Movie Revenue and Rating Prediction

Submitted by:
Aashi Nagpal - 102216118
Yash Yadav - 102216119

BE Third Year CSE

Submitted to:

Ms. Kudratdeep Aulakh

Assistant Professor

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

Computer Science and Engineering Department

Thapar Institute of Engineering and Technology, Patiala

INDEX

Introduction or Project Overview

1. Introduction

The movie industry is one of the largest entertainment sectors, generating billions of dollars annually. However, predicting the success of a movie remains a challenging task due to the multitude of factors influencing its revenue and audience reception. Production companies invest substantial resources into movies, often without a clear understanding of how well the final product will perform.

2. Project Overview

This project focuses on leveraging machine learning to predict two critical aspects of movies—revenue and audience ratings—which are essential for the decision-making processes of movie production companies. With the help of the TMDB 5000 Movie Dataset, the project combines historical data analysis and advanced predictive techniques to provide actionable insights for movie investments.

3. Scope and Methodology

The scope of this project extends beyond simple predictions of movie revenue and audience ratings. It provides actionable insights for various stakeholders like production companies, distributors and investors. By combining historical data and predictive analytics, this project aims to bridge the gap between creativity and business acumen, enabling data-informed decision-making and reducing risks associated with movie production. It not only helps optimize resources but also paves the way for innovation by identifying successful combinations of creative elements.

The methodology involves data preprocessing, exploratory data analysis (EDA), modeling, optimization and evaluation. This streamlined approach ensures reliable predictions, offering significant utility in planning, resource allocation, and marketing strategies.

Libraries Used

- **ast**: Provides tools for working with and modifying Python's abstract syntax tree.
- **matplotlib.pyplot**: A library for creating static, animated, and interactive visualizations in Python.
- **numpy**: Supports large, multi-dimensional arrays and matrices, along with mathematical operations on these structures.
- **pandas**: A data manipulation and analysis library, offering data structures like DataFrames.
- **pickle**: A module for serializing and deserializing Python objects.
- **seaborn**: A statistical data visualization library built on top of matplotlib.
- **sklearn.ensemble**: Provides ensemble methods like Random Forest, Gradient Boosting, etc., for machine learning.
- **sklearn.linear_model**: Contains linear models like linear regression and logistic regression.
- **sklearn.metrics**: Provides tools for evaluating the performance of machine learning models.
- **sklearn.model_selection**: Offers tools for splitting data into training and testing sets, cross-validation, etc.
- **sklearn.multioutput**: Provides tools for multi-output supervised learning.
- **sklearn.preprocessing**: Contains utilities for data preprocessing like scaling and encoding.
- **sklearn.tree**: Includes decision tree algorithms for classification and regression.

Algorithms Used

1) Linear Regression

Linear Regression is a fundamental algorithm for modeling the relationship between a dependent variable and one or more independent variables. It assumes a linear relationship, predicting outcomes by minimizing the sum of squared errors between predicted and actual values. Despite its simplicity, Linear Regression is effective for problems where the data follows linear trends, providing insights into feature significance through model coefficients.

2) Random Forest

Random Forest is an ensemble learning method that builds multiple decision trees during training and merges their outputs to improve accuracy and reduce overfitting. Each tree is trained on a random subset of the data and features, creating diversity in predictions. This robustness makes Random Forest effective for both classification and regression tasks.

3) Decision Tree

A Decision Tree splits the dataset into smaller subsets based on feature values, creating a tree structure with decision nodes and leaf nodes. It is intuitive and interpretable but can overfit without constraints. It supports both classification and regression tasks by modeling non-linear relationships in data.

Hyperparameter Tuning

Hyperparameter tuning optimizes model performance by systematically adjusting parameters like tree depth, learning rate, or the number of estimators. Techniques like Grid Search and Randomized Search test combinations of hyperparameter values, while advanced methods like Bayesian Optimization or automated tools (e.g., Optuna) efficiently explore the search space.

Overview of the Dataset used

Key Features

Numerical Features
Budget: The production budget of the movie (in dollars).
Popularity: A numerical score indicating the popularity of the movie based on audience engagement.
Runtime: The total runtime of the movie (in minutes).
Vote Count: The total number of audience votes received for the movie.
Revenue: The total earnings of the movie (in dollars).
Vote Average: The average audience rating for the movie (on a scale of 1 to 10).

Categorical Features

Genres: A list of genres associated with the movie (e.g., Action, Comedy, Drama).

Other Features (Excluded During Preprocessing):
Additional metadata such as production companies, release dates, and spoken languages were excluded to focus on core predictors and simplify the modeling process.

Target Variables:

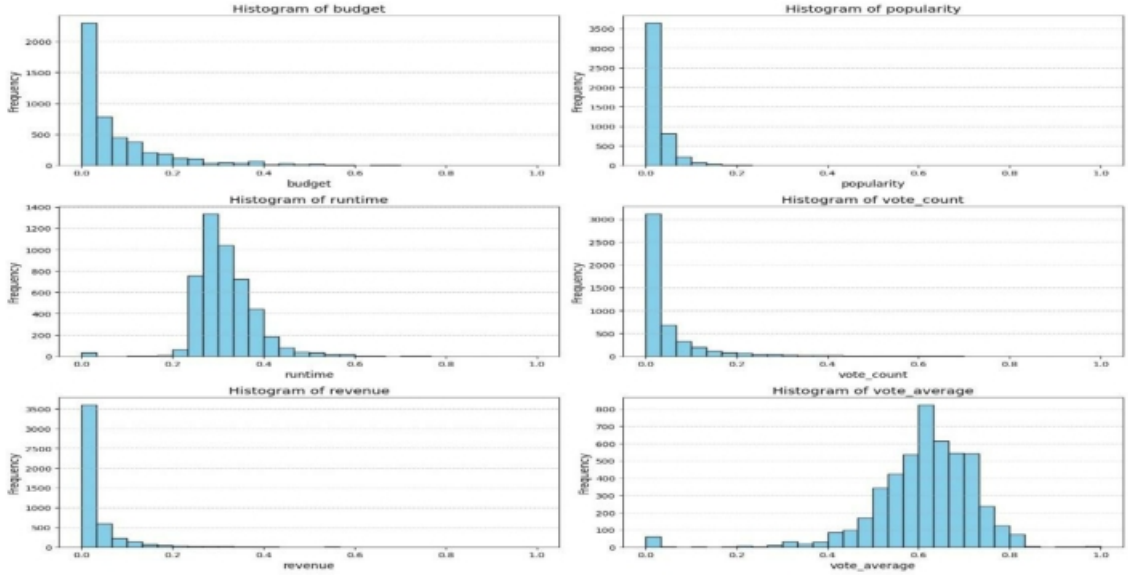
The project predicts two target variables:
Revenue: A measure of financial success.
Vote Average: A measure of audience reception.

Dataset Characteristics

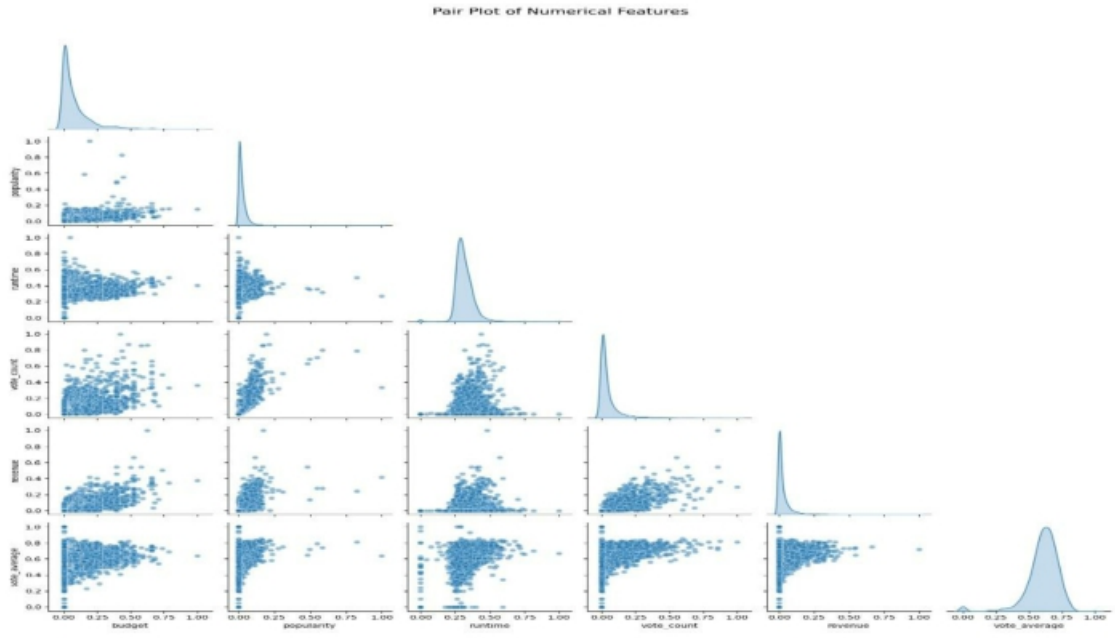
Size: The dataset contains 5000 entries, each representing a unique movie.
Variability: It includes movies from diverse genres, languages, and budgets, offering a rich mix of data for training robust models.
Representation: The dataset is representative of global cinema, capturing a wide spectrum of movie types, from blockbuster hits to independent films.
This dataset provides an excellent foundation for developing predictive models that can uncover meaningful patterns and insights about movie success.

Data Exploration

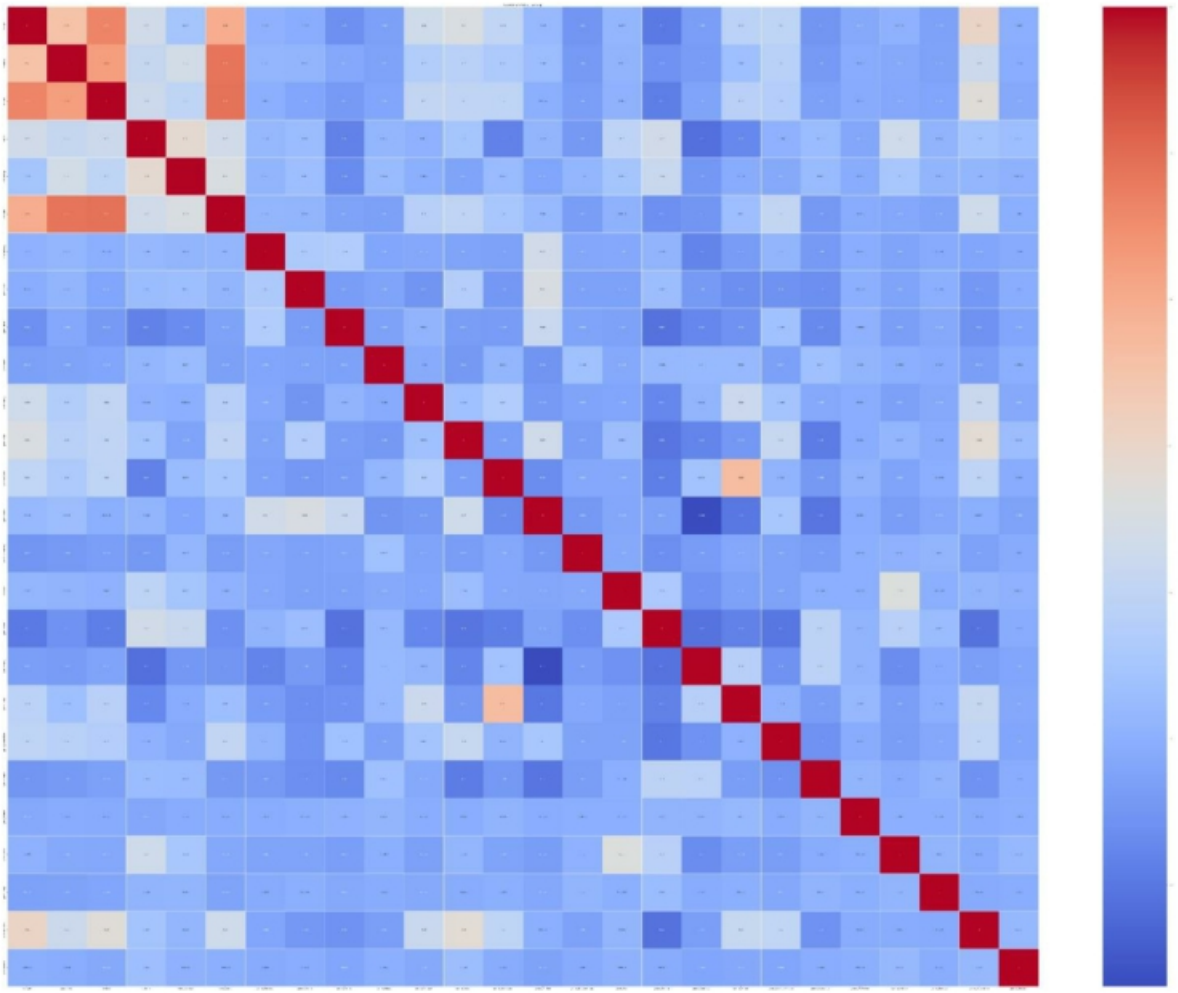
1. Histograms of Numerical Columns:



2.Pair Plot



3.Correlation Matrix



Code Snippets

The project was implemented through the following steps:

Data Preprocessing

1. **Handling Missing Values**: - The dataset was examined for missing values and no missing data was found.

```
movies.isna().any()
✓ 0.0s
budget                  False
popularity              False
revenue                 False
runtime                 False
vote_average            False
vote_count              False
genre_Mystery           False
genre_Crime             False
genre_Horror            False
genre_Music             False
genre_Fantasy           False
genre_Action            False
genre_Animation         False
genre_Thriller          False
genre_Documentary       False
genre_War               False
genre_Drama             False
genre_Comedy            False
genre_Family            False
genre_Science Fiction   False
genre_Romance           False
genre_TV Movie          False
genre_History           False
genre_Foreign           False
genre_Adventure         False
genre_Western           False
dtype: bool
```

2. **One Hot Encoding**: - The categorical feature genre is one-hot encoded and thus, a new feature is created for each category.

```
# Preprocess the `genres` column
def extract_genres(genres_str):
    try:
        return [genre['name'] for genre in ast.literal_eval(genres_str)]
    except (ValueError, TypeError):
        return []

movies['genres_list'] = movies['genres'].apply(extract_genres)
all_genres = set([genre for genres in movies['genres_list'] for genre in genres])
for genre in all_genres:
    movies[f'genre_{genre}'] = movies['genres_list'].apply(lambda x: 1 if genre in x else 0)

movies = movies.drop(columns=['genres', 'genres_list'])
```

3. **Normalization**: - Min-max scaler is used to normalize the feature values.

```
# Normalize numerical features
columns_to_normalize = ['budget', 'popularity', 'runtime', 'vote_count','revenue','vote_average']
scaler = MinMaxScaler()
movies[columns_to_normalize] = scaler.fit_transform(movies[columns_to_normalize])
```

Train Test Split

```
X = movies[['budget', 'popularity', 'runtime', 'vote_count']]
y = movies[['revenue', 'vote_average']]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Model Building

The following machine learning models were tested:

- Linear Regression
- Decision Tree Regressor
- Random Forest Regressor

Linear Regression

```
linear_model = MultiOutputRegressor(LinearRegression())
linear_model.fit(X_train, y_train)
y_pred = linear_model.predict(X_test)
```

Random Forest Regressor

```
rf_regressor = RandomForestRegressor(n_estimators=100,random_state=42)
rf_regressor.fit(X_train, y_train)
rf_predictions = rf_regressor.predict(X_test)
```

Decision Tree Regressor

```
dt_regressor = DecisionTreeRegressor(random_state=42)
dt_regressor.fit(X_train, y_train)
dt_predictions = dt_regressor.predict(X_test)
```

Hyperparameter Tuning

Optimized Random Forest Regressor

```
rf = RandomForestRegressor(random_state=42)
rf_param_grid = {
    'n_estimators': [50, 100, 200, 300],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
rf_random_search = RandomizedSearchCV(rf, rf_param_grid, n_iter=20, cv=3, scoring='neg_mean_squared_error', n_jobs=-1, random_state=42)
rf_random_search.fit(X_train, y_train)

best_rf = rf_random_search.best_estimator_
rf_predictions = best_rf.predict(X_test)
```

Optimized Decision Tree Regressor

```
dt = DecisionTreeRegressor(random_state=42)
dt_param_grid = {
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
dt_random_search = RandomizedSearchCV(dt, dt_param_grid, n_iter=20, cv=3, scoring='neg_mean_squared_error', n_jobs=-1, random_state=42)
dt_random_search.fit(X_train, y_train)

best_dt = dt_random_search.best_estimator_
dt_predictions = best_dt.predict(X_test)
```

Results

The performance of each model was evaluated using metrics such as Mean Squared Error (MSE), R-squared Score (R_2) and Mean Absolute Error (MAE). Here's a summary: -

	Mean Squared Error		Mean Absolute Error
Linear Regression	0.0055	0.5216	0.0410
Random Forest Regressor	0.0033	0.6696	0.0336
Decision Tree Regressor	0.0062	0.4009	0.0474
Random Forest Regressor (Hyperparameter Tuned)	0.0030	0.7185	0.0327
Decision Tree Regressor (Hyperparameter Tuned)	0.0039	0.6347	0.0372

Conclusion

This project successfully applied machine learning techniques to predict movie revenue and audience ratings using the TMDB 5000 Movie Dataset. Through models like Linear Regression, Decision Tree Regressor, and Random Forest Regressor, we explored key factors such as budget, popularity, runtime, and vote count, which significantly influence a movie's success.

The **Random Forest Regressor** emerged as the best-performing model, delivering the **lowest error and the highest R^2 score**. This highlights the strength of ensemble methods in capturing complex feature interactions and providing robust predictions.

These results demonstrate the capability of machine learning to provide actionable insights for stakeholders. By leveraging these predictions, production companies can optimize resource allocation, investors can minimize financial risks, and distributors can enhance their marketing strategies. This framework also has the potential to be adapted for other domains like TV shows and streaming content, further broadening its applicability in the entertainment industry.