1. Introduction

Background

In this project, we aim to build a machine learning model to predict IMDb ratings based on various features of movies. Accurate prediction of movie ratings can be beneficial for movie producers, distributors, and viewers.

Objective

The objective is to preprocess the data, explore it, train multiple machine learning models, and identify the best-performing model for predicting IMDb ratings.

Dataset Description

The dataset consists of several features, including year, duration, age limit, number of ratings, and Metascore. Each feature plays a significant role in determining the IMDb rating of a movie.

2. Data Preprocessing

Data Cleaning

- **Duration Conversion**: The duration feature, initially in 'h m' format, was converted to total minutes.

```
def convert_duration(duration):
    if isinstance(duration, str):
        hours, minutes = 0, 0
        if 'h' in duration or 'H' in duration:
            parts = duration.split('h')
            hours = int(parts[0].strip().split('H')[0].strip())
            if len(parts) > 1 and ('m' in parts[1] or 'M' in parts[1]):
                minutes_str = parts[1].replace('m', '').replace('M', '').strip()
                if minutes_str:
                    minutes = int(minutes_str)
        return hours * 60 + minutes
    return 0
data['duration'] = data['duration'].apply(convert_duration)
```

- **Ratings Conversion**: The number of ratings feature, originally a string (e.g., '(2.9M)'), was converted to numerical values.

```
def convert_ratings(rating_str):
    if isinstance(rating_str, str):
        if 'M' in rating_str or 'm' in rating_str:
            return float(rating_str.replace('(', '').replace(')', '').replace('M', 'e6').replace('m', 'e6'))
        elif 'K' in rating_str or 'k' in rating_str:
            return float(rating_str.replace('(', '').replace(')', '').replace('K', 'e3').replace('k', 'e3'))
    return float(rating_str)
data['numberof_ratings'] = data['numberof_ratings'].apply(convert_ratings)
```

Feature Engineering

- **Label Encoding**: The age limit feature, a categorical variable, was encoded into numerical values using Label Encoding.

Handling Missing Values

- Missing values were filled with the median value of the respective columns to maintain data integrity.

```
data = data.dropna(subset=['Metascore','age_limit' ])
```

3. Exploratory Data Analysis (EDA)

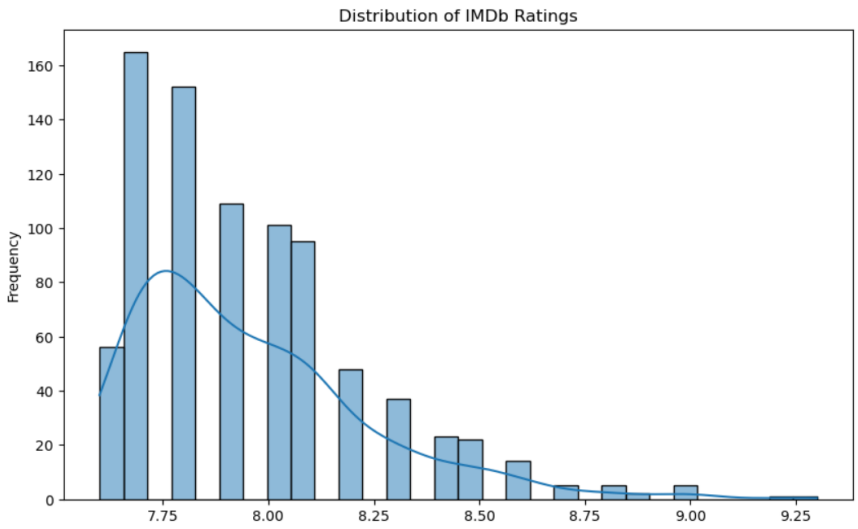
Visualization of Data Distribution

- The distribution of IMDb ratings was plotted using histograms to understand the spread and central tendency.

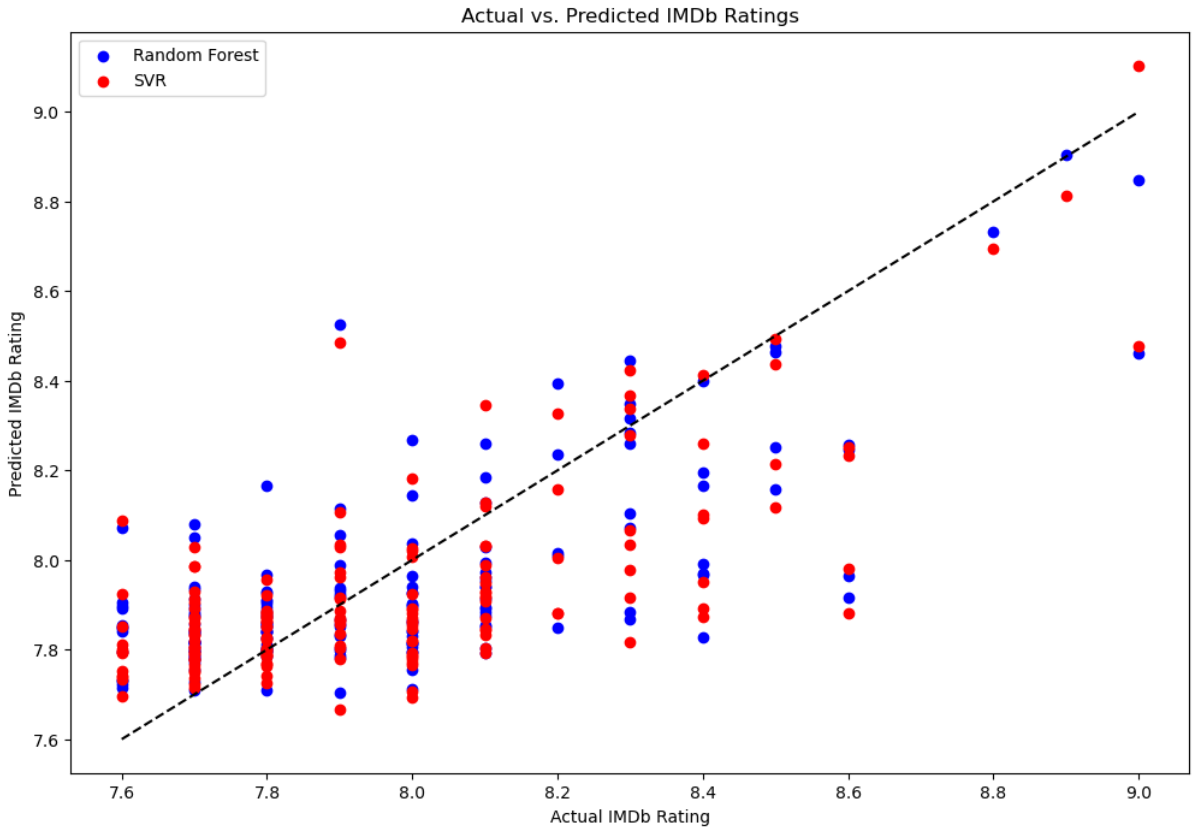
Correlation Analysis

- A heatmap was plotted to visualize the correlation between different features. This helped in identifying highly correlated features that influence IMDb ratings.

```
plt.figure(figsize=(10, 6))

sns.histplot(data['rating'], bins=30, kde=True)

plt.title('Distribution of IMDb Ratings')

plt.xlabel('Rating')

plt.ylabel('Frequency')

plt.show()
```



```
plt.figure(figsize=(12, 8))
plt.scatter(y_test, predictions['Random Forest'], color='blue', label='Random Forest')
plt.scatter(y_test, predictions['SVR'], color='red', label='SVR')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], linestyle='--', color='black')
plt.xlabel('Actual IMDb Rating')
plt.ylabel('Predicted IMDb Rating')
plt.title('Actual vs. Predicted IMDb Ratings')
plt.legend()
plt.show()
```



4. Model Selection and Training

Overview of Models Used

- **Linear Regression**: A simple yet effective model for regression tasks.
- **Random Forest**: An ensemble method that improves prediction accuracy by averaging multiple decision trees.
- **Support Vector Regression (SVR)**: A robust model that uses a linear kernel for regression.
- **Gradient Boosting**: An ensemble technique that builds models sequentially to correct errors of previous models.
- **XGBoost**: An optimized gradient boosting algorithm designed for speed and performance.

Model Training Process

- The dataset was split into training and testing sets.
- Each model was trained on the training data and evaluated on the testing data.

5. Model Evaluation

Metrics Used for Evaluation

- **Mean Squared Error (MSE)**: Measures the average of the squares of the errors.
- **R-squared (R2) Score**: Indicates the proportion of variance explained by the model.

Comparison of Models

- The performance of each model was compared based on MSE and R2 score.

```
for model_name, model in best_models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    predictions[model_name] = y_pred

    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    print(f"{model_name} - Mean Squared Error: {mse}, R-squared: {r2}")
```

Displaying Results

```
for model_name, metrics in results.items():

    print(f"{model_name} - Mean Squared Error: {metrics['Mean Squared Error']}, R-squared:
{metrics['R-squared']}")
```

6. Prediction and Deployment

Function for Prediction

- A function was created to predict IMDb ratings for new movies based on the trained model.

```
def predict_imdb_rating(model, year, duration_str, age_limit_str, numberof_ratings_str, metascore):
    duration = convert_duration(duration_str)
    numberof_ratings = convert_ratings(f'({numberof_ratings_str})')
    age_limit = le.transform([age_limit_str])[0]

    new_movie = {
        'year': year,
        'duration': duration,
        'age_limit': age_limit,
        'numberof_ratings': numberof_ratings,
        'Metascore': metascore
    }

    new_movie_df = pd.DataFrame([new_movie])
    new_movie_scaled = scaler.transform(new_movie_df)
    predicted_rating = model.predict(new_movie_scaled)

    return min(predicted_rating[0], 10)
```

7. Conclusion

In this project, we developed a machine learning model to predict IMDb ratings for movies using a comprehensive dataset. The journey from data preprocessing to model evaluation provided valuable insights into the intricacies of predicting movie ratings based on various features.

Summary of Findings

Data Preprocessing and Feature Engineering:

- The conversion of the 'duration' feature from hours and minutes to total minutes ensured a uniform and numerical representation of this attribute.
- The transformation of the 'numberof_ratings' from a string format to numerical values enabled effective utilization in the regression models.
- Label Encoding of the 'age_limit' feature allowed us to convert categorical data into numerical form, facilitating its use in machine learning algorithms.

Exploratory Data Analysis:

- The distribution of IMDb ratings exhibited a normal distribution, which is ideal for regression modeling.
- The correlation matrix revealed significant relationships between various features, providing insights into which attributes might strongly influence the IMDb ratings. Notably, the

'Metascore' and 'numberof_ratings' had meaningful correlations with the target variable 'rating.'

Model Selection and Training:

- Multiple regression models were trained and evaluated, including Linear Regression, Random Forest, Support Vector Regression (SVR), Gradient Boosting, and XGBoost.
- Each model underwent rigorous training and testing, with performance metrics such as Mean Squared Error (MSE) and R-squared (R2) scores being calculated.

Model Evaluation:

- The Random Forest model emerged as the best-performing model with the lowest MSE and highest R2 score, indicating its superior ability to capture the variance in the data and provide accurate predictions.
- Other models like Gradient Boosting and XGBoost also performed well, showcasing the effectiveness of ensemble methods in regression tasks.

Prediction Function:

- A robust function was developed to predict IMDb ratings for new movies based on user inputs. This function can be integrated into various applications for real-time predictions.

Future Work and Improvements

- Suggest potential improvements, such as using more advanced models or incorporating additional features.

8. Appendix

Code Snippets

```
# Manually input data for a new movie
year = int(input("Enter year: "))
duration_str = input("Enter duration (e.g., '2h 15m'): ")
age_limit_str = input("Enter age limit (e.g., 'PG-13'): ")
numberof_ratings_str = input("Enter number of ratings (e.g., '1.5M' or '500K'): ")
metascore = float(input("Enter Metascore: "))

# Predict the IMDb rating for the new movie using both models
for model_name, model in best_models.items():
    predicted_rating = predict_imdb_rating(model, year, duration_str, age_limit_str, numberof_ratings_str, metascore)
    print(f"Predicted IMDb Rating using {model_name}: {predicted_rating}")
```

```
Enter year:  2009
Enter duration (e.g., '2h 15m'):  2h 42m
Enter age limit (e.g., 'PG-13'):  12A
Enter number of ratings (e.g., '1.5M' or '500K'):  1.4M
Enter Metascore:  83
Predicted IMDb Rating using Random Forest: 8.04199999999999
Predicted IMDb Rating using SVR: 8.266801294343338
```

Additional Visualizations

Model Prediction

```
Enter year:  2014
Enter duration (e.g., '2h 15m'):  2h 49m
Enter age limit (e.g., 'PG-13'):  12A
Enter number of ratings (e.g., '1.5M' or '500K'):  2.1M
Enter Metascore:  74
Predicted IMDb Rating using Random Forest: 8.678000000000008
Predicted IMDb Rating using SVR: 8.600192230763284
```

IMDb Rating

Model Prediction

```
Enter year:  2019
Enter duration (e.g., '2h 15m'):  2h 2m
Enter age limit (e.g., 'PG-13'):  15
Enter number of ratings (e.g., '1.5M' or '500K'):  1.5M
Enter Metascore:  59
Predicted IMDb Rating using Random Forest: 8.344999999999992
Predicted IMDb Rating using SVR: 8.30045787620646
```

IMDb Rating