In [ ]:

```
import numpy as np
import pandas as pd
#import os
from matplotlib import pyplot as plt
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
import seaborn as sns
sns.set(style='white')
sns.set(style='whitegrid', color_codes=True)
```

In [ ]:

```
df = pd.read_csv("/content/Admission_Predict.csv")
print('Head\n',df.head())
print('Columns\n',df.columns)
df.head()
```

```
Head
    Serial No.  GRE Score  TOEFL Score  University Rating  SOP  LOR   CGPA
\
0           1        337          118                  4  4.5   4.5  9.65
1           2        324          107                  4  4.0   4.5  8.87
2           3        316          104                  3  3.0   3.5  8.00
3           4        322          110                  3  3.5   2.5  8.67
4           5        314          103                  2  2.0   3.0  8.21

   Research  Chance of Admit
0         1              0.92
1         1              0.76
2         1              0.72
3         1              0.80
4         0              0.65
Columns
 Index(['Serial No.', 'GRE Score', 'TOEFL Score', 'University Rating', 'SO
P',
       'LOR ', 'CGPA', 'Research', 'Chance of Admit '],
      dtype='object')
```

Out[36]:

	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 [ ]:

```
from google.colab import drive
drive.mount('/content/gdrive')
df = pd.read_csv('/content/gdrive/MyDrive/Admission_Predict.csv')
print('Head\n',df.head())
print('Columns\n',df.columns)
df.head()
```

In [ ]:

```
df.describe()
```

Out[7]:

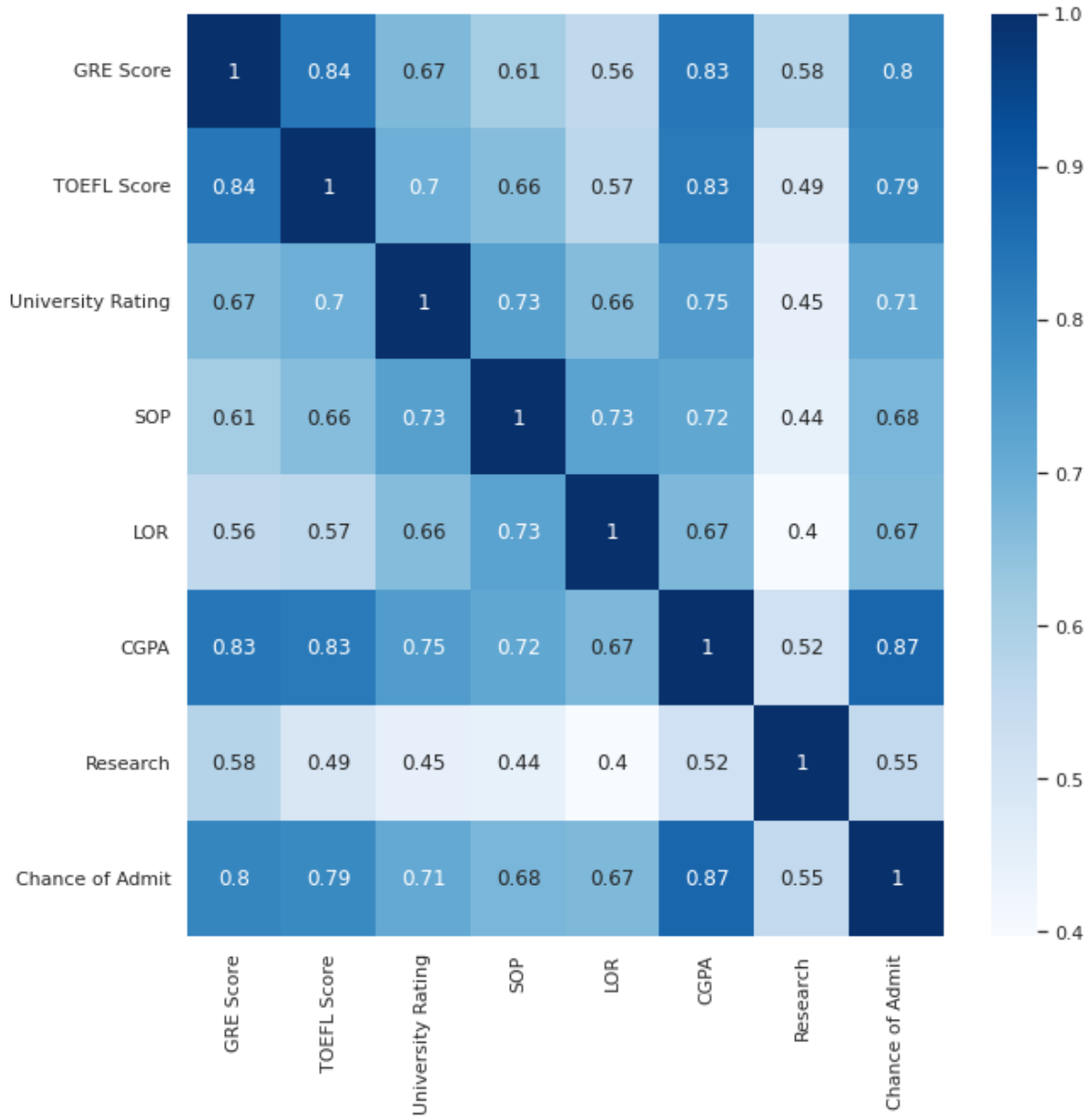
	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	R
count	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	40
mean	200.500000	316.807500	107.410000	3.087500	3.400000	3.452500	8.598925	
std	115.614301	11.473646	6.069514	1.143728	1.006869	0.898478	0.596317	
min	1.000000	290.000000	92.000000	1.000000	1.000000	1.000000	6.800000	
25%	100.750000	308.000000	103.000000	2.000000	2.500000	3.000000	8.170000	
50%	200.500000	317.000000	107.000000	3.000000	3.500000	3.500000	8.610000	
75%	300.250000	325.000000	112.000000	4.000000	4.000000	4.000000	9.062500	
max	400.000000	340.000000	120.000000	5.000000	5.000000	5.000000	9.920000	

In [ ]:

```
df.rename(columns = {'Chance of Admit ':'Chance of Admit', 'LOR ':'LOR'}, inplace=True)
df.drop(labels='Serial No.', axis=1, inplace=True)
```

In [ ]:

```
fig, ax = plt.subplots(figsize=(10,10))
sns.heatmap(df.corr(), annot=True, cmap='Blues');
```



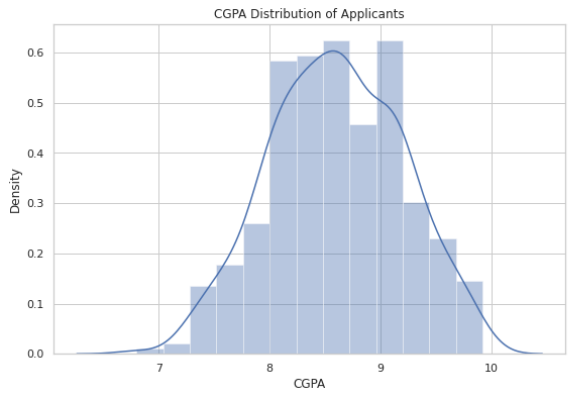
In [ ]:

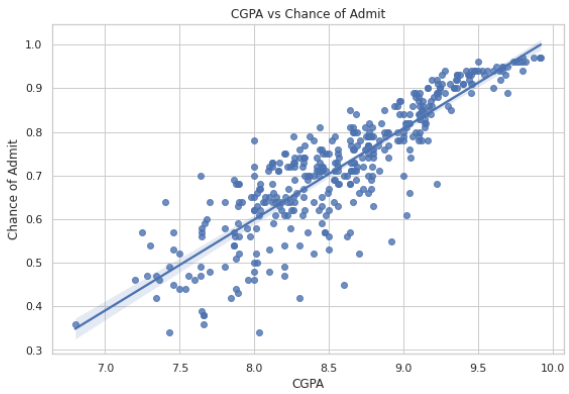
```
plt.figure(figsize=(20,6))
plt.subplot(1,2,1)
sns.distplot(df['CGPA'])
plt.title('CGPA Distribution of Applicants')

plt.subplot(1,2,2)
sns.regplot(df['CGPA'], df['Chance of Admit'])
plt.title('CGPA vs Chance of Admit')
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: Future
Warning: `distplot` is a deprecated function and will be removed in a future
version. Please adapt your code to use either `displot` (a figure-level func
tion with similar flexibility) or `histplot` (an axes-level function for his
tograms).
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarn
ing: Pass the following variables as keyword args: x, y. From version 0.12,
the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[39]:

```
Text(0.5, 1.0, 'CGPA vs Chance of Admit')
```





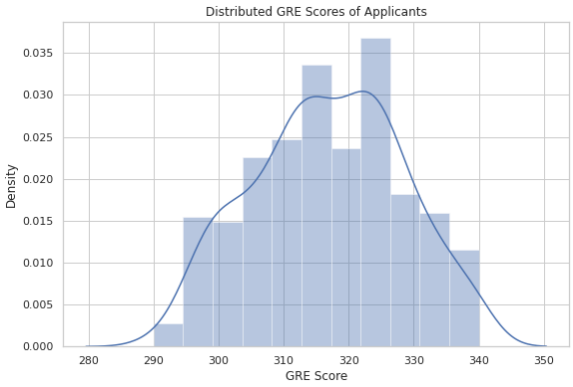
In [ ]:

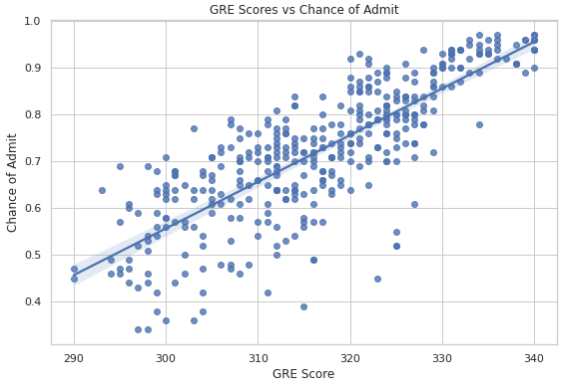
```
plt.figure(figsize=(20,6))
plt.subplot(1,2,1)
sns.distplot(df['GRE Score'])
plt.title('Distributed GRE Scores of Applicants')

plt.subplot(1,2,2)
sns.regplot(df['GRE Score'], df['Chance of Admit'])
plt.title('GRE Scores vs Chance of Admit')
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: Future
Warning: `distplot` is a deprecated function and will be removed in a future
version. Please adapt your code to use either `displot` (a figure-level func
tion with similar flexibility) or `histplot` (an axes-level function for his
tograms).
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarn
ing: Pass the following variables as keyword args: x, y. From version 0.12,
the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[40]:

```
Text(0.5, 1.0, 'GRE Scores vs Chance of Admit')
```





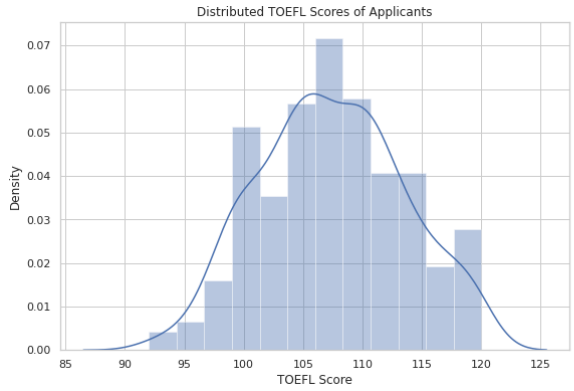
In [ ]:

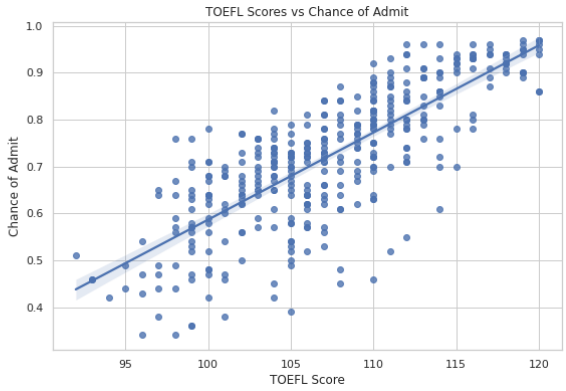
```
plt.figure(figsize=(20,6))
plt.subplot(1,2,1)
sns.distplot(df['TOEFL Score'])
plt.title('Distributed TOEFL Scores of Applicants')

plt.subplot(1,2,2)
sns.regplot(df['TOEFL Score'], df['Chance of Admit'])
plt.title('TOEFL Scores vs Chance of Admit')
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: Future
Warning: `distplot` is a deprecated function and will be removed in a future
version. Please adapt your code to use either `displot` (a figure-level func
tion with similar flexibility) or `histplot` (an axes-level function for his
tograms).
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarn
ing: Pass the following variables as keyword args: x, y. From version 0.12,
the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[41]:

```
Text(0.5, 1.0, 'TOEFL Scores vs Chance of Admit')
```





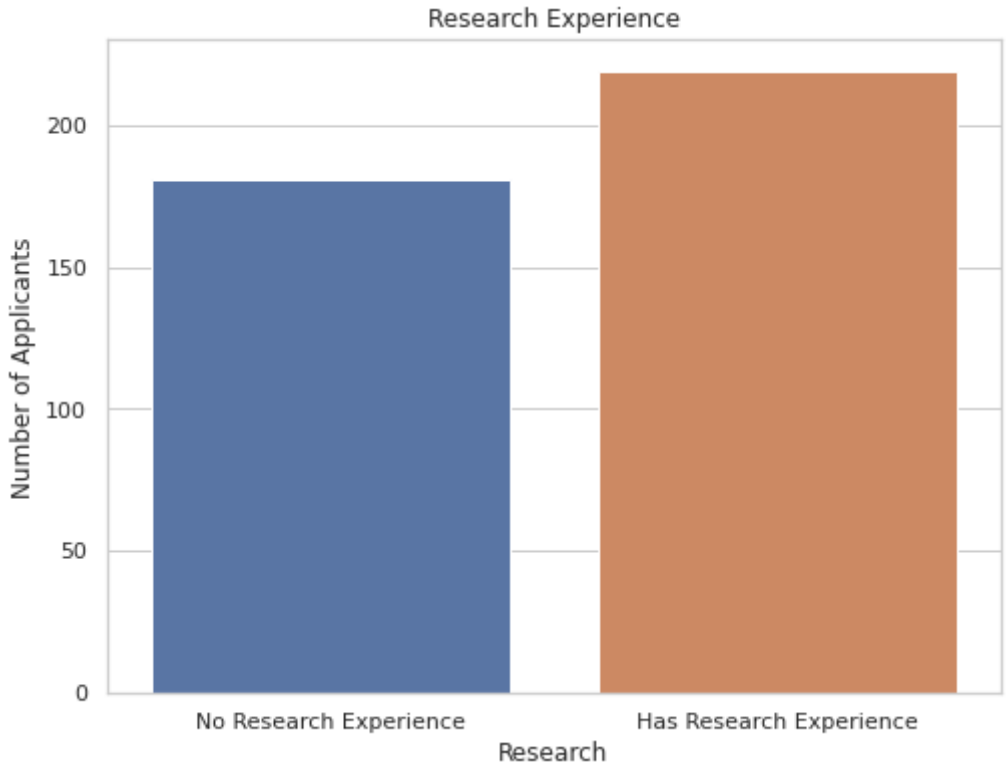
In [ ]:

```
fig, ax = plt.subplots(figsize=(8,6))
sns.countplot(df['Research'])
plt.title('Research Experience')
plt.ylabel('Number of Applicants')
ax.set_xticklabels(['No Research Experience', 'Has Research Experience'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarn
ing: Pass the following variable as a keyword arg: x. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments w
ithout an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[42]:

```
[Text(0, 0, 'No Research Experience'), Text(0, 0, 'Has Research Experienc
e')]
```



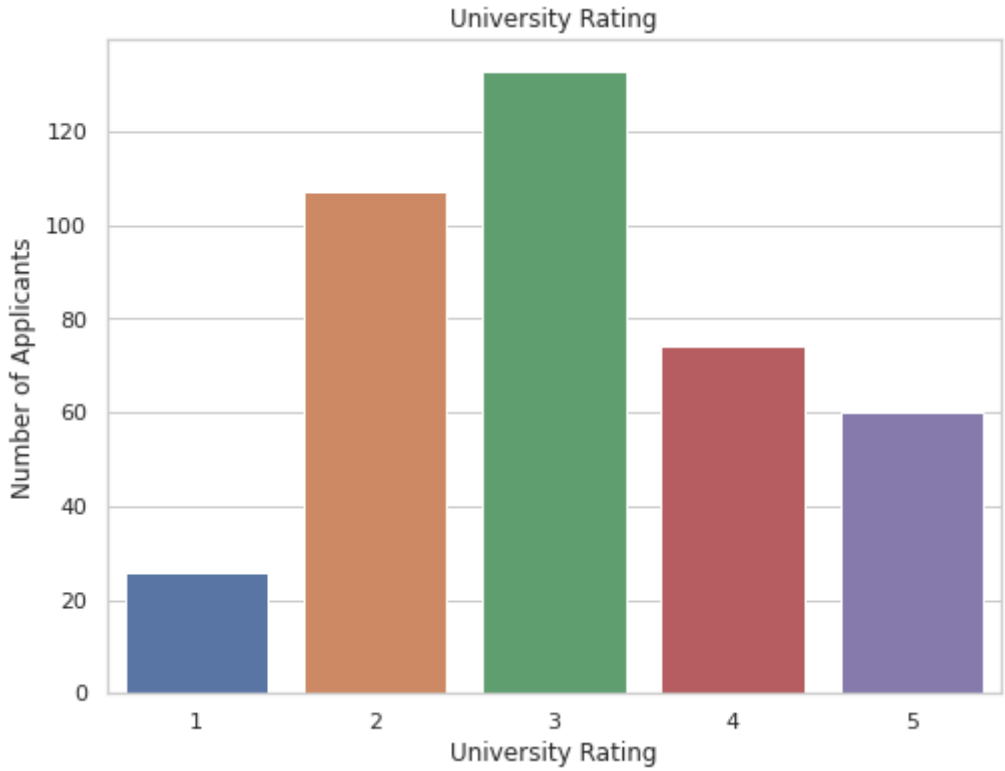
In [ ]:

```
fig, ax = plt.subplots(figsize=(8,6))
sns.countplot(df['University Rating'])
plt.title('University Rating')
plt.ylabel('Number of Applicants')
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarn
ing: Pass the following variable as a keyword arg: x. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments w
ithout an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[43]:

```
Text(0, 0.5, 'Number of Applicants')
```



Accuracy

In [ ]:

```
targets = df['Chance of Admit']
features = df.drop(columns = {'Chance of Admit'})

X_train, X_test, y_train, y_test = train_test_split(features, targets, test_size=0.2, randc
```

In [ ]:

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

In [ ]:

```
linreg = LinearRegression()
linreg.fit(X_train, y_train)
y_predict = linreg.predict(X_test)
linreg_score = (linreg.score(X_test, y_test))*100
linreg_score
```

Out[46]:

```
81.7386788111443
```

In [ ]:

```
dec_tree = DecisionTreeRegressor(random_state=0, max_depth=6)
dec_tree.fit(X_train, y_train)
y_predict = dec_tree.predict(X_test)
dec_tree_score = (dec_tree.score(X_test, y_test))*100
dec_tree_score
```

Out[47]:

```
73.99851580517213
```

In [ ]:

```
forest = RandomForestRegressor(n_estimators=110,max_depth=6,random_state=0)
forest.fit(X_train, y_train)
y_predict = forest.predict(X_test)
forest_score = (forest.score(X_test, y_test))*100
forest_score
```

Out[48]:

```
81.34052472373693
```

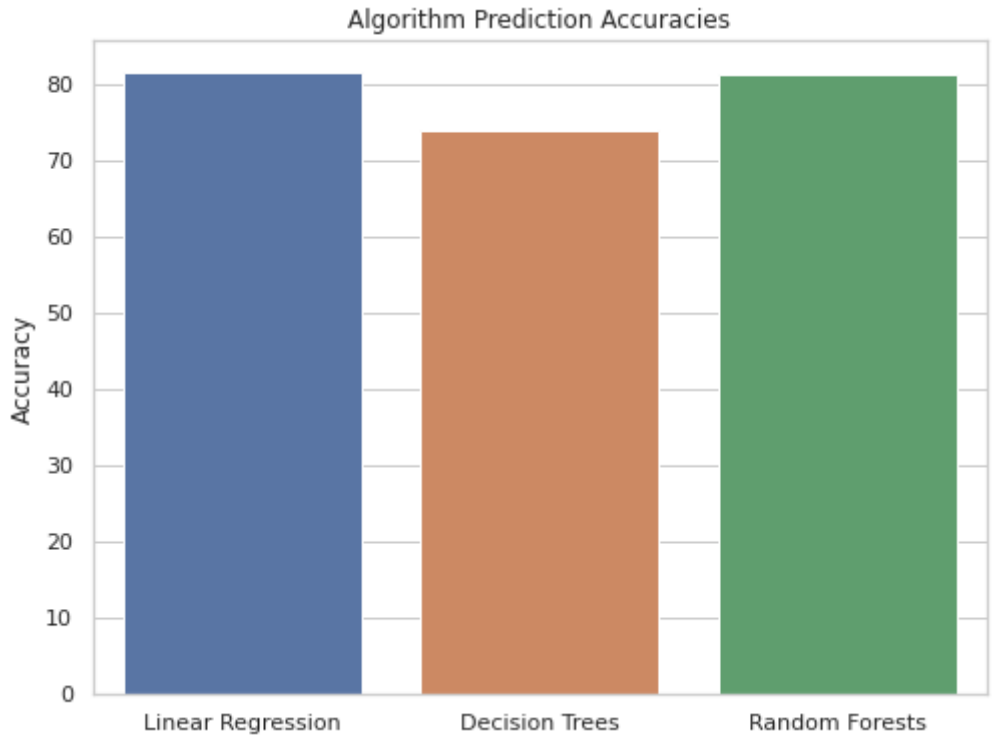
In [ ]:

```
Methods = ['Linear Regression', 'Decision Trees', 'Random Forests']
Scores = np.array([linreg_score, dec_tree_score, forest_score])

fig, ax = plt.subplots(figsize=(8,6))
sns.barplot(Methods, Scores)
plt.title('Algorithm Prediction Accuracies')
plt.ylabel('Accuracy')
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarn
ing: Pass the following variables as keyword args: x, y. From version 0.12,
the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[20]:

```
Text(0, 0.5, 'Accuracy')
```



Prediction

In [ ]:

```
x_input = np.matrix([df["GRE Score"].tolist(),df["TOEFL Score"].tolist(),df["University Rat
y_output = np.matrix(df["Chance of Admit"].tolist()).T
```

In [ ]:

```
x_input.shape
```

Out[50]:

```
(400, 7)
```

In [ ]:

```
y_output.shape
```

Out[51]:

```
(400, 1)
```

In [ ]:

```
def coefficient(x,y):
  beta = np.dot((np.dot((np.linalg.inv(np.dot((x.T),x))),x.T)),y)
  return  beta
```

In [ ]:

```
beta = coefficient(x_input,y_output).T
beta
```

Out[53]:

```
matrix([[-0.00291067,  0.00323154,  0.01990962,  0.00057609,  0.02319267,
          0.1308982 ,  0.05682043]])
```

In [ ]:

```
beta.shape
```

Out[54]:

```
(1, 7)
```

In [ ]:

```
pred_input = np.array([0,0,0,0,0,0,0])
while(1):
  x = float(input("Enter GRE score :"))
  if x>=0 and x<=340 :
    pred_input[0] = x
    break
  else:
    print("Enter Valid score?")
while(1):
  x = float(input("Enter TOEFL score :"))
  if x>=0 and x<=120 :
    pred_input[1] = x
    break
  else:
    print("Enter TOEFL score?")
while(1):
  x = float(input("Enter University Rating :"))
  if x>=0 and x<=5 :
    pred_input[2] = x
    break
  else:
    print("Enter Valid University Rating?")
while(1):
  x = float(input("Enter SOP Rating :"))
  if x>=0 and x<=5 :
    pred_input[3] = x
    break
  else:
    print("Enter Valid SOP score?")
while(1):
  x = float(input("Enter LOR Rating :"))
  if x>=0 and x<=5 :
    pred_input[4] = x
    break
  else:
    print("Enter Valid LOR score?")
while(1):
  x = float(input("Enter CGPA Rating :"))
  if x>=0 and x<=10 :
    pred_input[5] = x
    break
  else:
    print("Enter Valid CGPA score?")
while(1):
  x = float(input("Research: Enter 1 if yes else 0 :"))
  if x>=0 and x<=1 :
    pred_input[6] = x
    break
  else:
    print("Enter Valid Responce")
X_samp = [pred_input]
```

```
Enter GRE score :300
Enter TOEFL score :100
Enter University Rating :4.5
Enter SOP Rating :4.2
Enter LOR Rating :4.5
Enter CGPA Rating :9.2
Research: Enter 1 if yes else 0 :1
```

Output for above values using decision tree linear regression and random forest

In [ ]:

```
y_dec_tree = dec_tree.predict(X_samp)
print(y_dec_tree*100)
```

```
[96.5]
```

In [ ]:

```
y_predict = linreg.predict(X_samp)
print((y_predict/10)*100)
```

```
[93.70423489]
```

In [ ]:

```
y_predict = forest.predict(X_samp)
print(y_predict*100)
```

```
[96.66729021]
```

In [ ]:

```

```