PREDICTING IMDB SCORES

CONCEPT :

Continuously building an IMDb score prediction model involves several key steps, including feature engineering, model training, and evaluation. Below I will describe the progress of this project in detail:

DATASOURCE:
https://www.kaggle.com/datasets/luiscorter/netflix-original-films-imdb-scores

1. DATA COLLECTION:

- Collect movie database, including cast and crew details, genre, budget, year of release and other relevant information. This data set should also include the IMDb score, which will be the target variable for prediction.

2. DATA PROCESSING:

- Clean up and recycle the database. This includes handling missing values, removing duplicates, and changing data types as needed.

- Feature Selection: Determine which features are most likely to predict IMDb scores. Some features may have a stronger effect on the score than others.

3. TECHNICAL FEATURES:

- This is an important step in building a predictive model. Engineering features include:

- Create new features that can capture important information. For example, you can create a "director's score" based on the director's previous work.

- Categorical Coding Features: Convert categorical variables into numerical form (eg, one-hot coding or label coding).

- Scale and scale to keep digital features the same size.

- Text data manipulation: If you have a text-based feature (eg movie description or review), you can use natural language processing (NLP) techniques to extract useful information.

4. DISTRIBUTION OF INFORMATION:

- Split the database into training and test sets. The training set is used to train the model and the test set is used to evaluate its performance.

5. MODEL SELECTION:

- Select the appropriate machine learning model for the prediction problem. Common options include regression models such as linear regression, decision tree regression, random forest regression, or more advanced methods such as gradient boosting or neural networks.

6. STUDY MODEL:

- Train the selected model on the training database using the generated features. The model at this stage examines the relationship between features and IMDb scores.

7. EVALUATION MODEL:

- Evaluate the performance of the model using the test database. Common evaluation criteria for regression problems such as IMDb score prediction are:

- Mean Absolute Error (MAE): measures the average absolute difference between the predicted score and the actual score.

- Mean Squared Error (MSE): measures the average squared difference between the predicted score and the actual score.

- Root Mean Square Error (RMSE): The square root of MSE, which is a more intuitive measure.

- R-squared score (R2): Measures the proportion of variation in IMDb scores explained by the model.

- Visualization: Create scatterplots to visualize model predictions and actual scores.

8. CONFIGURE HYPERPARAMETERS(OPTIONAL):

- If the performance of the model is not satisfactory, consider hyperparameter tuning to optimize the model parameters for better results.

9. MODEL LOCATION(IF APPLICABLE):

- If you plan to use the model in a real-world application, position it so that it can make predictions about new, unseen data.

10. DOCUMENTS AND REPORTS:

- Document the entire project including data processing steps, feature engineering, model selection, training and evaluation.

Provide insights and conclusions about the performance of the model.

11. MAINTENANCE AND SUPERVISION:

- If the model is deployed, regularly monitor performance and update as needed to account for data distribution and model changes.

The success of the IMDb scoring model depends on data quality, feature engineering efficiency, appropriate model selection, and evaluation robustness. The process requires iteration and refinement to achieve optimal predictive accuracy.

CODING :

```
# Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import matplotlib.pyplot as plt

# Load your dataset with 'latin1' encoding
```

```
data = pd.read_csv("/content/NetflixOriginals.csv", encoding='latin1')

# List columns and display the first few rows of the dataset
print(data.columns)
print(data.head())

# Check for the correct column name for the dependent variable (target)
possible_column_names = ["IMDb_score", "IMDbRating", "IMDB_rating"]  # Add potential names

# Initialize y and X as None
y = None
X = None

# Try to find the correct column name for the target variable
for column_name in possible_column_names:
    if column_name in data.columns:
        y = data[column_name]
        X = data.drop(column_name, axis=1)
        break  # Break the loop if a valid column name is found
```

```
# Check if y and X have been assigned
if y is None or X is None:
    print("None of the potential column names were found in the dataset. Please inspect the dataset manually.")
else:
    # Split the 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)

    # Build and train a model
    model = LinearRegression()
    model.fit(X_train, y_train)

    # Make predictions
    y_pred = model.predict(X_test)

    # Evaluate the model
    mae = mean_absolute_error(y_test, y_pred)
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    # Print evaluation metrics
    print(f"Mean Absolute Error: {mae}")
```

```
print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")
# Create a residual plot to visualize prediction errors
plt.figure(figsize=(10, 6))
residuals = y_test - y_pred
plt.scatter(y_pred, residuals, alpha=0.5)
plt.title("Residual Plot")
plt.xlabel("Predicted IMDb Scores")
plt.ylabel("Residuals (Actual IMDb Score - Predicted IMDb Score)")
plt.axhline(0, color='red', linestyle='--')
plt.show()
```

OUTPUT :