Importing libraries

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import re   # for pattern matching and text manipulation
import string
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer, WordNetLemmatizer
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_extraction.text import CountVectorizer as CV
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

Loading Data

In [2]:
```
directory_path = "C:/Users/Megha Sharma/Desktop/MEGHA/MSC DATA SCIENCE/INTERNSHI

train_file_name = "train_data.txt"
test_file_name = "test_data.txt"

train_file_path = directory_path + train_file_name
test_file_path = directory_path + test_file_name

train_data = pd.read_csv(train_file_path, sep = ':::', names = ["title", "genre"
test_data = pd.read_csv( test_file_path, sep = ':::', names = ["title", "descrip
```

```
C:\Users\Megha Sharma\AppData\Local\Temp\ipykernel_9648\2503500492.py:9: ParserWa
rning: Falling back to the 'python' engine because the 'c' engine does not suppor
t 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_file_path, sep = ':::', names = ["title", "genr
e", "description"])
C:\Users\Megha Sharma\AppData\Local\Temp\ipykernel_9648\2503500492.py:10: ParserW
arning: Falling back to the 'python' engine because the 'c' engine does not suppo
rt 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_file_path, sep = ':::', names = ["title", "descri
ption"])
```

In [3]:
```
train_data.head()
```

Out[3]:

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

In [4]:
```
test_data.head()
```

Out[4]:

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

In [5]:
```
train_data.describe()
```

Out[5]:

	title	genre	description
count	54214	54214	54214
unique	54214	27	54086
top	Oscar et la dame rose (2009)	drama	Grammy - music award of the American academy ...
freq	1	13613	12

In [6]:
```
# Shape of train data
train_data.shape
```

Out[6]:
```
(54214, 3)
```

In [7]:
```
# Shape of test data
test_data.shape
```

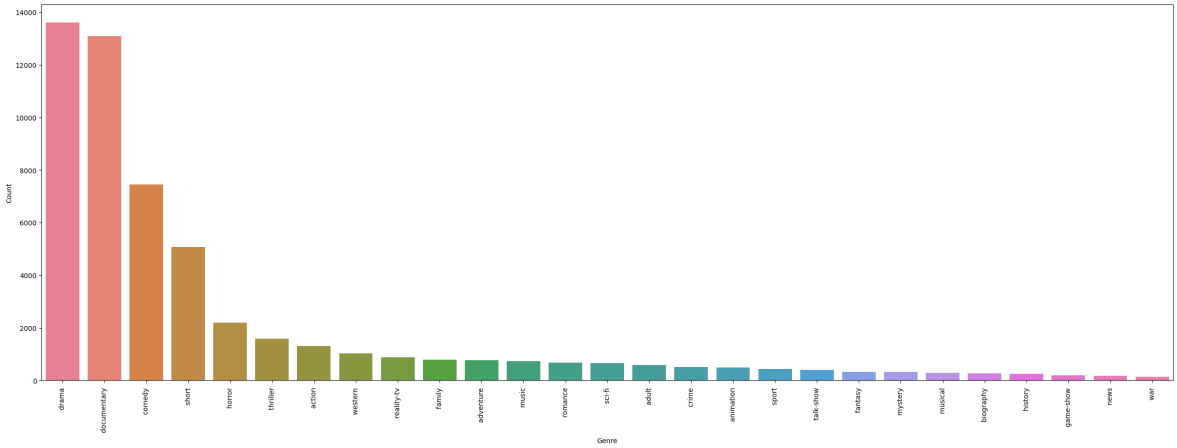
Out[7]:
```
(54200, 2)
```

In [8]:
```
train_data['genre'].value_counts()
```

Out[8]:
```
genre
 drama           13613
 documentary     13096
 comedy           7447
 short            5073
 horror           2204
 thriller         1591
 action           1315
 western          1032
 reality-tv        884
 family            784
 adventure         775
 music             731
 romance           672
 sci-fi            647
 adult             590
 crime             505
 animation         498
 sport             432
 talk-show         391
 fantasy           323
 mystery           319
 musical           277
 biography         265
 history           243
 game-show         194
 news              181
 war               132
Name: count, dtype: int64
```

In [9]:
```
plt.figure(figsize = (30,10))
counts = train_data['genre'].value_counts().sort_values(ascending = False)

custom_palette = sns.color_palette("husl", len(counts))

sns.barplot(x = counts.index, y = counts, palette = custom_palette)

plt.xlabel('Genre')
plt.ylabel('Count')
plt.xticks(rotation = 90)

plt.show()
```

```
C:\Users\Megha Sharma\AppData\Local\Temp\ipykernel_9648\3596272806.py:6: FutureWa
rning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.barplot(x = counts.index, y = counts, palette = custom_palette)
```



A bar plot that shows the distribution of genres in the 'train_data' DataFrame, with each bar represting the count of a specific genre.

In [10]:
```
# Checking for null values
train_data.isnull().sum()
```

Out[10]:
```
title          0
genre          0
description    0
dtype: int64
```

In [11]:
```
test_data.isnull().sum()
```

Out[11]:
```
title          0
description    0
dtype: int64
```

As there are no null values in the datasets so we can proceed further.

Text Cleaning

In [12]:
```
def clean_data(text):
    text = text.lower()  # Lowercase the text
    text = re.sub(r'@\S+|http\S+|[\w\.-]+@[\w\.-]+|pic.\S+|#|_|\[[^]]*\]', '', t
    text = re.sub(r"[^a-zA-Z+']", ' ', text)  # Keep only English characters and
    text = re.sub(r'\s+[a-zA-Z]\s+|\n', ' ', text)  # Remove single characters a
    text = "".join([char for char in text if char not in string.punctuation])  #
    text = re.sub("\s[\s]+", " ", text).strip()  # Remove repeated/leading/trail
    tokens = word_tokenize(text)  # Tokenize the text
    stop_words = set(stopwords.words('english'))  # Initialize stopwords
    text = " ".join([word for word in tokens if word not in stop_words and len(w

    return text
```

The 'clean_data()' function applies a series of text cleaning operations to preprocess the input text data, including removing unwanted characters, URLs, emails, stopwords, etc., to prepare it for further analysis or modeling tasks.

In [13]:
```
train_data['description_cleaned'] = train_data['description'].apply(clean_data)
test_data['description_cleaned'] = test_data['description'].apply(clean_data)
```

Stemming

In [14]:
```
st = PorterStemmer()
train_data['description_cleaned'] = train_data['description_cleaned'].apply(
    lambda x:' '.join([st.stem(word) for word in x.split()]))

test_data['description_cleaned'] = test_data['description_cleaned'].apply(
    lambda x: ' '.join([st.stem(word) for word in x.split()]))
```

In [15]:
```
train_data['length'] = train_data['description'].apply(len)
train_data['length_cleaned'] = train_data['description_cleaned'].apply(len)
train_data
```

Out[15]:

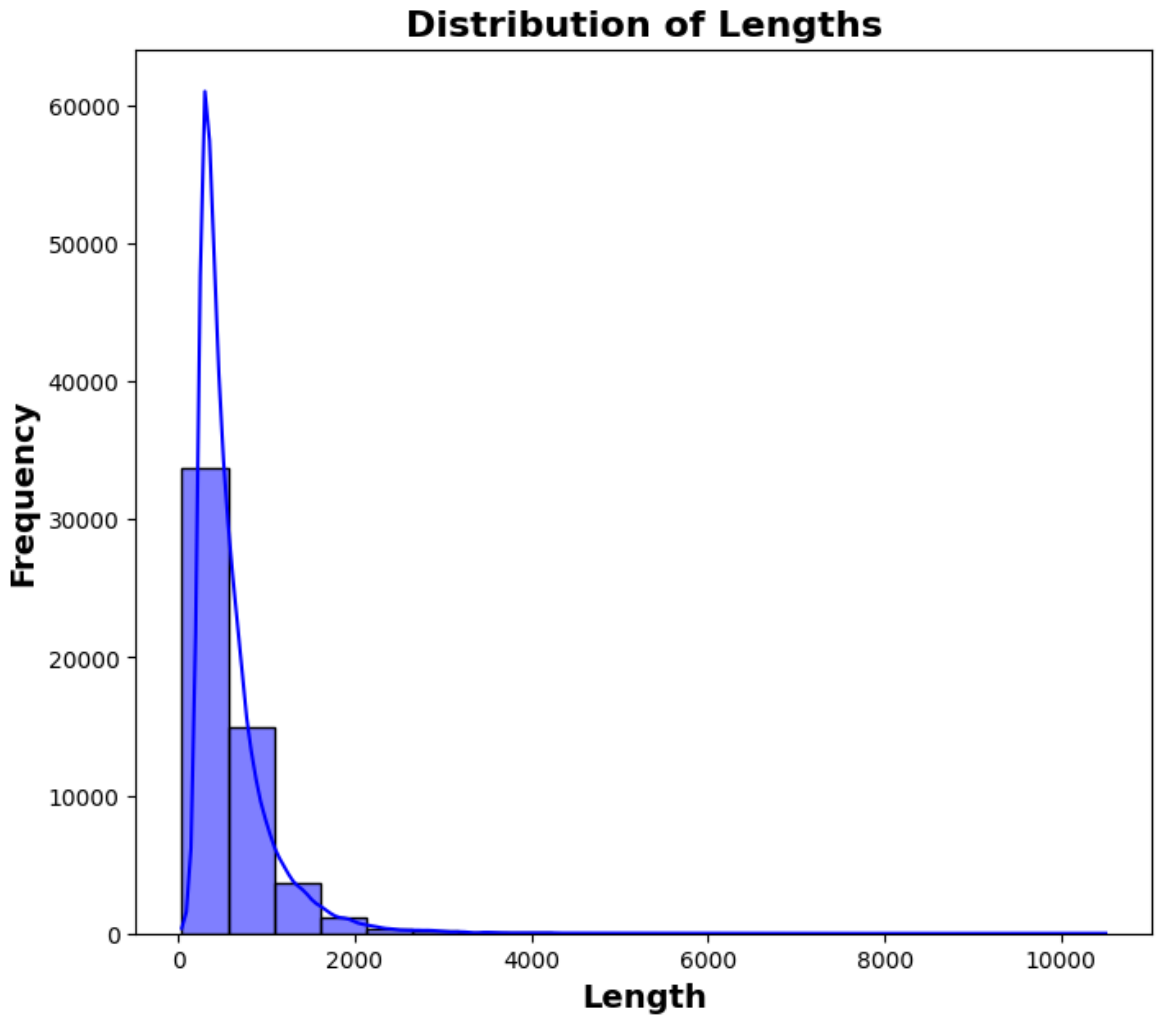
	title	genre	description	description_cleaned	length	length_cl
1	Oscar et la dame rose (2009)	drama	Listening in to a conversation between his do...	listen convers doctor parent year old oscar le...	546	
2	Cupid (1997)	thriller	A brother and sister with a past incestuous r...	brother sister past incestu relationship curre...	184	
3	Young, Wild and Wonderful (1980)	adult	As the bus empties the students for their fie...	bu empti student field trip museum natur histo...	650	
4	The Secret Sin (1915)	drama	To help their unemployed father make ends mee...	help unemploy father make end meet edith twin ...	1082	
5	The Unrecovered (2007)	drama	The film's title refers not only to the un-re...	film titl refer recov bodi ground zero also st...	625	
...	...	...	...	...	...	
54210	"Bonino" (1953)	comedy	This short-lived NBC live sitcom centered on ...	short live nbc live sitcom center bonino world...	507	
54211	Dead Girls Don't Cry (????)	horror	The NEXT Generation of EXPLOITATION. The sist...	next gener exploit sister kapa bay soror hous ...	781	
54212	Ronald Goedemondt: Ze bestaan echt (2008)	documentary	Ze bestaan echt, is a stand-up comedy about g...	bestaan echt stand comedi grow face fear freer...	255	
54213	Make Your Own Bed (1944)	comedy	Walter and Vivian live in the country and hav...	walter vivian live countri difficult time keep...	642	
54214	Nature's Fury: Storm of the Century (2006)	history	On Labor Day Weekend, 1935, the most intense ...	labor day weekend intens hurrican ever make la...	311	

54214 rows × 6 columns

In [16]:
```
print("Average Length of Text Before Cleaning: ", train_data['length'].mean())
print("Average Length of Text After Cleaning: ", train_data['length_cleaned'].me
```

```
Average Length of Text Before Cleaning:  600.4524292618142
Average Length of Text After Cleaning:  360.23698675618846
```

In [17]:
```
plt.figure(figsize = (8,7))

sns.histplot(data = train_data, x = 'length', bins = 20, kde = True, color = 'bl

plt.xlabel('Length', fontsize = 14, fontweight = 'bold')
plt.ylabel('Frequency', fontsize = 14, fontweight = 'bold')
plt.title('Distribution of Lengths', fontsize = 16, fontweight = 'bold')

plt.show()
```



Label Encodeing of the Target variable

In [18]:
```
le = LabelEncoder()
train_data['genre'] = le.fit_transform(train_data['genre'].values)

# keep only relevent columns
train_df = train_data.loc[:,['description_cleaned', 'genre']]
test_df = test_data.loc[:,['description_cleaned', 'title']]
train_df.head(10)
```

Out[18]:

	description_cleaned	genre
1	listen convers doctor parent year old oscar le...	8
2	brother sister past incestu relationship curre...	24
3	bu empti student field trip museum natur histo...	1
4	help unemploy father make end meet edith twin ...	8
5	film titl refer recov bodi ground zero also st...	8
6	qualiti control consist seri singl take shot f...	7
7	tough econom time max joey run idea discov sen...	5
8	ron petri keanu reev troubl teen whose life ha...	6
9	sudden calamit event caus great loss life dama...	18
10	four high school student embark terrifi journe...	13

Train Test Split

In [19]:
```
train_set , val_set , train_label , val_label = train_test_split(train_df['descr

print(f'Split data into train and eval sets')
print(f'Trani Set\t: {len(train_set)}\nValidation Set\t: {len(val_set)}')
```

```
Split data into train and eval sets
Trani Set       : 43371
Validation Set  : 10843
```

Feature Extraction

In [20]:
```
# using TF-IDF
vectorize = TfidfVectorizer(stop_words='english', max_features=100000)
train_set_tfidf = vectorize.fit_transform(train_set)
val_set_tfidf = vectorize.transform(val_set)
```

TF-IDF gets more accurate results in LR

Logistic Regression Model (LR)

In [21]:
```
LR_model = LogisticRegression()
LR_model.fit(train_set_tfidf, train_label)
predict_LR = LR_model.predict(val_set_tfidf)
print(classification_report(val_label, predict_LR))
LR_accuracy = accuracy_score(predict_LR,val_label)
print('Logistic Regression accuracy is: {:.2f}%'.format(LR_accuracy*100))
```

	precision	recall	f1-score	support
0	0.53	0.28	0.36	263
1	0.83	0.26	0.39	112
2	0.40	0.13	0.20	139
3	0.38	0.03	0.05	104
4	0.00	0.00	0.00	61
5	0.52	0.58	0.55	1443
6	0.20	0.02	0.03	107
7	0.66	0.84	0.74	2659
8	0.53	0.79	0.64	2697
9	0.42	0.09	0.15	150
10	0.00	0.00	0.00	74
11	0.94	0.42	0.59	40
12	0.00	0.00	0.00	45
13	0.66	0.57	0.61	431
14	0.65	0.47	0.55	144
15	0.00	0.00	0.00	50
16	0.00	0.00	0.00	56
17	0.00	0.00	0.00	34
18	0.51	0.14	0.21	192
19	0.50	0.01	0.01	151
20	0.64	0.21	0.32	143
21	0.50	0.30	0.37	1045
22	0.61	0.25	0.35	93
23	0.60	0.15	0.24	81
24	0.37	0.13	0.19	309
25	0.00	0.00	0.00	20
26	0.95	0.69	0.80	200
accuracy			0.58	10843
macro avg	0.42	0.24	0.27	10843
weighted avg	0.55	0.58	0.53	10843

```
Logistic Regression accuracy is: 57.90%
```

```
C:\Users\Megha Sharma\AppData\Roaming\Python\Python39\site-packages\sklearn\linea
r_model\_logistic.py:469: 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(
C:\Users\Megha Sharma\AppData\Roaming\Python\Python39\site-packages\sklearn\metri
cs\_classification.py:1497: UndefinedMetricWarning: Precision is ill-defined and
being set to 0.0 in labels with no predicted samples. Use `zero_division` paramet
er to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\Megha Sharma\AppData\Roaming\Python\Python39\site-packages\sklearn\metri
cs\_classification.py:1497: UndefinedMetricWarning: Precision is ill-defined and
being set to 0.0 in labels with no predicted samples. Use `zero_division` paramet
er to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\Megha Sharma\AppData\Roaming\Python\Python39\site-packages\sklearn\metri
cs\_classification.py:1497: UndefinedMetricWarning: Precision is ill-defined and
being set to 0.0 in labels with no predicted samples. Use `zero_division` paramet
er to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

Multinomial Naive Bayes Model (MultinomialNB)

In [22]:

```
# Train a Naive Bayes classifier
NB_model = MultinomialNB()
NB_model.fit(train_set_tfidf, train_label)
y_pred_naive = NB_model.predict(val_set_tfidf)
print(classification_report(val_label, y_pred_naive))
naive_accuracy = accuracy_score(y_pred_naive,val_label)
print('Naive Bayes model accuracy is: {:.2f}%'.format(naive_accuracy*100))
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       263
           1       0.00      0.00      0.00       112
           2       0.00      0.00      0.00       139
           3       0.00      0.00      0.00       104
           4       0.00      0.00      0.00        61
           5       0.67      0.05      0.09      1443
           6       0.00      0.00      0.00       107
           7       0.51      0.89      0.65      2659
           8       0.38      0.87      0.53      2697
           9       0.00      0.00      0.00       150
          10       0.00      0.00      0.00        74
          11       0.00      0.00      0.00        40
          12       0.00      0.00      0.00        45
          13       0.00      0.00      0.00       431
          14       0.00      0.00      0.00       144
          15       0.00      0.00      0.00        50
          16       0.00      0.00      0.00        56
          17       0.00      0.00      0.00        34
          18       0.00      0.00      0.00       192
          19       0.00      0.00      0.00       151
          20       0.00      0.00      0.00       143
          21       0.00      0.00      0.00      1045
          22       0.00      0.00      0.00        93
          23       0.00      0.00      0.00        81
          24       0.00      0.00      0.00       309
          25       0.00      0.00      0.00        20
          26       0.00      0.00      0.00       200

    accuracy                           0.44     10843
   macro avg       0.06      0.07      0.05     10843
weighted avg       0.31      0.44      0.30     10843

Naive Bayes model accuracy is: 44.11%
```

```
C:\Users\Megha Sharma\AppData\Roaming\Python\Python39\site-packages\sklearn\metri
cs\_classification.py:1497: UndefinedMetricWarning: Precision is ill-defined and
being set to 0.0 in labels with no predicted samples. Use `zero_division` paramet
er to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\Megha Sharma\AppData\Roaming\Python\Python39\site-packages\sklearn\metri
cs\_classification.py:1497: UndefinedMetricWarning: Precision is ill-defined and
being set to 0.0 in labels with no predicted samples. Use `zero_division` paramet
er to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\Megha Sharma\AppData\Roaming\Python\Python39\site-packages\sklearn\metri
cs\_classification.py:1497: UndefinedMetricWarning: Precision is ill-defined and
being set to 0.0 in labels with no predicted samples. Use `zero_division` paramet
er to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

Decision Tree Model (ID3)

In [23]:

```
DT = DecisionTreeClassifier(max_depth=(1), random_state=0)
DT.fit(train_set_tfidf, train_label)
predict_ID3 = DT.predict(val_set_tfidf)
print(classification_report(val_label, predict_ID3))
```

```
ID3_accuracy = accuracy_score(predict_ID3,val_label)
print('ID3 model accuracy is: {:.2f}%'.format(ID3_accuracy*100))
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       263
           1       0.00      0.00      0.00       112
           2       0.00      0.00      0.00       139
           3       0.00      0.00      0.00       104
           4       0.00      0.00      0.00        61
           5       0.00      0.00      0.00      1443
           6       0.00      0.00      0.00       107
           7       0.82      0.31      0.45      2659
           8       0.27      0.99      0.43      2697
           9       0.00      0.00      0.00       150
          10       0.00      0.00      0.00        74
          11       0.00      0.00      0.00        40
          12       0.00      0.00      0.00        45
          13       0.00      0.00      0.00       431
          14       0.00      0.00      0.00       144
          15       0.00      0.00      0.00        50
          16       0.00      0.00      0.00        56
          17       0.00      0.00      0.00        34
          18       0.00      0.00      0.00       192
          19       0.00      0.00      0.00       151
          20       0.00      0.00      0.00       143
          21       0.00      0.00      0.00      1045
          22       0.00      0.00      0.00        93
          23       0.00      0.00      0.00        81
          24       0.00      0.00      0.00       309
          25       0.00      0.00      0.00        20
          26       0.00      0.00      0.00       200

    accuracy                           0.32     10843
   macro avg       0.04      0.05      0.03     10843
weighted avg       0.27      0.32      0.22     10843

ID3 model accuracy is: 32.30%
```

```
C:\Users\Megha Sharma\AppData\Roaming\Python\Python39\site-packages\sklearn\metri
cs\_classification.py:1497: UndefinedMetricWarning: Precision is ill-defined and
being set to 0.0 in labels with no predicted samples. Use `zero_division` paramet
er to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\Megha Sharma\AppData\Roaming\Python\Python39\site-packages\sklearn\metri
cs\_classification.py:1497: UndefinedMetricWarning: Precision is ill-defined and
being set to 0.0 in labels with no predicted samples. Use `zero_division` paramet
er to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\Megha Sharma\AppData\Roaming\Python\Python39\site-packages\sklearn\metri
cs\_classification.py:1497: UndefinedMetricWarning: Precision is ill-defined and
being set to 0.0 in labels with no predicted samples. Use `zero_division` paramet
er to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

Support Vector Machine Model (SVC)

In [24]:
```
# Train a SVC classifier
from sklearn.svm import LinearSVC
```

```
svm_model = LinearSVC()
svm_model.fit(train_set_tfidf, train_label)
predict = svm_model.predict(val_set_tfidf)

print(classification_report(val_label, predict))
svm_accuracy = accuracy_score(predict,val_label)
print('SVC model accuracy is: {:.2f}%'.format(svm_accuracy*100))
```

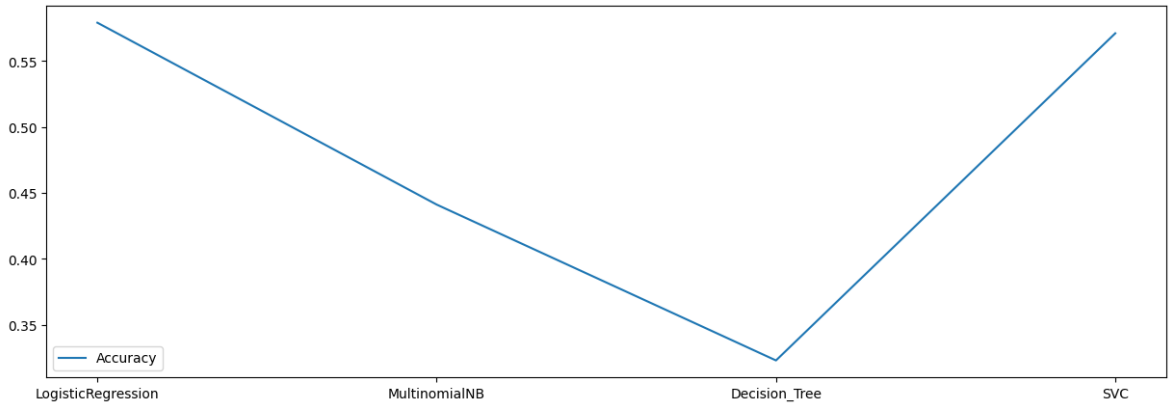
```
C:\Users\Megha Sharma\AppData\Roaming\Python\Python39\site-packages\sklearn\svm\_
classes.py:31: FutureWarning: The default value of `dual` will change from `True`
to `'auto'` in 1.5. Set the value of `dual` explicitly to suppress the warning.
  warnings.warn(
              precision    recall  f1-score   support

           0       0.46      0.33      0.38       263
           1       0.68      0.44      0.53       112
           2       0.39      0.21      0.27       139
           3       0.37      0.13      0.20       104
           4       0.00      0.00      0.00        61
           5       0.53      0.58      0.55      1443
           6       0.22      0.06      0.09       107
           7       0.68      0.80      0.73      2659
           8       0.55      0.71      0.62      2697
           9       0.38      0.15      0.22       150
          10       0.40      0.05      0.10        74
          11       0.83      0.62      0.71        40
          12       0.25      0.02      0.04        45
          13       0.60      0.63      0.61       431
          14       0.55      0.53      0.54       144
          15       0.27      0.06      0.10        50
          16       0.00      0.00      0.00        56
          17       0.29      0.06      0.10        34
          18       0.48      0.29      0.36       192
          19       0.28      0.06      0.10       151
          20       0.49      0.34      0.40       143
          21       0.43      0.34      0.38      1045
          22       0.61      0.44      0.51        93
          23       0.55      0.27      0.36        81
          24       0.27      0.16      0.20       309
          25       0.50      0.05      0.09        20
          26       0.85      0.83      0.84       200

    accuracy                           0.57     10843
   macro avg       0.44      0.30      0.33     10843
weighted avg       0.54      0.57      0.54     10843

SVC model accuracy is: 57.10%
```

In [25]:
```
columns=['LogisticRegression', 'MultinomialNB', 'Decision_Tree','SVC']
accuracy= [LR_accuracy, naive_accuracy, ID3_accuracy, svm_accuracy]

FinalResult=pd.DataFrame({'Algorithm':columns, 'Accuracy':accuracy})

FinalResult
```

Out[25]:

	Algorithm	Accuracy
0	LogisticRegression	0.578991
1	MultinomialNB	0.441114
2	Decision_Tree	0.322973
3	SVC	0.570967

In [26]:
```
fig,ax=plt.subplots(figsize=(15,5))
plt.plot(FinalResult.Algorithm,accuracy,label="Accuracy")
plt.legend()
plt.show()
```



In [ ]: