In [1]:
```
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.preprocessing import StandardScaler

from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor

from sklearn import metrics
```

In [2]:
```
dataset = pd.read_csv('Admission_Predict.csv')
```

In [3]:
```
dataset.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]:
```
dataset.tail()
```

Out[4]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
395	396	324	110	3	3.5	3.5	9.04	1	0.82
396	397	325	107	3	3.0	3.5	9.11	1	0.84
397	398	330	116	4	5.0	4.5	9.45	1	0.91
398	399	312	103	3	3.5	4.0	8.78	0	0.67
399	400	333	117	4	5.0	4.0	9.66	1	0.95

In [5]:
```
dataset.shape
```

Out[5]:
```
(400, 9)
```

In [6]:
```
print('Number of Rows:',dataset.shape[0])
print('Number of Columns:',dataset.shape[1])
```

```
Number of Rows: 400
Number of Columns: 9
```

In [7]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Serial No.         400 non-null    int64  
 1   GRE Score          400 non-null    int64  
 2   TOEFL Score        400 non-null    int64  
 3   University Rating  400 non-null    int64  
 4   SOP                400 non-null    float64
 5   LOR                400 non-null    float64
 6   CGPA               400 non-null    float64
 7   Research           400 non-null    int64  
 8   Chance of Admit    400 non-null    float64
dtypes: float64(4), int64(5)
memory usage: 28.2 KB
```

In [8]:
```
dataset.isnull().sum()
```

Out[8]:
```
Serial No.           0
GRE Score            0
TOEFL Score          0
University Rating    0
SOP                  0
LOR                  0
CGPA                 0
Research             0
Chance of Admit      0
dtype: int64
```

In [9]:
```
dataset.duplicated().sum()
```

Out[9]:
```
0
```

In [10]:
```
dataset.describe()
```

Out[10]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Rese
count	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.00
mean	200.500000	316.807500	107.410000	3.087500	3.400000	3.452500	8.598925	0.54
std	115.614301	11.473646	6.069514	1.143728	1.006869	0.898478	0.596317	0.49
min	1.000000	290.000000	92.000000	1.000000	1.000000	1.000000	6.800000	0.00
25%	100.750000	308.000000	103.000000	2.000000	2.500000	3.000000	8.170000	0.00
50%	200.500000	317.000000	107.000000	3.000000	3.500000	3.500000	8.610000	1.00
75%	300.250000	325.000000	112.000000	4.000000	4.000000	4.000000	9.062500	1.00
max	400.000000	340.000000	120.000000	5.000000	5.000000	5.000000	9.920000	1.00

In [11]:
```
dataset.corr()
```

Out[11]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	C of
Serial No.	1.000000	-0.097526	-0.147932	-0.169948	-0.166932	-0.088221	-0.045608	-0.063138	0.0
GRE Score	-0.097526	1.000000	0.835977	0.668976	0.612831	0.557555	0.833060	0.580391	0.8
TOEFL Score	-0.147932	0.835977	1.000000	0.695590	0.657981	0.567721	0.828417	0.489858	0.7
University Rating	-0.169948	0.668976	0.695590	1.000000	0.734523	0.660123	0.746479	0.447783	0.7
SOP	-0.166932	0.612831	0.657981	0.734523	1.000000	0.729593	0.718144	0.444029	0.6
LOR	-0.088221	0.557555	0.567721	0.660123	0.729593	1.000000	0.670211	0.396859	0.6
CGPA	-0.045608	0.833060	0.828417	0.746479	0.718144	0.670211	1.000000	0.521654	0.8
Research	-0.063138	0.580391	0.489858	0.447783	0.444029	0.396859	0.521654	1.000000	0.5
Chance of Admit	0.042336	0.802610	0.791594	0.711250	0.675732	0.669889	0.873289	0.553202	1.0

In [12]:
```
dataset.cov()
```

Out[12]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Re
Serial No.	13366.666667	-129.369674	-103.807018	-22.472431	-19.432331	-9.164160	-3.144373	-3.6
GRE Score	-129.369674	131.644555	58.216967	8.778791	7.079699	5.747726	5.699742	3.3
TOEFL Score	-103.807018	58.216967	36.838997	4.828697	4.021053	3.095965	2.998337	1.4
University Rating	-22.472431	8.778791	4.828697	1.308114	0.845865	0.678352	0.509117	0.2
SOP	-19.432331	7.079699	4.021053	0.845865	1.013784	0.660025	0.431183	0.2
LOR	-9.164160	5.747726	3.095965	0.678352	0.660025	0.807262	0.359084	0.1
CGPA	-3.144373	5.699742	2.998337	0.509117	0.431183	0.359084	0.355594	0.1
Research	-3.637845	3.318690	1.481729	0.255232	0.222807	0.177701	0.155026	0.2
Chance of Admit	0.698020	1.313271	0.685179	0.116009	0.097028	0.085834	0.074265	0.0

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

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

In [14]:
```
dataset = dataset.drop('Serial No.',axis=1)
```

In [15]:
```
X = dataset.drop('Chance of Admit ',axis=1)
y = dataset['Chance of Admit ']
```

In [28]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.20,random_sta
```

In [29]:
```
sc = StandardScaler()
```

In [30]:
```
X_train=sc.fit_transform(X_train)
X_test=sc.transform(X_test)
```

In [31]:
```
lr =LinearRegression()
lr.fit(X_train,y_train)

svm = SVR()
svm.fit(X_train,y_train)

rf = RandomForestRegressor()
rf.fit(X_train,y_train)

gr = GradientBoostingRegressor()
gr.fit(X_train,y_train)
```

Out[31]:
```
▾ GradientBoostingRegressor
GradientBoostingRegressor()
```

In [20]:
```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X_train,y_train, train_size
print('Train/Test Sets Sizes:', X_train.shape,  X_test.shape, Y_train.shape, Y_
```

```
Train/Test Sets Sizes: (256, 7) (64, 7) (256,) (64,)
```

In [21]:
```
from sklearn.ensemble import BaggingRegressor

bag_regressor = BaggingRegressor(random_state = 1)
bag_regressor.fit(X_train, Y_train)
```

Out[21]:
```
BaggingRegressor
BaggingRegressor(random_state=1)
```

In [22]:
```
Y_preds = bag_regressor.predict(X_test)

print('Training Coefficient of R2: %.3f'%bag_regressor.score(X_train, Y_train))
print('Test Coefficient of R2: %.3f'%bag_regressor.score(X_test, Y_test))
```

```
Training Coefficient of R2: 0.962
Test Coefficient of R2: 0.598
```

In [32]:
```
y_pred1 = lr.predict(X_test)
y_pred2 = svm.predict(X_test)
y_pred3 = rf.predict(X_test)
y_pred4 = gr.predict(X_test)
```

In [33]:
```
score1 = metrics.r2_score(y_test,y_pred1)
score2 = metrics.r2_score(y_test,y_pred2)
score3 = metrics.r2_score(y_test,y_pred3)
score4 = metrics.r2_score(y_test,y_pred4)
```

In [34]:
```
print(score1,score2,score3,score4)
```

```
0.8212082591486991 0.7597814848647667 0.8054682660745024 0.7937547395153746
```

In [35]:
```
final_data = pd.DataFrame({'Models':['LR','SVR','RF','GR'],
                           'R2_Score':[score1,score2,score3,score4]})
final_data
```

Out[35]:

	Models	R2_Score
0	LR	0.821208
1	SVR	0.759781
2	RF	0.805468
3	GR	0.793755

In [ ]:
```
# sns.barplot(final_data['Models'],final_data['R2_Score'])
# plt.show()
```

Classification

In [36]:
```
dataset.head()
```

Out[36]:

	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 [37]:
```
y_train = [1 if value>0.8 else 0 for value in y_train]
y_test = [1 if value>0.8 else 0 for value in y_test]

y_train = np.array(y_train)
y_test = np.array(y_test)
```

In [38]:
```
from sklearn.linear_model import LogisticRegression
from sklearn import svm
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
```

In [39]:
```
lr = LogisticRegression()
lr.fit(X_train,y_train)
y_pred1= lr.predict(X_test)
print(accuracy_score(y_test,y_pred1))
```

```
0.925
```

In [40]:
```
svm = svm.SVC()
svm.fit(X_train,y_train)
y_pred2 = svm.predict(X_test)
print(accuracy_score(y_test,y_pred2))
```

```
0.925
```

In [41]:
```
knn=KNeighborsClassifier()
knn.fit(X_train,y_train)
y_pred3 = knn.predict(X_test)
print(accuracy_score(y_test,y_pred3))
```

```
0.8875
```

In [42]:
```
rf = RandomForestClassifier()
rf.fit(X_train,y_train)
y_pred4 = rf.predict(X_test)
print(accuracy_score(y_test,y_pred4))
```

```
0.95
```

In [43]:
```
gr = GradientBoostingClassifier()
gr.fit(X_train,y_train)
y_pred5 = gr.predict(X_test)
print(accuracy_score(y_test,y_pred5))
```

```
0.975
```

In [44]:
```
final_data = pd.DataFrame({'Models':['LR','SVC','KNN','RF','GBC'],
```

```
                          'Accuracy_Score':[accuracy_score(y_test,y_pred1),
                                            accuracy_score(y_test,y_pred2),
                                            accuracy_score(y_test,y_pred3),
                                            accuracy_score(y_test,y_pred4),
                                            accuracy_score(y_test,y_pred5)]})
final_data
```

Out[44]:

	Models	Accuracy_Score
0	LR	0.9250
1	SVC	0.9250
2	KNN	0.8875
3	RF	0.9500
4	GBC	0.9750

In [ ]:
```
# sns.barplot(final_data['Models'],final_data['Accuracy_Score'])
# plt.show()
```

In [45]:
```
from sklearn.model_selection import StratifiedKFold
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.calibration import CalibratedClassifierCV
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.linear_model import SGDClassifier
from sklearn.svm import SVC
import xgboost as xgb
```

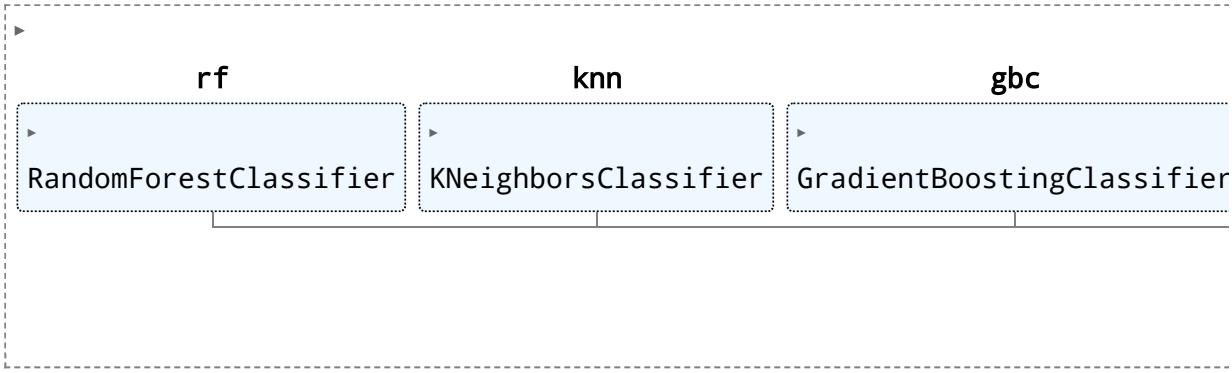
In [46]:
```
estimators = [
              ('rf', RandomForestClassifier(n_estimators = 10, random_state = 42
              ('knn', KNeighborsClassifier(n_neighbors = 10)),
              ('gbc', GradientBoostingClassifier()),
              ('lr', LogisticRegression()),
              ('ccv', CalibratedClassifierCV()),
              ('mlp', MLPClassifier()),
              ('dt', DecisionTreeClassifier()),
              ('lda', LinearDiscriminantAnalysis()),
              ('gnb', GaussianNB()),
              ('adb', AdaBoostClassifier()),
              ('etc', ExtraTreesClassifier()),
              ('sgd', SGDClassifier()),
              ('svm', SVC()),
              ('xgb', xgb.XGBClassifier(n_estimators= 10, random_state = 42))
              ]
```

In [47]:
```
from sklearn.ensemble import StackingClassifier

clf = StackingClassifier(
                         estimators = estimators,
```

```
                          final_estimator = LogisticRegression(),
                          cv = 10,
                          stack_method='predict',
                          n_jobs= -1
                          )
```

In [48]:
```
clf.fit(X_train,y_train)
```

Out[48]:



In [49]:
```
y_pred = clf.predict(X_test)
```

In [50]:
```
accuracy_score(y_test,y_pred)
```

Out[50]:
```
0.9375
```

In [60]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.model_selection import GridSearchCV
```

In [54]:
```
DTC_parameters = {
 'criterion' : ['gini', 'entropy', 'log_loss'],
 'splitter' : ['best', 'random'],
 'max_depth' : range(1,10,1),
 'min_samples_split' : range(2,10,2),
 'min_samples_leaf' : range(1,5,1),
 'max_features' : ['auto', 'sqrt', 'log2']
}

Bagging_parameters = {
 'n_estimators' : [5, 10, 15],
 'max_samples' : range(2, 10, 1),
 'max_features' : range(2, 10, 3)
}

RFC_parameters = {
 'criterion' : ['gini', 'entropy', 'log_loss'],
 'max_depth' : range(1, 10, 1),
 'min_samples_split' : range(2, 10, 2),
 'min_samples_leaf' : range(1, 10, 1),
}

ETC_parameters = {
 'n_estimators' : [10,20,30],
 'criterion' : ['gini', 'entropy', 'log_loss'],
```

```
 'max_depth' : range(2,10,1),
 'min_samples_split' : range(2,10,2),
 'min_samples_leaf' : range(1,5,1),
 'max_features' : ['sqrt', 'log2']
}
```

In [68]:
```
hyper1 = GridSearchCV(estimator = DecisionTreeClassifier(), param_grid = DTC_pa
hyper2 = GridSearchCV(estimator = BaggingClassifier(), param_grid = Bagging_par
hyper3 = GridSearchCV(estimator = RandomForestClassifier(), param_grid = RFC_pa
hyper4 = GridSearchCV(estimator = ExtraTreesClassifier(), param_grid = ETC_para
```

In [78]:
```
hyper1.fit(X_train,y_train)
hyper2.fit(X_train,y_train)
hyper3.fit(X_train,y_train)
hyper4.fit(X_train,y_train)
```

In [82]:
```
h_pred1 = hyper1.predict(X_test)
h_pred2 = hyper1.predict(X_test)
h_pred3 = hyper1.predict(X_test)
h_pred4 = hyper1.predict(X_test)
```

In [86]:
```
accuracy_score(y_test,h_pred1)
accuracy_score(y_test,h_pred2)
accuracy_score(y_test,h_pred3)
accuracy_score(y_test,h_pred4)
```

In [89]:
```
X = dataset.drop('Chance of Admit ',axis=1)
y = dataset['Chance of Admit ']
```

In [90]:
```
y  = [1 if value>0.8 else 0 for value in y]
```

In [91]:
```
y = np.array(y)
```

In [92]:
```
X = sc.fit_transform(X)
```

In [93]:
```
gr = GradientBoostingClassifier()
gr.fit(X,y)
```

Out[93]:
```
▾ GradientBoostingClassifier
GradientBoostingClassifier()
```

In [94]:
```
import joblib
```

In [95]:
```
joblib.dump(gr,'admission_model')
```

Out[95]:
```
['admission_model']
```

In [96]:
```
model = joblib.load('admission_model')
```

In [97]:
```
model.predict(sc.transform([[337,118,4,4.5,4.5,9.65,1]]))
```

```
C:\Users\prasad jadhav\AppData\Local\Programs\Python\Python310\lib\site-packag
es\sklearn\base.py:450: UserWarning: X does not have valid feature names, but
StandardScaler was fitted with feature names
  warnings.warn(
```

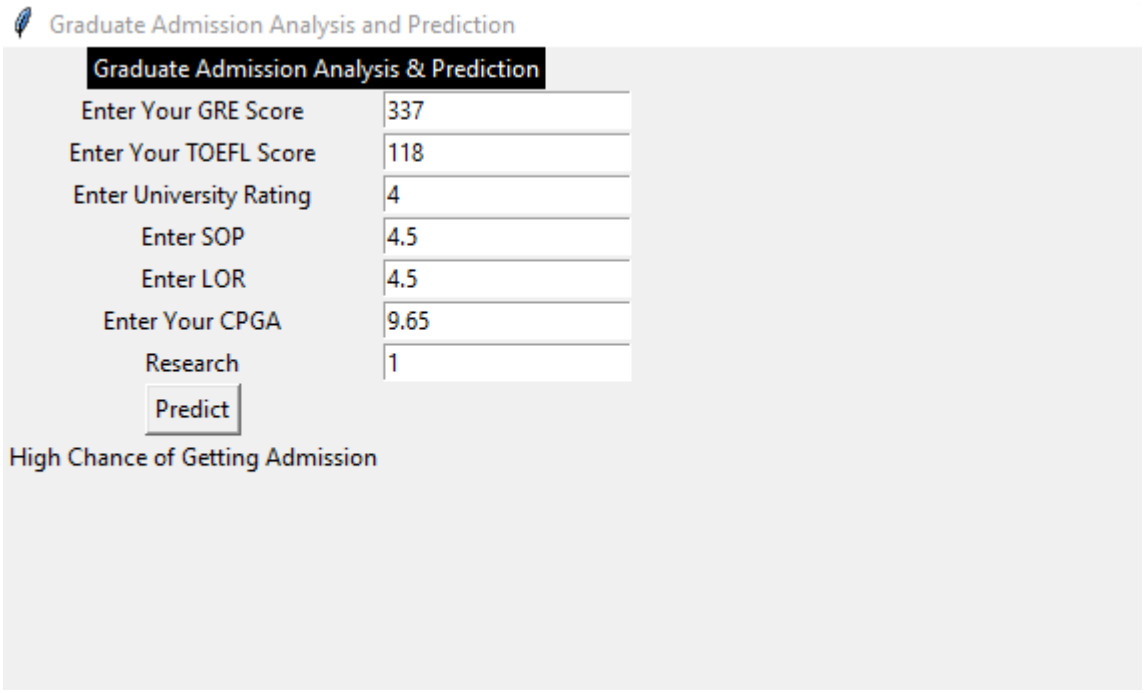
Out[97]:
```
array([1])
```

In [98]:
```
from tkinter import *
import joblib
from sklearn.preprocessing import StandardScaler
```

In [100…
```
def show_entry():

    p1 = float(e1.get())
    p2 = float(e2.get())
    p3 = float(e3.get())
    p4 = float(e4.get())
    p5 = float(e5.get())
    p6 = float(e6.get())
    p7 = float(e6.get())

    model = joblib.load('admission_model')
    result = model.predict(sc.transform([[p1,p2,p3,p4,p5,p6,p7]]))

    if result == 1:
        Label(master, text='High Chance of Getting Admission').grid(row=31)
    else:
        Label(master, text='You May Get Admission').grid(row=31)

master =Tk()
master.title('Graduate Admission Analysis and Prediction')
label = Label(master,text = 'Graduate Admission Analysis & Prediction',bg = "bl
               fg = "white").grid(row=0,columnspan=2)

Label(master,text = 'Enter Your GRE Score').grid(row=1)
Label(master,text = 'Enter Your TOEFL Score').grid(row=2)
Label(master,text = 'Enter University Rating').grid(row=3)
Label(master,text = 'Enter SOP').grid(row=4)
Label(master,text = 'Enter LOR').grid(row=5)
Label(master,text = 'Enter Your CPGA').grid(row=6)
Label(master,text = 'Research').grid(row=7)


e1 = Entry(master)
e2 = Entry(master)
e3 = Entry(master)
e4 = Entry(master)
e5 = Entry(master)
e6 = Entry(master)
e7 = Entry(master)

e1.grid(row=1,column=1)
e2.grid(row=2,column=1)
e3.grid(row=3,column=1)
e4.grid(row=4,column=1)
e5.grid(row=5,column=1)
e6.grid(row=6,column=1)
e7.grid(row=7,column=1)
```

```
Button(master,text='Predict',command=show_entry).grid()

mainloop()
```

```
C:\Users\prasad jadhav\AppData\Local\Programs\Python\Python310\lib\site-packag
es\sklearn\base.py:450: UserWarning: X does not have valid feature names, but
StandardScaler was fitted with feature names
  warnings.warn(
```



Thank You

CONNECT WITH ME :

LinkedIn GitHub kaggle Medium

PRASADMJADHAV2