MOVIE RATING PREDICTION WITH PYTHON

Project Overview:

This project focuses on predicting the IMDb ratings of movies using a dataset containing features such as movie title, genre, director, cast, year, and more. The workflow involves:

- ❖ Data Preprocessing
- ❖ Exploratory Data Analysis (EDA)
- ❖ Feature Engineering
- ❖ Model Building
- ❖ Model Evaluation

Data Preprocessing:

- ➔ Handle missing values.
- ➔ Convert categorical variables into numerical formats.
- ➔ Clean and process columns like Duration, Votes, and Year.

```
print(data.isnull().sum())
data = data.dropna(subset=['Rating'])
categorical_cols = ['Genre', 'Director', 'Actor 1', 'Actor 2', 'Actor 3']

for col in categorical_cols:
    data[col] = data[col].fillna('Unknown')
data['Duration'] = data['Duration'].str.replace(' min',
'').astype('float').fillna(data['Duration'].mean())
data['Votes'] = data['Votes'].str.replace(',', '').astype('float').fillna(data['Votes'].mean())
data['Year'] =
data['Year'].str.extract(r'(\d{4})').astype(float).fillna(data['Year'].mode()[0])
```

Exploratory Data Analysis (EDA):

➔ Visualize key relationships in the dataset to understand features like genre distribution, correlation with ratings, and more.

```
import matplotlib.pyplot as plt
import seaborn as sns
sns.histplot(data['Rating'], bins=20, kde=True)
plt.title('Distribution of IMDb Ratings')
plt.show()
top_genres =
data.groupby('Genre')['Rating'].mean().sort_values(ascending=False).head(10)
top_genres.plot(kind='bar', color='skyblue')
plt.title('Top 10 Genres by Average Rating')
plt.show()
numeric_cols = ['Duration', 'Votes', 'Year', 'Rating']
sns.heatmap(data[numeric_cols].corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```

Feature Engineering:

➔ Convert categorical features like Genre and Director into numeric formats using one-hot encoding or label encoding.

```
from sklearn.preprocessing import OneHotEncoder, LabelEncoder
label_cols = ['Genre', 'Director', 'Actor 1', 'Actor 2', 'Actor 3']
for col in label_cols:
    le = LabelEncoder()
    data[col] = le.fit_transform(data[col])
features = ['Year', 'Duration', 'Votes', 'Genre', 'Director', 'Actor 1', 'Actor 2', 'Actor 3']
X = data[features]
y = data['Rating']
```

Model Building:

- ➔ Split the dataset into training and testing sets. Build and train models like Linear Regression, Random Forest, or Gradient Boosting.

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

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

Model Evaluation:

- ➔ Evaluate the model using metrics like RMSE and R^2 score.

```
rmse = mean_squared_error(y_test, y_pred, squared=False)
r2 = r2_score(y_test, y_pred)

print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"R² Score: {r2}")
```

Program:

```
# Importing necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import LabelEncoder
```

```
# Load the dataset
dataset_path = "IMDb Movies India.csv"  # Update with your dataset path
data = pd.read_csv(dataset_path, encoding="latin1")

# Data Preprocessing
# Drop rows with missing target variable (Rating)
data = data.dropna(subset=["Rating"])

# Fill missing categorical columns with 'Unknown'
categorical_cols = ["Genre", "Director", "Actor 1", "Actor 2", "Actor 3"]
for col in categorical_cols:
    data[col] = data[col].fillna("Unknown")

# Handle missing numeric columns
data["Duration"] = data["Duration"].str.replace(" min",
"").astype(float).fillna(data["Duration"].mean())
data["Votes"] = data["Votes"].str.replace(",",
"").astype(float).fillna(data["Votes"].mean())
data["Year"] =
data["Year"].str.extract(r"(\d{4})").astype(float).fillna(data["Year"].mode()[0])

# Exploratory Data Analysis (EDA)
# Plot distribution of Ratings
sns.histplot(data["Rating"], bins=20, kde=True)
plt.title("Distribution of IMDb Ratings")
plt.xlabel("Rating")
plt.ylabel("Frequency")
plt.show()

# Top Genres by Average Rating
top_genres =
data.groupby("Genre")["Rating"].mean().sort_values(ascending=False).head(10)
top_genres.plot(kind="bar", color="skyblue")
plt.title("Top 10 Genres by Average Rating")
plt.xlabel("Genre")
plt.ylabel("Average Rating")
plt.show()

# Correlation heatmap
numeric_cols = ["Duration", "Votes", "Year", "Rating"]
sns.heatmap(data[numeric_cols].corr(), annot=True, cmap="coolwarm")
plt.title("Correlation Heatmap")
plt.show()

# Feature Engineering
# Encode categorical columns
label_cols = ["Genre", "Director", "Actor 1", "Actor 2", "Actor 3"]
for col in label_cols:
    le = LabelEncoder()
    data[col] = le.fit_transform(data[col])

# Select features and target
features = ["Year", "Duration", "Votes", "Genre", "Director", "Actor 1", "Actor 2",
"Actor 3"]
X = data[features]
y = data["Rating"]

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Model Building
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Predictions
y_pred = model.predict(X_test)
```

```
# Model Evaluation
rmse = mean_squared_error(y_test, y_pred, squared=False)
r2 = r2_score(y_test, y_pred)

print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"R² Score: {r2}")

# Feature Importance Visualization
importances = model.feature_importances_
feature_names = X.columns
sns.barplot(x=importances, y=feature_names, color="blue")
plt.title("Feature Importance")
plt.xlabel("Importance")
plt.ylabel("Feature")
plt.show()
```

Results and Conclusion:

➔ Summarize findings, such as the most important features influencing movie ratings.

➔ Discuss the model's performance metrics.

```
importances = model.feature_importances_
feature_names = X.columns
sns.barplot(x=importances, y=feature_names, color='blue')
plt.title('Feature Importance')
plt.show()
```