In [7]:
```
import numpy as np
import pandas as pd
from sklearn.metrics import r2_score,mean_squared_error
from math import sqrt
```

In [8]:
```
df = pd.read_csv("C:\\Users\\Pooja Agarwal\\Downloads\\Admission_Predict.csv")
#printing first 5 rows of data frame
df.head()
```

Out[8]:

	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 [10]:
```
df.shape
```

Out[10]:
```
(400, 9)
```

In [12]:
```
df.dtypes
```

Out[12]:
```
Serial No.            int64
GRE Score             int64
TOEFL Score           int64
University Rating     int64
SOP                 float64
LOR                 float64
CGPA                float64
Research              int64
Chance of Admit     float64
dtype: object
```

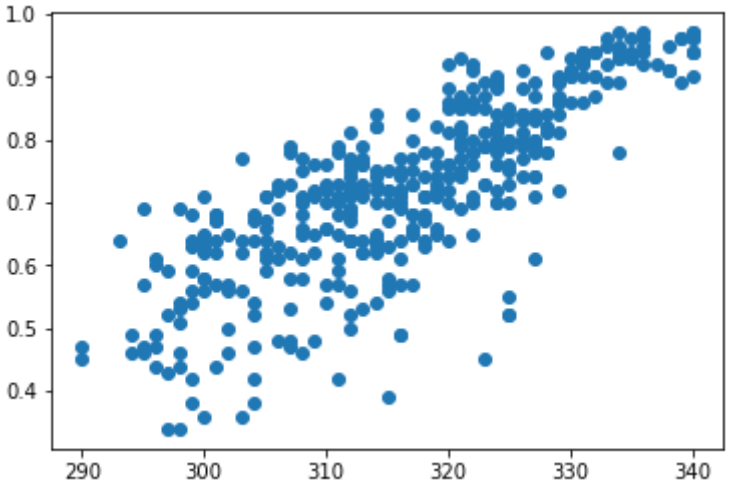
In [13]:
```
df.columns
```

Out[13]:
```
Index(['Serial No.', 'GRE Score', 'TOEFL Score', 'University Rating', 'SOP',
       'LOR ', 'CGPA', 'Research', 'Chance of Admit '],
      dtype='object')
```

In [22]:
```
X=df[['GRE Score', 'TOEFL Score', 'University Rating', 'SOP','LOR ', 'CGPA', 'Resear
y=df['Chance of Admit ']
```

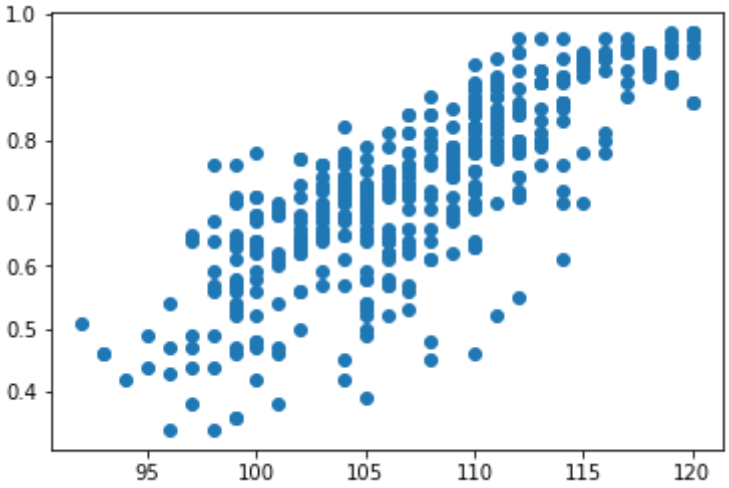
In [16]:
```
import matplotlib.pyplot as plt
plt.scatter(X['GRE Score'],y)
```

Out[16]:
```
<matplotlib.collections.PathCollection at 0x1880dcf2910>
```



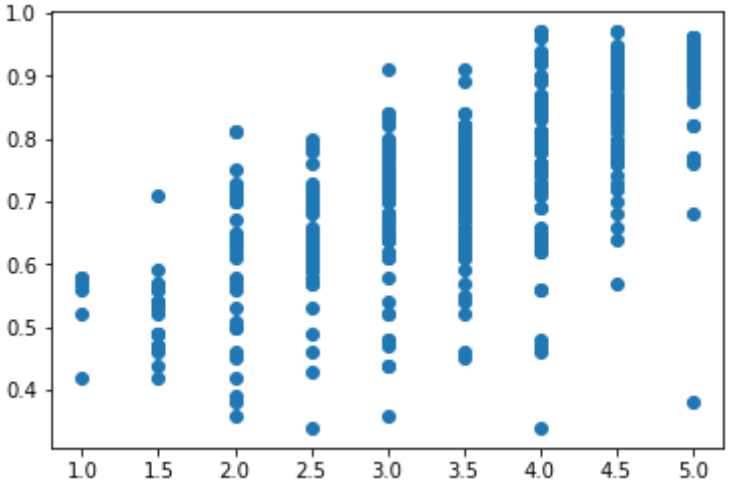
In [17]:
```
plt.scatter(X['TOEFL Score'],y)
```

Out[17]: <matplotlib.collections.PathCollection at 0x1880e03a460>



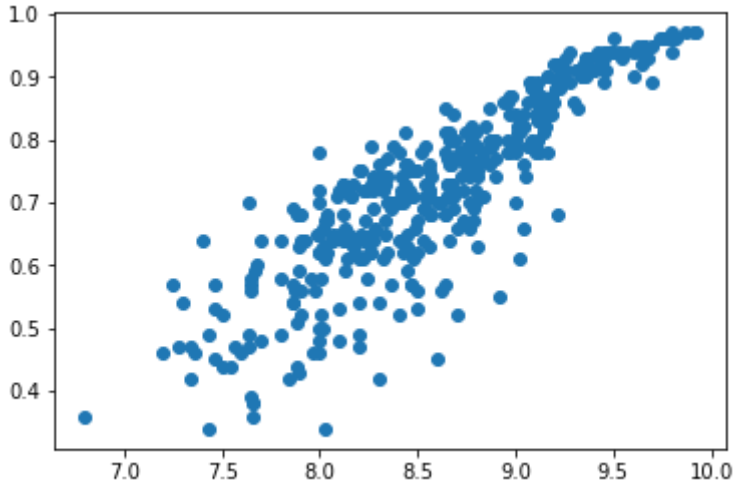
In [18]:
```
plt.scatter(X['SOP'],y)
```

Out[18]: <matplotlib.collections.PathCollection at 0x1880e095bb0>



In [19]:
```
plt.scatter(X['CGPA'],y)
```

Out[19]: <matplotlib.collections.PathCollection at 0x1880e0fc610>



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=0.2,random_state=14)
```

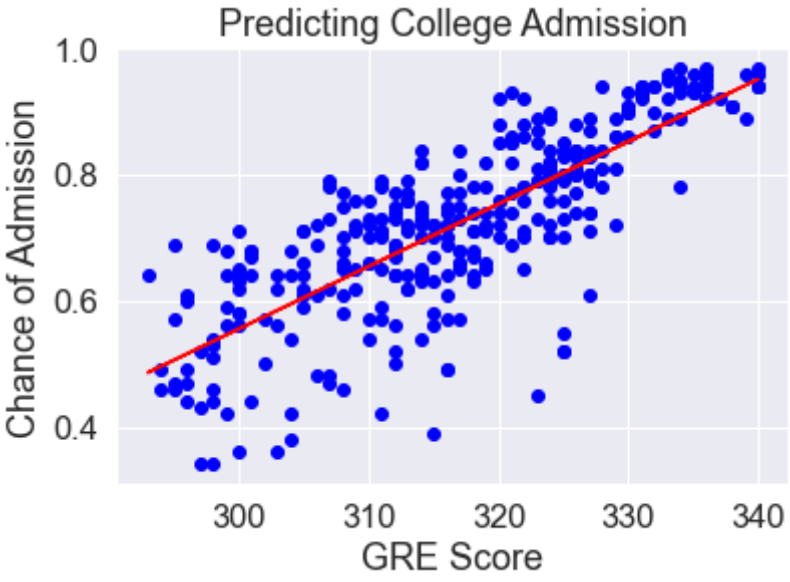
In [24]:
```
X_tested=X_test[['GRE Score']]
X_trained=X_train[['GRE Score']]
```

In [25]:
```
#Simple Linear Regression

from sklearn.linear_model import LinearRegression
LR=LinearRegression()
LR.fit(X_trained,y_train)
yhat=LR.predict(X_tested)
print(np.sqrt(mean_squared_error(y_test,yhat)))#out of sample accuracy
print(r2_score(y_test,yhat))
```

```
0.07226833478856222
0.7350482562603644
```

In [63]:
```
yhated=LR.predict(X_trained)
plt.scatter(X_trained,y_train,color='blue')
plt.plot(X_trained,yhated,color='red')
plt.title('Predicting College Admission')
plt.xlabel('GRE Score')
plt.ylabel('Chance of Admission')
plt.show()
```

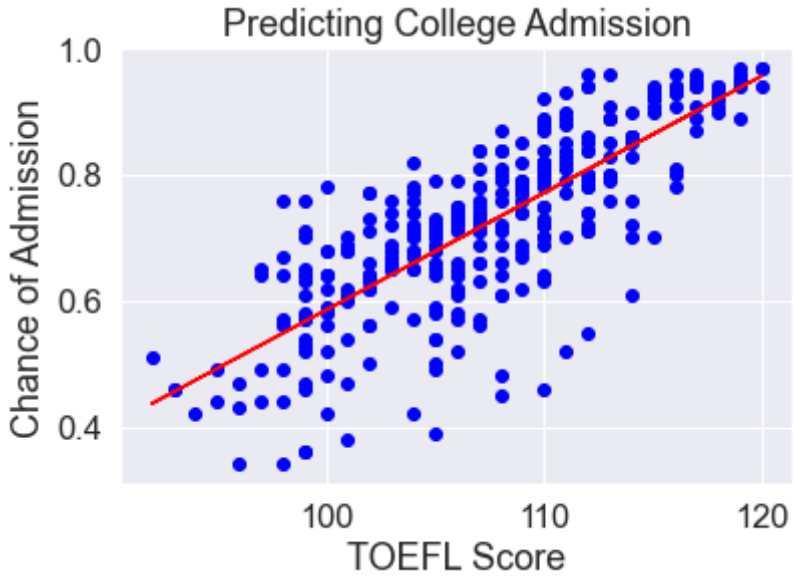


In [64]:
```
X_tested=X_test[['TOEFL Score']]
X_trained=X_train[['TOEFL Score']]
```

In [65]:
```
from sklearn.linear_model import LinearRegression
LR=LinearRegression()
LR.fit(X_trained,y_train)
yhat=LR.predict(X_tested)
print(np.sqrt(mean_squared_error(y_test,yhat)))#out of sample accuracy
print(r2_score(y_test,yhat))
```

```
0.08252344213607452
0.6545180552654432
```

In [66]:
```
yhated=LR.predict(X_trained)
plt.scatter(X_trained,y_train,color='blue')
plt.plot(X_trained,yhated,color='red')
plt.title('Predicting College Admission')
plt.xlabel('TOEFL Score')
plt.ylabel('Chance of Admission')
plt.show()
```

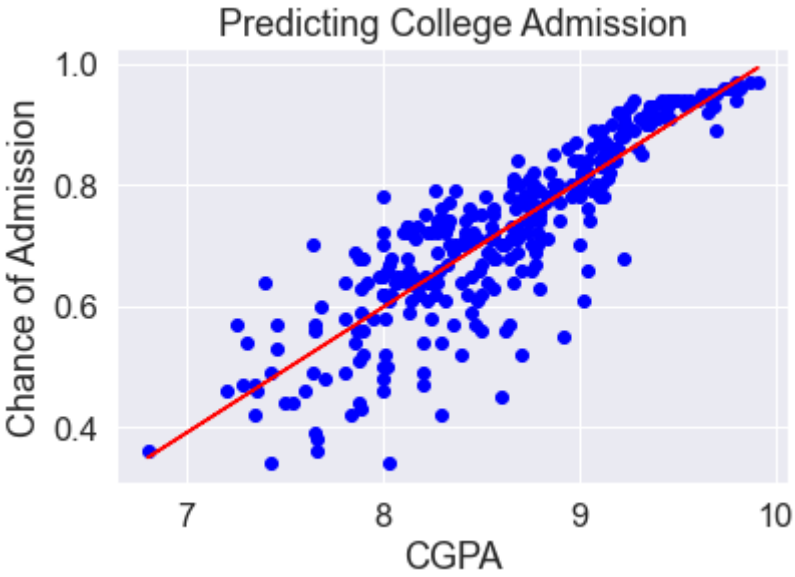


In [71]:
```
X_tested=X_test[['CGPA']]
X_trained=X_train[['CGPA']]
```

In [72]:
```
from sklearn.linear_model import LinearRegression
LR=LinearRegression()
LR.fit(X_trained,y_train)
yhat=LR.predict(X_tested)
print(np.sqrt(mean_squared_error(y_test,yhat)))#out of sample accuracy
print(r2_score(y_test,yhat))
```

```
0.06223449324321656
0.8035133710605582
```

In [73]:
```
yhated=LR.predict(X_trained)
plt.scatter(X_trained,y_train,color='blue')
plt.plot(X_trained,yhated,color='red')
plt.title('Predicting College Admission')
plt.xlabel('CGPA')
plt.ylabel('Chance of Admission')
plt.show()
```



In [27]:
```
#Multiple Linear Regression

X_test1=X_test[['GRE Score','TOEFL Score','SOP','CGPA','University Rating']]
X_train1=X_train[['GRE Score','TOEFL Score','SOP','CGPA','University Rating']]
```

In [29]:
```
mlr=LinearRegression()
mlr.fit(X_train1,y_train)
yhatmul=mlr.predict(X_test1)
```

In [30]:
```
print(np.sqrt(mean_squared_error(y_test,yhatmul)))#out of sample accuracy
print(r2_score(y_test,yhatmul))
```

```
0.05716606389378742
0.8342142236850849
```

In [77]:
```
data1=df[['GRE Score','TOEFL Score','CGPA','SOP','Chance of Admit ']]
data1=data1.rename(columns={'GRE Score':'GRE', 'TOEFL Score':'TOEFL','CGPA':'CGPA','
data1.head()
```

Out[77]:

	GRE	TOEFL	CGPA	SOP	Chance_of_Admit
0	337	118	9.65	4.5	0.92
1	324	107	8.87	4.0	0.76
2	316	104	8.00	3.0	0.72
3	322	110	8.67	3.5	0.80
4	314	103	8.21	2.0	0.65

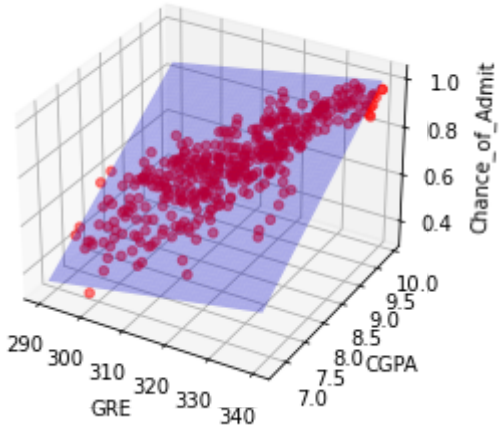
In [78]:
```
import matplotlib.pyplot as plt
import statsmodels.formula.api as smf
model = smf.ols(formula='Chance_of_Admit  ~ GRE + CGPA + TOEFL + SOP', data=data1)
results_formula = model.fit()
results_formula.params
```

Out[78]:
```
Intercept   -1.502194
GRE          0.002327
CGPA         0.135224
TOEFL        0.002681
SOP          0.011307
dtype: float64
```

In [37]:
```
x_surf, y_surf = np.meshgrid(np.linspace(data1.GRE.min(), data1.GRE.max(), 100),np.l
```

```
onlyX = pd.DataFrame({'GRE': x_surf.ravel(), 'CGPA': y_surf.ravel()})
fittedY=results_formula.predict(exog=onlyX)
fittedY=np.array(fittedY)
```

In [38]:
```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(data1['GRE'],data1['CGPA'],data1['Chance_of_Admit'],c='red', marker='o',
ax.plot_surface(x_surf,y_surf,fittedY.reshape(x_surf.shape), color='b', alpha=0.3)
ax.set_xlabel('GRE')
ax.set_ylabel('CGPA')
ax.set_zlabel('Chance_of_Admit')
plt.show()
```



In [39]:
```
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
Xs=ss.fit_transform(X)
from sklearn.model_selection import train_test_split
X_trains,X_tests,y_trains,y_tests=train_test_split(Xs,y,test_size=0.2,random_state=1
mlrs=LinearRegression()
mlrs.fit(X_trains,y_trains)
yhats=mlrs.predict(X_tests)
print(np.sqrt(mean_squared_error(y_tests,yhats)))
print(r2_score(y_tests,yhats))
```

```
0.05758406168074793
0.8317809114577306
```

In [48]:
```
results_formula.rsquared
```

Out[48]:
```
0.7810684887220862
```

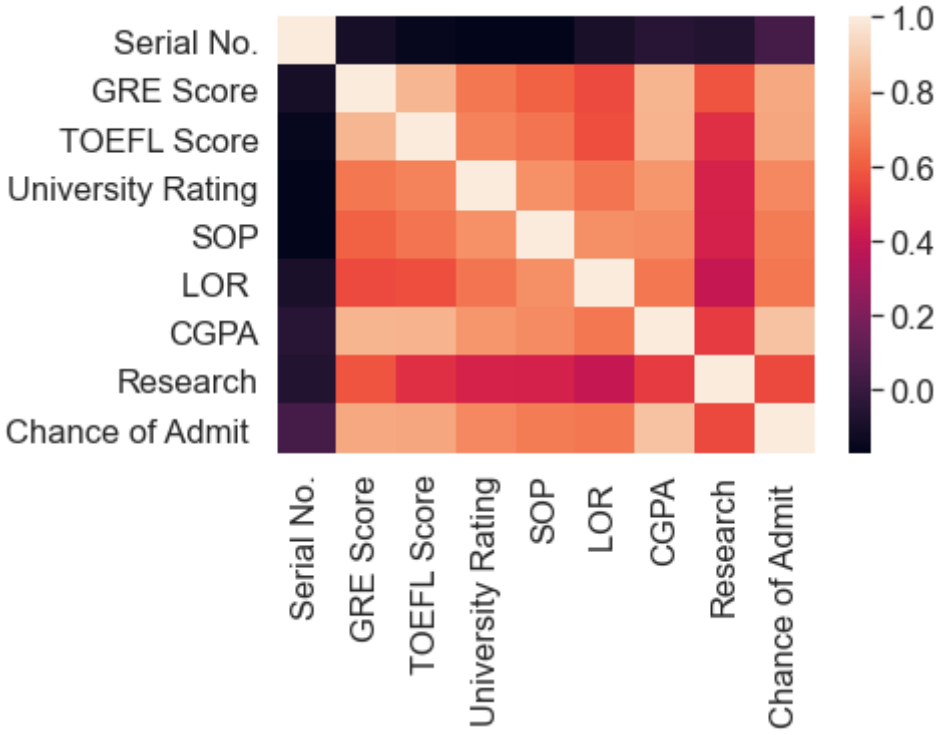
In [50]:
```
results_formula.rsquared_adj
```

Out[50]:
```
0.7799655591942378
```

In [52]:
```
import seaborn as sns
sns.set(style="white",color_codes=True)
sns.set(font_scale=1.5)
```

In [60]:
```
import seaborn as sns
corr = df.corr()
sns.heatmap(corr,
            xticklabels=corr.columns.values,
            yticklabels=corr.columns.values)
```

Out[60]:
```
<AxesSubplot:>
```



In [79]:
```
results_formula.pvalues
```

Out[79]:
```
Intercept    1.444868e-34
GRE          9.471928e-05
CGPA         1.558924e-25
TOEFL        1.669822e-02
SOP          1.845946e-02
dtype: float64
```

In [82]:
```
model = smf.ols(formula='Chance_of_Admit  ~ GRE + CGPA + TOEFL', data=data1)
results_formula = model.fit()
results_formula.rsquared
```

Out[82]:
```
0.7853531466375188
```

In [83]:
```
model = smf.ols(formula='Chance_of_Admit  ~ GRE + CGPA', data=data1)
results_formula = model.fit()
results_formula.rsquared
```

Out[83]:
```
0.7810684887220862
```

In [84]:
```
model = smf.ols(formula='Chance_of_Admit  ~ GRE', data=data1)
results_formula = model.fit()
results_formula.rsquared
```

Out[84]:
```
0.6441835498438335
```

In [ ]:
```
#As the no. of variables decrease, the rsquared value also decreases
```