Mellors MSDS610 Week6 Assignment - Notebook 1: Splitting and Running the Model

Loading Libraries and Data

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
from sqlalchemy import create_engine

import joblib
from sklearn.model_selection import train_test_split
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
import xgboost as xgb
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from imblearn.over_sampling import SMOTE
```

In [2]:
```
host = r'127.0.0.1'
db = r'MSDS610'
user = r'postgres'
pw = r'postgres'
port = r'5432'
schema = r'clean'
```

In [3]:
```
db_conn = create_engine("postgresql://{}:{}@{}:{}/{}".format(user, pw, host, port, db))
```

In [4]:
```
table_name = r'movies_cleaned'
schema = r'cleaned'
```

In [5]:
```
df = pd.read_sql_table(table_name, db_conn, schema)
```

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4803 entries, 0 to 4802
Data columns (total 16 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   budget             4803 non-null   int64  
 1   popularity         4803 non-null   float64
 2   revenue            4803 non-null   int64  
 3   title              4803 non-null   object 
 4   vote_average       4803 non-null   float64
 5   vote_count         4803 non-null   int64  
 6   clean_genres       4803 non-null   object 
 7   clean_keywords     4803 non-null   object 
 8   clean_tagline      4803 non-null   object 
 9   clean_overview     4803 non-null   object 
 10  financial_success  4803 non-null   int64  
 11  critical_success   4803 non-null   int64  
 12  audience_success   4803 non-null   int64  
 13  budget_score       4803 non-null   float64
 14  success_score      4803 non-null   float64
 15  worth_funding      4803 non-null   int64  
dtypes: float64(4), int64(7), object(5)
memory usage: 600.5+ KB
```

Splitting the Data (3-way)

In [7]:
```
text_features = ["clean_genres", "clean_keywords", "clean_tagline", "clean_overview"]
y = df["worth_funding"]
```

In [8]:
```
df["combined_text"] = df[text_features].apply(lambda x: " ".join(x.astype(str)), axis=1)
```

In [9]:
```
X_train, X_temp, y_train, y_temp = train_test_split(df["combined_text"], y, test_size=0.2, random_state=4, stratify=y
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=4, stratify=y_temp)
```

In [10]:
```
X_temp.info()
```

```
<class 'pandas.core.series.Series'>
Index: 961 entries, 155 to 3088
Series name: combined_text
Non-Null Count  Dtype 
--------------  ----- 
961 non-null    object
dtypes: object(1)
memory usage: 15.0+ KB
```

In [11]:
```
X_temp.head()
```

Out[11]:
```
155     fantasy  action  adventure  family  comedy dyr...
1953    comedy  drama  romance sex  aquarium  nudity  ...
1567    mystery  thriller chicago  fbi  menace  covere...
3770    adventure  action  western robbery  miner  tre...
3905    drama  foreign female nudity  pagan  aztec ind...
Name: combined_text, dtype: object
```

In [12]:
```
y_temp.info()
```

```
<class 'pandas.core.series.Series'>
Index: 961 entries, 155 to 3088
Series name: worth_funding
Non-Null Count  Dtype
--------------  -----
961 non-null    int64
dtypes: int64(1)
memory usage: 15.0 KB
```

In [13]:
```
y_temp.sample(10)
```

Out[13]:
```
1452    2
4099    2
621     0
3293    0
2701    1
480     1
2333    0
2306    1
3738    1
1169    0
Name: worth_funding, dtype: int64
```

In [14]:
```
y_temp.value_counts()
```

Out[14]:
```
worth_funding
0    482
1    354
2    125
Name: count, dtype: int64
```

In [15]:
```python
print(X_test.shape)
X_test.head()
```

Out[15]:
```
(481,)
2860    drama  thriller  mystery toronto  lesbian  rem...
1843    drama  family  fantasy  adventure based on nov...
366     comedy politics  politician  election campaign...
4272    drama  mystery secret  nightclub  pet shop  in...
2447    drama  mystery  thriller audio tape  hitman  p...
Name: combined_text, dtype: object
```

In [16]:
```python
print(X_val.shape)
X_val.head()
```

Out[16]:
```
(480,)
3932    action  comedy  horror monster  pub  duringcre...
3440    comedy  drama  mystery independent film every ...
1417    action  comedy  crime new york  money launderi...
3133    comedy hotel  infidelity  onenight stand  frie...
2397    comedy alcohol  baby  party  family  fraternit...
Name: combined_text, dtype: object
```

Building the Model

In [17]:
```python
vectorizer = TfidfVectorizer(max_features=12000)
X_train_tfidf = vectorizer.fit_transform(X_train)
X_val_tfidf = vectorizer.transform(X_val)
```

In [18]:
```python
smote = SMOTE(sampling_strategy="auto", random_state=4)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train_tfidf, y_train)
```

In [19]:
```
xgb_model = xgb.XGBClassifier(
    objective="multi:softprob",
    eval_metric="mlogloss",
    reg_lambda=6.0,
    reg_alpha=4.0,
    max_depth=3,
    learning_rate=0.03,
    n_estimators=500,
    subsample=0.7,
    colsample_bytree=0.7)
xgb_model.fit(X_train_resampled, y_train_resampled)
```

Out[19]:
```
▾                              XGBClassifier                              i

XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=0.7, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric='mlogloss',
              feature_types=None, gamma=None, grow_policy=None,
              importance_type=None, interaction_constraints=None,
              learning_rate=0.03, max_bin=None, max_cat_threshold=None,
              max_cat_to_onehot=None, max_delta_step=None, max_depth=3,
              max_leaves=None, min_child_weight=None, missing=nan,
              monotone_constraints=None, multi_strategy=None, n_estimators=500,
              n_jobs=None, num_parallel_tree=None, objective='multi:softprob', ...)
```

In [20]:
```
y_train_pred = xgb_model.predict(X_train_resampled)
```

In [21]:
```
train_accuracy = accuracy_score(y_train_resampled, y_train_pred)
train_report = classification_report(y_train_resampled, y_train_pred)
train_conf_matrix = confusion_matrix(y_train_resampled, y_train_pred)
```

In [22]:
```
print(f"Training Accuracy: {train_accuracy:.4f}")
print("Training Classification Report:\n", train_report)
print("Training Confusion Matrix:\n", train_conf_matrix)
```

```
Training Accuracy: 0.7229
Training Classification Report:
               precision    recall  f1-score   support

           0       0.60      0.82      0.69      1926
           1       0.77      0.57      0.65      1926
           2       0.88      0.78      0.82      1926

    accuracy                           0.72      5778
   macro avg       0.75      0.72      0.72      5778
weighted avg       0.75      0.72      0.72      5778

Training Confusion Matrix:
 [[1588  292   46]
 [ 673 1092  161]
 [ 387   42 1497]]
```

Exporting Data for Validation (Notebook 2)

Saving the Model and the Vectorizor

In [23]:
```
joblib.dump(xgb_model, "xgb_model.pkl")
joblib.dump(vectorizer, "tfidf_vectorizer.pkl")
```

Out[23]:
```
['tfidf_vectorizer.pkl']
```

Saving the Validation Set

In [24]:
```
X_val.to_csv("X_val.csv", index=False, header=True)
y_val.to_csv("y_val.csv", index=False, header=True)
```

In [39]:
```
df.to_csv("clean_movie_df.csv", index=False)
```