```
Code Way Internship Task 1

Name :- Sampurna Dey

Task Name :- predict the genre of a
movie
```

Code Way Internship Task 1

Name :- Sampurna Dey

Task Name :- predict the genre of a movie

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
train_data = pd.read_csv('/content/train_data.txt',sep=':::', names =['ID','TITLE','GENRE','DESCRIPTION'])
display(train_data.head())
train_data.shape
```

```
<ipython-input-2-4a73d4479885>:1: ParserWarning: Falling back to the 'python' engine because the 'c' engine does not support regex s
  train_data = pd.read_csv('/content/train_data.txt',sep=':::', names =['ID','TITLE','GENRE','DESCRIPTION'])
```

	ID	TITLE	GENRE	DESCRIPTION
0	1	Oscar et la dame rose (2009)	drama	Listening in to a conversation between his do...
1	2	Cupid (1997)	thriller	A brother and sister with a past incestuous r...
2	3	Young, Wild and Wonderful (1980)	adult	As the bus empties the students for their fie...
3	4	The Secret Sin (1915)	drama	To help their unemployed father make ends mee...
4	5	The Unrecovered (2007)	drama	The film's title refers not only to the un-re...

```
(3178, 4)
```

```
test_data = pd.read_csv('/content/test_data.txt',sep=':::', names=['ID','TITLE','GENRE','DESCRIPTION'])
display(test_data.head())
test_data.shape
```

```
<ipython-input-3-cfe35ffed055>:1: ParserWarning: Falling back to the 'python' engine because the 'c' engine does not support regex s
  test_data = pd.read_csv('/content/test_data.txt',sep=':::', names=['ID','TITLE','GENRE','DESCRIPTION'])
```

	ID	TITLE	GENRE	DESCRIPTION
0	1	Edgar's Lunch (1998)	L.R. Brane loves his life - his car, his apar...	NaN
1	2	La guerra de papá (1977)	Spain, March 1964: Quico is a very naughty ch...	NaN
2	3	Off the Beaten Track (2010)	One year in the life of Albin and his family ...	NaN
3	4	Meu Amigo Hindu (2015)	His father has died, he hasn't spoken with hi...	NaN
4	5	Er nu zhai (1955)	Before he was known internationally as a mart...	NaN

```
(9822, 4)
```

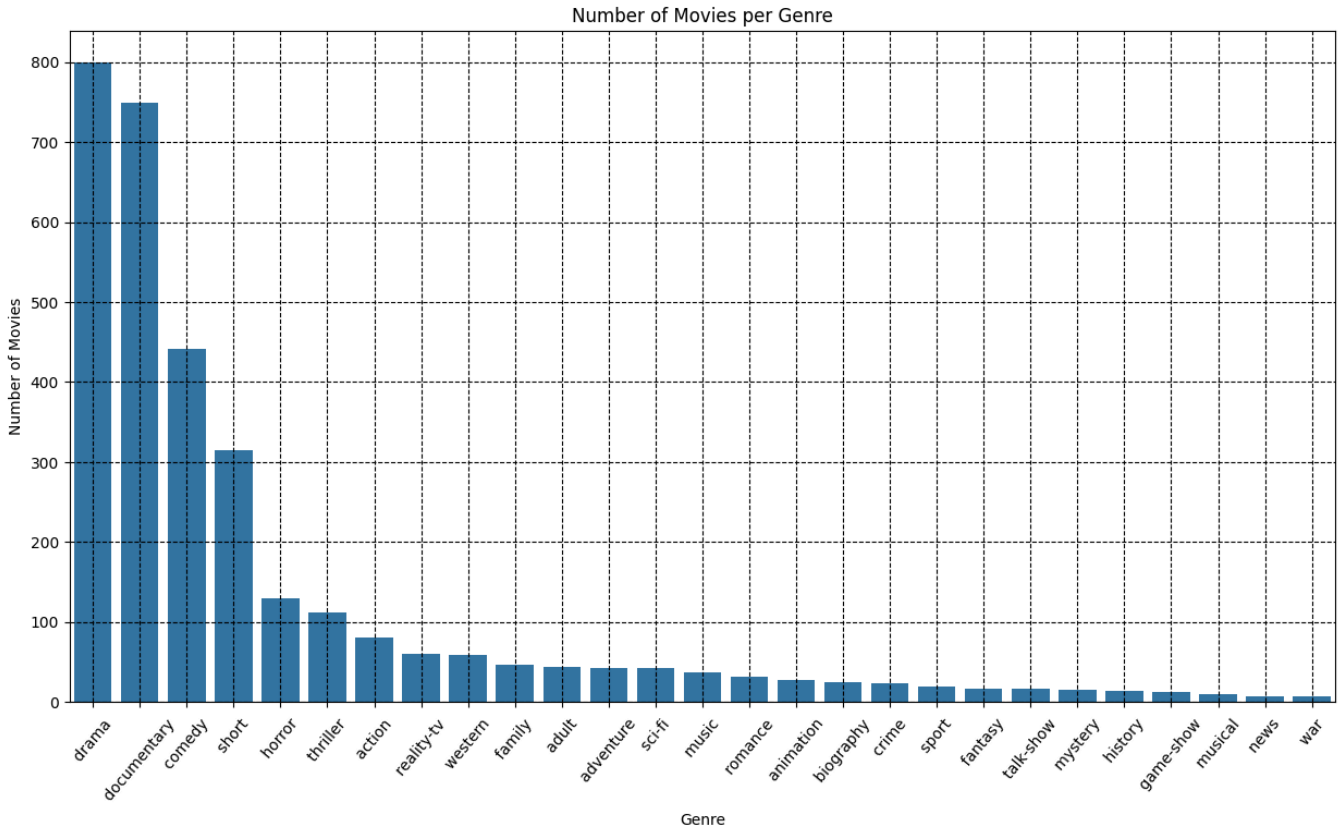
```
test_solution_data = pd.read_csv('/content/test_data_solution.txt',sep=':::', names=['ID','TITLE','GENRE','DESCRIPTION'])
display(test_solution_data.head())
test_solution_data.shape
```

```
<ipython-input-4-018f6a31a9ab>:1: ParserWarning: Falling back to the 'python' engine because the 'c' engine does not support regex s
  test_solution_data = pd.read_csv('/content/test_data_solution.txt',sep=':::', names=['ID','TITLE','GENRE','DESCRIPTION'])
```

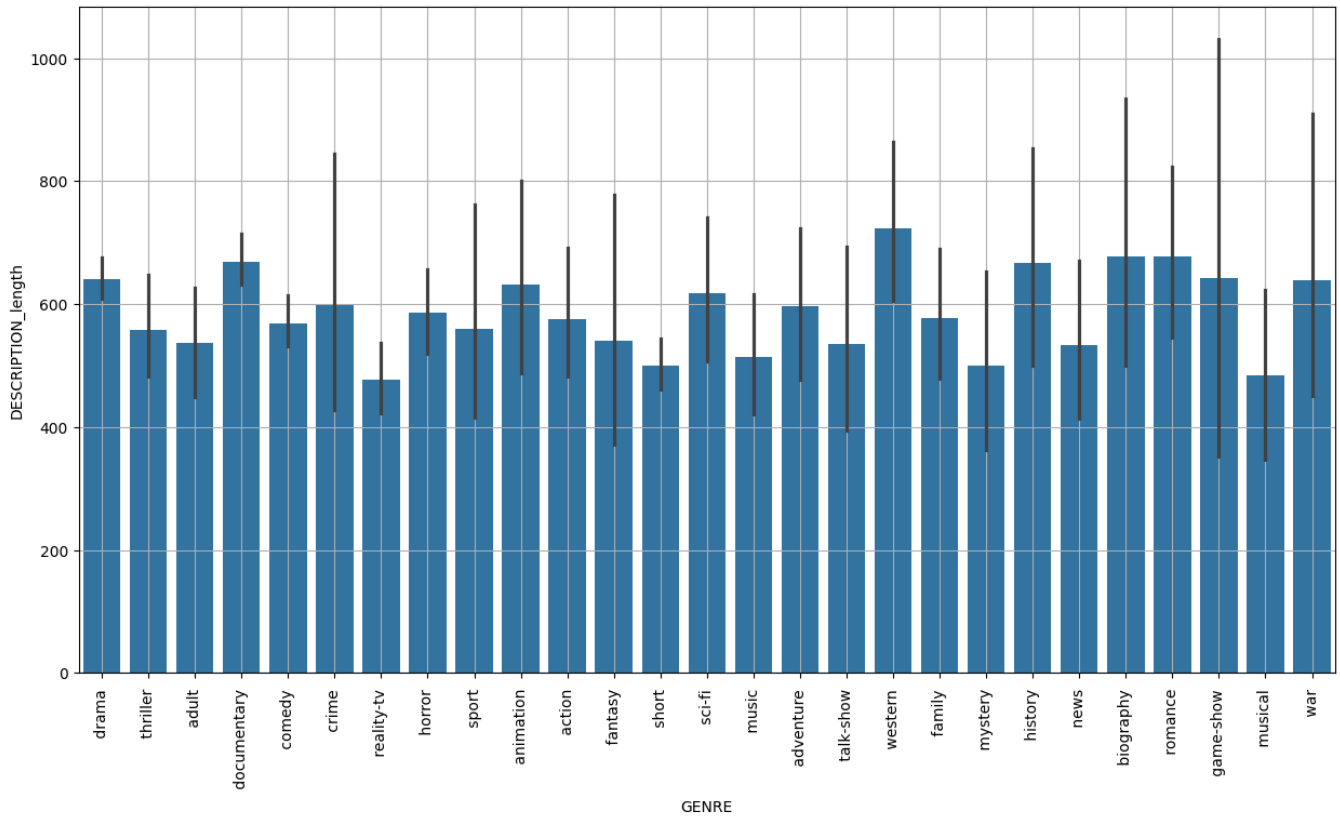
	ID	TITLE	GENRE	DESCRIPTION
0	1	Edgar's Lunch (1998)	thriller	L.R. Brane loves his life - his car, his apar...
1	2	La guerra de papá (1977)	comedy	Spain, March 1964: Quico is a very naughty ch...
2	3	Off the Beaten Track (2010)	documentary	One year in the life of Albin and his family ...
3	4	Meu Amigo Hindu (2015)	drama	His father has died, he hasn't spoken with hi...
4	5	Er nu zhai (1955)	drama	Before he was known internationally as a mart...

```
(3202, 4)
```

```
plt.figure(figsize=(15,8))
sns.countplot(x = train_data['GENRE'], order = train_data['GENRE'].value_counts().index)
plt.xticks(rotation=50)
plt.grid(linestyle='--',color='black')
plt.title('Number of Movies per Genre')
plt.ylabel('Number of Movies')
plt.xlabel('Genre')
plt.show()
```



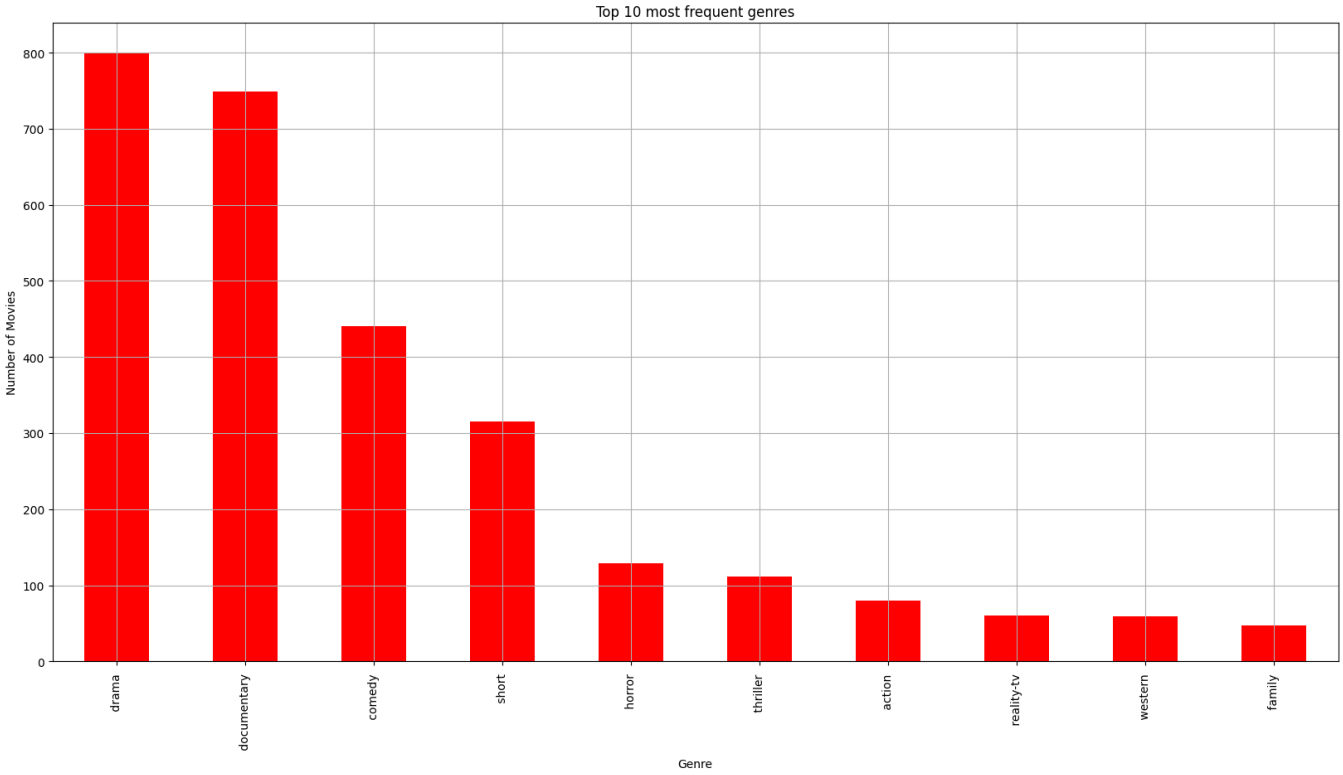

```
train_data['DESCRIPTION_length'] = train_data['DESCRIPTION'].apply(len)
plt.figure(figsize=(15,8))
sns.barplot(x='GENRE',y='DESCRIPTION_length', data=train_data)
plt.xticks(rotation=90)
plt.grid()
plt.show()
```




```
top_genre = train_data['GENRE'].value_counts().head(10)
top_genre
```

```
drama           799
documentary     749
comedy          441
short           315
horror          129
thriller        112
action           80
reality-tv       60
western          59
family           47
Name: GENRE, dtype: int64
```

```
plt.figure(figsize=(20,10))
top_genre.plot(kind='bar', color='red')
plt.title('Top 10 most frequent genres')
plt.grid()
plt.ylabel('Number of Movies')
plt.xlabel('Genre')
plt.show()
```




```
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,confusion_matrix
```

```
train_data['DESCRIPTION'].fillna("", inplace=True)
test_data['DESCRIPTION'].fillna("", inplace=True)
```

```
train_data.isnull().sum()
```

```
ID                  0
TITLE               0
GENRE               0
DESCRIPTION         0
DESCRIPTION_length  0
dtype: int64
```

```
test_data.isnull().sum()
```

```
ID             0
TITLE          0
GENRE          0
DESCRIPTION    0
dtype: int64
```

```
tvf = TfidfVectorizer(stop_words='english', max_features=100000)
x_train =tvf.fit_transform(train_data['DESCRIPTION'])
x_test =tvf.transform(test_data['DESCRIPTION'])
```

```
x_train
```

```
<3178x26830 sparse matrix of type '<class 'numpy.float64'>'
        with 145577 stored elements in Compressed Sparse Row format>
```

```
x_test
```

```
<9822x26830 sparse matrix of type '<class 'numpy.float64'>'
        with 0 stored elements in Compressed Sparse Row format>
```

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

```
y_train
```

```
array([ 8, 24,  1, ...,  7,  7, 26])
```

```
y_train
```

```
array([ 8, 24,  1, ...,  7,  7, 26])
```

```
x_train_sub, x_Val, y_train_sub, y_Val = train_test_split(x_train,y_train,
test_size=0.2,random_state=111)
print(x_train_sub.shape)
print(x_Val.shape)
print(y_train_sub.shape)
print(y_Val.shape)
```

```
(2542, 26830)
(636, 26830)
(2542,)
(636,)
```

```
clf = LinearSVC()
clf.fit(x_train_sub,y_train_sub)
y_val_predict = clf.predict(x_Val)
```

```
print('Validation Accuracy:',accuracy_score(y_Val,y_val_predict))
print('validation classification report : \n',classification_report(y_Val,y_val_predict))
```

```
Validation Accuracy: 0.5094339622641509
validation classification report :
               precision    recall  f1-score   support

           0       0.00      0.00      0.00         8
           1       0.00      0.00      0.00        13
           2       1.00      0.12      0.22         8
           3       0.00      0.00      0.00         7
           4       0.00      0.00      0.00         4
           5       0.43      0.41      0.42        87
           6       0.00      0.00      0.00         5
           7       0.61      0.87      0.72       153
           8       0.48      0.77      0.59       165
           9       1.00      0.08      0.15        12
          10       0.00      0.00      0.00         4
          11       0.00      0.00      0.00         2
          12       0.00      0.00      0.00         2
          13       0.67      0.30      0.41        27
          14       0.00      0.00      0.00         7
          15       0.00      0.00      0.00         1
          16       0.00      0.00      0.00         2
          17       0.00      0.00      0.00         4
          18       0.50      0.10      0.17        10
          19       0.00      0.00      0.00        11
          20       0.00      0.00      0.00         6
          21       0.32      0.21      0.26        57
          22       0.00      0.00      0.00         6
          23       0.00      0.00      0.00         1
          24       0.00      0.00      0.00        24
          25       0.00      0.00      0.00         2
          26       0.83      0.62      0.71         8

    accuracy                           0.51       636
   macro avg       0.22      0.13      0.14       636
weighted avg       0.44      0.51      0.44       636

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are i
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are i
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are i
  _warn_prf(average, modifier, msg_start, len(result))
```

```
y_pred = clf.predict(x_test)
print('test classification report : \n', classification_report(y_test,y_test))
```

```
test classification report :
                precision    recall  f1-score   support

             0       1.00      1.00      1.00        88
             1       1.00      1.00      1.00        32
             2       1.00      1.00      1.00        48
             3       1.00      1.00      1.00        42
             4       1.00      1.00      1.00        15
             5       1.00      1.00      1.00       428
             6       1.00      1.00      1.00        25
             7       1.00      1.00      1.00       734
             8       1.00      1.00      1.00       842
             9       1.00      1.00      1.00        33
            10       1.00      1.00      1.00        11
            11       1.00      1.00      1.00         6
            12       1.00      1.00      1.00        19
            13       1.00      1.00      1.00       134
            14       1.00      1.00      1.00        47
            15       1.00      1.00      1.00        10
            16       1.00      1.00      1.00        13
            17       1.00      1.00      1.00         8
            18       1.00      1.00      1.00        46
            19       1.00      1.00      1.00        48
            20       1.00      1.00      1.00        37
            21       1.00      1.00      1.00       323
            22       1.00      1.00      1.00        28
            23       1.00      1.00      1.00        18
            24       1.00      1.00      1.00        92
            25       1.00      1.00      1.00        10
            26       1.00      1.00      1.00        65

      accuracy                           1.00      3202
     macro avg       1.00      1.00      1.00      3202
  weighted avg       1.00      1.00      1.00      3202
```

```
from sklearn.naive_bayes import MultinomialNB
```