```python
# import csv file , paste file path in place of movies.csv
import pandas as pd
movies = pd.read_csv('movies.csv')
print(movies)
```

```
                      Movie                                   Genre
Rating  \
0                   3 Idiots                            Comedy, Drama
8.4
1                      Anand                                   Drama
8.3
2   Baahubali: The Beginning    Action, Adventure, Drama, Fantasy
8.3
3          Bajrangi Bhaijaan                  Action, Comedy, Drama
8.0
4                      Black                          Drama, Thriller
8.2
..                       ...                                     ...
...
75                 Virumaandi                   Action, Crime, Drama
8.1
76               Wake Up Sid                 Comedy, Drama, Romance
7.6
77                     Wazir                          Crime, Thriller
7.0
78                   Welcome                            Comedy, Drama
6.4
79                   Pathaan        Action, Adventure, Thriller\n
8.2

    Year Box Office Performance Box Office Collection
0   2009              Super Hit              439 crore
1   1971                    Hit              1.5 crore
2   2015             Blockbuster              650 crore
3   2015             Blockbuster              969 crore
4   2005                    Hit             24.9 crore
..   ...                    ...                    ...
75  2004                   Flop               40 crore
76  2009                    Hit               55 crore
77  2016                   Flop             52.5 crore
78  2007             Blockbuster           1.35 billion
79  2023             Blockbuster           1.05 billion

[80 rows x 6 columns]
```

```python
# importing json file
import pandas as pd
import json
movies = pd.read_csv('movies.csv')
# Define a function to handle JSON parsing errors
```

```python
def parse_genre(x):
    try:
        return list(json.loads(x).values())
    except (json.JSONDecodeError, TypeError):
        print(f"Error parsing JSON in row: {x}")
        return None
# Apply the function to the 'Genre' column
movies['Genre'] = movies['Genre'].apply(parse_genre)
print(movies)
```

```
Error parsing JSON in row:   Comedy, Drama
Error parsing JSON in row:   Drama
Error parsing JSON in row:   Action, Adventure, Drama, Fantasy
Error parsing JSON in row:   Action, Comedy, Drama
Error parsing JSON in row:   Drama, Thriller
Error parsing JSON in row:   Drama, Sport
Error parsing JSON in row:   Action, Comedy, Drama
Error parsing JSON in row:   Biography, Drama, Sport
Error parsing JSON in row:   Comedy, Drama, Romance
Error parsing JSON in row:   Action, Crime, Thriller
Error parsing JSON in row:   Crime, Drama, Thriller

Error parsing JSON in row:   Comedy, Drama, Romance
Error parsing JSON in row:   Action, Romance, Thriller
Error parsing JSON in row:   Comedy, Drama
Error parsing JSON in row:   Action, Drama, Romance, Thriller
Error parsing JSON in row:   Comedy, Drama, Romance
Error parsing JSON in row:   Action, Crime, Drama, Thriller
Error parsing JSON in row:   Comedy, Crime
Error parsing JSON in row:   Crime, Drama
Error parsing JSON in row:   Adventure, Crime, Drama, Romance
Error parsing JSON in row:   Comedy, Drama, Romance
Error parsing JSON in row:   Drama, Family
Error parsing JSON in row:   Comedy, Drama, Romance
Error parsing JSON in row: History, Romance, War
Error parsing JSON in row: Drama, Musical, Romance
Error parsing JSON in row: Crime, Drama, Mystery, Thriller
Error parsing JSON in row: Drama, Sport
Error parsing JSON in row: Comedy, Drama, Romance
Error parsing JSON in row: Action, Drama, History, War
Error parsing JSON in row: Action, Comedy, Crime, Thriller
Error parsing JSON in row: Action, Adventure, Sci-Fi, Thriller
Error parsing JSON in row: Comedy, Drama, Romance
Error parsing JSON in row: Drama, Romance, Sport
Error parsing JSON in row: Comedy, Drama
Error parsing JSON in row: Drama, Romance
Error parsing JSON in row: Action, Drama, Thriller
Error parsing JSON in row: Action, Crime, Drama, Thriller
Error parsing JSON in row: Drama, Romance
Error parsing JSON in row: Drama
```

```
Error parsing JSON in row: Comedy, Drama
Error parsing JSON in row: Crime, Drama, Thriller
Error parsing JSON in row: Action, Drama, Thriller
Error parsing JSON in row: Biography, Drama, Thriller
Error parsing JSON in row: Crime, Drama, Thriller
Error parsing JSON in row: Comedy, Drama
Error parsing JSON in row: Drama, History, Romance
Error parsing JSON in row: Action, Crime, Drama, Thriller
Error parsing JSON in row: Comedy, Drama, Sci-Fi
Error parsing JSON in row: Comedy, Drama
Error parsing JSON in row: Comedy, Drama, Romance
Error parsing JSON in row: Comedy, Drama
Error parsing JSON in row: Drama, History
Error parsing JSON in row: Action, Drama, Thriller
Error parsing JSON in row: Drama, Music, Romance
Error parsing JSON in row: Action, Comedy, Crime
Error parsing JSON in row: Drama, Romance
Error parsing JSON in row: Action, Crime, Thriller
Error parsing JSON in row: Comedy, Romance
Error parsing JSON in row: Crime, Drama, Thriller
Error parsing JSON in row: Drama
Error parsing JSON in row: Drama
Error parsing JSON in row: Comedy, Drama, Romance
Error parsing JSON in row: Action, Comedy
Error parsing JSON in row: Drama, Romance
Error parsing JSON in row: Action, Drama, Thriller
Error parsing JSON in row: Drama, Romance
Error parsing JSON in row: Comedy
Error parsing JSON in row: Drama, Thriller
Error parsing JSON in row:   Crime, Drama
Error parsing JSON in row:   Horror, Fantasy
Error parsing JSON in row:   Action, Mystery, Thriller
Error parsing JSON in row:   Drama
Error parsing JSON in row:   Action, War
Error parsing JSON in row:   Comedy, Romance
Error parsing JSON in row:   Romance, Drama
Error parsing JSON in row:   Action, Crime, Drama
Error parsing JSON in row:   Comedy, Drama, Romance
Error parsing JSON in row:   Crime, Thriller
Error parsing JSON in row:   Comedy, Drama
Error parsing JSON in row:   Action, Adventure, Thriller

                       Movie Genre  Rating  Year Box Office
Performance  \
0                   3 Idiots  None     8.4  2009              Super
Hit
1                      Anand  None     8.3  1971
Hit
2   Baahubali: The Beginning  None     8.3  2015
```

```
Blockbuster
3          Bajrangi Bhaijaan  None     8.0  2015
Blockbuster
4                      Black  None     8.2  2005
Hit
..                       ...   ...     ...   ...                     ..
.
75                Virumaandi  None     8.1  2004
Flop
76               Wake Up Sid  None     7.6  2009
Hit
77                     Wazir  None     7.0  2016
Flop
78                   Welcome  None     6.4  2007
Blockbuster
79                   Pathaan  None     8.2  2023
Blockbuster

   Box Office Collection
0              439 crore
1              1.5 crore
2              650 crore
3              969 crore
4             24.9 crore
..                   ...
75              40 crore
76              55 crore
77            52.5 crore
78          1.35 billion
79          1.05 billion

[80 rows x 6 columns]
```

```python
# Assuming Movies is our existing DataFrame
import pandas as pd
import json
def parse_genre(x):
    try:
        return list(json.loads(x).values())
    except (json.JSONDecodeError, TypeError):
        return None  # Handle parsing errors by returning None
movies['Genre'] = movies['Genre'].apply(parse_genre)
# Filter out rows with None values in the 'Genre' column
movies_new = movies.dropna(subset=['Genre'])
print(movies_new)
```

```
Empty DataFrame
Columns: [Movie, Genre, Rating, Year, Box Office Performance, Box
Office Collection]
Index: []
```

```
# Remember Use Your Data In Place Of Movie & Genre that is in your csv file
x = movies_new['Movie']
y = movies_new['Genre']
print("Input Feature (x):")
print(x.head())
print("\nTarget Variable (y):")
print(y.head())
```

```
Input Feature (x):
Series([], Name: Movie, dtype: object)

Target Variable (y):
Series([], Name: Genre, dtype: object)
```

```
# Split data into train and validation sets
import pandas as pd
from sklearn.model_selection import train_test_split

data = pd.read_csv('movies.csv')

x = data[['Movie', 'Rating', 'Year', 'Box Office Performance', 'Box Office Collection']]
y = data['Genre']

xtrain, xval, ytrain, yval = train_test_split(x, y, test_size=0.2, random_state=42)

# Print the shapes of the train and validation sets
print("Shape of xtrain:", xtrain.shape)
print("Shape of xval:", xval.shape)
print("Shape of ytrain:", ytrain.shape)
print("Shape of yval:", yval.shape)
```

```
Shape of xtrain: (64, 5)
Shape of xval: (16, 5)
Shape of ytrain: (64,)
Shape of yval: (16,)
```

```
# Create TF-IDF features
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf_vectorizer = TfidfVectorizer(max_df=0.8, max_features=10000)
xtrain_tfidf = tfidf_vectorizer.fit_transform(xtrain)
xval_tfidf = tfidf_vectorizer.transform(xval)

print("Shape of xtrain_tfidf:", xtrain_tfidf.shape)
print("Shape of xval_tfidf:", xval_tfidf.shape)

print("TF-IDF matrix for xtrain:")
print(xtrain_tfidf.toarray())
```

```
Shape of xtrain_tfidf: (5, 7)
Shape of xval_tfidf: (5, 7)
TF-IDF matrix for xtrain:
[[0.         0.         1.         0.         0.         0.
  0.        ]
 [0.         0.         0.         0.         0.         1.
  0.        ]
 [0.         0.         0.         0.         0.         0.
  1.        ]
 [0.53177225 0.         0.         0.53177225 0.659118   0.
  0.        ]
 [0.53177225 0.659118   0.         0.53177225 0.         0.
  0.        ]]
```

```python
# Fit the model on the train set
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression

data = pd.read_csv('movies.csv')

x = data[['Movie', 'Rating', 'Year', 'Box Office Performance', 'Box
Office Collection']]
y = data['Genre']

xtrain, xval, ytrain, yval = train_test_split(x, y, test_size=0.2,
random_state=42)
s
tfidf_vectorizer = TfidfVectorizer(max_df=0.8, max_features=10000)
xtrain_tfidf = tfidf_vectorizer.fit_transform(xtrain['Movie'])
xval_tfidf = tfidf_vectorizer.transform(xval['Movie'])
print("Shape of xtrain_tfidf:", xtrain_tfidf.shape)
print("Shape of xval_tfidf:", xval_tfidf.shape)

clf = LogisticRegression()

clf.fit(xtrain_tfidf, ytrain)

print("Model fitting complete.")

print("TF-IDF matrix for xtrain:")
print(xtrain_tfidf.toarray())

print("Model coefficients:")
print(clf.coef_)
```

```
Shape of xtrain_tfidf: (64, 119)
Shape of xval_tfidf: (16, 119)
Model fitting complete.
```

```
TF-IDF matrix for xtrain:
[[0.         0.         0.         ... 0.         0.         0.
]
 [0.         0.         0.         ... 0.57735027 0.         0.
]
 [0.         0.         0.         ... 0.         0.         0.
]
 ...
 [0.         0.         0.         ... 0.         0.         0.
]
 [0.         0.         0.         ... 0.         0.         0.
]
 [0.         0.         0.         ... 0.         0.         0.
]]
Model coefficients:
[[-0.01042013 -0.00711175 -0.01088444 ... -0.00868449 -0.01504197
  -0.00851848]
 [-0.01059416 -0.00723056 -0.01106616 ... -0.00882951 -0.01529315
  -0.00866075]
 [-0.02076874 -0.01418108 -0.02168186 ... -0.01730474 -0.02997269
  -0.01697823]
 ...
 [-0.0105291  -0.00718614 -0.01099822 ... -0.00877529 -0.01519925
  -0.00860756]
 [-0.01059416 -0.00723056 -0.01106616 ... -0.00882951 -0.01529315
  -0.00866075]
 [-0.01059416 -0.00723056  0.67778238 ... -0.00882951 -0.01529315
  -0.00866075]]

# Make predictions on the validation set
y_pred_prob = clf.predict_proba(xval_tfidf)
t = 0.3  # threshold value
y_pred_new = (y_pred_prob >= t).astype(int)

print("Predicted Probabilities:")
print(y_pred_prob)
print("\nBinary Predictions (Threshold = 0.3):")
print(y_pred_new)

Predicted Probabilities:
[[0.01551928 0.01579293 0.03139041 0.01579293 0.01579293 0.01579293
  0.01579293 0.01555837 0.01579293 0.01579293 0.01579293 0.03139041
  0.07766032 0.01579293 0.01579293 0.03139041 0.01579293 0.01557507
  0.01579293 0.01579293 0.01579293 0.01579293 0.01579293 0.03139041
  0.01579293 0.04531168 0.01579293 0.01545044 0.06237381 0.03139041
  0.01579293 0.01575032 0.01579293 0.04691401 0.04594178 0.01571
  0.01579293 0.01579293 0.07656212 0.01569034 0.01579293 0.01579293]
 [0.01551928 0.01579293 0.03139041 0.01579293 0.01579293 0.01579293
  0.01579293 0.01555837 0.01579293 0.01579293 0.01579293 0.03139041
  0.07766032 0.01579293 0.01579293 0.03139041 0.01579293 0.01557507
```

```
 0.01579293 0.01579293 0.01579293 0.01579293 0.01579293 0.03139041
 0.01579293 0.04531168 0.01579293 0.01545044 0.06237381 0.03139041
 0.01579293 0.01575032 0.01579293 0.04691401 0.04594178 0.01571
 0.01579293 0.01579293 0.07656212 0.01569034 0.01579293 0.01579293]
[0.01519286 0.01547869 0.03053902 0.01547869 0.01547869 0.01547869
 0.01547869 0.01523408 0.01547869 0.01547869 0.01547869 0.03053902
 0.07406909 0.01547869 0.01547869 0.03053902 0.01547869 0.01526672
 0.01547869 0.01547869 0.01547869 0.01547869 0.01547869 0.03053902
 0.01547869 0.04353478 0.01547869 0.01511967 0.05986209 0.03053902
 0.01547869 0.01543724 0.01547869 0.04532431 0.07340882 0.015398
 0.01547869 0.01547869 0.07258968 0.01537888 0.01547869 0.01547869]
[0.0148957  0.01515627 0.02988416 0.01515627 0.01515627 0.01515627
 0.01515627 0.01493292 0.01515627 0.01515627 0.01515627 0.02988416
 0.12162093 0.01515627 0.01515627 0.02988416 0.01515627 0.01494881
 0.01515627 0.01515627 0.01515627 0.01515627 0.01515627 0.02988416
 0.01515627 0.04283757 0.01515627 0.01483013 0.05847781 0.02988416
 0.01515627 0.0151157  0.01515627 0.04431728 0.04341969 0.0150773
 0.01515627 0.01515627 0.07129623 0.01505859 0.01515627 0.01515627]
[0.01551928 0.01579293 0.03139041 0.01579293 0.01579293 0.01579293
 0.01579293 0.01555837 0.01579293 0.01579293 0.01579293 0.03139041
 0.07766032 0.01579293 0.01579293 0.03139041 0.01579293 0.01557507
 0.01579293 0.01579293 0.01579293 0.01579293 0.01579293 0.03139041
 0.01579293 0.04531168 0.01579293 0.01545044 0.06237381 0.03139041
 0.01579293 0.01575032 0.01579293 0.04691401 0.04594178 0.01571
 0.01579293 0.01579293 0.07656212 0.01569034 0.01579293 0.01579293]
[0.01551928 0.01579293 0.03139041 0.01579293 0.01579293 0.01579293
 0.01579293 0.01555837 0.01579293 0.01579293 0.01579293 0.03139041
 0.07766032 0.01579293 0.01579293 0.03139041 0.01579293 0.01557507
 0.01579293 0.01579293 0.01579293 0.01579293 0.01579293 0.03139041
 0.01579293 0.04531168 0.01579293 0.01545044 0.06237381 0.03139041
 0.01579293 0.01575032 0.01579293 0.04691401 0.04594178 0.01571
 0.01579293 0.01579293 0.07656212 0.01569034 0.01579293 0.01579293]
[0.01551928 0.01579293 0.03139041 0.01579293 0.01579293 0.01579293
 0.01579293 0.01555837 0.01579293 0.01579293 0.01579293 0.03139041
 0.07766032 0.01579293 0.01579293 0.03139041 0.01579293 0.01557507
 0.01579293 0.01579293 0.01579293 0.01579293 0.01579293 0.03139041
 0.01579293 0.04531168 0.01579293 0.01545044 0.06237381 0.03139041
 0.01579293 0.01575032 0.01579293 0.04691401 0.04594178 0.01571
 0.01579293 0.01579293 0.07656212 0.01569034 0.01579293 0.01579293]
[0.01551928 0.01579293 0.03139041 0.01579293 0.01579293 0.01579293
 0.01579293 0.01555837 0.01579293 0.01579293 0.01579293 0.03139041
 0.07766032 0.01579293 0.01579293 0.03139041 0.01579293 0.01557507
 0.01579293 0.01579293 0.01579293 0.01579293 0.01579293 0.03139041
 0.01579293 0.04531168 0.01579293 0.01545044 0.06237381 0.03139041
 0.01579293 0.01575032 0.01579293 0.04691401 0.04594178 0.01571
 0.01579293 0.01579293 0.07656212 0.01569034 0.01579293 0.01579293]
[0.01551928 0.01579293 0.03139041 0.01579293 0.01579293 0.01579293
 0.01579293 0.01555837 0.01579293 0.01579293 0.01579293 0.03139041
 0.07766032 0.01579293 0.01579293 0.03139041 0.01579293 0.01557507
```

```
 0.01579293 0.01579293 0.01579293 0.01579293 0.01579293 0.03139041
 0.01579293 0.04531168 0.01579293 0.01545044 0.06237381 0.03139041
 0.01579293 0.01575032 0.01579293 0.04691401 0.04594178 0.01571
 0.01579293 0.01579293 0.07656212 0.01569034 0.01579293 0.01579293]
[0.01551928 0.01579293 0.03139041 0.01579293 0.01579293 0.01579293
 0.01579293 0.01555837 0.01579293 0.01579293 0.01579293 0.03139041
 0.07766032 0.01579293 0.01579293 0.03139041 0.01579293 0.01557507
 0.01579293 0.01579293 0.01579293 0.01579293 0.01579293 0.03139041
 0.01579293 0.04531168 0.01579293 0.01545044 0.06237381 0.03139041
 0.01579293 0.01575032 0.01579293 0.04691401 0.04594178 0.01571
 0.01579293 0.01579293 0.07656212 0.01569034 0.01579293 0.01579293]
[0.01551928 0.01579293 0.03139041 0.01579293 0.01579293 0.01579293
 0.01579293 0.01555837 0.01579293 0.01579293 0.01579293 0.03139041
 0.07766032 0.01579293 0.01579293 0.03139041 0.01579293 0.01557507
 0.01579293 0.01579293 0.01579293 0.01579293 0.01579293 0.03139041
 0.01579293 0.04531168 0.01579293 0.01545044 0.06237381 0.03139041
 0.01579293 0.01575032 0.01579293 0.04691401 0.04594178 0.01571
 0.01579293 0.01579293 0.07656212 0.01569034 0.01579293 0.01579293]
[0.01446174 0.01471319 0.02883394 0.01471319 0.01471319 0.01471319
 0.01471319 0.01449767 0.01471319 0.01471319 0.01471319 0.02883394
 0.06855573 0.01471319 0.01471319 0.02883394 0.01471319 0.01451299
 0.01471319 0.01471319 0.01471319 0.01471319 0.01471319 0.02883394
 0.01471319 0.0411136  0.01471319 0.01439846 0.13942646 0.02883394
 0.01471319 0.01467404 0.01471319 0.04250816 0.04166239 0.01463698
 0.01471319 0.01471319 0.06764668 0.01461893 0.01471319 0.01471319]
[0.01551928 0.01579293 0.03139041 0.01579293 0.01579293 0.01579293
 0.01579293 0.01555837 0.01579293 0.01579293 0.01579293 0.03139041
 0.07766032 0.01579293 0.01579293 0.03139041 0.01579293 0.01557507
 0.01579293 0.01579293 0.01579293 0.01579293 0.01579293 0.03139041
 0.01579293 0.04531168 0.01579293 0.01545044 0.06237381 0.03139041
 0.01579293 0.01575032 0.01579293 0.04691401 0.04594178 0.01571
 0.01579293 0.01579293 0.07656212 0.01569034 0.01579293 0.01579293]
[0.01551928 0.01579293 0.03139041 0.01579293 0.01579293 0.01579293
 0.01579293 0.01555837 0.01579293 0.01579293 0.01579293 0.03139041
 0.07766032 0.01579293 0.01579293 0.03139041 0.01579293 0.01557507
 0.01579293 0.01579293 0.01579293 0.01579293 0.01579293 0.03139041
 0.01579293 0.04531168 0.01579293 0.01545044 0.06237381 0.03139041
 0.01579293 0.01575032 0.01579293 0.04691401 0.04594178 0.01571
 0.01579293 0.01579293 0.07656212 0.01569034 0.01579293 0.01579293]
[0.01551928 0.01579293 0.03139041 0.01579293 0.01579293 0.01579293
 0.01579293 0.01555837 0.01579293 0.01579293 0.01579293 0.03139041
 0.07766032 0.01579293 0.01579293 0.03139041 0.01579293 0.01557507
 0.01579293 0.01579293 0.01579293 0.01579293 0.01579293 0.03139041
 0.01579293 0.04531168 0.01579293 0.01545044 0.06237381 0.03139041
 0.01579293 0.01575032 0.01579293 0.04691401 0.04594178 0.01571
 0.01579293 0.01579293 0.07656212 0.01569034 0.01579293 0.01579293]
[0.01551928 0.01579293 0.03139041 0.01579293 0.01579293 0.01579293
 0.01579293 0.01555837 0.01579293 0.01579293 0.01579293 0.03139041
 0.07766032 0.01579293 0.01579293 0.03139041 0.01579293 0.01557507
```

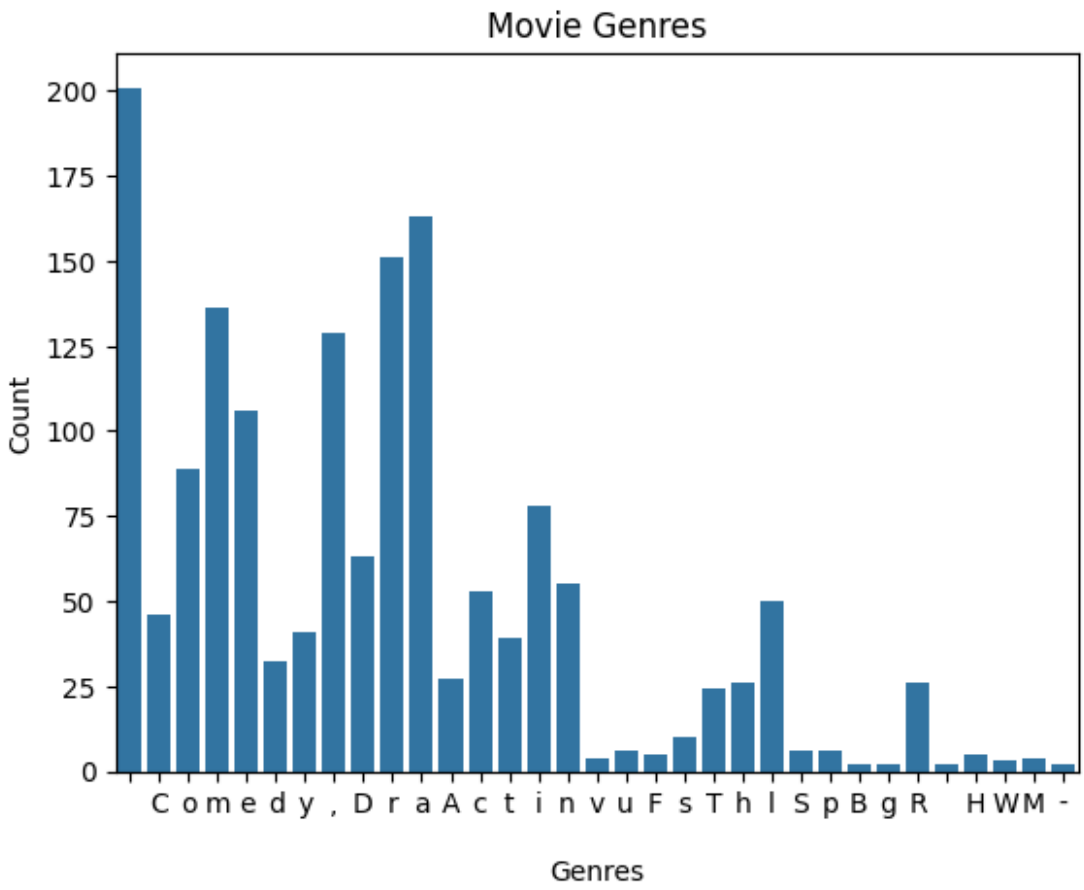
```
  0.01579293 0.01579293 0.01579293 0.01579293 0.01579293 0.03139041
  0.01579293 0.04531168 0.01579293 0.01545044 0.06237381 0.03139041
  0.01579293 0.01575032 0.01579293 0.04691401 0.04594178 0.01571
  0.01579293 0.01579293 0.07656212 0.01569034 0.01579293 0.01579293]]

Binary Predictions (Threshold = 0.3):
[[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0
  0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0
  0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0
  0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0
  0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0
  0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0
  0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0
  0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0
  0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0
  0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0
  0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0
  0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0
  0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0
  0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0
  0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
0 0
  0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0
  0 0 0 0 0 0]]
```

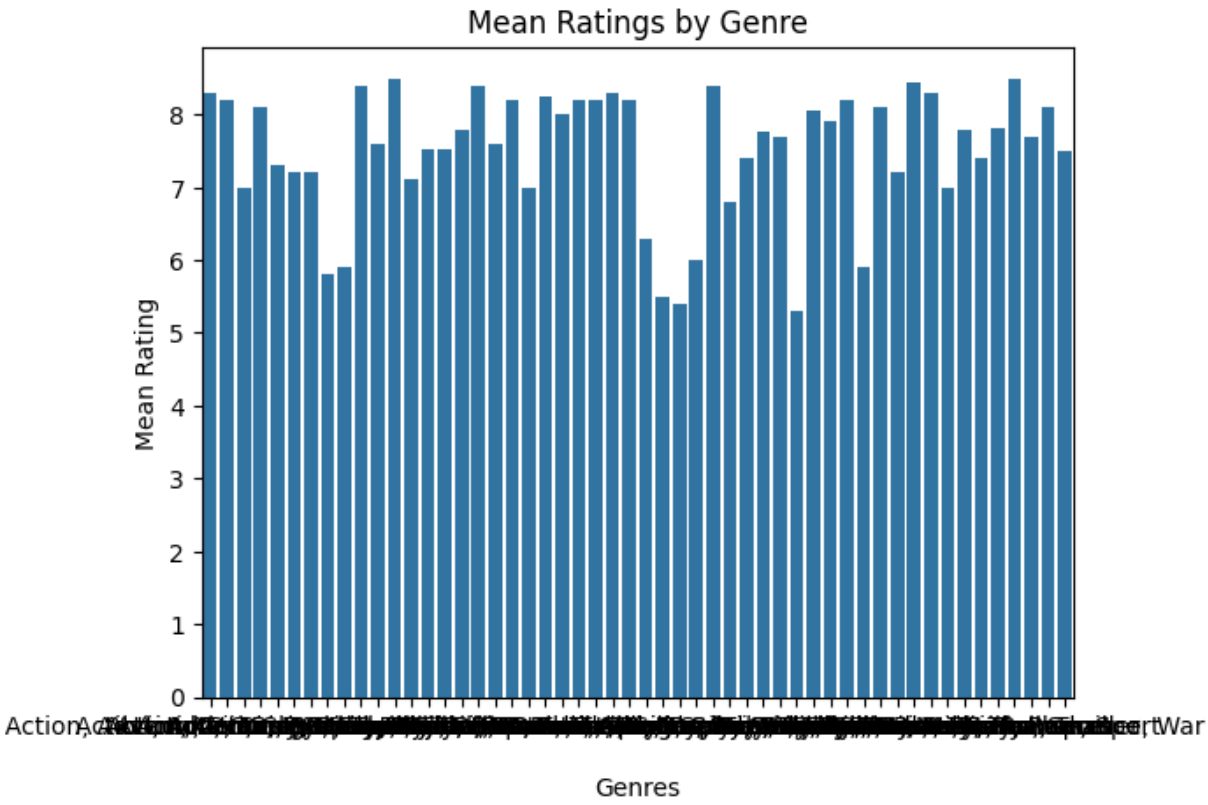
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import f1_score

data = pd.read_csv('movies.csv')

x = data[['Movie', 'Rating', 'Year', 'Box Office Performance', 'Box
Office Collection']]
y = data['Genre']

xtrain, xval, ytrain, yval = train_test_split(x, y, test_size=0.2,
random_state=42)

tfidf_vectorizer = TfidfVectorizer(max_df=0.8, max_features=10000)
xtrain_tfidf = tfidf_vectorizer.fit_transform(xtrain['Movie'])
xval_tfidf = tfidf_vectorizer.transform(xval['Movie'])

y_pred_new = clf.predict(xval_tfidf)
f1 = f1_score(yval, y_pred_new, average="macro")
print("F1 score:", f1)
```

```
F1 score: 0.075
```

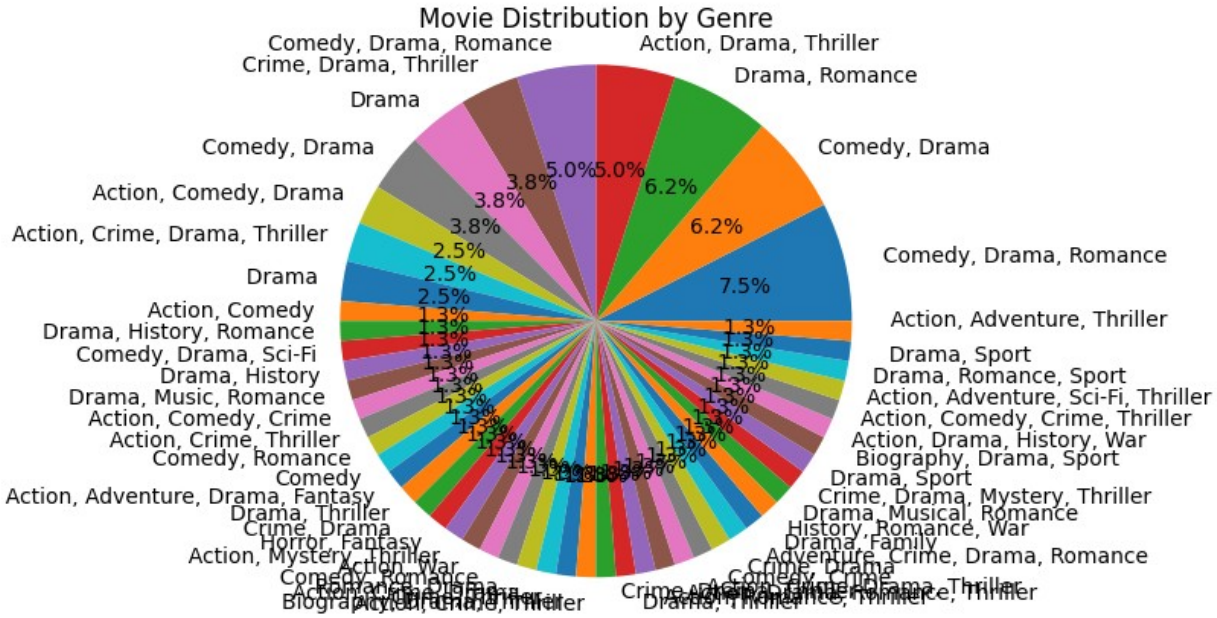
```
import seaborn as sns
import matplotlib.pyplot as plt
data = pd.read_csv('movies.csv')
data_new = data.dropna(subset=['Genre'])
genres = [genre for sublist in data_new['Genre'] for genre in sublist]
sns.countplot(x=genres)
plt.title("Movie Genres")
plt.xlabel("Genres")
plt.ylabel("Count")
plt.show()
```




```
# Calculate the mean rating for each genre
mean_ratings = data_new.explode('Genre').groupby('Genre')
['Rating'].mean()
sns.barplot(x=mean_ratings.index, y=mean_ratings.values)
plt.title("Mean Ratings by Genre")
plt.xlabel("Genres")
plt.ylabel("Mean Rating")
plt.show()
```




```
import matplotlib.pyplot as plt
genre_counts = data_new['Genre'].explode().value_counts()
plt.pie(genre_counts, labels=genre_counts.index, autopct='%1.1f%%')
plt.axis('equal')
plt.title("Movie Distribution by Genre")
plt.show()
```




```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression

def predict_genre(title, description):
    clf = LogisticRegression()
    clf.fit(xtrain_tfidf, ytrain)
    tfidf_vectorizer = TfidfVectorizer(max_df=0.8, max_features=10000)
    tfidf_vectorizer.fit(xtrain['Movie'])

    movie_tfidf = tfidf_vectorizer.transform([title + ' ' +
description])

    # Make predictions on the input movie
    y_pred_prob = clf.predict_proba(movie_tfidf)
    y_pred_new = (y_pred_prob >= 0.3).astype(int)
    genre_pred = clf.predict(movie_tfidf)

    print("Predicted Probabilities:")
    print(y_pred_prob)
    print("\nBinary Predictions (Threshold = 0.3):")
    print(y_pred_new)
    print("\nPredicted Genre:")
    print(genre_pred)

# Example
title = "The Matrix"
description = "A computer hacker learns from mysterious rebels about
the true nature of his reality and his role in the war against its
```

```
controllers."
predict_genre(title, description)
```

```
Predicted Probabilities:
[[0.02121001 0.01477981 0.02818218 0.01477981 0.01477981 0.01477981
  0.01477981 0.02003884 0.01477981 0.01477981 0.01477981 0.02818218
  0.06242863 0.01477981 0.01477981 0.02818218 0.01477981 0.01457634
  0.01477981 0.01477981 0.01477981 0.01477981 0.01477981 0.02818218
  0.01477981 0.074557   0.01477981 0.02363583 0.05190743 0.02818218
  0.01477981 0.01474159 0.01477981 0.04050929 0.05337834 0.01470544
  0.01477981 0.01477981 0.09514955 0.01753527 0.01477981 0.01477981]]

Binary Predictions (Threshold = 0.3):
[[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0
  0 0 0 0 0 0]]

Predicted Genre:
['Drama, Romance']
```