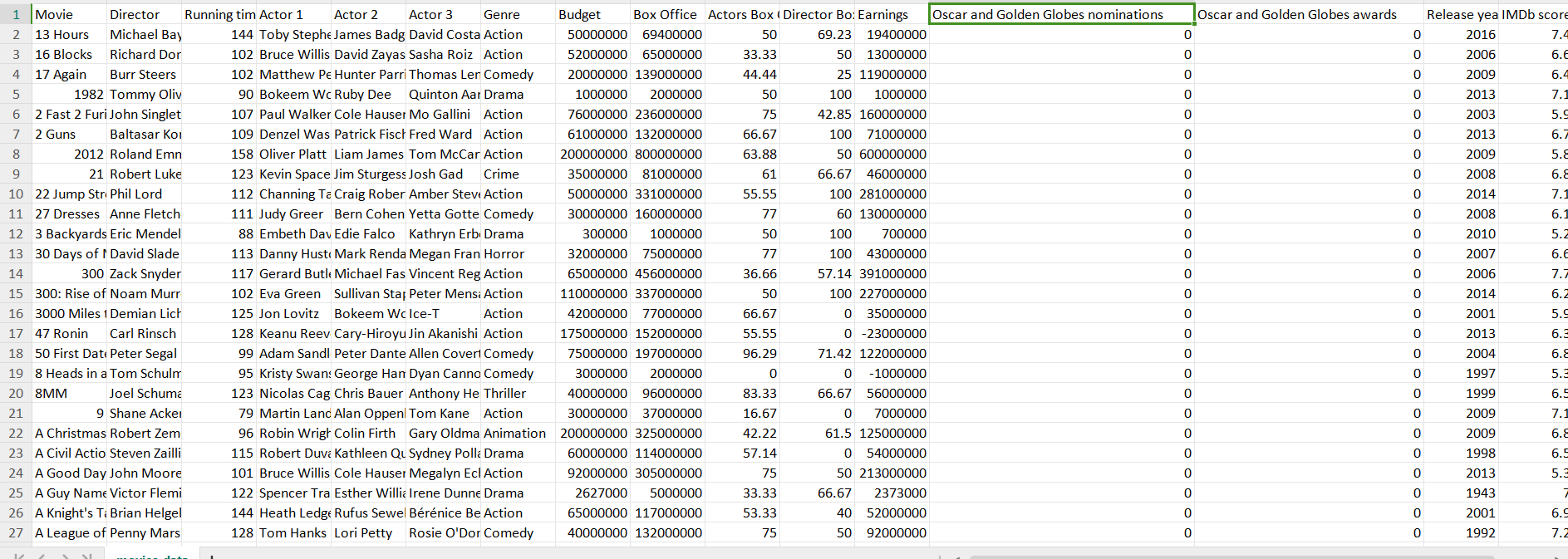
**Project Report: Movie Nominations and IMDb Score Prediction**

Link to the dataset : <https://www.kaggle.com/datasets/delfinaoliva/movies>

Snippet of the dataset :



## **1. Main Objective of the Analysis**

The main objective of this project is to use machine learning techniques to predict two outcomes:

1. Whether a movie is **nominated** for an award (Oscar or Golden Globes).
2. Whether a movie has a **high IMDb score** (≥ 7.0).

For both objectives, we aim to develop models that will provide insightful predictions based on movie attributes such as genre, budget, director, and box office earnings. The analysis seeks to help stakeholders (such as producers and marketers) predict the likelihood of movie nominations and high IMDb ratings, potentially informing marketing strategies and production decisions.

## **2. Data Description**

The dataset contains 3,974 movie records with 16 attributes each. The relevant features include:

* **Movie Information**: Director, Actors, Running Time, Genre.
* **Financial Information**: Budget, Box Office Earnings, Earnings as percentages for the actors and director.
* **Award Information**: Number of nominations and awards for Oscars and Golden Globes.
* **IMDb Score**: Numeric score representing the overall rating on IMDb.

The target variables are:

* **Nominated**: A binary classification of whether a movie was nominated for an award (1 = Nominated, 0 = Not Nominated).
* **High IMDb Score**: A binary classification based on whether a movie's IMDb score is ≥ 7.0 (1 = High Score, 0 = Low Score).

### **Summary of Dataset:**

* **Number of Records**: 3,974
* **Number of Features**: 16 (including both categorical and numerical attributes)

## **3. Data Exploration and Cleaning**

### **Data Exploration:**

* The dataset contains both categorical (e.g., Genre, Director) and numerical (e.g., Budget, IMDb score) features.
* Some movies have missing values for awards, which were handled by either removing unnecessary columns or focusing on available data.

### **Data Cleaning Steps:**

1. **Feature Selection**: Removed unnecessary columns such as actor names, awards, and IMDb scores for predictive modeling.
2. **One-Hot Encoding**: Converted categorical variables (e.g., Genre, Director) into dummy variables to ensure compatibility with machine learning algorithms.
3. **Feature Scaling**: Applied **StandardScaler** to normalize the numerical features (e.g., Budget, Box Office).
4. **Binary Target Creation**: Created binary target columns:
   * **Nominated**: 1 if the movie had nominations, 0 otherwise.
   * **High IMDb Score**: 1 if the IMDb score was ≥ 7.0, 0 otherwise.

## **4. Model Training**

We trained three different classifiers to predict both movie nominations and IMDb scores:

1. **Logistic Regression**: A linear model often used for binary classification.
2. **Decision Tree**: A tree-based model that recursively splits data based on feature importance.
3. **Random Forest**: An ensemble model that builds multiple decision trees and aggregates their outputs for a more robust prediction.

### **Data Split:**

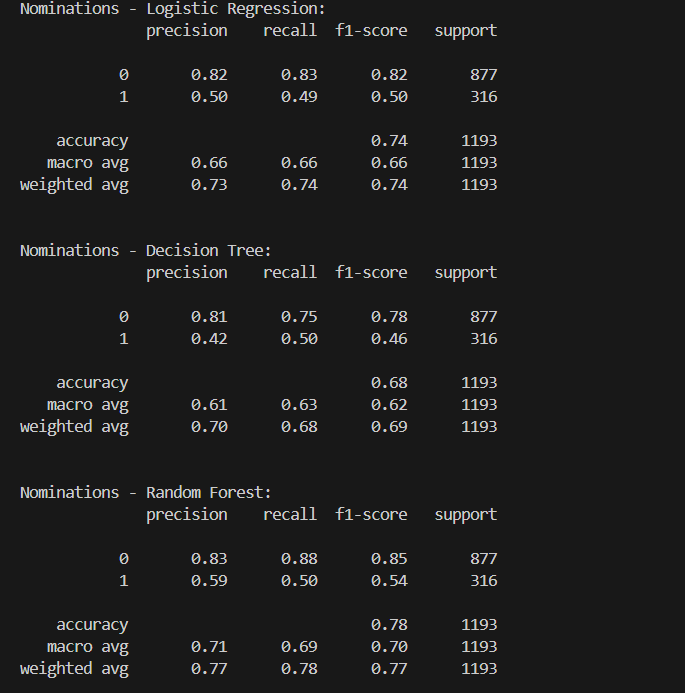
* **Training Set**: 70% of the data (used to train the models).
* **Test Set**: 30% of the data (used to evaluate the models).

### **Summary of Models:**

All models were trained using the same data split and evaluated using metrics such as accuracy, precision, recall, and F1-score.

## **5. Model Comparison and Selection**

### **Nominations Prediction**



The Random Forest model performed the best for predicting nominations, with the highest accuracy and balanced performance between precision and recall. The classification reports for each model are shown below:

#### **Logistic Regression:**

* **Accuracy**: 74%
* **Precision (Class 1)**: 50%
* **Recall (Class 1)**: 49%
* **F1-Score (Class 1)**: 50%

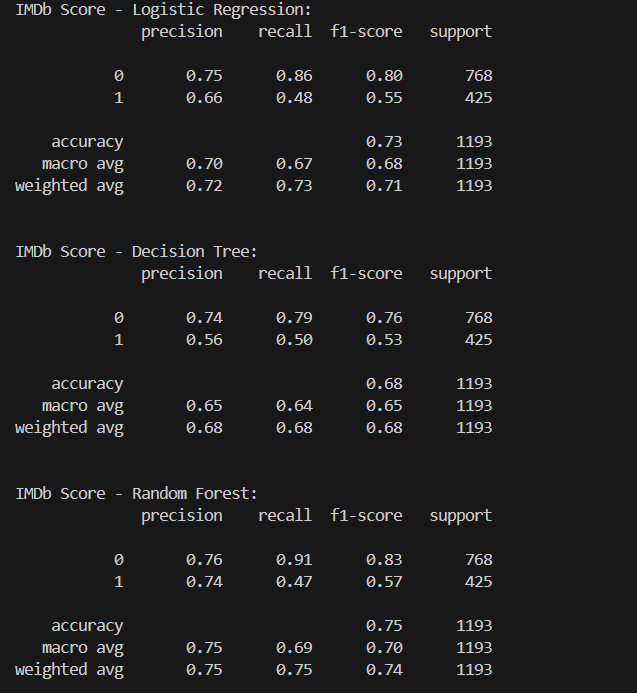
#### **Decision Tree:**

* **Accuracy**: 68%
* **Precision (Class 1)**: 42%
* **Recall (Class 1)**: 50%
* **F1-Score (Class 1)**: 46%

#### **Random Forest:**

* **Accuracy**: 78%
* **Precision (Class 1)**: 59%
* **Recall (Class 1)**: 50%
* **F1-Score (Class 1)**: 54%

### **IMDb Score Prediction**



Similarly, the Random Forest model performed the best in predicting high IMDb scores, showing better overall performance compared to Logistic Regression and Decision Trees.

#### **Logistic Regression:**

* **Accuracy**: 73%
* **Precision (Class 1)**: 66%
* **Recall (Class 1)**: 48%
* **F1-Score (Class 1)**: 55%

#### **Decision Tree:**

* **Accuracy**: 68%
* **Precision (Class 1)**: 56%
* **Recall (Class 1)**: 50%
* **F1-Score (Class 1)**: 53%

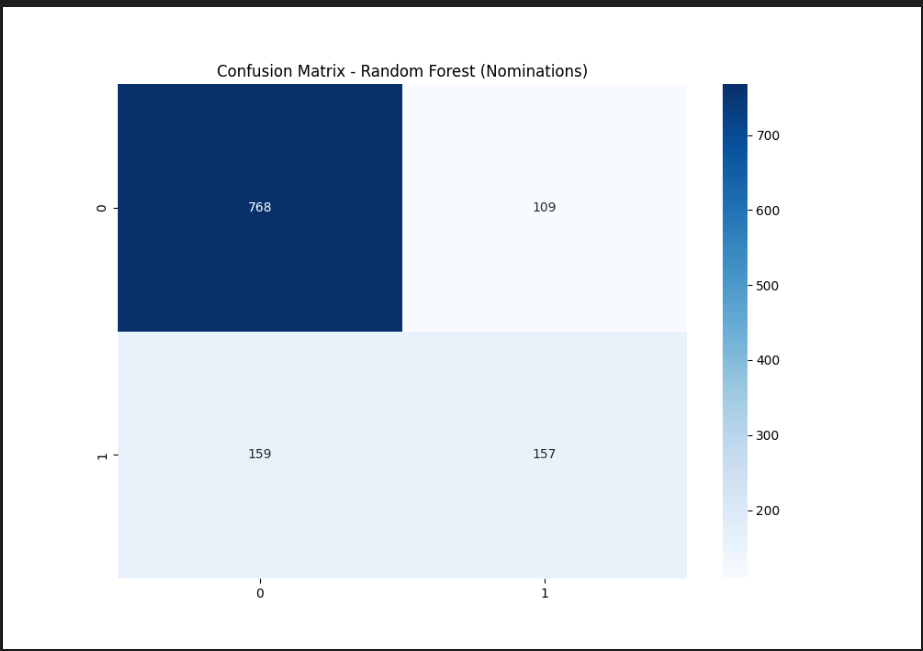
#### **Random Forest:**

* **Accuracy**: 75%
* **Precision (Class 1)**: 74%
* **Recall (Class 1)**: 47%
* **F1-Score (Class 1)**: 57%

### **Confusion Matrices:**

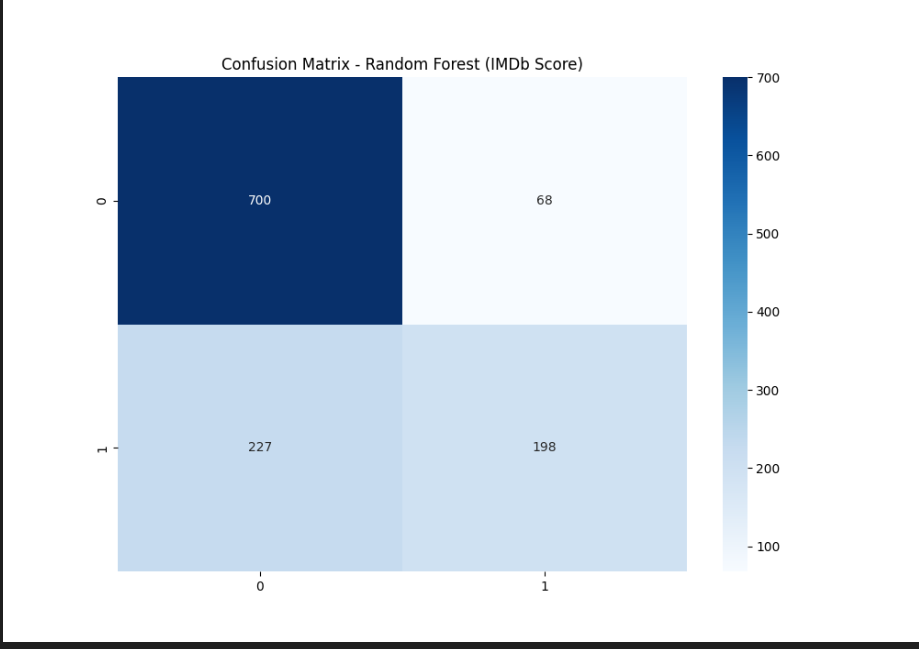
#### **Components of the Confusion Matrix:**

* **True Positives (TP)**: Correctly predicted **positive class** (bottom-right quadrant, in this case, movies with high IMDb scores).
* **True Negatives (TN)**: Correctly predicted **negative class** (top-left quadrant, in this case, movies with low IMDb scores).
* **False Positives (FP)**: Incorrectly predicted **positive class** (top-right quadrant, in this case, movies predicted to have high IMDb scores but actually have low scores).
* **False Negatives (FN)**: Incorrectly predicted **negative class** (bottom-left quadrant, in this case, movies predicted to have low IMDb scores but actually have high scores).



### **Interpretation:**

1. **True Negatives (TN = 700)**:
   * The model correctly predicted **700** movies as having low IMDb scores when they actually do have low scores.
2. **False Positives (FP = 68)**:
   * The model incorrectly predicted **68** movies as having high IMDb scores when they actually have low scores. These are false alarms.
3. **False Negatives (FN = 227)**:
   * The model incorrectly predicted **227** movies as having low IMDb scores when they actually have high scores. These are missed opportunities where the model failed to identify high-rated movies.
4. **True Positives (TP = 198)**:
   * The model correctly predicted **198** movies as having high IMDb scores when they actually have high scores.



**True Negatives (TN)**: 768

* The model correctly predicted **768** movies as **not nominated** when they were actually **not nominated**.

**False Positives (FP)**: 109

* The model incorrectly predicted **109** movies as **nominated**, but these movies were actually **not nominated** (false alarm).

**False Negatives (FN)**: 159

* The model predicted **159** movies as **not nominated**, but they were actually **nominated** (missed opportunity).

**True Positives (TP)**: 157

* The model correctly predicted **157** movies as **nominated**, which were indeed **nominated**.

## **6. Key Findings and Insights**

* **Random Forest** consistently outperformed the other models across both prediction tasks, achieving the highest accuracy and balanced F1-scores.
* The models performed better at predicting **non-nominated** movies (Class 0) and **low IMDb score** movies (Class 0), with higher precision and recall in these classes. This indicates that the model may have challenges predicting nominations and high IMDb scores due to data imbalance.
* **Logistic Regression** and **Decision Trees** struggled with precision for Class 1 (nominated or high IMDb score), with relatively lower F1-scores compared to Random Forest.

## **7. Suggestions for Future Analysis**

### **Model Improvements:**

1. **Class Imbalance**: The models show lower performance for predicting Class 1 (nominations and high IMDb scores). This could be improved by addressing class imbalance through techniques such as **oversampling** or **undersampling**.
2. **Hyperparameter Tuning**: For Random Forest, performance could be further optimized through **grid search** or **randomized search** to find the best hyperparameters.
3. **Feature Engineering**: Additional features such as the **profit margin** (Box Office - Budget) or interaction terms between variables (e.g., Genre and Director) could be explored to provide more predictive power.

### **Additional Features:**

* Consider incorporating additional data sources, such as movie reviews or external award data, which could provide more insights into the factors that influence nominations and high IMDb scores.

## **8. Conclusion**

In conclusion, the Random Forest model was the best performer for both predicting movie nominations and IMDb scores. Future improvements could focus on addressing class imbalance and further tuning the models to enhance their predictive capabilities.

1. **Code :**

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix

import seaborn as sns

import matplotlib.pyplot as plt

import matplotlib

matplotlib.use('Agg')

# Load the dataset

file\_path = 'movies\_data.csv'

movies\_data = pd.read\_csv(file\_path, encoding='ISO-8859-1')

print(movies\_data.columns)

# Display the first few rows and summary information of the dataset

movies\_data\_info = movies\_data.info()

movies\_data\_head = movies\_data.head()

movies\_data\_info, movies\_data\_head

# Create binary target for nominations (1 if nominated, 0 if not)

movies\_data['Nominated'] = movies\_data['Oscar and Golden Globes nominations'].apply(lambda x : 1 if x > 0 else 0)

# Create binary target for high IMDb score (1 if score >= 7.0, 0 if score < 7.0)

movies\_data['High\_IMDb\_Score'] = movies\_data['IMDb score'].apply(lambda x : 1 if x>= 7.0 else 0)

# Dropping unnecessary columns

movies\_cleaned = movies\_data.drop(columns = ['Movie', 'Actor 1', 'Actor 2' , 'Actor 3', 'Oscar and Golden Globes nominations', 'Oscar and Golden Globes awards' , 'IMDb score' ])

# Handling categorical columns: One-Hot Encoding for 'Genre' and 'Director'

movies\_cleaned = pd.get\_dummies(movies\_cleaned, columns=['Genre', 'Director'] , drop\_first=True)

# Define features (all columns except target) and targets (Nominated and High\_IMDb\_Score)

X = movies\_cleaned.drop(columns=['Nominated','High\_IMDb\_Score'])

y\_nominated = movies\_cleaned['Nominated']

y\_imdb = movies\_cleaned['High\_IMDb\_Score']

X\_train\_nom, X\_test\_nom, y\_train\_nom, y\_test\_nom = train\_test\_split (X, y\_nominated, test\_size=0.3, random\_state=42 )

X\_train\_imdb, X\_test\_imdb, y\_train\_imdb, y\_test\_imdb = train\_test\_split(X, y\_imdb, test\_size=0.3, random\_state=42)

scaler = StandardScaler()

X\_train\_nom = scaler.fit\_transform(X\_train\_nom)

X\_test\_nom = scaler.fit\_transform(X\_test\_nom)

X\_train\_imdb = scaler.fit\_transform(X\_train\_imdb)

X\_test\_imdb = scaler.fit\_transform(X\_test\_imdb)

# Initialize models

log\_reg = LogisticRegression(max\_iter=1000)

decision\_tree = DecisionTreeClassifier(random\_state=42)

random\_forest = RandomForestClassifier(random\_state=42)

# Train and evaluate models for predicting nominations

log\_reg.fit(X\_train\_nom, y\_train\_nom)

dec\_tree\_nom = decision\_tree.fit(X\_train\_nom, y\_train\_nom)

rand\_forest\_nom = random\_forest.fit(X\_train\_nom, y\_train\_nom)

# Train and evaluate models for predicting IMDb score

log\_reg.fit(X\_train\_imdb, y\_train\_imdb)

dec\_tree\_imdb = decision\_tree.fit(X\_train\_imdb, y\_train\_imdb)

rand\_forest\_imdb = random\_forest.fit(X\_train\_imdb, y\_train\_imdb)

# Predict nominations

y\_pred\_log\_nom = log\_reg.predict(X\_test\_nom)

y\_pred\_tree\_nom = dec\_tree\_nom.predict(X\_test\_nom)

y\_pred\_forest\_nom = rand\_forest\_nom.predict(X\_test\_nom)

# Predict IMDb scores

y\_pred\_log\_imdb = log\_reg.predict(X\_test\_imdb)

y\_pred\_tree\_imdb = dec\_tree\_imdb.predict(X\_test\_imdb)

y\_pred\_forest\_imdb = rand\_forest\_imdb.predict(X\_test\_imdb)

# Generate classification reports and accuracy for both tasks

nominations\_report = {

    'Logistic Regression': classification\_report(y\_test\_nom, y\_pred\_log\_nom, output\_dict=True),

    'Decision Tree': classification\_report(y\_test\_nom, y\_pred\_tree\_nom, output\_dict=True),

    'Random Forest': classification\_report(y\_test\_nom, y\_pred\_forest\_nom, output\_dict=True)

}

imdb\_report = {

    'Logistic Regression': classification\_report(y\_test\_imdb, y\_pred\_log\_imdb, output\_dict=True),

    'Decision Tree': classification\_report(y\_test\_imdb, y\_pred\_tree\_imdb, output\_dict=True),

    'Random Forest': classification\_report(y\_test\_imdb, y\_pred\_forest\_imdb, output\_dict=True)

}

# You should use the dictionaries you created directly:

reports = {

    "Nominations - Logistic Regression": nominations\_report['Logistic Regression'],

    "Nominations - Decision Tree": nominations\_report['Decision Tree'],

    "Nominations - Random Forest": nominations\_report['Random Forest'],

    "IMDb Score - Logistic Regression": imdb\_report['Logistic Regression'],

    "IMDb Score - Decision Tree": imdb\_report['Decision Tree'],

    "IMDb Score - Random Forest": imdb\_report['Random Forest']

}

# Display the reports

# Instead of just returning the reports, print each one for better readability

# Print classification reports for nominations prediction

print("Nominations - Logistic Regression:")

print(classification\_report(y\_test\_nom, y\_pred\_log\_nom))

print("\nNominations - Decision Tree:")

print(classification\_report(y\_test\_nom, y\_pred\_tree\_nom))

print("\nNominations - Random Forest:")

print(classification\_report(y\_test\_nom, y\_pred\_forest\_nom))

# Print classification reports for IMDb score prediction

print("\nIMDb Score - Logistic Regression:")

print(classification\_report(y\_test\_imdb, y\_pred\_log\_imdb))

print("\nIMDb Score - Decision Tree:")

print(classification\_report(y\_test\_imdb, y\_pred\_tree\_imdb))

print("\nIMDb Score - Random Forest:")

print(classification\_report(y\_test\_imdb, y\_pred\_forest\_imdb))

# Plot confusion matrix for Random Forest on nominations

cf\_matrix\_nom = confusion\_matrix(y\_test\_nom, y\_pred\_forest\_nom)

plt.figure(figsize=(10, 7))

sns.heatmap(cf\_matrix\_nom, annot=True, fmt='d', cmap='Blues')

plt.title("Confusion Matrix - Random Forest (Nominations)")

plt.savefig('confusion\_matrix\_nominations.png')  # Save plot

# Plot confusion matrix for Random Forest on IMDb score

cf\_matrix\_imdb = confusion\_matrix(y\_test\_imdb, y\_pred\_forest\_imdb)

plt.figure(figsize=(10, 7))

sns.heatmap(cf\_matrix\_imdb, annot=True, fmt='d', cmap='Blues')

plt.title("Confusion Matrix - Random Forest (IMDb Score)")

plt.savefig('confusion\_matrix\_imdb.png')  # Save plot

#python moviesml.py