task-1

February 6, 2024

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```
train_data = pd.read_csv('train_data.txt',sep=':::', names␣
 ↪=['ID','TITLE','GENRE','DESCRIPTION'])
display(train_data.head())
train_data.shape
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_10276\61894218.py:1: 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…
```

[2]:
```
(54214, 4)
```

[3]:
```
test_data = pd.read_csv('test_data.txt',sep=':::', names␣
 ↪=['ID','TITLE','GENRE','DESCRIPTION'])
display(test_data.head())
test_data.shape
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_10276\3327045876.py:1:
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
```

```
[3]: (54200, 4)
```

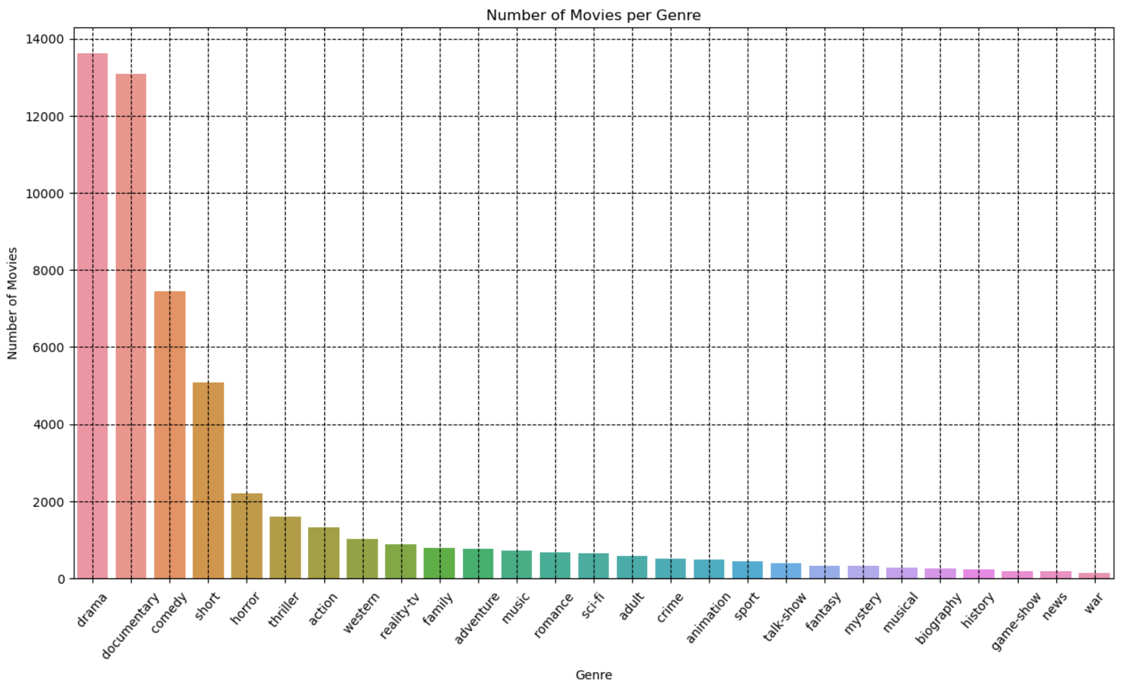
```
[4]: test_solution_data = pd.read_csv('test_data_solution.txt',sep=':::', names␣
      ↪=['ID','TITLE','GENRE','DESCRIPTION'])
     display(test_solution_data.head())
     test_solution_data.shape
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_10276\3836370518.py:1:
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'])
```

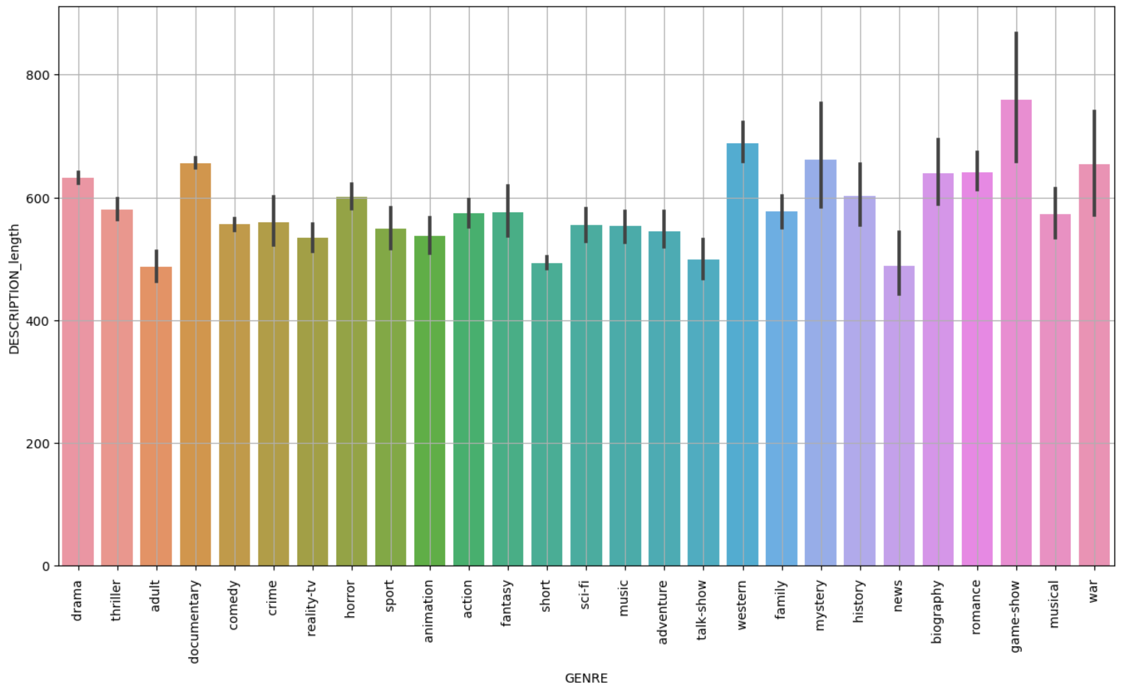
```
   ID                          TITLE          GENRE  \
0   1           Edgar's Lunch (1998)       thriller
1   2       La guerra de papá (1977)         comedy
2   3    Off the Beaten Track (2010)    documentary
3   4         Meu Amigo Hindu (2015)          drama
4   5              Er nu zhai (1955)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

[4]: (54200, 4)

[5]:
```
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()
```



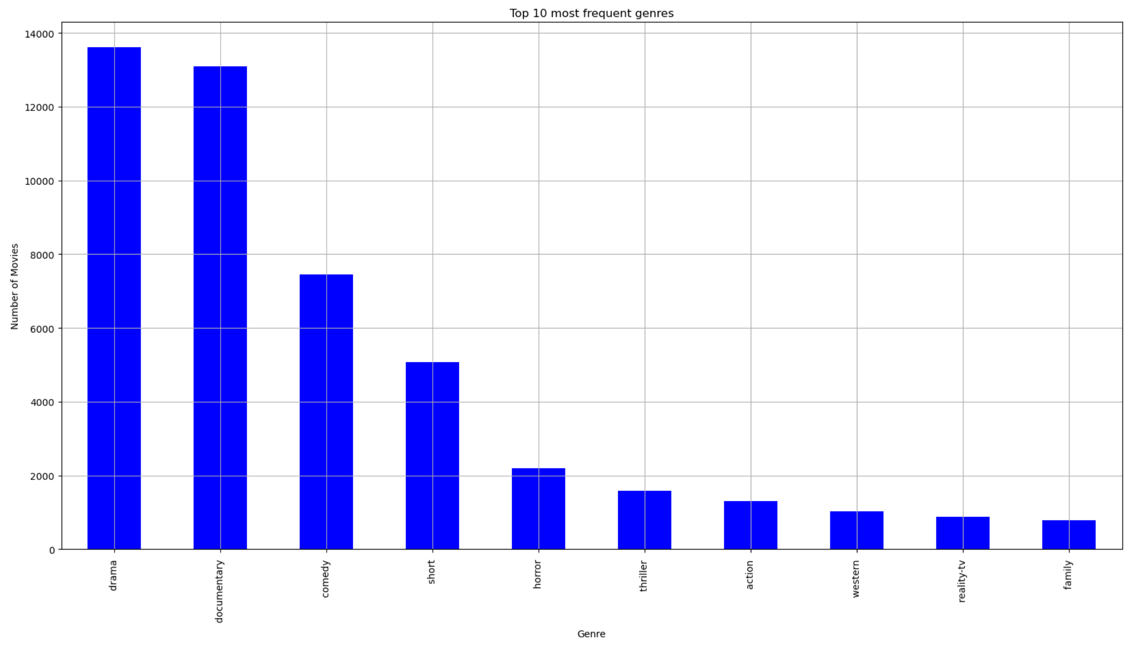
[6]:
```
train_data['DESCRIPTION_length'] = train_data['DESCRIPTION'].apply(len)
plt.figure(figsize=(15,8))
sns.barplot(x='GENRE',y='DESCRIPTION_length', data=train_data)
plt.xticks(rotation=90)
plt.grid()
plt.show()
```



[7]:
```
top_genre = train_data['GENRE'].value_counts().head(10)
top_genre
```

[7]:
```
 drama           13613
 documentary     13096
 comedy           7447
 short            5073
 horror           2204
 thriller         1591
 action           1315
 western          1032
 reality-tv        884
 family            784
Name: GENRE, dtype: int64
```

[8]:
```
plt.figure(figsize=(20,10))
top_genre.plot(kind='bar', color='blue')
plt.title('Top 10 most frequent genres')
plt.grid()
plt.ylabel('Number of Movies')
plt.xlabel('Genre')
plt.show()
```



```
[9]: 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
```

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

```
[11]: train_data.isnull().sum()
```

```
[11]: ID                    0
      TITLE                 0
      GENRE                 0
      DESCRIPTION           0
      DESCRIPTION_length    0
      dtype: int64
```

```
[12]: test_data.isnull().sum()
```

```
[12]: ID             0
      TITLE          0
      GENRE          0
      DESCRIPTION    0
      dtype: int64
```

[13]:
```
tvf  = TfidfVectorizer(stop_words='english', max_features=100000)
x_train =tvf.fit_transform(train_data['DESCRIPTION'])
x_test =tvf.transform(test_data['DESCRIPTION'])
```

[14]:
```
x_train
```

[14]:
```
<54214x100000 sparse matrix of type '<class 'numpy.float64'>'
        with 2442599 stored elements in Compressed Sparse Row format>
```

[15]:
```
x_test
```

[15]:
```
<54200x100000 sparse matrix of type '<class 'numpy.float64'>'
        with 0 stored elements in Compressed Sparse Row format>
```

[16]:
```
label = LabelEncoder()
y_train = label.fit_transform(train_data['GENRE'])
y_test = label.transform(test_solution_data['GENRE'])
```

[17]:
```
y_train
```

[17]:
```
array([ 8, 24,  1, …,  7,  5, 12])
```

[18]:
```
y_test
```

[18]:
```
array([24,  5,  7, …,  1,  8,  8])
```

[19]:
```
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)
```

```
(43371, 100000)
(10843, 100000)
(43371,)
(10843,)
```

[20]:
```
clf = LinearSVC()
clf.fit(x_train_sub,y_train_sub)

y_val_predict = clf.predict(x_Val)
```

```
C:\Users\Admin\anaconda3\Lib\site-packages\sklearn\svm\_classes.py:32:
FutureWarning: The default value of `dual` will change from `True` to `'auto'`
in 1.5. Set the value of `dual` explicitly to suppress the warning.
  warnings.warn(
[21]: print('Validation Accuracy:',accuracy_score(y_Val,y_val_predict))
      print('validation classification report : \n',␣
       ↪classification_report(y_Val,y_val_predict))
```

```
Validation Accuracy: 0.5819422669003044
validation classification report :
               precision    recall  f1-score   support

           0       0.43      0.32      0.36       258
           1       0.74      0.39      0.51       127
           2       0.42      0.19      0.26       162
           3       0.65      0.17      0.27       101
           4       0.00      0.00      0.00        54
           5       0.53      0.57      0.55      1499
           6       0.15      0.03      0.05       108
           7       0.69      0.84      0.76      2649
           8       0.55      0.71      0.62      2693
           9       0.39      0.11      0.18       149
          10       0.07      0.02      0.03        54
          11       0.93      0.66      0.77        38
          12       0.00      0.00      0.00        50
          13       0.62      0.65      0.63       463
          14       0.63      0.46      0.53       138
          15       0.00      0.00      0.00        48
          16       0.20      0.01      0.03        70
          17       0.38      0.15      0.21        34
          18       0.51      0.26      0.34       169
          19       0.22      0.04      0.07       128
          20       0.51      0.30      0.37       132
          21       0.43      0.35      0.38       993
          22       0.63      0.42      0.51        73
          23       0.56      0.29      0.39        78
          24       0.34      0.20      0.25       324
          25       0.67      0.06      0.11        32
          26       0.84      0.84      0.84       219

    accuracy                           0.58     10843
   macro avg       0.45      0.30      0.33     10843
weighted avg       0.55      0.58      0.55     10843


C:\Users\Admin\anaconda3\Lib\site-
packages\sklearn\metrics\_classification.py:1471: 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\Admin\anaconda3\Lib\site-
packages\sklearn\metrics\_classification.py:1471: 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\Admin\anaconda3\Lib\site-
packages\sklearn\metrics\_classification.py:1471: 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))
```

[22]:
```
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_test))
```

```
test Accuracy: 0.09357933579335793
test classification report :
               precision    recall  f1-score   support

           0       1.00      1.00      1.00      1314
           1       1.00      1.00      1.00       590
           2       1.00      1.00      1.00       775
           3       1.00      1.00      1.00       498
           4       1.00      1.00      1.00       264
           5       1.00      1.00      1.00      7446
           6       1.00      1.00      1.00       505
           7       1.00      1.00      1.00     13096
           8       1.00      1.00      1.00     13612
           9       1.00      1.00      1.00       783
          10       1.00      1.00      1.00       322
          11       1.00      1.00      1.00       193
          12       1.00      1.00      1.00       243
          13       1.00      1.00      1.00      2204
          14       1.00      1.00      1.00       731
          15       1.00      1.00      1.00       276
          16       1.00      1.00      1.00       318
          17       1.00      1.00      1.00       181
          18       1.00      1.00      1.00       883
          19       1.00      1.00      1.00       672
          20       1.00      1.00      1.00       646
          21       1.00      1.00      1.00      5072
          22       1.00      1.00      1.00       431
          23       1.00      1.00      1.00       391
          24       1.00      1.00      1.00      1590
          25       1.00      1.00      1.00       132
          26       1.00      1.00      1.00      1032

    accuracy                           1.00     54200
   macro avg       1.00      1.00      1.00     54200
weighted avg       1.00      1.00      1.00     54200
```

[23]:
```
from sklearn.naive_bayes  import MultinomialNB
mnb = MultinomialNB()
mnb.fit(x_train,y_train)
```

[23]:
```
MultinomialNB()
```

[24]:
```
mnb.predict(x_test)
```

[24]:
```
array([8, 8, 8, …, 8, 8, 8])
```

[25]:
```
from sklearn.linear_model import LogisticRegression
lr= LogisticRegression()
lr.fit(x_train, y_train)
```

```
C:\Users\Admin\anaconda3\Lib\site-
packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

[25]:
```
LogisticRegression()
```

[26]:
```
y_pred = lr.predict(x_test)
```

[27]:
```
print('Accuracy score:', accuracy_score(y_test,y_pred))
```

```
Accuracy score: 0.2511439114391144
```

[ ]:
```
```