svm-classification-1

April 5, 2024

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

[2]:
```
# Loading the dataset
df = pd.read_csv(r'C:\Users\ntpc\Desktop\Movie_classification.csv')
df.head()
```

[2]:

	Marketing expense	Production expense	Multiplex coverage	Budget	Movie_length	Lead_ Actor_Rating	Lead_Actress_rating	Director_rating	Producer_rating	Critic_rating	Trailer_views	3D_available	Time_taken	Twitter_hastags	Genre	Avg_age_actors	Num_multiplex	Collection
0	20.1264	59.62	0.462	36524.125	138.7	7.825	8.095	7.910	7.995	7.94	527367	YES	109.60	223.840	Thriller	23	494	48000
1	20.5462	69.14	0.531	35668.655	152.4	7.505	7.650	7.440	7.470	7.44	494055	NO	146.64	243.456	Drama	42	462	43200
2	20.5458	69.14	0.531	39912.675	134.6	7.485	7.570	7.495	7.515	7.44	547051	NO	147.88	2022.400	Comedy	38	458	69400
3	20.6474	59.36	0.542	38873.890	119.3	6.895	7.035	6.920	7.020	8.26	516279	YES	185.36	225.344	Drama	45	472	66800
4	21.3810	59.36	0.542	39701.585	127.7	6.920	7.070	6.815	7.070	8.26	531448	NO	176.48	225.792	Drama	55	395	72400

```
   Start_Tech_Oscar
0                 1
1                 0
2                 1
3                 1
4                 1
```

0.1 Missing Value Imputation

[3]:
```
df['Time_taken'].mean()
```

[3]:
```
157.3914979757085
```

[4]:
```
df['Time_taken'].fillna(value = df['Time_taken'].mean(), inplace = True)
```

```
C:\Users\ntpc\AppData\Local\Temp\ipykernel_13416\720848667.py:1: FutureWarning:
A value is trying to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df['Time_taken'].fillna(value = df['Time_taken'].mean(), inplace = True)
```

[5]:
```
df.dtypes
```

[5]:
```
Marketing expense      float64
Production expense     float64
Multiplex coverage     float64
Budget                 float64
Movie_length           float64
Lead_ Actor_Rating     float64
Lead_Actress_rating    float64
Director_rating        float64
Producer_rating        float64
Critic_rating          float64
Trailer_views            int64
3D_available            object
Time_taken             float64
Twitter_hastags        float64
Genre                   object
Avg_age_actors           int64
```

```
Num_multiplex            int64
Collection               int64
Start_Tech_Oscar         int64
dtype: object
```

[6]:
```
df = pd.get_dummies(df,columns = ["3D_available","Genre"],drop_first = True)
```

0.2 X-y split

[7]:
```
X = df.loc[:,df.columns!="Start_Tech_Oscar"]
```

[8]:
```
y = df["Start_Tech_Oscar"]
```

[9]:
```
from sklearn.model_selection import train_test_split
```

[10]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=0.
↪2,random_state=0)
```

[11]:
```
from sklearn.preprocessing import StandardScaler
```

[12]:
```
sc = StandardScaler().fit(X_train)
```

[13]:
```
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)
X_test_std
```

[13]:
```
array([[-0.40835869, -1.12872913,  0.83336883, …,  1.50268577,
        -0.48525664, -0.75225758],
       [ 0.71925111,  0.9988844 , -0.65283979, …,  1.50268577,
        -0.48525664, -0.75225758],
       [-0.40257488,  0.39610829,  0.05115377, …,  1.50268577,
        -0.48525664, -0.75225758],
       …,
       [-0.3982601 , -0.85812418,  0.89420778, …, -0.66547513,
        -0.48525664,  1.3293319 ],
       [-0.39934279, -0.07637654,  0.58132175, …,  1.50268577,
        -0.48525664, -0.75225758],
       [-0.40088071, -0.36702631,  0.31189212, …, -0.66547513,
        -0.48525664, -0.75225758]])
```

[14]:
```
from sklearn import svm
```

[15]:
```
clf_svm_l = svm.SVC(kernel='linear', C=100)
clf_svm_l.fit(X_train_std, y_train)
```

[15]:
```
SVC(C=100, kernel='linear')
```

[16]:
```
y_train_pred = clf_svm_l.predict(X_train_std)
y_test_pred = clf_svm_l.predict(X_test_std)
```

[17]:
```
from sklearn.metrics import accuracy_score, confusion_matrix
```

[18]:
```
confusion_matrix(y_test, y_test_pred)
```

[18]:
```
array([[25, 19],
       [25, 33]], dtype=int64)
```

[19]:
```
accuracy_score(y_test, y_test_pred)
```

[19]:
```
0.5686274509803921
```