

# Support vector regression on Admission prediction dataset

In [1]:

```
#importing libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
a=pd.read_csv('https://raw.githubusercontent.com/srinivasav22/Graduate-Admission-Prediction/')
```

In [3]:

```
data.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 [4]:

data.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 9 columns):
#   Column                Non-Null Count  Dtype
---  ---
0   Serial No.            500 non-null   int64
1   GRE Score              500 non-null   int64
2   TOEFL Score            500 non-null   int64
3   University Rating      500 non-null   int64
4   SOP                    500 non-null   float64
5   LOR                    500 non-null   float64
6   CGPA                   500 non-null   float64
7   Research                500 non-null   int64
8   Chance of Admit        500 non-null   float64
dtypes: float64(4), int64(5)
memory usage: 35.3 KB
```

In [6]:

data.columns

Out[6]:

```
Index(['Serial No.', 'GRE Score', 'TOEFL Score', 'University Rating', 'SOP',
      'LOR ', 'CGPA', 'Research', 'Chance of Admit '],
      dtype='object')
```

In [8]:

```
#We dont need serial no feature so we removing it
data=data.drop('Serial No.', axis=1)
```

In [9]:

data.head()

Out[9]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	337	118	4	4.5	4.5	9.65	1	0.92
1	324	107	4	4.0	4.5	8.87	1	0.76
2	316	104	3	3.0	3.5	8.00	1	0.72
3	322	110	3	3.5	2.5	8.67	1	0.80
4	314	103	2	2.0	3.0	8.21	0	0.65

In [10]:

```
data.describe()
```

Out[10]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	C of
count	500.000000	500.000000	500.000000	500.000000	500.000000	500.000000	500.000000	500.
mean	316.472000	107.192000	3.114000	3.374000	3.48400	8.576440	0.560000	0.
std	11.295148	6.081868	1.143512	0.991004	0.92545	0.604813	0.496884	0
min	290.000000	92.000000	1.000000	1.000000	1.00000	6.800000	0.000000	0.
25%	308.000000	103.000000	2.000000	2.500000	3.00000	8.127500	0.000000	0.
50%	317.000000	107.000000	3.000000	3.500000	3.50000	8.560000	1.000000	0.
75%	325.000000	112.000000	4.000000	4.000000	4.00000	9.040000	1.000000	0.
max	340.000000	120.000000	5.000000	5.000000	5.00000	9.920000	1.000000	0.



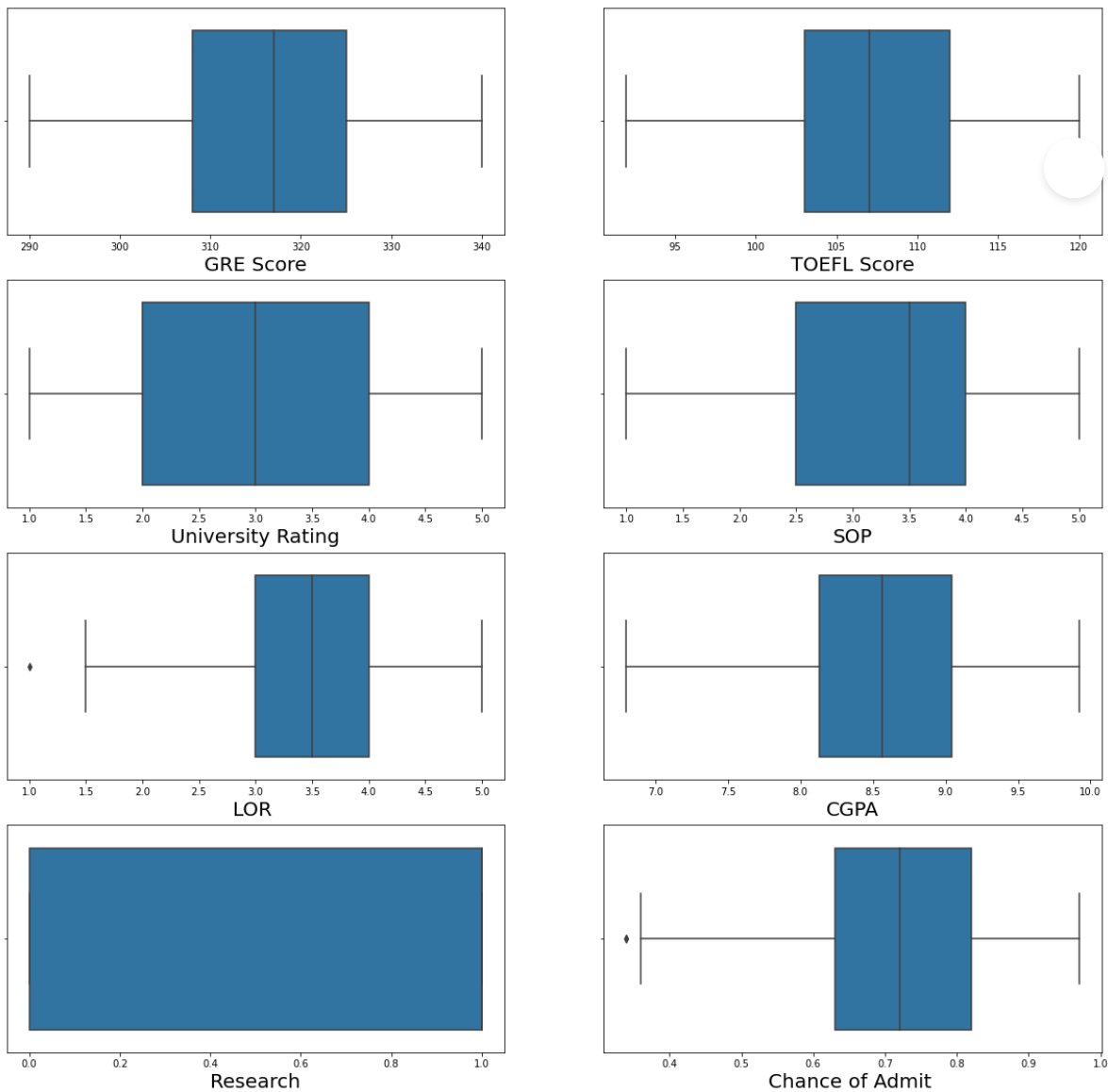
In [20]:

*#We are plotting boxplot to check outlier present in the dataset*

```
plt.figure(figsize=(20,40), facecolor='white')
plotnumber=1
```

```
for column in data:
    if plotnumber<=12 :
        ax = plt.subplot(8,2,plotnumber)
        sns.boxplot(data[column])
        plt.xlabel(column,fontsize=20)
```

```
    plotnumber+=1
plt.show()
```



In [23]:

```
#We are checking outliers present in the dataset
data.isnull().sum()
```

Out[23]:

```
GRE Score      0
TOEFL Score    0
University Rating 0
SOP            0
LOR            0
CGPA           0
Research        0
Chance of Admit 0
dtype: int64
```

In [25]:

```
#Checking correlation using heatmap
plt.figure(figsize=(16,10))
sns.heatmap(data=data.corr(), annot=True)
```

Out[25]:

&lt;AxesSubplot:&gt;



## Now creating dependent and independent features

In [27]:

```
X= data.drop('Chance of Admit ', axis=1)
```

In [28]:

```
y = data['Chance of Admit ']
```

In [29]:

```
X
```

Out[29]:

	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
...	...	...	...	...	...	...	...
495	332	108	5	4.5	4.0	9.02	1
496	337	117	5	5.0	5.0	9.87	1
497	330	120	5	4.5	5.0	9.56	1
498	312	103	4	4.0	5.0	8.43	0
499	327	113	4	4.5	4.5	9.04	0

500 rows × 7 columns

In [30]:

```
y
```

Out[30]:

```
0    0.92
1    0.76
2    0.72
3    0.80
4    0.65
...
495   0.87
496   0.96
497   0.93
498   0.73
499   0.84
```

Name: Chance of Admit , Length: 500, dtype: float64

In [31]:

```
#Splitting data into training data and testing data
from sklearn.model_selection import train_test_split
```

In [32]:

```
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.25, random_state=42)
```

In [33]:

```
#Feature Scaling  
from sklearn.preprocessing import StandardScaler
```

In [34]:

```
scaler=StandardScaler()
```

In [35]:

```
X_train=scaler.fit_transform(X_train)  
X_test=scaler.transform(X_test)
```


In [36]:

```
#Importing SVR  
from sklearn.svm import SVR  
model=SVR()
```

In [37]:

```
model.fit(X_train,y_train)
```

Out[37]:



SVR()  
SVR()

In [38]:

```
y_pred=model.predict(X_test)
```

In [40]:

```
model.score(X_test,y_test)
```

Out[40]:

0.7407908837691202