IMDb Score Prediction

Project Title: IMDb Score Prediction

Problem Statement: Develop a machine learning model to predict the IMDb scores of movies available on Films based on their genre, premiere date, runtime, and language. The model aims to accurately estimate the popularity of movies to assist users in discovering highly rated films that align with their preferences.

Problem Definition:IMDb scores are determined by user ratings and can change over time as more users rate the movie or show.Here are some common factors that can influence IMDb scores:

User Ratings: IMDb scores are primarily based on user ratings and reviews. The more users rate a movie or show, the more reliable the score tends to be. Higher user ratings typically result in higher IMDb scores.

Critic Reviews: Professional critic reviews can also impact IMDb scores. Movies and shows that receive positive reviews from critics may have higher scores.

Box Office Performance: Box office success can sometimes correlate with higher IMDb scores, as it suggests that a larger audience enjoyed the film.

Awards and Nominations: Movies and shows that receive awards or nominations from prestigious organizations may receive higher IMDb scores as a result.

Genre and Audience: Different genres appeal to different audiences, and IMDb scores can vary widely depending on the target audience. For example, a niche horror film might have a lower score compared to a mainstream blockbuster.

Marketing and Hype: The marketing and pre-release buzz surrounding a movie or show can influence IMDb scores. High expectations can lead to disappointment and lower ratings if the final product doesn't meet them.

Word of Mouth: Positive word-of-mouth recommendations from viewers can help boost IMDb scores over time.

Personal Taste: IMDb scores are subjective and reflect the opinions of the users who rate them. Personal taste plays a significant role in determining the score for any given individual.

Creating a machine learning model to predict IMDb scores for movies based on genre, premiere date, runtime, and language is a complex task that involves multiple steps. We will also need a dataset

containing relevant information about movies and their IMDb scores.

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

Predicting IMDb scores involves several steps, including data collection, preprocessing, feature selection, model training, and evaluation, etc,…Here's a high-level overview of the process:

- Data Collection
- Data preprocessing
- Feature Engineering:
- Split the Data
- Model Selection
- Model Training
- Model Evaluation
- Hyperparameter Tuning
- Model Interpretation
- Final Model Selection
- Test the Model
- Deployment:
- Monitoring and Maintenance

1. Data Collection: Data collection is the process of gathering and measuring information on variables of interest in a systematic and organized manner.

Data Collection Methods:

- **Surveys:** Questionnaires or interviews are used to gather information from individuals or groups.
- **Observations:** Researchers directly watch and record behaviors, events, or phenomena.
- **Experiments:**Controlled tests are conducted to gather data under specific conditions.
- **Secondary Data:** Existing data from sources like government databases, academic studies, or company records can be utilized.

Sensor Data: Sensors and IoT devices can collect real-time data on various physical parameters.

Web Scraping: Automated techniques can extract data from websites.

First, you need a dataset containing movie information, including features like genre, premiere date, runtime, language, and IMDb scores.
Websites like

- IMDb,
- Kaggle, or
- APIs like OMDB can be useful for gathering such data.

2. Data Preprocessing:

- Clean and preprocess the data. This may involve handling missing values, encoding categorical variables (like genres and languages), and normalizing numerical features.
- Clean the dataset by handling missing values, removing duplicates, and dealing with outliers.

- Convert categorical variables like genres and languages into numerical format using techniques like one-hot encoding or label encoding.
- Normalize or scale numerical features like runtime to ensure they have the same range.
- Once you have the data, preprocess it to make it suitable for machine learning:

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
data = pd.read_csv('your_dataset.csv')
label_encoders = {}
for feature in ['genre', 'language']:
    le = LabelEncoder()
    data[feature] = le.fit_transform(data[feature])
    label_encoders[feature] = le
data['premiere_date'] = pd.to_datetime(data['premiere_date'])
data['days_since_premiere'] = (data['premiere_date'] -
data['premiere_date'].min()).dt.days
data.drop(['premiere_date'], axis=1, inplace=True)
X = data[['genre', 'days_since_premiere', 'runtime', 'language']]
y = data['imdb_score']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

3.Feature Engineering:

Create additional features if necessary. For example, you might extract features like the month of the premiere date, the year of the premiere, or the number of genres associated with a movie.

4. Split the Data:

- Divide your dataset into training, validation, and test sets to evaluate the model's performance.
- Typically, an 80-10-10 or 70-15-15 split is common.

5. Model Selection:

- Choose an appropriate machine learning algorithm for regression, as this is a regression problem (predicting IMDb scores). Common choices include:
- Linear Regression
- Decision Trees
- Random Forests
- Gradient Boosting (e.g., XGBoost, LightGBM)
- Neural Networks (Deep Learning)

6. Model Training:

- Train your selected model(s) on the training data using appropriate hyperparameters.
- Choose a machine learning algorithm suitable for regression tasks. Here, I'll use XGBoost as an example:

```python
from xgboost import XGBRegressor
from sklearn.metrics import mean_squared_error
```

```
model = XGBRegressor()
model.fit(X_train, y_train)
predictions = model.predict(X_test)
rmse = mean_squared_error(y_test, predictions, squared=False)
print("Root Mean Squared Error: {:.2f}".format(rmse))
```

7. Model Evaluation:

- Evaluate your model(s) on the validation set using relevant evaluation metrics for regression tasks, such as

✓ Mean Absolute Error (MAE),
✓ Root Mean Squared Error (RMSE), and
✓ R-squared (R2).

8. Hyperparameter Tuning:

- Experiment with different hyperparameters to optimize your model's performance.
- Techniques like grid search or random search can be used for this purpose.

9. Model Interpretation (Optional):

If using a complex model like a neural network, consider using techniques like feature importance or SHAP values to understand how different features impact the predictions.

10. Final Model Selection:

After hyperparameter tuning and validation, select the best-performing model.

11. Test the Model:

Evaluate the final model on the test set to get an unbiased estimate of its performance.

12. Deployment:

If the model meets your performance requirements, you can deploy it as an application or service that takes input data (movie information) and returns IMDb score predictions.

13. Monitoring and Maintenance:

Continuously monitor the model's performance in production and update it as needed with new data or improvements

Here's a simplified Python code example using scikit-learn for this task:

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
data = pd.read_csv('movie_data.csv')
X = data.drop(columns=['IMDb_Score'])
y = data['IMDb_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)
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')
feature_importances = model.feature_importances_
print('Feature Importances:')
for feature, importance in zip(X.columns, feature_importances):
   print(f'{feature}: {importance}'}

   ‘’’
```

Here the quality and size of our dataset, as well as the choice of features and model, will greatly impact the predictive performance of our IMDb score prediction model. Experimentation and iteration are often necessary to achieve the best results.