering model. Centroids were initialized randomly, no_of_clusters were assigned, and the seed for random number generation for reproducibility were assigned. The algorithm was run 50 times with different centroid seeds. After fitting the model, Principal Component Analysis (PCA) was applied to reduce the dimensionality of the feature space to 2 dimensions. This was done for visualizing the clusters.

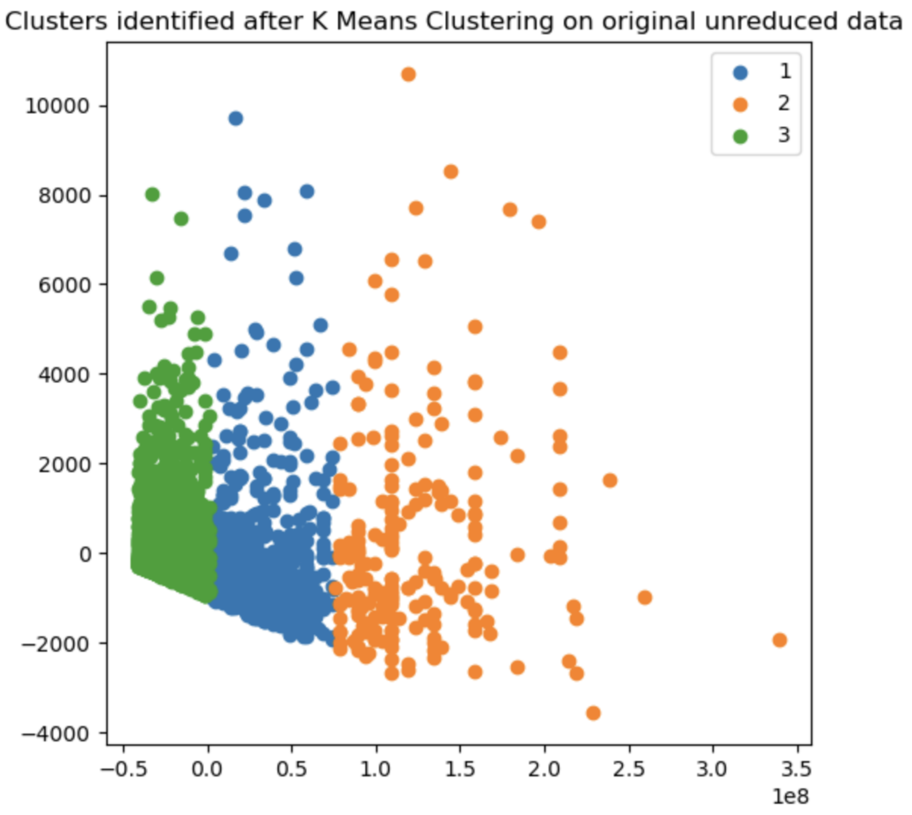

Fig 1.7: Clusters k=3

Section 2: Explanation and Analysis

2.1 Logistic Regression

The reasons of using Logistic Regression:

- In addition to getting results, we also wanted to understand the impact of each variable on the prediction. This feature is exhibited by logistic regression as it provides a straightforward and interpretable output. The coefficients represent the log-odds of the target variable, making it easy to understand the impact of each feature on the prediction.
- Since the project is performed with personal use laptops, we also had to make sure our computing resources are enough for model training. As Logistic regression is computationally efficient and can handle large datasets and a high number of features without requiring extensive computational resources, it aided our decision of using it.
- We have a multiclass classification problem(3 classes) and logistic regression naturally extends to handle both binary and multiclass classification problems.
- Our dataset had features of different scales. In fact, some features range within 100, while others range within billions. As Logistic regression is less sensitive to the scale of input features compared to some other algorithms, we decided to use it.

Feature Selection

For **feature selection, RFE(Recursive Feature Elimination**) Technique was used.

- The importance or contribution of each feature is evaluated based on a predefined criterion (e.g. coefficients in the case of linear models or feature importances in the case of tree-based models).
- The least important features (according to the ranking) are removed from the dataset.
- The above two steps are repeated iteratively until the desired number of features is reached. For our case we set the desired **number of features to 5**.

After selecting 5 features, we got the following classification scores:

> ***Accuracy***: *0.9080068143100511*
> ***Precision***: *0.9015729301410587*
> ***Recall***: *0.9080068143100511*
> ***F1 Score***: *0.9042608197452746*

Which is almost equal to the model that was trained using 26 features.

Hyperparameter Tuning on Logistic Regression:
We performed Grid Search Cross Validation Technique for Hyperparameter Tuning. We created a grid of hyperparameter values(**param_grid**) to search through. We explored different regularization types ('penalty'), inverse regularization strengths ('C'), and optimization algorithms ('solver').

The GridSearchCV algorithm systematically explores the hyperparameter space defined in **param_grid**. It trains the model with different combinations of hyperparameters, performs cross-validation to assess performance, and identifies the set of hyperparameters that yield the best performance. This process helps optimize the model for better generalization to unseen data. After tuning, the following best hyperparameters and classification scores were achieved. The scores achieved were **slightly higher** than what we achieved without tuning the model.

> ***Best Hyperparameters***: *{'C': 0.1, 'penalty': 'l1', 'solver': 'saga'}*
> ***Accuracy***: *0.909710391822828*
> ***Precision***: *0.9013446093940131*
> ***Recall***: *0.909710391822828*
> ***F1 Score***: *0.9047313852270022*

2.2 XGBoost Classifier

The reasons of using XGBoost:

- XGBoost provides parameters that allow for better handling of imbalanced datasets. **This was crucial in our classification task as we had popularity classes in the ratio of 1:4:30 and hit_miss_or_average classes in the ratio 1:1:3.**
- XGBoost is known for its high predictive accuracy. It is highly effective for structured/tabular data and can handle a mix of numerical and categorical features without the need for extensive pre-processing.
- XGBoost provides a built-in feature importance analysis, allowing you to understand the contribution of each feature to the model's predictions. This feature is valuable for feature selection and understanding the underlying patterns in the data.

For feature selection we used the same technique(RFE) that we used for Logistic Regression.

Hyperparameter Tuning on XGBoost:
We considered different values for the learning rate, the number of estimators (n_estimators), maximum depth of trees, subsample ratio and colsample_bytree. **Colsample_bytree** is the fraction of features (columns) to be randomly sampled and used when constructing each tree in the ensemble. It is a regularization technique designed to add diversity to the individual trees and reduce overfitting. Then we performed Grid Search Cross Validation. We achieved the following best hyperparameter values and slightly better classification results after tuning.

> ***Best Hyperparameters****: {'colsample_bytree': 0.8, 'learning_rate': 0.01, 'max_depth': 3, 'n_estimators': 100, 'subsample': 0.8}*
> ***Accuracy****: 0.9080068143100511*
> ***Precision****: 0.9023521324606234*
> ***Recall****: 0.9080068143100511*
> ***F1 Score****: 0.904384206834352*

2.3 Decision Tree
The reasons of using Decision Tree:
- The decision tree performs well for non linear datasets and is scalable. Since our dataset has 25+ features we needed a model that is scalable.
- Decision Tree can be used to classify multi class dataset. In our dataset we are categorizing the popularity into 3 bins, making it a multi class problem.
- Decision trees are fairly robust to outliers.

Training and tuning Decision Tree:
- We used the top 11 features for the decision tree. This was done to reduce the dimensionality of the data and eliminate features that are highly correlated.
- GridSearchCV was used to tune the parameter, It is an exhaustive search over specified parameter values for an estimator.
- The data was scaled using Standard scaler to improve the performance of the model.

Significant Findings:
To classify popularity, our model gave a higher importance vote_count and runtime among the top 11 best features, which are:
- vote_count
- runtime
- vote_average
- num_of_production_companies
- budget
- Adventure
- Comedy
- Science Fiction
- Action
- Fantasy
- Animation

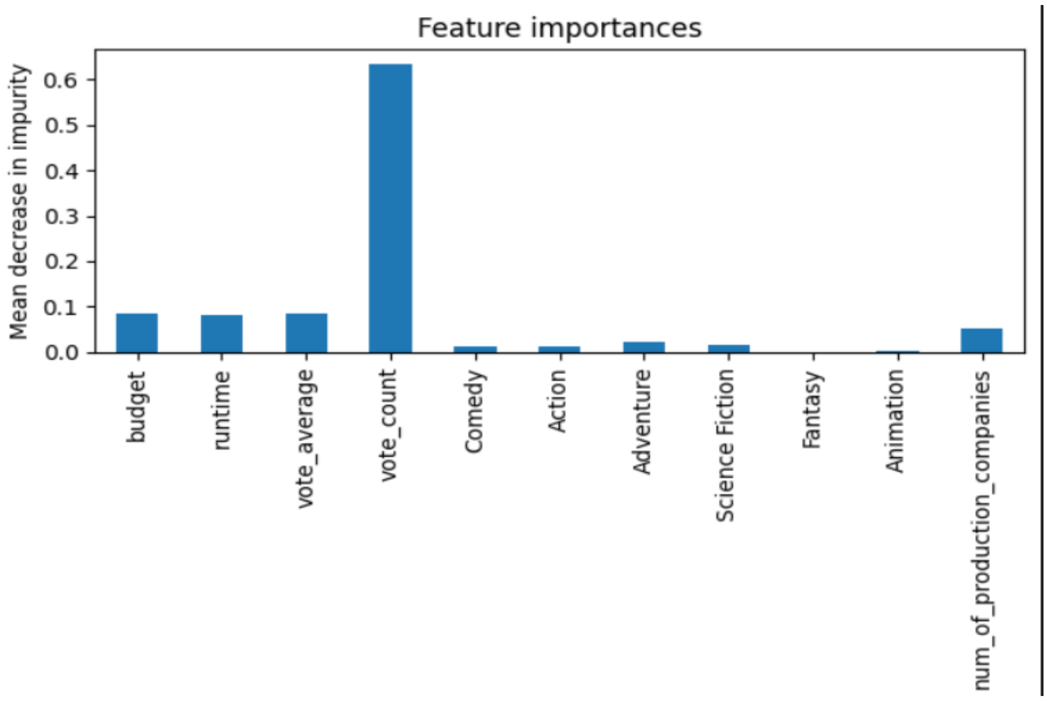

Fig 2.1: Feature Importances

2.4 Random Forest

The reasons of using Random Forest:

- Random forest performs well for non linear datasets and is scalable. Since our dataset has 25+ features this was necessary.
- Random reduces overfitting as it averages the predictions of multiple trees
- Random forest provides feature importance which helps us assess what features are highly influential to predict revenue.

Training and tuning Random Forest:

- We used the top 11 features. This was done to reduce the dimensionality of the data and eliminate features that are highly correlated.
- GridSearchCV was used to tune the parameter, It is an exhaustive search over specified parameter values for an estimator.
- The data was scaled using Standard scaler to improve the performance of the model.

Significant Findings:

To classify popularity, our model gave a higher importance vote_count and budget among the top 11 best feature, which are:

- vote_count
- Budget
- Thriller
- num_of_production_companies
- Adventure
- Romance
- Crime

- Action
- Family
- Animation
- Documentary
- Sentiment

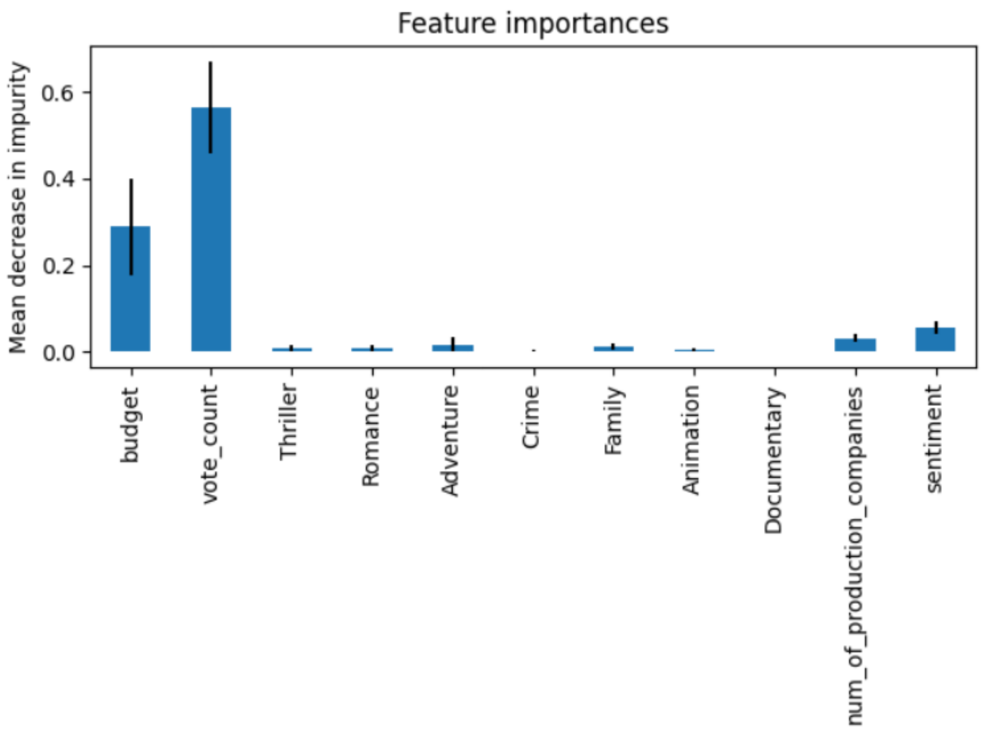

Fig 2.2: Feature Importances

Model can moderately predict movies that would have a high revenue, since there is a lot of variability in the data. The model tries to capture as much as possible. Overall, the model does fairly predict which movie would have a high revenue and which would do poorly in terms of revenue, which could help us answer our problem.

2.5 Support Vector Machines(SVMs)

The SVM algorithm was selected for its ability to handle complex nonlinear decision boundaries and categorical features present in the problem statement. It is also less prone to overfitting compared to other sophisticated models. The model was tuned by scaling the features, optimizing hyperparameters like C, kernel and class weights using **grid search cross-validation** to maximize the F1 score. This accounts for the class imbalance. The SVM achieved an **accuracy of 71%, precision of 73%, recall of 71% and F1 score of 72%** highlighting excellent performance in correctly classifying popularity with low false positives and negatives. The SVM performed with almost 8% better accuracy than Naive Bayes. On top of that we also tested with xgboost and logistic regression in predicting the fate of a movie, both of these models gave accuracy metrics similar to that of Naive Bayes. These observations demonstrate that the tuned SVM model was effective for predicting the fate of a movie.

2.6 Naive Bayes

The Naive Bayes algorithm was chosen for its suitability for classification tasks with discrete features and high dimensional data by selecting relevant features. It is also very fast to train compared to complex models. The model was tuned by adding a constant to avoid zeros and using the Multinomial variant suitable for discrete data. However, the **Naive Bayes only achieved 63% accuracy, 61% precision, 63% recall and F1-score of 62%** indicating reasonable but inferior performance compared to SVM. The lower scores show higher misclassifications across all classes. **While Naive Bayes has some advantages, its assumption of independent features likely caused poorer effectiveness for the complex patterns in movie hit/miss prediction.**

2.7 K-means Clustering

The reason to perform K-means clustering was basically to find how many bins is proper for making our problem a classification problem. For finding the optimal value of k we performed the elbow method. We ran K-means clustering for a range of values of k and plotted the sum of squared distances (**WCSS** or Within Cluster Sum of Squares of Distances) against k. The "elbow" of the curve represents a point where the rate of decrease in WCSS slows down. The optimal k is often chosen at the point where adding more clusters does not significantly reduce WCSS.

We saw the best value for k to be 3-5. For our problem we used k=3 because not much difference was seen between 3 and 5.

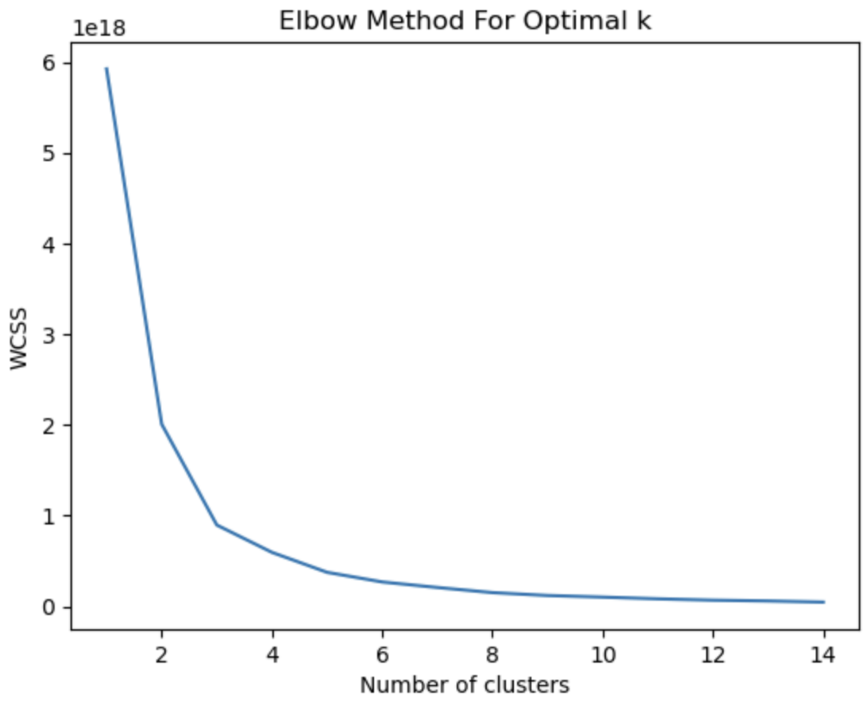

Fig 2.3: Finding optimal k using wcss and elbow method

Section 3: Conclusion

To sum up, our machine learning models, trained on a diverse set of features, demonstrated promising performance in predicting movie popularity, revenue and fate. The accuracy of SVM over other models by almost 8% on the test set indicates its effectiveness in capturing underlying patterns. In conclusion, our work offers valuable insights into the dynamics of movie success. By leveraging these findings, filmmakers and studios can make informed decisions in the realms of production, marketing, and distribution, ultimately increasing the likelihood of creating movies that captivate audiences and generate substantial revenue.

References:

- https://raw.githubusercontent.com/rashida048/Datasets/master/home_data.csv [Data Source]
- https://www.geeksforgeeks.org/decision-tree/#
- https://scikit-learn.org/stable/modules/grid_search.html#grid-search
- https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html
- https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier
- https://matplotlib.org/
- https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html
- https://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html
- https://www.investopedia.com/terms/r/r-squared.asp
- https://www.ibm.com/topics/random-forest
- https://matplotlib.org/
- https://stackoverflow.com/questions/51262395/plotting-the-r-square-error-for-regression-model-results
- https://xgboost.readthedocs.io/en/stable/python/python_api.html
- https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
- slides/shamsad_parvin