Assignment on Classification technique

The counselor of the firm is supposed check whether the student will get an admission or not based on his/her GRE score and Academic Score. So to help the counselor to take appropriate decisions build a machine learning model classifier using SVM to predict whether a student will get admission or not.

1. Apply Data pre-processing (Label Encoding, Data Transformation....) techniques if necessary.
2. Perform data-preparation (Train-Test Split)
3. Apply Machine Learning Algorithm
4. Evaluate Model.

Importing required libraries

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
```

Reading the dataset

In [3]:
```
A = pd.read_csv(r'C:\Users\DELL\Downloads\Admission_Predict.csv')
A.head()
```

Out[3]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	1	337	118	4	4.5	4.5	9.65	1	0.92
1	2	324	107	4	4.0	4.5	8.87	1	0.76
2	3	316	104	3	3.0	3.5	8.00	1	0.72
3	4	322	110	3	3.5	2.5	8.67	1	0.80
4	5	314	103	2	2.0	3.0	8.21	0	0.65

In [5]:
```
#dropping Serail No. column as it has no significance
A.drop('Serial No.', axis=1, inplace=True)
```

Analysing the dataset

In [6]:
```
A.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   GRE Score          400 non-null    int64  
 1   TOEFL Score        400 non-null    int64  
 2   University Rating  400 non-null    int64  
 3   SOP                400 non-null    float64
 4   LOR                400 non-null    float64
 5   CGPA               400 non-null    float64
 6   Research           400 non-null    int64  
 7   Chance of Admit    400 non-null    float64
dtypes: float64(4), int64(4)
memory usage: 25.1 KB
```

In [7]:
```
# three is no null value in the dataset
```

In [8]:
```
A.describe()
```

Out[8]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chanc Ac
count	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.00(
mean	316.807500	107.410000	3.087500	3.400000	3.452500	8.598925	0.547500	0.72
std	11.473646	6.069514	1.143728	1.006869	0.898478	0.596317	0.498362	0.142
min	290.000000	92.000000	1.000000	1.000000	1.000000	6.800000	0.000000	0.34(
25%	308.000000	103.000000	2.000000	2.500000	3.000000	8.170000	0.000000	0.64(
50%	317.000000	107.000000	3.000000	3.500000	3.500000	8.610000	1.000000	0.73(
75%	325.000000	112.000000	4.000000	4.000000	4.000000	9.062500	1.000000	0.83(
max	340.000000	120.000000	5.000000	5.000000	5.000000	9.920000	1.000000	0.97(

In [9]:
```
A.columns
```

Out[9]:
```
Index(['GRE Score', 'TOEFL Score', 'University Rating', 'SOP', 'LOR ', 'CGPA',
       'Research', 'Chance of Admit '],
      dtype='object')
```

In [11]:
```
# converting chance of admit column into binary values
A['Chance of Admit '] = A['Chance of Admit '].apply(lambda x: 1 if x >= 0.50 else 0
```

In [12]:
```
A['Chance of Admit '].value_counts()
```

Out[12]:
```
1    367
0     33
Name: Chance of Admit , dtype: int64
```

Splitting the dataset

In [13]:
```
X = A.drop('Chance of Admit ', axis=1)
Y = A['Chance of Admit ']
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_sta
```

In [14]:
```
X.head()
```

Out[14]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research
0	337	118	4	4.5	4.5	9.65	1
1	324	107	4	4.0	4.5	8.87	1
2	316	104	3	3.0	3.5	8.00	1
3	322	110	3	3.5	2.5	8.67	1
4	314	103	2	2.0	3.0	8.21	0

In [15]:
```
Y.head()
```

Out[15]:
```
0    1
1    1
2    1
3    1
4    1
Name: Chance of Admit , dtype: int64
```

In [16]:
```
print("shape of X_train: ", X_train.shape)
print("shape of X_test: ", X_test.shape)
print("shape of y_train: ", Y_train.shape)
print("shape of y_test: ", Y_test.shape)
```

```
shape of X_train:  (320, 7)
shape of X_test:  (80, 7)
shape of y_train:  (320,)
shape of y_test:  (80,)
```

Creating Model

Linear kernel

In [37]:
```
clf1 = SVC(kernel='linear')
clf1.fit(X_train, Y_train)
y1_pred = clf1.predict(X_test)
```

Polynomial Kernel

In [38]:
```
clf2 = SVC(kernel='poly')
clf2.fit(X_train, Y_train)
y2_pred = clf2.predict(X_test)
```

RBF Kernel

In [39]:
```
clf3 = SVC(kernel='rbf')
clf3.fit(X_train, Y_train)
y3_pred = clf3.predict(X_test)
```

Sigmoid kernel

In [40]:
```
clf4 = SVC(kernel='sigmoid')
clf4.fit(X_train, Y_train)
y4_pred = clf4.predict(X_test)
```

Evaluation

In [43]:
```
final_output_diff = pd.DataFrame({'Acutal': Y_test, 'Linear Kernel': y1_pred, 'Poly
```

In [44]:
```
final_output_diff.head(10)
```

Out[44]:

	Acutal	Linear Kernel	Polynomial kernel	rbf kernel	sigmoid kernel
209	1	1	1	1	1
280	1	1	1	1	1
33	1	1	1	1	1
210	1	1	1	1	1
93	0	1	1	1	1
84	1	1	1	1	1
329	0	0	1	1	1
94	0	0	1	1	1
266	1	1	1	1	1
126	1	1	1	1	1

In [45]:
```
# print accuracy score
print("Accuracy score of Linear kernel: ", accuracy_score(Y_test, y1_pred))
print("Accuracy score of Polynomial kernel: ", accuracy_score(Y_test, y2_pred))
print("Accuracy score of RBF kernel: ", accuracy_score(Y_test, y3_pred))
print("Accuracy score of Sigmoid kernel: ", accuracy_score(Y_test, y4_pred))
```

```
Accuracy score of Linear kernel:  0.9375
Accuracy score of Polynomial kernel:  0.875
Accuracy score of RBF kernel:  0.875
Accuracy score of Sigmoid kernel:  0.875
```

In [46]:
```
# comparing accurency scores
accuracy_df = pd.DataFrame({'Linear Kernel': accuracy_score(Y_test, y1_pred), 'Poly
```

In [47]:
```
accuracy_df
```

Out[47]:

	Linear Kernel	Polynomial kernel	RBF kernel	Sigmoid kernel
0	0.9375	0.875	0.875	0.875

In [48]:
```
# from above result we can see that our dataset is linear hence Linear kernel is be
```

Visualising Linear Model

In [49]:
```
# visualising linear kernel
cm = confusion_matrix(Y_test, y1_pred)
sns.heatmap(cm, annot=True)
```

Out[49]: <AxesSubplot:>