In [2]:

```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [3]:

```
df = pd.read_csv('Movie_classification.csv')
df
```

Out[3]:

	Marketing expense	Production expense	Multiplex coverage	Budget	Movie_length	Lead_ Actor_Rating	Lead_Actress_rating	Director_rating	Producer_rating	Critic_ratin
0	20.1264	59.62	0.462	36524.125	138.7	7.825	8.095	7.910	7.995	7.
1	20.5462	69.14	0.531	35668.655	152.4	7.505	7.650	7.440	7.470	7.
2	20.5458	69.14	0.531	39912.675	134.6	7.485	7.570	7.495	7.515	7.
3	20.6474	59.36	0.542	38873.890	119.3	6.895	7.035	6.920	7.020	8.
4	21.3810	59.36	0.542	39701.585	127.7	6.920	7.070	6.815	7.070	8.
...	...	...	...	...	...	...	...	...	...	
501	21.2526	78.86	0.427	36624.115	142.6	8.680	8.775	8.620	8.970	6.
502	20.9054	78.86	0.427	33996.600	150.2	8.780	8.945	8.770	8.930	7.
503	21.2152	78.86	0.427	38751.680	164.5	8.830	8.970	8.855	9.010	7.
504	22.1918	78.86	0.427	37740.670	162.8	8.730	8.845	8.800	8.845	6.
505	20.9482	78.86	0.427	33496.650	154.3	8.640	8.880	8.680	8.790	6.

506 rows × 19 columns

In [4]:

```
y = df.iloc[:,-1]
y
```

Out[4]:

```
0      1
1      0
2      1
3      1
4      1
      ..
501    0
502    0
503    0
504    0
505    0
Name: Start_Tech_Oscar, Length: 506, dtype: int64
```

In [6]:

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
df['3D_available'] = le.fit_transform(df['3D_available'])
df['Genre'] = le.fit_transform(df['Genre'])
```

In [19]:

```
df.isna().sum()
```

Out[19]:

```
Marketing expense       0
Production expense      0
Multiplex coverage      0
Budget                  0
Movie_length            0
Lead_ Actor_Rating      0
Lead_Actress_rating     0
Director_rating         0
Producer_rating         0
Critic_rating           0
Trailer_views           0
3D_available            0
Time_taken             12
Twitter_hastags         0
Genre                   0
Avg_age_actors          0
Num_multiplex           0
Collection              0
Start_Tech_Oscar        0
dtype: int64
```

In [21]:

```
del df['Time_taken']
```

In [22]:

```
x = df.iloc[:,0:-1].values
```

In [23]:

```
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.20,random_state=0)
```

In [31]:

```
from sklearn.cluster import KMeans
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans.fit(x_train,y_train)
    wcss.append(kmeans.inertia_)
```

In [32]:

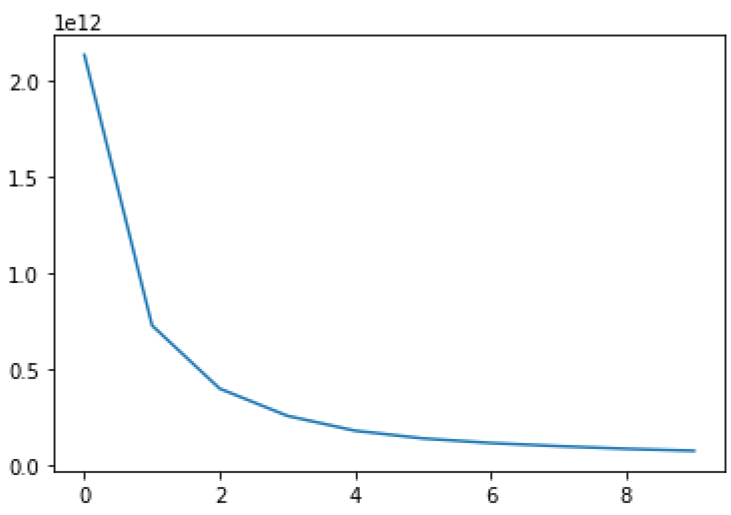
```
import matplotlib.pyplot as plt
```

In [33]:

```
plt.plot(wcss)
```

Out[33]:

```
[<matplotlib.lines.Line2D at 0x2cea7dac7f0>]
```



In [ ]:

In [56]:

```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters = 2, init = 'k-means++', random_state = 42)
kmeans.fit(x_train,y_train)
```

Out[56]:

```
KMeans(n_clusters=2, random_state=42)
```

In [57]:

```
y_pred = kmeans.predict(x_test)
```

In [58]:

```
y_pred
```

Out[58]:

```
array([1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0,
       1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0,
       1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1])
```

In [59]:

```
from sklearn.metrics import confusion_matrix,ConfusionMatrixDisplay
cm = confusion_matrix(y_test,y_pred, labels=[1,0])
disp = ConfusionMatrixDisplay(confusion_matrix=cm , display_labels=['True','Spam'])
```

In [60]:

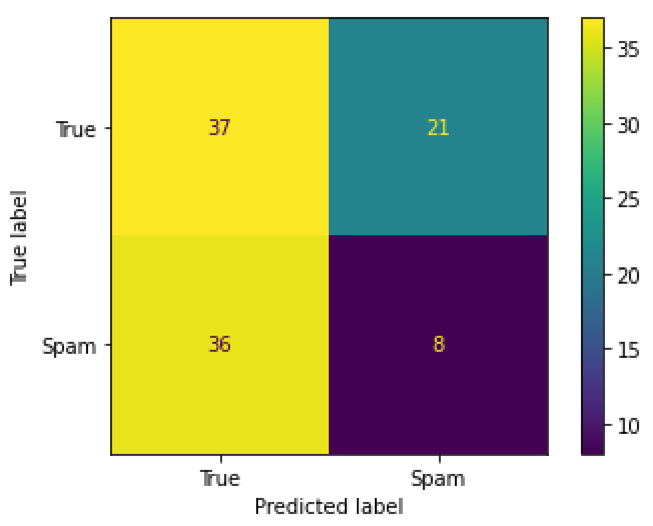
```
import seaborn as sns
```

In [61]:

```
disp.plot()
```

Out[61]:

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x2cea7b72170>
```



In [62]:

```
from sklearn.metrics import classification_report

cr = classification_report(y_test, y_pred)
print(cr)
```

```
              precision    recall  f1-score   support

           0       0.28      0.18      0.22        44
           1       0.51      0.64      0.56        58

    accuracy                           0.44       102
   macro avg       0.39      0.41      0.39       102
weighted avg       0.41      0.44      0.42       102
```

In [ ]:

In [ ]:

In [ ]: