Task-01_MOVIE GENRE CLASSIFICATION

January 3, 2024

[26]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import LabelEncoder

from sklearn.svm import LinearSVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
```

[27]:
```
# Importing the data
train_data = pd.read_csv("train_data.txt",sep=':::', names=['ID', 'TITLE',␣
 ↪'GENRE', 'DESCRIPTION'])
display(train_data.head())
print(train_data.shape)


test_data = pd.read_csv("test_data.txt",sep=':::', names=['ID', 'TITLE',␣
 ↪'GENRE', 'DESCRIPTION'])
print(display(test_data.head()))
print(test_data.shape)

test_solution_data = pd.read_csv("test_data_solution.txt",sep=':::',␣
 ↪names=['ID', 'TITLE', 'GENRE', 'DESCRIPTION'])
print(display(test_solution_data.head()))
print(test_solution_data.shape)
```

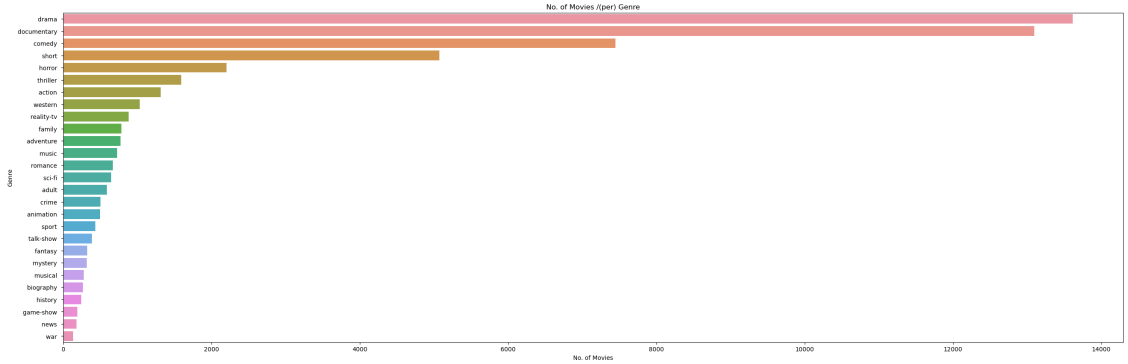
```
C:\Users\COMPUTER GARDEN\AppData\Local\Temp\ipykernel_16900\4082559861.py:2:
ParserWarning: Falling back to the 'python' engine because the 'c' engine does
not support regex separators (separators > 1 char and different from '\s+' are
interpreted as regex); you can avoid this warning by specifying engine='python'.
  train_data = pd.read_csv("train_data.txt",sep=':::', names=['ID', 'TITLE',
'GENRE', 'DESCRIPTION'])
```

	ID	TITLE	GENRE	\
0	1	Oscar et la dame rose (2009)	drama	
1	2	Cupid (1997)	thriller	
2	3	Young, Wild and Wonderful (1980)	adult	

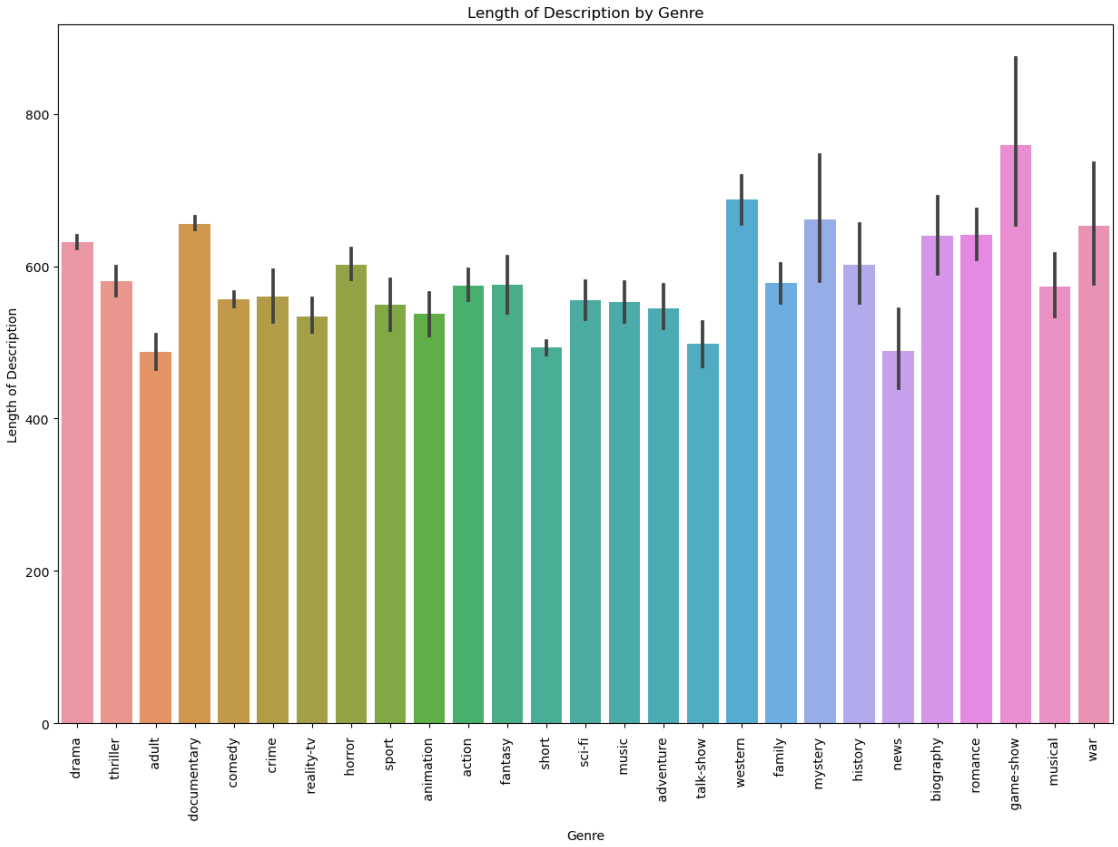
```
3   4              The Secret Sin (1915)       drama
4   5             The Unrecovered (2007)       drama

                                        DESCRIPTION
0   Listening in to a conversation between his do…
1   A brother and sister with a past incestuous r…
2   As the bus empties the students for their fie…
3   To help their unemployed father make ends mee…
4   The film's title refers not only to the un-re…

(54214, 4)

C:\Users\COMPUTER GARDEN\AppData\Local\Temp\ipykernel_16900\4082559861.py:7:
ParserWarning: Falling back to the 'python' engine because the 'c' engine does
not support regex separators (separators > 1 char and different from '\s+' are
interpreted as regex); you can avoid this warning by specifying engine='python'.
  test_data = pd.read_csv("test_data.txt",sep=':::', names=['ID', 'TITLE',
'GENRE', 'DESCRIPTION'])

   ID                          TITLE  \
0   1          Edgar's Lunch (1998)
1   2      La guerra de papá (1977)
2   3   Off the Beaten Track (2010)
3   4        Meu Amigo Hindu (2015)
4   5             Er nu zhai (1955)

                                             GENRE  DESCRIPTION
0   L.R. Brane loves his life - his car, his apar…          NaN
1   Spain, March 1964: Quico is a very naughty ch…          NaN
2   One year in the life of Albin and his family …          NaN
3   His father has died, he hasn't spoken with hi…          NaN
4   Before he was known internationally as a mart…          NaN

None
(54200, 4)

C:\Users\COMPUTER GARDEN\AppData\Local\Temp\ipykernel_16900\4082559861.py:11:
ParserWarning: Falling back to the 'python' engine because the 'c' engine does
not support regex separators (separators > 1 char and different from '\s+' are
interpreted as regex); you can avoid this warning by specifying engine='python'.
  test_solution_data = pd.read_csv("test_data_solution.txt",sep=':::',
names=['ID', 'TITLE', 'GENRE', 'DESCRIPTION'])

   ID                          TITLE          GENRE  \
0   1          Edgar's Lunch (1998)       thriller
1   2      La guerra de papá (1977)         comedy
2   3   Off the Beaten Track (2010)    documentary
3   4        Meu Amigo Hindu (2015)          drama
4   5             Er nu zhai (1955)          drama

                                        DESCRIPTION
```

```
0   L.R. Brane loves his life - his car, his apar…
1   Spain, March 1964: Quico is a very naughty ch…
2   One year in the life of Albin and his family …
3   His father has died, he hasn't spoken with hi…
4   Before he was known internationally as a mart…

None
(54200, 4)
```

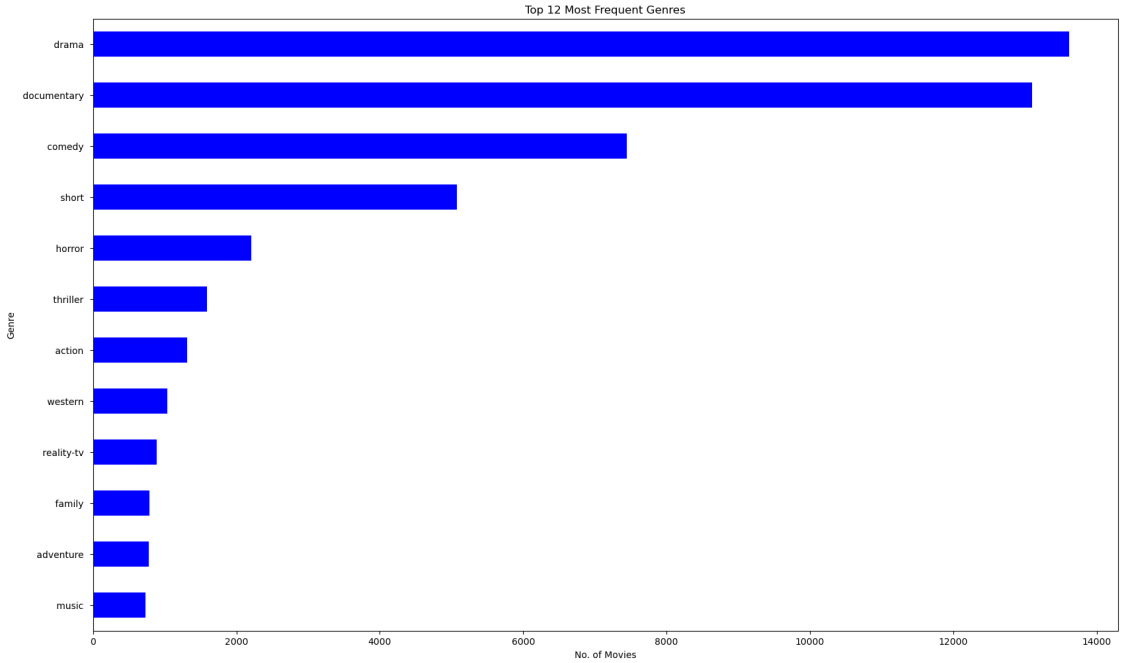
[28]:
```
# Performing some data visualization method
plt.figure(figsize=(32,10))
sns.countplot(y=train_data['GENRE'], order = train_data['GENRE'].value_counts().
  ↪index)
plt.title('No. of Movies /(per) Genre')
plt.xlabel('No. of Movies')
plt.ylabel('Genre')
plt.show()
```



[29]:
```
# Bar Representation of Genre vs Description length
train_data['Length of Description'] = train_data['DESCRIPTION'].apply(len)
plt.figure(figsize=(15, 10))
sns.barplot(x='GENRE', y='Length of Description', data=train_data)
plt.title('Length of Description by Genre')
plt.xticks(rotation=90)
plt.xlabel('Genre')
plt.ylabel('Length of Description')
plt.show()
```



[30]:
```
# Top genre which mostly people watched
top_genres = train_data['GENRE'].value_counts().head(12)

plt.figure(figsize=(20, 12))
top_genres.plot(kind='barh', color='blue')
plt.title('Top 12 Most Frequent Genres')
plt.xlabel('No. of Movies')
plt.ylabel('Genre')
plt.gca().invert_yaxis()  # Invert y-axis to have the genre with the most␣
 ↪movies at the top
plt.show()
```



[31]:
```
# Training and Testing of the data

# Handle any potential missing values
train_data['DESCRIPTION'].fillna("", inplace=True)
test_data['DESCRIPTION'].fillna("", inplace=True)

t_v = TfidfVectorizer(stop_words='english', max_features=100000)
X_train = t_v.fit_transform(train_data['DESCRIPTION'])
X_test = t_v.transform(test_data['DESCRIPTION'])

label_encoder = LabelEncoder()
y_train = label_encoder.fit_transform(train_data['GENRE'])
y_test = label_encoder.transform(test_solution_data['GENRE'])
```

[32]:
```
X_train_sub, X_val, y_train_sub, y_val = train_test_split(X_train, y_train,␣
 ↪test_size=0.2, random_state=42)

clf = LinearSVC()
clf.fit(X_train_sub, y_train_sub)

y_val_pred = clf.predict(X_val)
print("Validation Accuracy:", accuracy_score(y_val, y_val_pred))
print("Validation Classification Report:\n", classification_report(y_val,␣
 ↪y_val_pred))
```

```
Validation Accuracy: 0.5836945494789265
```

```
Validation Classification Report:
              precision    recall  f1-score   support

           0       0.44      0.32      0.37       263
           1       0.74      0.44      0.55       112
           2       0.45      0.21      0.28       139
           3       0.47      0.15      0.23       104
           4       0.00      0.00      0.00        61
           5       0.53      0.59      0.56      1443
           6       0.39      0.07      0.11       107
           7       0.69      0.81      0.75      2659
           8       0.56      0.72      0.63      2697
           9       0.36      0.17      0.23       150
          10       0.13      0.03      0.04        74
          11       0.82      0.68      0.74        40
          12       0.00      0.00      0.00        45
          13       0.65      0.66      0.66       431
          14       0.61      0.53      0.57       144
          15       0.25      0.04      0.07        50
          16       0.43      0.05      0.10        56
          17       0.20      0.06      0.09        34
          18       0.49      0.25      0.33       192
          19       0.36      0.06      0.10       151
          20       0.50      0.28      0.36       143
          21       0.44      0.36      0.40      1045
          22       0.60      0.41      0.49        93
          23       0.62      0.25      0.35        81
          24       0.30      0.16      0.21       309
          25       0.50      0.05      0.09        20
          26       0.85      0.83      0.84       200

    accuracy                           0.58     10843
   macro avg       0.46      0.30      0.34     10843
weighted avg       0.56      0.58      0.56     10843
```

[33]:
```
y_pred = clf.predict(X_test)
print("Test Accuracy:", accuracy_score(y_test, y_pred))
print("Test Classification Report:\n", classification_report(y_test, y_pred))
```

```
Test Accuracy: 0.09357933579335793
Test Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00      1314
           1       0.00      0.00      0.00       590
           2       0.00      0.00      0.00       775
           3       0.00      0.00      0.00       498
```

```
           4       0.00      0.00      0.00       264
           5       0.00      0.00      0.00      7446
           6       0.00      0.00      0.00       505
           7       0.00      0.00      0.00     13096
           8       0.00      0.00      0.00     13612
           9       0.00      0.00      0.00       783
          10       0.00      0.00      0.00       322
          11       0.00      0.00      0.00       193
          12       0.00      0.00      0.00       243
          13       0.00      0.00      0.00      2204
          14       0.00      0.00      0.00       731
          15       0.00      0.00      0.00       276
          16       0.00      0.00      0.00       318
          17       0.00      0.00      0.00       181
          18       0.00      0.00      0.00       883
          19       0.00      0.00      0.00       672
          20       0.00      0.00      0.00       646
          21       0.09      1.00      0.17      5072
          22       0.00      0.00      0.00       431
          23       0.00      0.00      0.00       391
          24       0.00      0.00      0.00      1590
          25       0.00      0.00      0.00       132
          26       0.00      0.00      0.00      1032

    accuracy                           0.09     54200
   macro avg       0.00      0.04      0.01     54200
weighted avg       0.01      0.09      0.02     54200
```

```
C:\Users\COMPUTER GARDEN\anaconda3\lib\site-
packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\COMPUTER GARDEN\anaconda3\lib\site-
packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\COMPUTER GARDEN\anaconda3\lib\site-
packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

[34]:

```python
from sklearn.naive_bayes import MultinomialNB
Mnb_classifier = MultinomialNB()
Mnb_classifier.fit(X_train, y_train)
```

[34]:
```
MultinomialNB()
```

[35]:
```
Mnb_classifier.predict(X_test)
```

[35]:
```
array([8, 8, 8, …, 8, 8, 8])
```

[38]:
```
from sklearn.linear_model import LogisticRegression
lr_classifier = LogisticRegression(max_iter=432)
lr_classifier.fit(X_train, y_train)
```

[38]:
```
LogisticRegression(max_iter=432)
```

[39]:
```
lr_classifier.predict(X_test)
```

[39]:
```
array([8, 8, 8, …, 8, 8, 8])
```

[40]:
```
def predict_movie(description):
    t_v1 = t_v.transform([description])
    pred_label = clf.predict(t_v1)
    return label_encoder.inverse_transform(pred_label)[0]

sample_descr_for_movie = "Edouard is a pianist, married with Caroline. This␣
 ↪evening, they are invited to Claude's. Claude is the snobbish uncle of␣
 ↪Caroline, his son Alain (as snobbish as his father) is in love with Caroline.
 ↪ They spite Edouard a little because he's poor. At the party, Edouard must␣
 ↪play the piano to make himself known by Claude's important acquaintances.␣
 ↪But just before leaving, Edouard and Caroline quarrel about clothes, and␣
 ↪Edouard goes alone..."
print(predict_movie(sample_descr_for_movie))

sample_descr_for_movie = "A movie where police cashes the criminal and shoot␣
 ↪him"
print(predict_movie(sample_descr_for_movie))

sample_descr_for_movie = "This is a horror trilogy about an antique shop where␣
 ↪the pieces of the furniture each have their own stories from the past. The␣
 ↪new owners get their little piece of horror, says Joey Reyes. In Kama, Aleli␣
 ↪purchases an antique bed, not realizing that its previous owner of the bed␣
 ↪is out to claim her son. Aparador is about a couple who purchased an antique␣
 ↪cabinet where a young man died inside before. In Tokador, an independent␣
 ↪woman gets a possessed dresser out to take her soul."
print(predict_movie(sample_descr_for_movie))

sample_descr_for_movie1 = "A movie where person cashes a girl too get marry␣
 ↪with him but girl refuses him."
print(predict_movie(sample_descr_for_movie1))
```

```
sample_descr_for_movie = "A busy motel becomes a fantasy ground for its␣
 ↪occupants. At first, porn producer Paul gives an impromptu casting couch to␣
 ↪a chambermaid. Newlywed Vicky wants her husband Harold to spend their␣
 ↪honeymoon just in bed. Shy cowboy Beau hires his first escort Sally Thompson.
 ↪ Young runaway lovers Reggie and Catherine unite in a store room. Friends␣
 ↪Nick and Ritchie hire a hooker. Nick wins her company via a coin toss, but␣
 ↪she gets him to invite Ritchie along. Buxom aunt Mrs. Robinson gets her␣
 ↪younger nephew in his sleep, but it's later revealed it was a hired act.␣
 ↪Porn fans Alex and Jeff invite porn stars Karen and Charmel to their room.␣
 ↪After Karen and Charmel enjoy each other, Alex gets Charmel and Jeff gets␣
 ↪Karen."
print(predict_movie(sample_descr_for_movie))
```

```
comedy
action
horror
drama
adult
```