```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import re
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

```
train_data_path="train_data.txt"
train_data = pd.read_csv("train_data.txt", header=None, sep=":::", names=["ID", "Title", "Genres","Descriptic
train_data.head()
```

	ID	Title	Genres	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...

Next steps: View recommended plots

```
train_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 54214 entries, 0 to 54213
Data columns (total 4 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   ID           54214 non-null  int64 
 1   Title        54214 non-null  object
 2   Genres       54214 non-null  object
 3   Description  54214 non-null  object
dtypes: int64(1), object(3)
memory usage: 1.7+ MB
```

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

```
ID             0
Title          0
Genres         0
Description    0
dtype: int64
```

```
test_path = "test_data.txt"

test_data = pd.read_csv(test_path, sep=":::", names=["ID", "Title","Description"], engine="python")

test_data.head()
```

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

Next steps: View recommended plots

```
test_data.info()
```

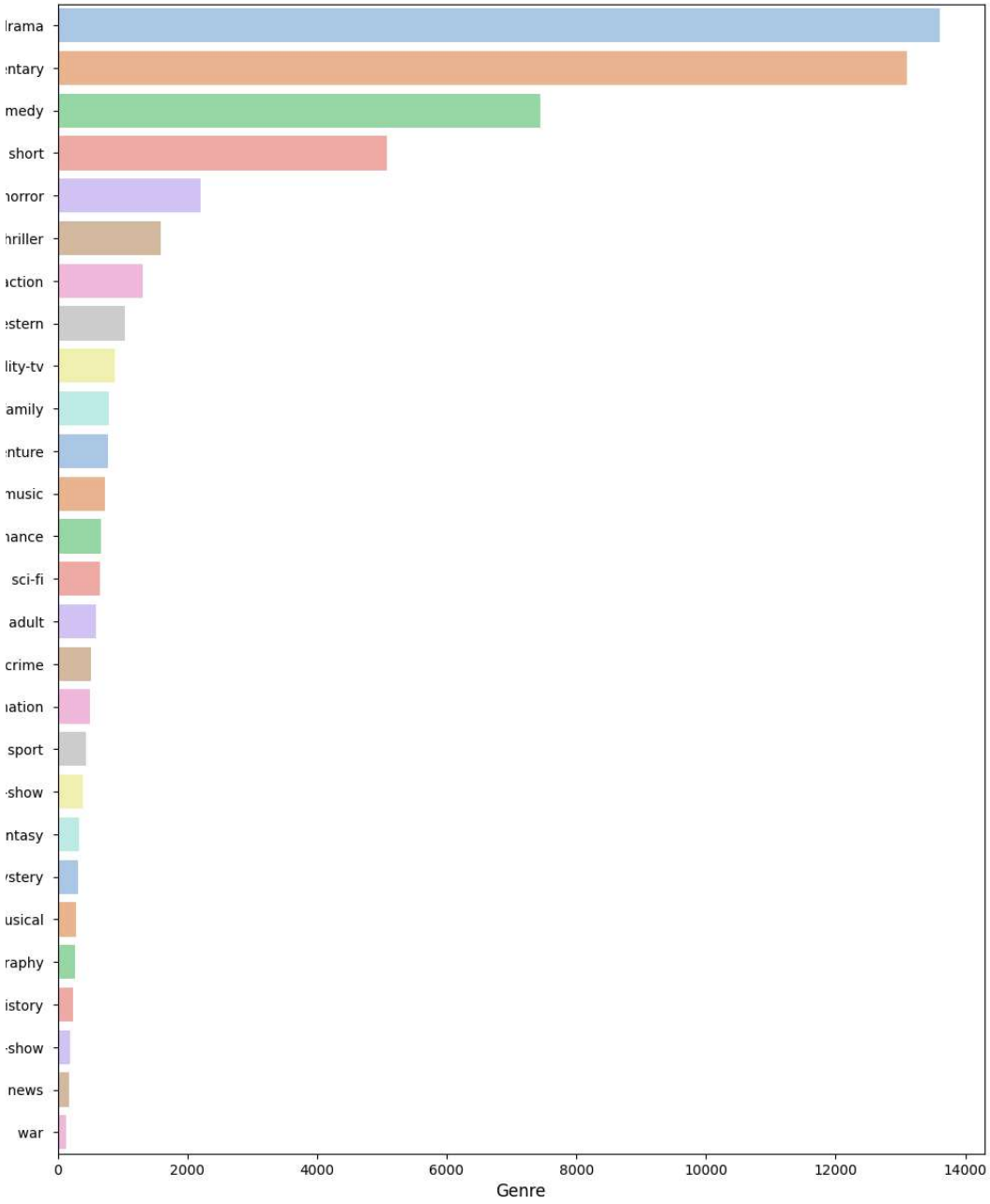
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 54200 entries, 0 to 54199
Data columns (total 3 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   ID           54200 non-null  int64 
 1   Title        54200 non-null  object
 2   Description  54200 non-null  object
dtypes: int64(1), object(2)
memory usage: 1.2+ MB
```

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

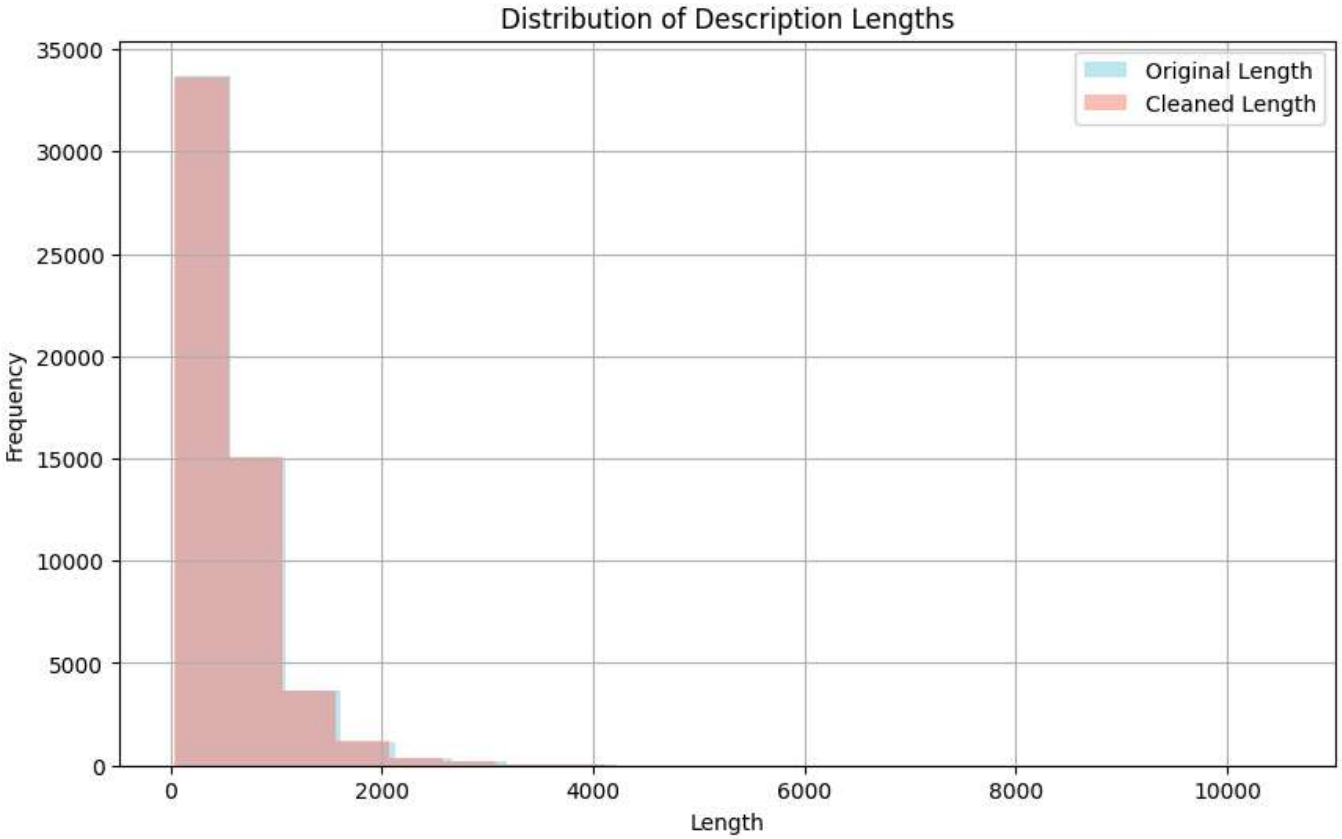
```
ID             0
Title          0
Description    0
dtype: int64
```

```
# Function to clean description
def clean_description(text):
    # Remove special characters, punctuation, and extra whitespaces
    text = re.sub(r'[^\w\s]', '', text)  # Remove special characters and punctuation
    text = re.sub(r'\s+', ' ', text)  # Remove extra whitespaces
    text = re.sub(r"\s+", " ", text).strip()  # Replace multiple spaces with a single space

    return text

# Apply cleaning function to Description column
train_data['Clean_Description'] = train_data['Description'].apply(clean_description)
test_data['Clean_Description']=test_data['Description'].apply(clean_description)


# Plot genre counts
palette = sns.color_palette("pastel")

plt.figure(figsize=(12, 15))
sns.countplot(data=train_data, y="Genres", order=train_data["Genres"].value_counts().index, palette=palette
plt.xlabel('Genre', fontsize=12)
plt.ylabel('Count', fontsize=12)
plt.xticks(fontsize=10)
plt.show()
```

```
ɪ-input-9-e5c69e6448c0>:5: FutureWarning:

 `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign

ɔuntplot(data=train_data, y="Genres", order=train_data["Genres"].value_counts().index, p
ɪ-input-9-e5c69e6448c0>:5: UserWarning:
ɛtte list has fewer values (10) than needed (27) and will cycle, which may produce an un
ɔuntplot(data=train_data, y="Genres", order=train_data["Genres"].value_counts().index, p
```



```
# Calculate length of original and cleaned descriptions
train_data['Original_Length'] = train_data['Description'].apply(len)
train_data['Cleaned_Length'] = train_data['Clean_Description'].apply(len)

# Plotting
plt.figure(figsize=(10, 6))

# Plot original lengths
plt.hist(train_data['Original_Length'], bins=20, color='skyblue', alpha=0.5, label='Original Length')

# Plot cleaned lengths
plt.hist(train_data['Cleaned_Length'], bins=20, color='salmon', alpha=0.5, label='Cleaned Length')

plt.title('Distribution of Description Lengths')
plt.xlabel('Length')
plt.ylabel('Frequency')
plt.legend()
plt.grid(True)
plt.show()

# Calculate count of removed characters
removed_characters = sum(train_data['Original_Length'] - train_data['Cleaned_Length'])
print("Total characters removed during cleaning:", removed_characters)
```




```
Total characters removed during cleaning: 925669
```

```
# Splitting the data into train and validation sets
X = train_data['Description']
y = train_data['Genres']

X_train, X_val, y_train, y_val = train_test_split(X, y, test_size= 0.2, random_state=123)


# Convert text to numerical features using TF-IDF
vectorize = TfidfVectorizer()

X_train_tfidf  = vectorize.fit_transform(X_train)
X_test_tfidf = vectorize.transform(test_data['Clean_Description'])
X_val_tfidf  = vectorize.transform(X_val)


# Training the SVM classifier
svm_classifier = SVC()
svm_classifier.fit(X_val_tfidf, y_val)
```

▾ SVC

SVC()

```
# Making predictions on the validation set
y_pred_val = svm_classifier.predict(X_val_tfidf)
valAccuracy = accuracy_score(y_val, y_pred_val)
print("Validation Accuracy:", valAccuracy)
```

```
Validation Accuracy: 0.8758646131144517
```