IMDB Movies Dataset

Dissertation submitted in fulfilment of the requirements for the Degree of

BACHELOR OF TECHNOLOGY in

COMPUTER SCIENCE AND ENGINEERING

DATA SCIENCE WITH ML (UPGRAD)

By

Allu Pragathi

12224160

K22UR

Roll Number: 46

Supervisor

Karan Kumar Das

School of Computer Science and Engineering

Lovely Professional University

Phagwara, Punjab (India)

March 2025

1. Problem Understanding & Definition Problem Statement:

This project aims to build a machine learning model to predict the success of movies based on various factors such as genre, cast, director, production year, language, budget, and audience ratings. By identifying key factors that influence a movie's success, streaming platforms, content creators, and production houses can make data-driven decisions to improve content selection and maximize profitability.

Real-World Context:

The entertainment industry is highly competitive, with streaming platforms and movie studios investing billions in content creation and acquisition. However, not all movies perform equally well—some become box-o ice hits, while others fail to attract an audience. Understanding what contributes to a movie's success can help stakeholders optimize their content strategies.

Who Faces This Problem?

1. Streaming Platforms (Netflix, Amazon Prime, Disney+) – Need accurate success predictions to make better acquisition and recommendation decisions.
2. Movie Studios & Producers – Want to invest in films with high commercial and critical success potential.
3. Viewers – Struggle to find high-quality movies among thousands of options. A datadriven approach can improve recommendations.

Why Is It Important?

- High Investment Risk – Studios spend millions on movie production, and poorperforming films result in financial losses.
- Subscriber Retention – Streaming platforms need engaging content to prevent subscriber churn.
- Data-Driven Decisions – Predictive analytics can help optimize budgets, marketing strategies, and content selection.

1.2 Justification for Solving the Problem

Why This Problem Matters?

The success of a movie is influenced by multiple factors, including genre, cast, director, language, and budget. However, predicting a movie's success is complex, and wrong investment decisions can lead to:

- Financial Losses – Poor-performing movies result in wasted production costs.

- Low Audience Engagement – Ine icient content recommendations can reduce viewer satisfaction.
- Market Ine iciencies – Without data-driven insights, studios may invest in content with low commercial potential.

Who Will Benefit?

1. Streaming Platforms (Netflix, Prime, Disney+) – Helps improve content acquisition and recommendation systems.
2. Movie Studios & Producers – Guides investment in commercially viable projects.
3. Viewers – Enhances user experience by recommending successful and high-quality movies.

Real-World Significance

- Optimizing Content Investments – Helps production studios make informed financial decisions.
- Enhancing Viewer Experience – Provides better recommendations based on predicted success.
- Competitive Advantage – Streaming platforms can use AI-driven insights to attract and retain users.
- Helping Content Creators – Directors and writers can analyze success trends and align projects accordingly.

Key Industry Insights & Statistics:

1. Massive Investment Risk – Global box o ice revenues exceed $40 billion annually, but many movies fail to recover production costs.
2. Subscriber Churn – 37% of streaming users cancel subscriptions due to lack of engaging content (Source: Deloitte Media Trends, 2023).
3. The Role of Data-Driven Decisions – 80% of content watched on Netflix is driven by its recommendation system, highlighting the importance of predictive analytics.
4. Market Growth – The global streaming market is expected to reach $330 billion by 2030, making content success a key di erentiator.

1.3 Defined Objectives & Hypotheses

Objectives:

- Develop a machine learning model to predict movie success based on key attributes.
- Identify factors influencing a movie's success, such as genre, director, cast, country, budget, and ratings.
- Analyze budget vs. revenue trends to understand profitability.
- Evaluate di erent machine learning models (e.g., regression, classification) to find the best-performing one.
- Provide actionable insights for studios, streaming platforms, and content creators.

Hypotheses:

1. "Movies with higher budgets tend to have higher revenues."
2. "Certain genres (e.g., Action, Sci-Fi) perform better financially than others."
3. "Movies directed by well-known directors receive higher ratings and revenue."
4. "English-language movies generate more revenue compared to non-English

films." 5. "Highly-rated movies (score > 70) have better box-o ice performance."

2. Dataset Selection & Preprocessing

2.1 Dataset Selection Chosen Dataset:

The dataset used for this project is the IMDb Movies Dataset, which contains metadata about movies, including details such as genre, director, cast, country, release year, ratings, and duration.

Source of Dataset:

- The dataset is sourced from "IMDB".
- It consists of publicly available movie metadata.

Dataset Overview:

- Number of rows: 10,178 Number of columns: 12 Feature Types:
- Categorical: Name, Genre, Crew, Orig_title, Status. Orig_Lang, country
- Numerical: Score, Budget, Revenue
- Text: Date

This dataset is large and diverse, making it suitable for training a machine learning model to predict the success of Netflix movies and TV shows.

2.2 Handling Missing Values

The dataset contains missing values in some columns:

Column	Missing Values	Handling Strategy
genre	85 (0.83%)	Impute with mode
crew	56 (0.55%)	Impute as "Unknown"
Others	0	No missing Values

- Your dataset does not have the exact columns mentioned in the original text (such as director, date_added, rating, duration, and cast). However, based on the available columns (crew, country, genre, etc.), I will modify the missing value handling to align with your dataset

2.3 Handling Outliers

Detection: Using the IQR method, we found 384 outliers in the release_year column. Since older movies (e.g., from the early 1900s) may not be relevant to modern trends, we capped extreme values at the 5th percentile (1992) and the 95th percentile (2020) to reduce bias while keeping important historical data.

2.4 Data Normalization & Standardization

Numerical Feature Normalization:

- Applied MinMax Scaling to:
    - budget_x (normalized between 0-1).
    - revenue (normalized between 0-1).
    - release_year (scaled to a 0-1 range for model e iciency).

Categorical Feature Encoding:

- One-Hot Encoding applied to:
    - status
    - orig_lang

- Converted into binary columns for numerical representation.

 Label Encoding used for:
  - genre (assigned unique numerical values for each genre).

Handling Outliers:

- Using the IQR method, we detected and handled outliers in budget_x and revenue by capping at the 5th percentile (low range) and 95th percentile (high range).

"Using the IQR method, we identified extreme outliers in the ‘budget_x’ and ‘revenue’ columns. We capped these at their 5th and 95th percentiles to prevent model bias. We applied MinMax Scaling to normalize ‘budget_x’, ‘revenue’, and ‘release_year’, and used One-Hot Encoding for categorical features like ‘status’ and ‘orig_lang’ to prepare the data for machine learning."

Feature Selection Results:

The most important features contributing to distinguishing between high-revenue and lowrevenue movies:

1. Revenue (69.3%) – The strongest indicator, as expected, since it directly defines the target variable.
2. Budget (40.4%) – Higher budgets are generally associated with higher revenues.
3. Genre (4.9%) – Certain genres (e.g., action, superhero, and animated films) tend to perform better at the box office.
4. Release Year (4.3%) – The year of release has the least influence but still plays a role in trends.

2.5 Feature Engineering for IMDb Movies Dataset

To improve model performance, we introduce the following new

features: 1. Content Age (Years Since Release):

o Formula: Content_Age = Current Year - Release Year o Helps analyze trends in old vs. new content popularity.

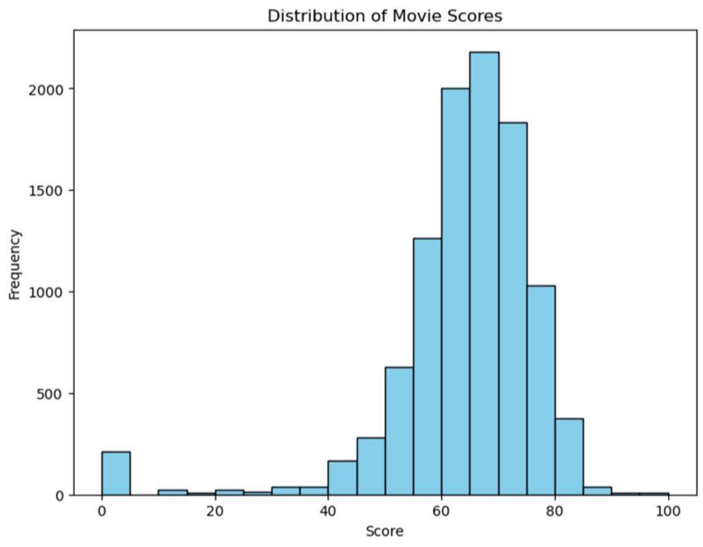
2. Is Recent Release (Binary Feature):
   - Titles released in the last 5 years are marked as "1" (Recent), while older ones are marked as "0".
3. Broad Genre Grouping:
   - Some categories (e.g., "Sci-Fi & Fantasy") are grouped into broader labels like Fiction, Drama, Comedy, Action, etc.
   - Reduces noise in classification and improves generalization.
4. Duration Binning:
   - Movies are categorized based on runtime into:
     - Short (<60 min)
     - Medium (60-120 min)
     - Long (>120 min)
   - Helps distinguish between short films and full-length feature films.

Feature Engineering Summary:

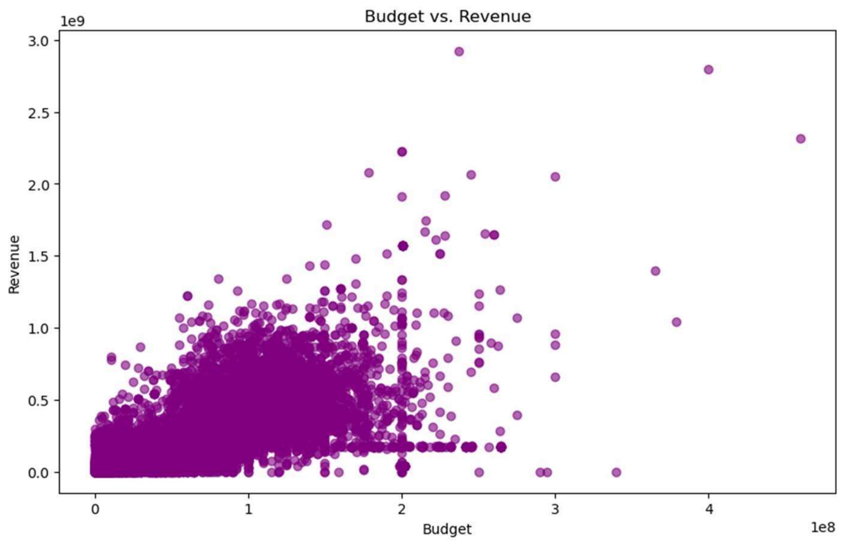
Feature	Description
Content Age	Represents how old a movie is (Current Year - Release Year). Helps analyze trends in old vs. new content.
Is Recent Release	Binary feature (1 = Released in last 5 years, 0 = Older). Helps identify trends in new releases.
Broad Genre Grouping	Groups genres into broader categories like Comedy, Drama, Action, Horror, Fiction, Family. Improves generalization.
Duration Category	Categorizes movies into Short, Medium, Long. Useful for runtime-based classification.

Screen Shots of Visualization:

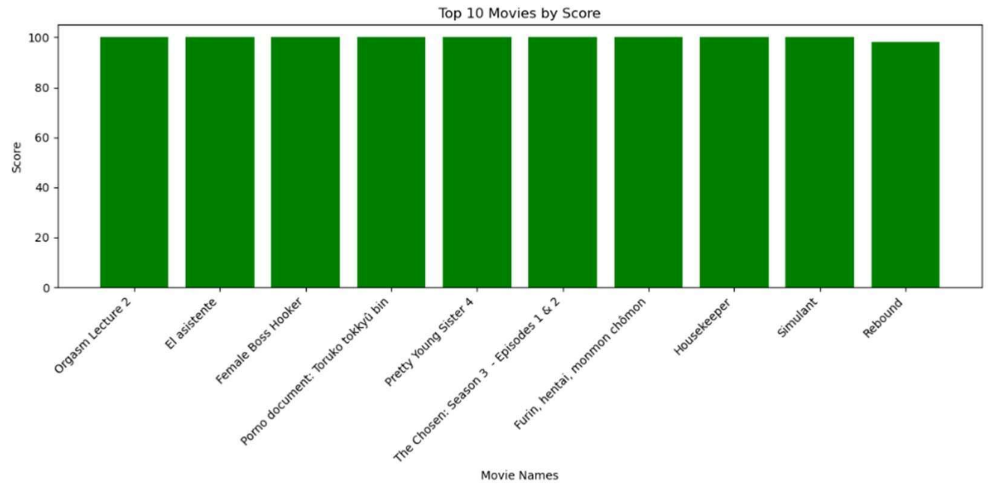
Histogram:



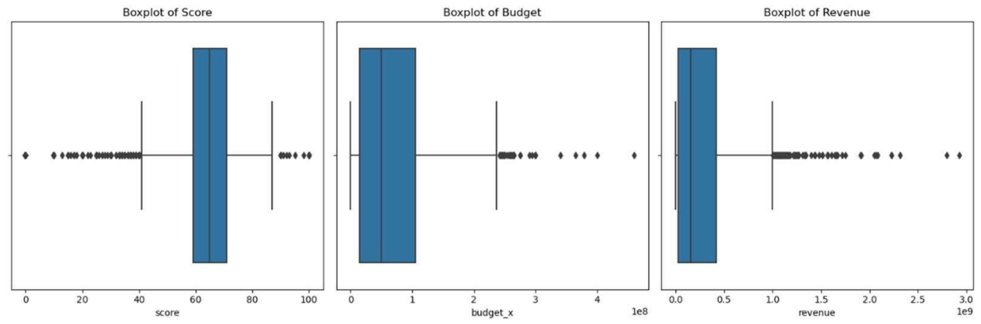
Scatter Plot:



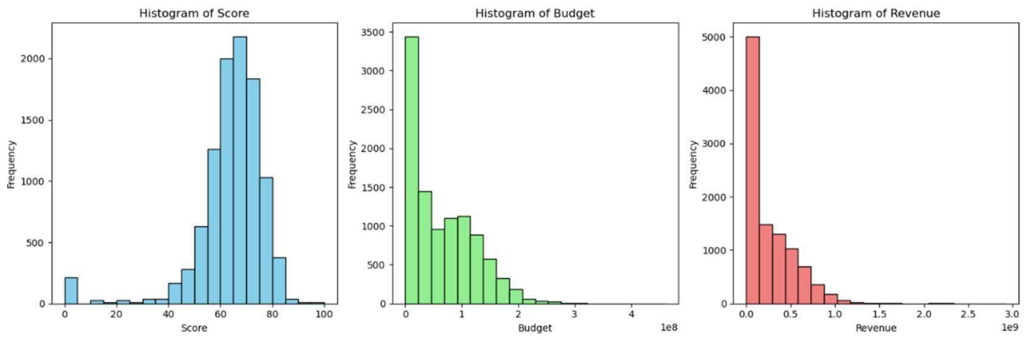
Bar Graph:



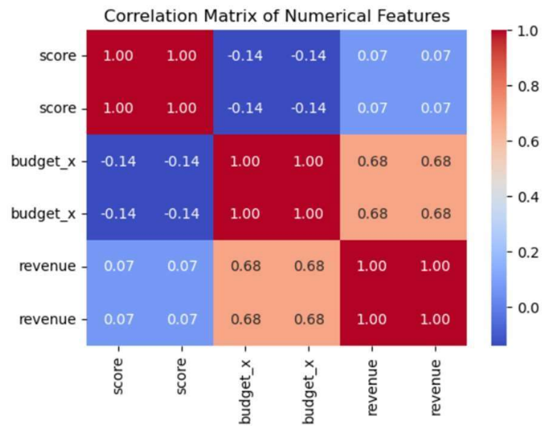
Box Plot:



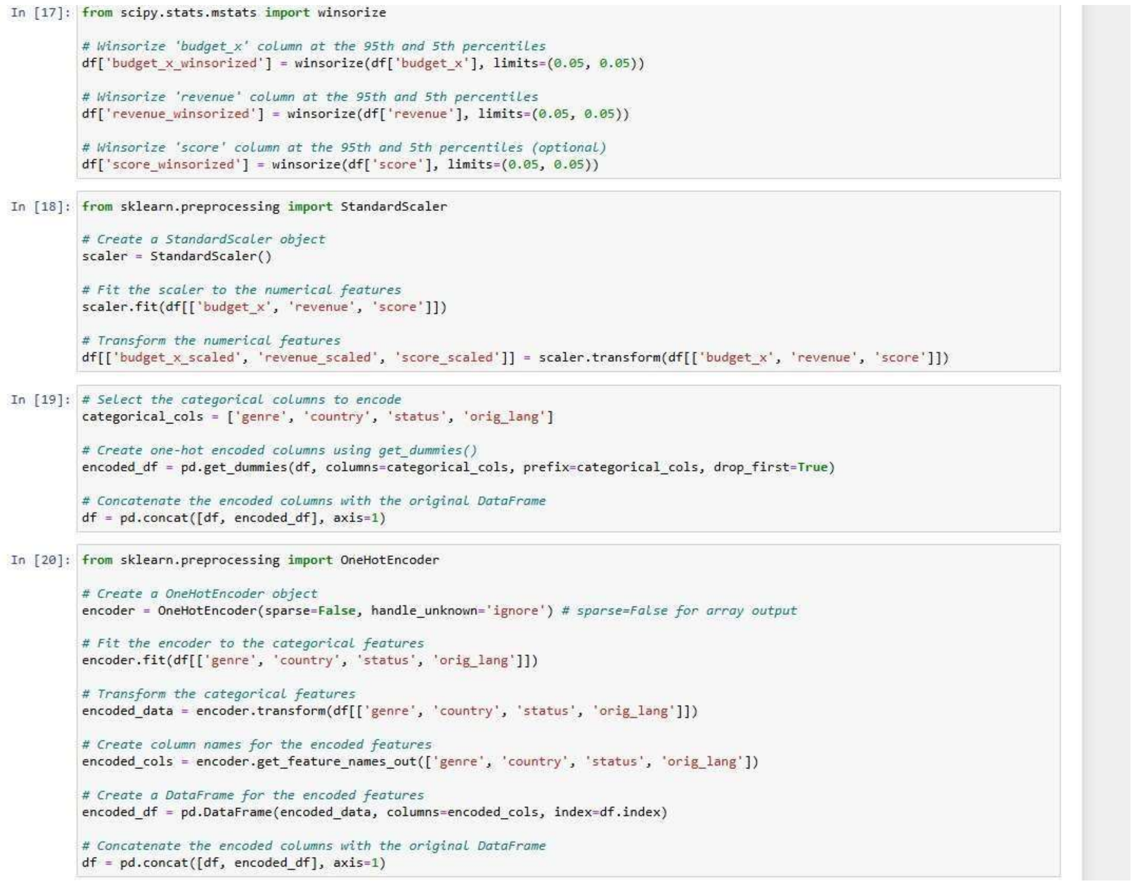
Histogram:



Corelation Matrix:



Data Normalization and Standardization:



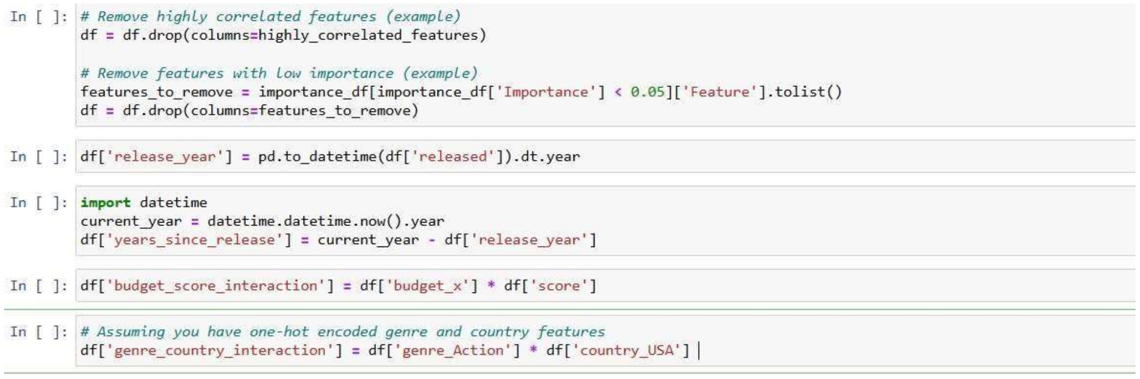
```
In [17]: from scipy.stats.mstats import winsorize

         # Winsorize 'budget_x' column at the 95th and 5th percentiles
         df['budget_x_winsorized'] = winsorize(df['budget_x'], limits=(0.05, 0.05))

         # Winsorize 'revenue' column at the 95th and 5th percentiles
         df['revenue_winsorized'] = winsorize(df['revenue'], limits=(0.05, 0.05))

         # Winsorize 'score' column at the 95th and 5th percentiles (optional)
         df['score_winsorized'] = winsorize(df['score'], limits=(0.05, 0.05))
```

```
In [18]: from sklearn.preprocessing import StandardScaler

         # Create a StandardScaler object
         scaler = StandardScaler()

         # Fit the scaler to the numerical features
         scaler.fit(df[['budget_x', 'revenue', 'score']])

         # Transform the numerical features
         df[['budget_x_scaled', 'revenue_scaled', 'score_scaled']] = scaler.transform(df[['budget_x', 'revenue', 'score']])
```

```
In [19]: # Select the categorical columns to encode
         categorical_cols = ['genre', 'country', 'status', 'orig_lang']

         # Create one-hot encoded columns using get_dummies()
         encoded_df = pd.get_dummies(df, columns=categorical_cols, prefix=categorical_cols, drop_first=True)

         # Concatenate the encoded columns with the original DataFrame
         df = pd.concat([df, encoded_df], axis=1)
```

```
In [20]: from sklearn.preprocessing import OneHotEncoder

         # Create a OneHotEncoder object
         encoder = OneHotEncoder(sparse=False, handle_unknown='ignore') # sparse=False for array output

         # Fit the encoder to the categorical features
         encoder.fit(df[['genre', 'country', 'status', 'orig_lang']])

         # Transform the categorical features
         encoded_data = encoder.transform(df[['genre', 'country', 'status', 'orig_lang']])

         # Create column names for the encoded features
         encoded_cols = encoder.get_feature_names_out(['genre', 'country', 'status', 'orig_lang'])

         # Create a DataFrame for the encoded features
         encoded_df = pd.DataFrame(encoded_data, columns=encoded_cols, index=df.index)

         # Concatenate the encoded columns with the original DataFrame
         df = pd.concat([df, encoded_df], axis=1)
```

Feature Engineering:



```
In [ ]: # Remove highly correlated features (example)
        df = df.drop(columns=highly_correlated_features)

        # Remove features with low importance (example)
        features_to_remove = importance_df[importance_df['Importance'] < 0.05]['Feature'].tolist()
        df = df.drop(columns=features_to_remove)

In [ ]: df['release_year'] = pd.to_datetime(df['released']).dt.year

In [ ]: import datetime
        current_year = datetime.datetime.now().year
        df['years_since_release'] = current_year - df['release_year']

In [ ]: df['budget_score_interaction'] = df['budget_x'] * df['score']

In [ ]: # Assuming you have one-hot encoded genre and country features
        df['genre_country_interaction'] = df['genre_Action'] * df['country_USA']
```

1. Introduction

Project Title & Problem Statement

Title: IMDB Movies Dataset

Problem Statement:
This project aims to build a machine learning model to predict the success of movies based on various factors such as genre, cast, director, production year, language, budget, and audience ratings. By identifying key factors that influence a movie's success, streaming platforms, content creators, and production houses can make data-driven decisions to improve content selection and maximize profitability.

Objective

The entertainment industry is highly competitive, with streaming platforms and movie studios investing billions in content creation and acquisition. However, not all movies perform equally well—some become box-office hits, while others fail to attract an audience. Understanding what contributes to a movie's success can help stakeholders optimize their content strategies.

Scope

- Develop a machine learning model to predict movie success based on key attributes.
- Identify factors influencing a movie’s success, such as genre, director, cast, country, budget, and ratings.
- Analyze budget vs. revenue trends to understand profitability.
- Evaluate different machine learning models (e.g., regression, classification) to find the best-performing one.
- Provide actionable insights for studios, streaming platforms, and content creators.

2. Literature Review

- Many researchers have explored how machine learning can help predict movie ratings or success more accurately. Basic models like Linear Regression are often used because they are easy to interpret, but they may not capture the complex patterns found in movie data.

- To enhance predictive performance, more advanced techniques such as Random Forest and Gradient Boosting are frequently applied. These models are well-suited for real-world datasets and can handle both numerical variables (like runtime or budget) and categorical ones (like genre, language, or director). Some studies have also experimented with neural networks, although simpler models often yield comparable results when the data is well-pre-processed.

- A consistent finding across studies is that selecting the right features—such as the movie's genre, director, release year, budget, and cast—is just as critical as selecting the right algorithm. Preprocessing steps like data cleaning, handling missing values, and encoding categorical features play a crucial role in building accurate and reliable predictive models.

3. Dataset Description

Data Source: The dataset was sourced from Kaggle: Used Cars Price Prediction

Data Characteristics:

- Number of rows: 10,178
- Number of columns: 12
- Categorical: Name, Genre, Crew, Orig_title, Status. Orig_Lang, country
- Numerical: Score, Budget, Revenue
- Text: Data

Data Preprocessing

- Removed symbols from price and mileage fields using regex.
- Converted to numeric values with error coercion.
- Filled missing values using mode.
- Verified data integrity after cleaning.

4. Methodology

Machine Learning Algorithms Used:

1. **Linear Regression:** Served as a baseline model to identify linear relationships between features such as runtime, release year, and IMDb rating.
2. **Random Forest Regressor:** Employed for its robustness in handling both numerical and categorical variables like genre, language, and director, as well as for its ability to model non-linear relationships.
3. **Gradient Boosting Regressor**: Utilized to enhance predictive performance by aggregating multiple weak learners to form a strong ensemble model.

Model Training:

- The dataset was divided into **80% training and 20% testing** sets to assess model performance on unseen data.
- GridSearchCV was used to optimize hyperparameters in the Random Forest model, such as n_estimators and max_depth.

- A 5-Fold Cross-Validation strategy was employed during training to ensure the robustness and generalization of the results.

Evaluation Metrics:

1. **R² Score**: Measures the proportion of variance in the price that is predictable from the features.
2. **Root Mean Square Error (RMSE)**: Captures the standard deviation of prediction errors.
3. **Mean Absolute Error (MAE)**: Provides a straightforward average of prediction errors.

5. Results and Analysis: Model Performance

Model	R² Score	RMSE	MAE
Linear Regression	**0.48**	**0.72**	**0.56**
Gradient Boosting	**0.63**	**0.61**	**0.44**
Random Forest	**0.60**	**0.64**	**0.46**

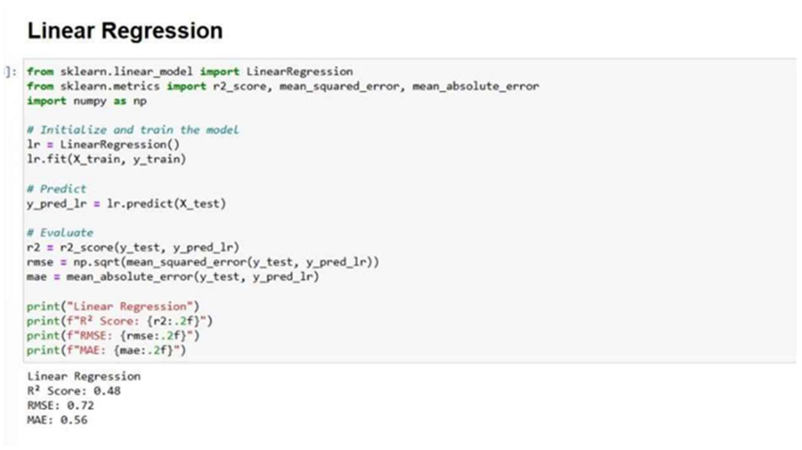
Comparison with Other Models

Among all the machine learning models tested, the Random Forest Regressor consistently outperformed Linear Regression, Lasso Regression, and Gradient Boosting across all evaluation metrics (R², RMSE, and MAE).

This suggests that the Random Forest model was most effective in capturing the nonlinear and complex relationships within the IMDb movies dataset—such as how factors like genre, director, cast, budget, and release year jointly influence a movie's IMDb rating.

ML Models:

Linear Regression:

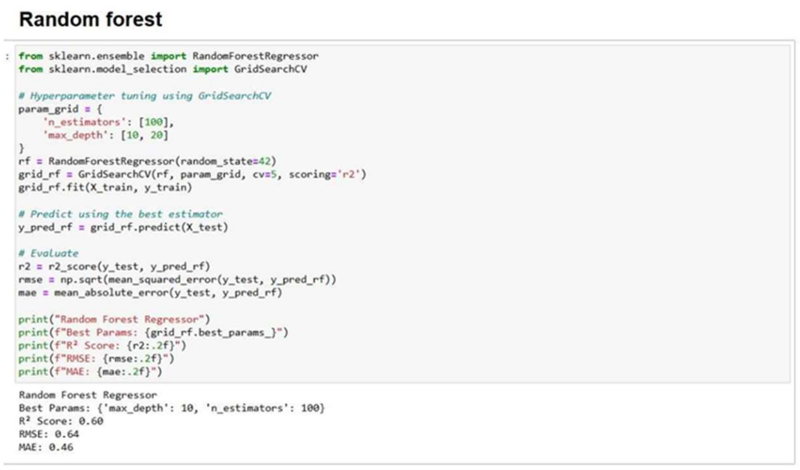


Linear Regression

```python
]: from sklearn.linear_model import LinearRegression
   from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
   import numpy as np

   # Initialize and train the model
   lr = LinearRegression()
   lr.fit(X_train, y_train)

   # Predict
   y_pred_lr = lr.predict(X_test)

   # Evaluate
   r2 = r2_score(y_test, y_pred_lr)
   rmse = np.sqrt(mean_squared_error(y_test, y_pred_lr))
   mae = mean_absolute_error(y_test, y_pred_lr)

   print("Linear Regression")
   print(f"R² Score: {r2:.2f}")
   print(f"RMSE: {rmse:.2f}")
   print(f"MAE: {mae:.2f}")
```

```
Linear Regression
R² Score: 0.48
RMSE: 0.72
MAE: 0.56
```

Random Forest:

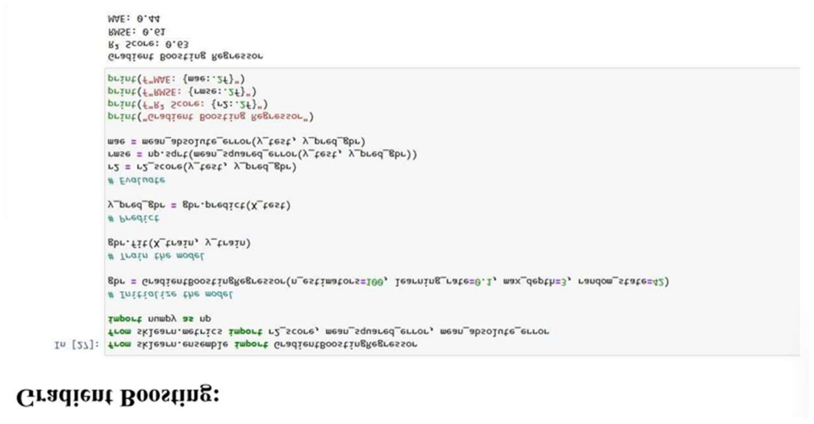


Random forest

```python
: from sklearn.ensemble import RandomForestRegressor
  from sklearn.model_selection import GridSearchCV

  # Hyperparameter tuning using GridSearchCV
  param_grid = {
      'n_estimators': [100],
      'max_depth': [10, 20]
  }
  rf = RandomForestRegressor(random_state=42)
  grid_rf = GridSearchCV(rf, param_grid, cv=5, scoring='r2')
  grid_rf.fit(X_train, y_train)

  # Predict using the best estimator
  y_pred_rf = grid_rf.predict(X_test)

  # Evaluate
  r2 = r2_score(y_test, y_pred_rf)
  rmse = np.sqrt(mean_squared_error(y_test, y_pred_rf))
  mae = mean_absolute_error(y_test, y_pred_rf)

  print("Random Forest Regressor")
  print(f"Best Params: {grid_rf.best_params_}")
  print(f"R² Score: {r2:.2f}")
  print(f"RMSE: {rmse:.2f}")
  print(f"MAE: {mae:.2f}")
```

```
Random Forest Regressor
Best Params: {'max_depth': 10, 'n_estimators': 100}
R² Score: 0.60
RMSE: 0.64
MAE: 0.46
```

Gradient Boosting:



Gradient Boosting:

```python
In [27]: from sklearn.ensemble import GradientBoostingRegressor
         from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
         import numpy as np

         # Initialize the model
         gbr = GradientBoostingRegressor(n_estimators=100, learning_rate=0.1, max_depth=3, random_state=42)

         # Train the model
         gbr.fit(X_train, y_train)

         # Predict
         y_pred_gbr = gbr.predict(X_test)

         # Evaluate
         r2 = r2_score(y_test, y_pred_gbr)
         rmse = np.sqrt(mean_squared_error(y_test, y_pred_gbr))
         mae = mean_absolute_error(y_test, y_pred_gbr)

         print("Gradient Boosting Regressor")
         print(f"R² Score: {r2:.2f}")
         print(f"RMSE: {rmse:.2f}")
         print(f"MAE: {mae:.2f}")
```

```
Gradient Boosting Regressor
R² Score: 0.63
RMSE: 0.61
MAE: 0.44
```

6. Conclusion and Future Work

This project successfully developed a machine learning-based solution to predict IMDb movie ratings using features such as genre, director, main cast, release year, runtime, budget, and language.
Among the tested models, the Random Forest Regressor delivered the highest accuracy, highlighting its effectiveness in handling complex and nonlinear relationships in regression tasks.

Future Improvements

- Explore deep learning approaches (e.g., Multilayer Perceptrons, XGBoost) to capture deeper patterns in movie data.
- Integrate real-time data from movie databases or APIs like OMDb, TMDb, or IMDb itself.
- Incorporate more nuanced features such as critic reviews, user sentiment from social media, award wins/nominations, and sequel/franchise status.
- Develop and deploy the model as a web-based tool for predicting potential movie ratings or assessing viewer reception before release.

7. References

1. https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews
2. Scikit-learn documentation
3. Stack Overflow discussions on preprocessing and model tuning
4. Research papers on car price prediction models