مقدمه:

در این پروژه هدف یادگیری و استفاده از لایبرری های زبان Python استکه در آن از Matplot, Numpy, Pandas نیز استفاده شده است.
در این پروژه یک دیتا ست از دانشجو ها با ۸ ستون داده و تعداد ۴۰۰ تا ردیف در اختیار داریم که این دیتا ست ناقص است و ابتدا باید آن را کامل کنیم و سپس شانس قبولی برخی دانشجو ها که در دیتا ست معلوم نشده است را با استفاده از نمودار و یک تقریب خطی از آن بدست اوریم.

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

قسمت اول:

ابتدا فایل هاوی دیتا ست را با استفاده از Pandas می خوانیم و ان را به صورت یک دیتا فریم pandas در می اوریم:

```
df = pd.read_csv("AdmissionPredict.csv")
print(df)
```

سپس برای این دیتا فریم ۴ تابع Head, Tail, Describe, Info را صدا می کنیم:
Head, Tail: تابع head تعداد ردیف های ورودی خود را از شماره ۱ دیتافریم (از سر آن) جدا می کند و تابع tail بر عکس آن است یعنی تعداد ورودی خود را از آخر دیتا فریم جدا می کند مانند زیر:

```
# Using Head/Tail Function to see 5 first Rows/2 Last Row of DataFrame
print(df.head(5))
print(df.tail(2))
```

```
[400 rows x 9 columns]
   Serial No.  GRE Score  TOEFL Score  ...  CGPA  Research  Chance of Admit
0           1      337.0        118.0  ...  9.65         1              0.92
1           2      324.0        107.0  ...  8.87         1               NaN
2           3      316.0          NaN  ...  8.00         1              0.72
3           4        NaN        110.0  ...  8.67         1              0.80
4           5      314.0        103.0  ...  8.21         0              0.65

[5 rows x 9 columns]
     Serial No.  GRE Score  TOEFL Score  ...  CGPA  Research  Chance of Admit
398         399      312.0        103.0  ...  8.78         0              0.67
399         400      333.0        117.0  ...  9.66         1              0.95
>>>
```

Info: تابع info برای یک دیتافریم نوع دیتا فریم تعداد ردیف های آن تعداد ستون های آن به همراه نام های ستون ها تعداد NaN ها از هر ستون و نوع دیتا تایو آن ستون را مشخص می کند همچنین مموری استفاده شده برای ذخیره این دیتا فریم را نیز معلوم می کند مانند زیر:

```
# Using info to find How many NaN are in each columns
print(df.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          378 non-null    float64
 2   TOEFL Score        380 non-null    float64
 3   University Rating  400 non-null    int64
 4   SOP                400 non-null    float64
 5   LOR                400 non-null    float64
 6   CGPA               380 non-null    float64
 7   Research           400 non-null    int64
 8   Chance of Admit    384 non-null    float64
dtypes: float64(6), int64(3)
memory usage: 28.2 KB
None
```

Describe: تابع describe برای یک دیتا فریم برای هر ستون آن تعداد ردیف ها (به جز NaN ها) مقدار میانگین آن مینیموم ۲۵٪ ۵۰٪ ۷۵٪ و ماکسیموم آن را مشخص می کند مانند زیر:

```
       Serial No.   GRE Score  ...    Research  Chance of Admit
count  400.000000  378.000000  ...  400.000000       384.000000
mean   200.500000  316.759259  ...    0.547500         0.724375
std    115.614301   11.415599  ...    0.498362         0.142964
min      1.000000  290.000000  ...    0.000000         0.340000
25%    100.750000  308.250000  ...    0.000000         0.640000
50%    200.500000  317.000000  ...    1.000000         0.730000
75%    300.250000  325.000000  ...    1.000000         0.830000
max    400.000000  340.000000  ...    1.000000         0.970000
```

برای بدست آوردن تعداد مقادیر NaN هر ستون می توان از همان تابع Info استفاده کرد ولی به صورت جدا و فقط مجموع این تعداد را نیز می توان به صورت زیر نیز بدست آورد:

```
# Because of info we know which columns have NaN Data
print("GRE Score NaN Number is:", df["GRE Score"].isna().sum())
print("TOEFL NaN Number is:", df["TOEFL Score"].isna().sum())
print("CGPA NaN Number is:", df["CGPA"].isna().sum())
```

که در این صورت در خروجی داریم:

```
GRE Score NaN Number is: 22
TOEFL NaN Number is: 20
CGPA NaN Number is: 20
```

سپس می خواهیم این مقادیر NaN را در هر ستون جایگزین کنیم
برای این کار از مقداری میانگین تمام عضای عمان ستون بدون در نظر گرفتن مقدار NaN ها استفاده می کنیم و این مقدار را برای تمامی مقادیر NaN هر ستون قرار می دهیم مانند زیر:

```
# Filling Every NaN with its column mean except last column
df = df.fillna(df.loc[:, ["GRE Score", "TOEFL Score", "CGPA"]].mean())
print("DataFrame with NaN Replaced")
print(df)  # to Show NaN are Replaced
```

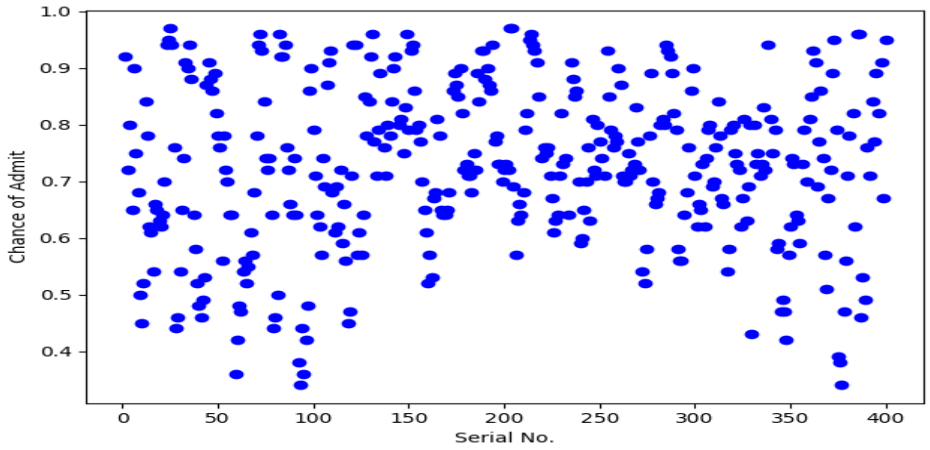
که در این صورت تمام مقادیر درست جایگزین می شوند و می دانیم که Chance of Admit جایگزین نکردیم زیرا این مقدار مجهول ما هست و در قسمت های بعدی باید آن را بدست آوریم

قسمت دوم:

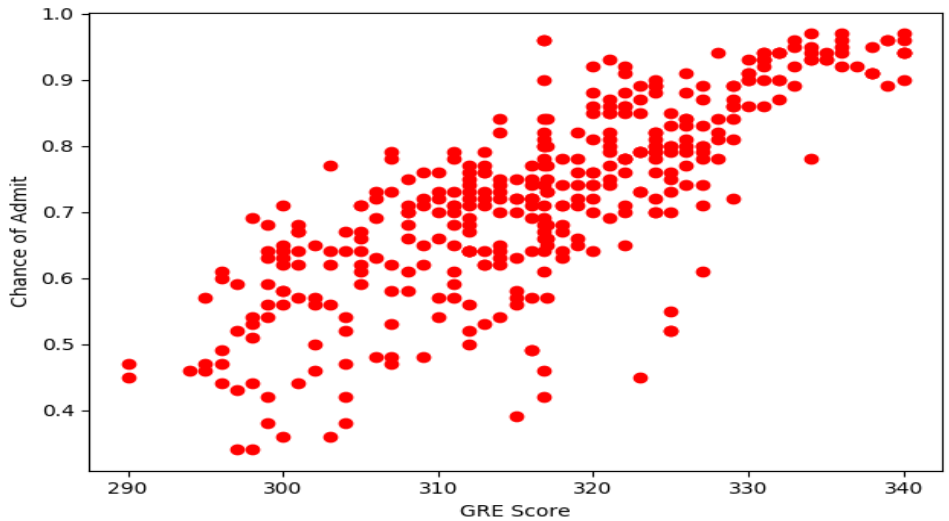
در این قسمت می خواهیم برای هر یک از مشخضه هایی که داریم نسبت به شانس پذیرش آن یک نمودار scatter رسم کنیم و مشخصه که بیشترین هم بستگی به آن را دارد در یابیم

برای رسم نمودار ها از Matplotlib.pyplot که برای رسم نمودار است استفاده می کنیم و با دادن نام برای محور های x y هر نمودار و شکل نقاط آن این نمودار ها را رسم می کنیم در این صورت ۸ نمودار که هر کدام یک ستون بر حسب شانس پذیرش را نشان می دهد داریم:

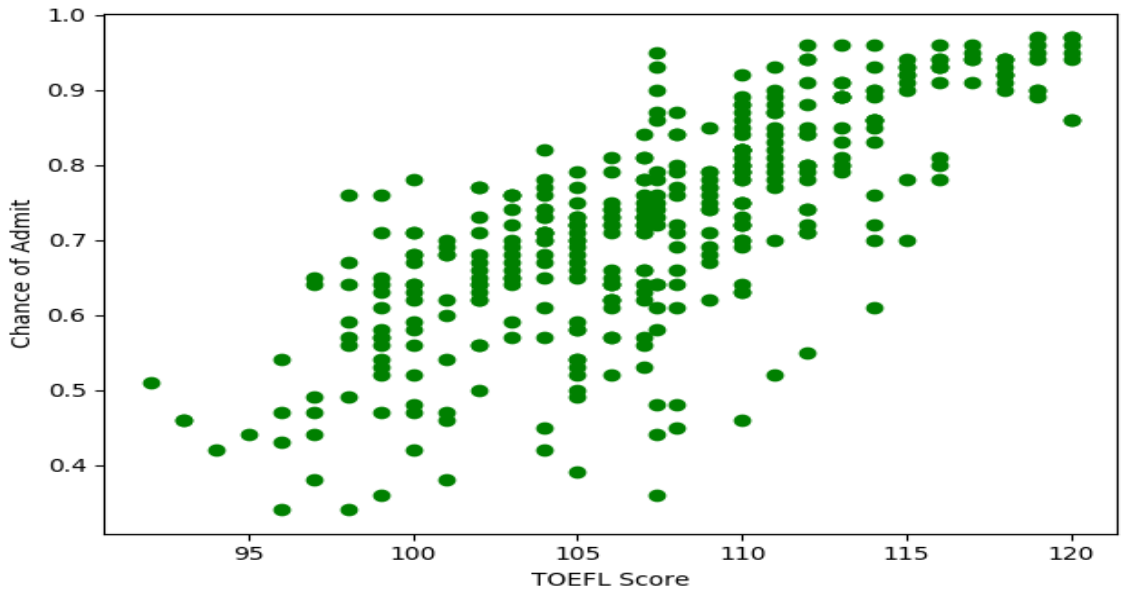
```
plt.xlabel('Serial No.')
plt.ylabel('Chance of Admit')
plt.plot(df["Serial No."], df["Chance of Admit"], 'bo')
plt.show()
```



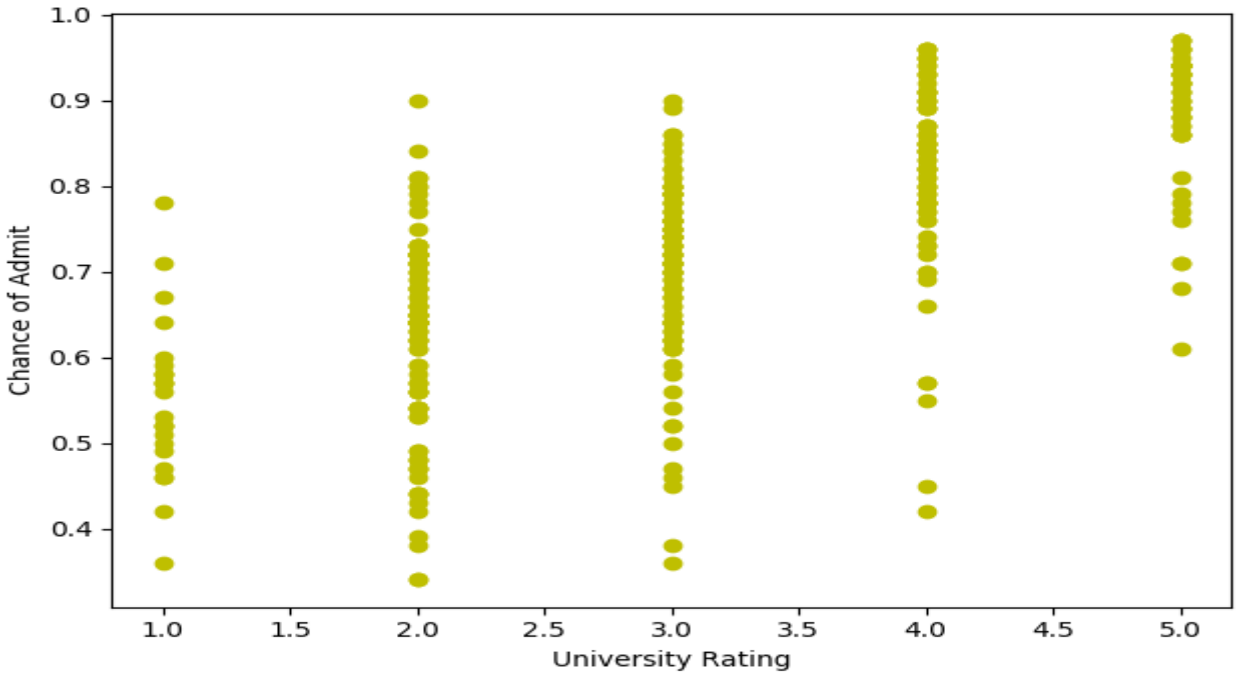

```
plt.xlabel('GRE Score')
plt.ylabel('Chance of Admit')
plt.plot(df["GRE Score"], df["Chance of Admit"], 'ro')
plt.show()
```



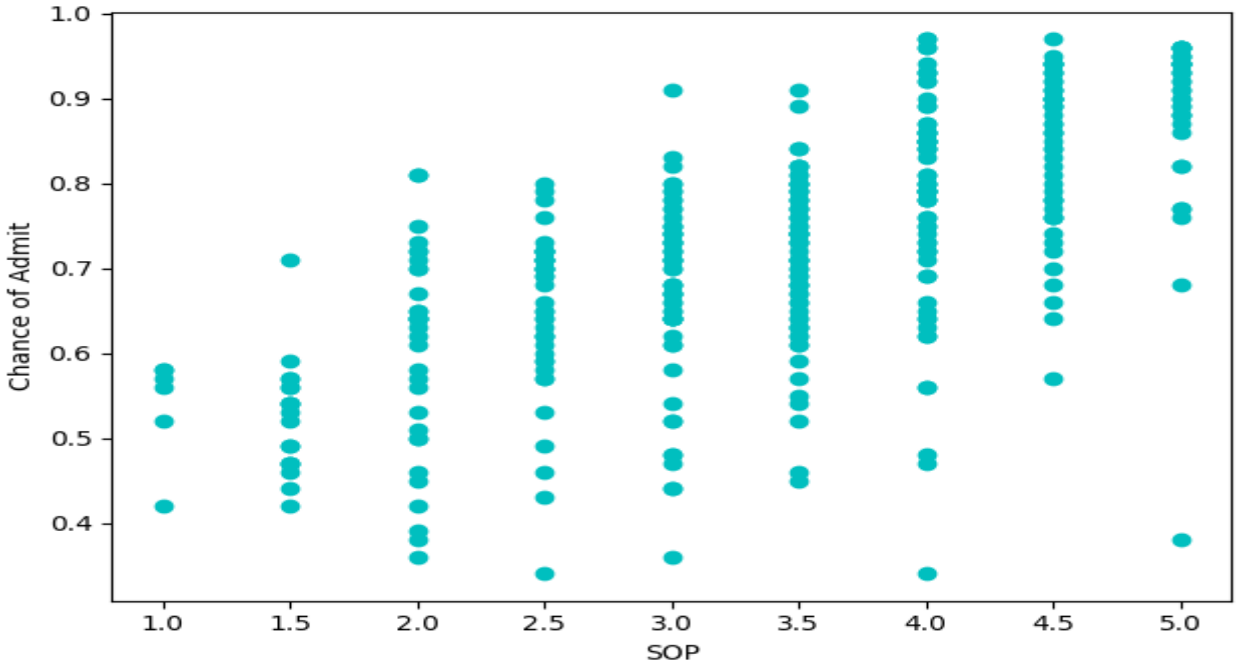
```
plt.xlabel('TOEFL Score')
plt.ylabel('Chance of Admit')
plt.plot(df["TOEFL Score"], df["Chance of Admit"], 'go')
plt.show()
```



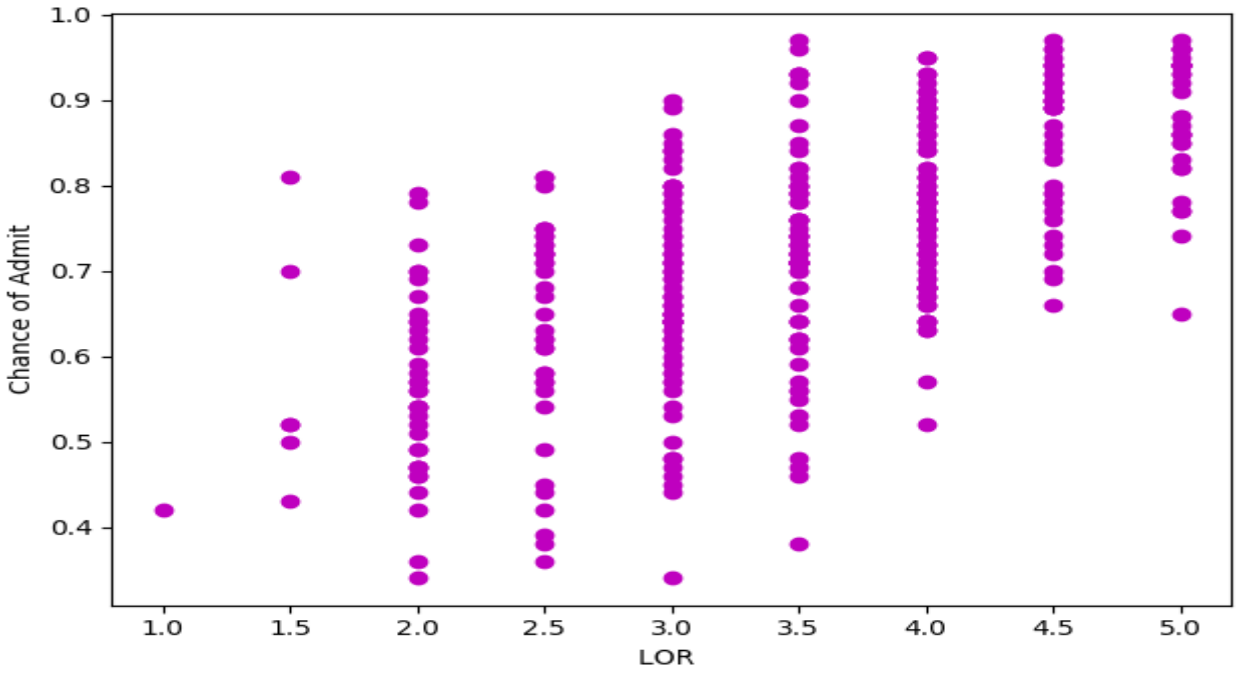

```
plt.xlabel('University Rating')
plt.ylabel('Chance of Admit')
plt.plot(df["University Rating"], df["Chance of Admit"], 'yo')
plt.show()
```



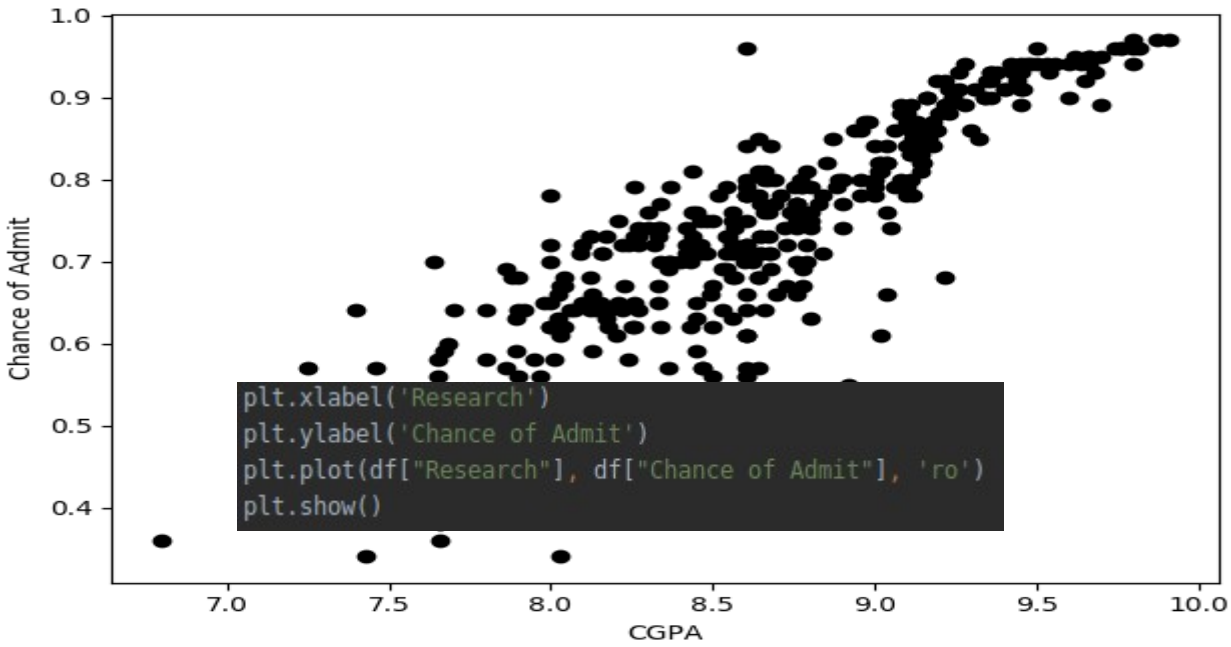
```
plt.xlabel('SOP')
plt.ylabel('Chance of Admit')
plt.plot(df["SOP"], df["Chance of Admit"], 'co')
plt.show()
```



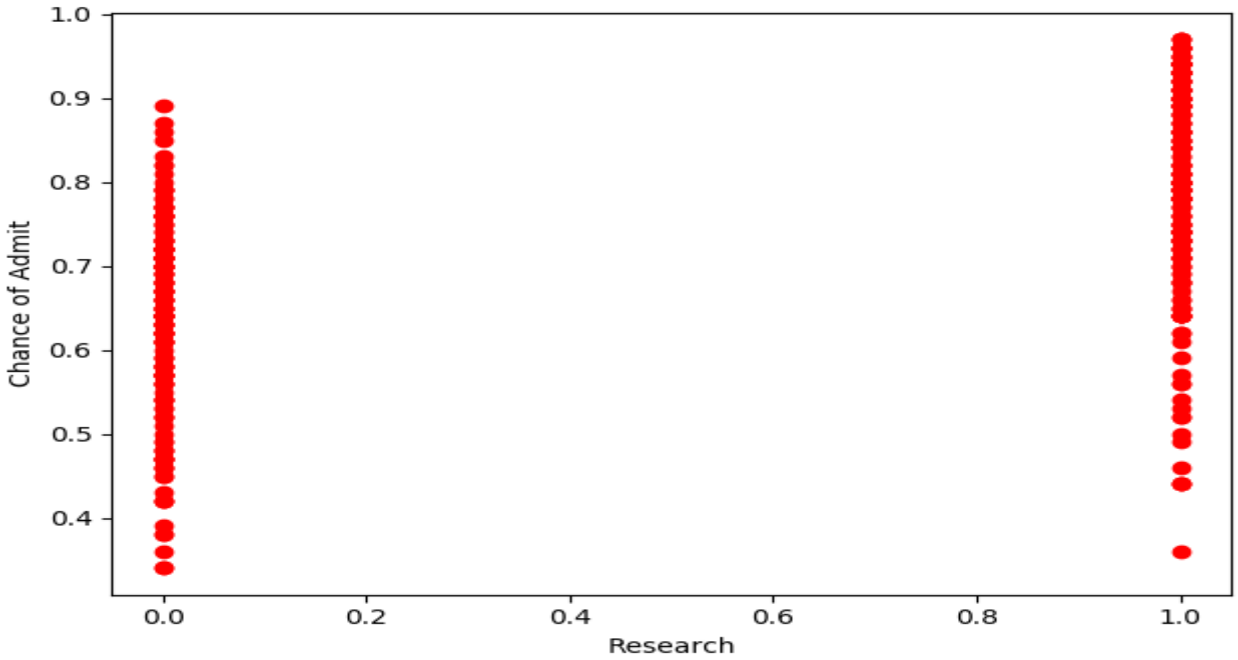

```
plt.xlabel('LOR ')
plt.ylabel('Chance of Admit')
plt.plot(df["LOR "], df["Chance of Admit"], 'mo')
plt.show()
```




```
plt.xlabel('CGPA')
plt.ylabel('Chance of Admit')
plt.plot(df["CGPA"], df["Chance of Admit"], 'ko')
plt.show()
```




```
plt.xlabel('Research')
plt.ylabel('Chance of Admit')
plt.plot(df["Research"], df["Chance of Admit"], 'ro')
plt.show()
```



همان طور که از نمودار ها معلوم است بیشترین همبستگی به طور چشمی مربوط به متغیر CGPA است. این متغیر به طور چشمی چون هر چقدر بیشتر می شود شانس قبولی نیز بیشتر می شود بیشترین همبستگی را دارد در بقیه نمودار ها این همبستگی کمتر است یعنی در آن ها در بعضی موارد با بیشتر شدن آن متغیر شانس قبولی ان دانشجو کمتر شده است که از میزان همبستگی آن متغیر با شانس پذیرش می کاهد

قسمت سوم:

در این قسمت می خواهیم برخی قسمت های دیتافریم را فیلتر کنیم که در آن با کتابخانه های Pandas Numpy استفاده کرده ایم

در قسمت اول می خواهیم دانشجویان با CPGA بالای ۹ و TOEFL بالای ۱۱۰ را از بقیه جدا کنیم باری این کار از کد زیر استفاده می کنیم:

```
# # Filter Accepted Students
FilterAccepted = df[(df['CGPA'] >= 9) & (df['TOEFL Score'] >= 110)]
print(FilterAccepted)
print("Number of Accepted Students are " + str(FilterAccepted['Serial No.'].count()))
```

که تعداد آن و خود آنها در خروجی مانند زیر آورده شده است:

```
     Serial No.   GRE Score  TOEFL Score  ...  CGPA  Research  Chance of Admit
0              1  337.000000        118.0  ...  9.65         1              0.92
5              6  330.000000        115.0  ...  9.34         1              0.90
11            12  316.759259        111.0  ...  9.00         1              0.84
12            13  328.000000        112.0  ...  9.10         1              0.78
22            23  328.000000        116.0  ...  9.50         1              0.94
..           ...         ...          ...  ...   ...       ...               ...
385          386  316.759259        117.0  ...  9.82         1              0.96
394          395  329.000000        111.0  ...  9.23         1              0.89
395          396  324.000000        110.0  ...  9.04         1              0.82
397          398  330.000000        116.0  ...  9.45         1              0.91
399          400  333.000000        117.0  ...  9.66         1              0.95

[97 rows x 9 columns]
Number of Accepted Students are 97
```

و برای قسمت بعد می خواهیم به ازای هر دانشگاه میانگین GRE ان را بدست آوریم که داریم:

```
for i in range(1, 6):
    x = df.loc[df['University Rating'] == i]['GRE Score'].mean()
    print("University Rate " + str(i) + " Mean GRE is " + str(x))
```

و در خروجی داریم:

```
University Rate 1 Mean GRE is 303.15384615384613
University Rate 2 Mean GRE is 309.7528556593977
University Rate 3 Mean GRE is 315.9346978557505
University Rate 4 Mean GRE is 324.07507507507506
University Rate 5 Mean GRE is 327.9546296296296
```

قسمت چهارم:

در این قسمت می خواهیم یک رگرسیون تک متغیره از CGPA که در قسمت قبل فهمیدیم بیشترین همبستگی را به شانس پذیرش دارد به دست اوریم
این خط باید منطبق بر نقاط دیگر باشد و یک معیاری از کل متغیر ها ی آن.
با استفاده از این خط می توانیم شانس آن دسته از دانشجویانی که شانس پذیرش آن ها مشخص نیست را با توجه به مقدار CGPA انها به صورت تقریبی بدست آوریم.

خطی که می خواهیم رسم کنیم به شکل زیر است:

$$y = teta1 * x + teta0$$

که x همان CGPA است و با این خط می توان شانس پذیرش را تخمین زد
برای پیدا کردن teta0 و teta1 باید خطای تابع هزینه را مینیموم کنیم تا به بهترین خط دست یابیم
برای این کار من از روش gradient decent استفاده کرده ام که با دادن ورودی های ان که در واقع ۳۸۴ تا CGPA مربوط به دانشجو ها هست مقدار تابع هزینه را مینیموم می کند
در این روش با پیدا کردن weight و bias که به ترتیب همان teta1 و teta0 است تابع هزینه را مینیموم می کند و در نتیجه بهترین خط که روی نمودار CGPA منطبق می شود را بدست آورده ایم
در این روش برای پیدا کردن weight و bias ابتدا یک تابع gradient نوشته و با دادن ورودی و train کردن ان بهترین خط را بدست می اوریم.

برای این کار ابتدا یک دیتا فریم جدید از CPGA و شانس پذیرش دانشجو ها درست می کنیم مانند زیر:

```
df_New = df[["CGPA", "Chance of Admit"]]
```

و سپس مقادیر ی که NaN هستندرا حذف می کنیم و دوباره ایندکس هارا ریست می کنیم و یک دیتافریم کامل بدست می اوریم:

```
df_New = df_New.dropna()
df_New = df_New.reset_index(drop=True)
```

سپس تابع هزینه را نوشته:

```
def cost_function(radio, sales, weight, bias):
    companies = len(radio)
    total_error = 0.0
    for i in range(companies):
        total_error += (-sales[i] + (weight * radio[i] + bias)) ** 2
    return total_error / (2 * companies)
```

و سپس تابع gradient را می نویسیم که برای update کردن همان weight و bias که همان teta0 و teta1 است مانند زیر:

```
def update_weights(radio, sales, weight, bias, learning_rate):
    weight_deriv = 0
    bias_deriv = 0
    companies = len(radio)

    for i in range(companies):
        # Calculate partial derivatives
        # -2x(y - (mx + b))
        weight_deriv += -2 * radio[i] * (sales[i] - (weight * radio[i] + bias))

        # -2(y - (mx + b))
        bias_deriv += -2 * (sales[i] - (weight * radio[i] + bias))

    # We subtract because the derivatives point in direction of steepest ascent
    weight -= (weight_deriv / companies) * learning_rate
    bias -= (bias_deriv / companies) * learning_rate

    return weight, bias
```

و سپس تابع trainرا نوشته که با گرفتن ورودی ۲ تابع بالارا صدا می کند و مقدار تابع هزینه را مینیموم می کند.

```
def train(radio, sales, weight, bias, learning_rate, iters):
    cost_history = []

    for i in range(iters):
        weight, bias = update_weights(radio, sales, weight, bias, learning_rate)

        # Calculate cost for auditing purposes
        cost = cost_function(radio, sales, weight, bias)
        cost_history.append(cost)

        # Log Progress
        if i % 10 == 0:
            print("iter={:d}    weight={:.2f}    bias={:.4f}    cost={:.2}".format(i, weight, bias, cost))

    return weight, bias, cost_history
```

که به ان همان ۳۸۴ ورودی و شانس پذیرش را داده و خطارا بدست می آوریم

برای train کردن آن ها ابتدا یک مقدار اولیه به teta0 و teta1 میدهیم و سپس داده ها را ورودی میدهیم مانند شکل زیر:

```
teta0 = 1
teta1 = -1
teta0, teta1, co                                                    tal, 0.01, 384)
```

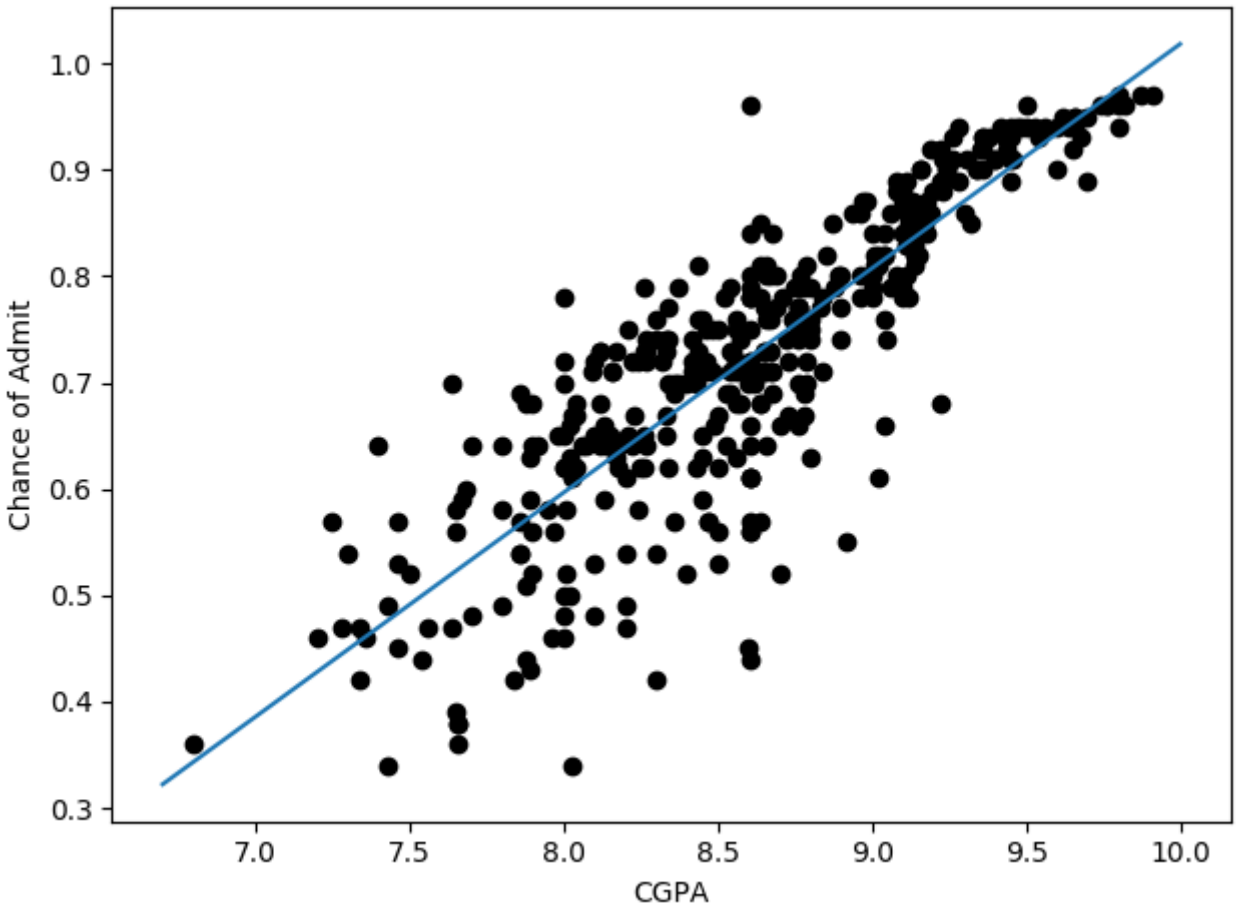
و در خروجی داریم:

```
iter=100    weight=0.21    bias=-1.0912    cost=0.0027
iter=110    weight=0.21    bias=-1.0912    cost=0.0027
iter=120    weight=0.21    bias=-1.0911    cost=0.0027
iter=130    weight=0.21    bias=-1.0911    cost=0.0027
iter=140    weight=0.21    bias=-1.0911    cost=0.0027
iter=150    weight=0.21    bias=-1.0911    cost=0.0027
iter=160    weight=0.21    bias=-1.0911    cost=0.0027
iter=170    weight=0.21    bias=-1.0911    cost=0.0027
iter=180    weight=0.21    bias=-1.0911    cost=0.0027
iter=190    weight=0.21    bias=-1.0911    cost=0.0027
iter=200    weight=0.21    bias=-1.0911    cost=0.0027
iter=210    weight=0.21    bias=-1.0911    cost=0.0027
iter=220    weight=0.21    bias=-1.0910    cost=0.0027
iter=230    weight=0.21    bias=-1.0910    cost=0.0027
iter=240    weight=0.21    bias=-1.0910    cost=0.0027
iter=250    weight=0.21    bias=-1.0910    cost=0.0027
iter=260    weight=0.21    bias=-1.0910    cost=0.0027
iter=270    weight=0.21    bias=-1.0910    cost=0.0027
iter=280    weight=0.21    bias=-1.0910    cost=0.0027
iter=290    weight=0.21    bias=-1.0910    cost=0.0027
iter=300    weight=0.21    bias=-1.0910    cost=0.0027
iter=310    weight=0.21    bias=-1.0910    cost=0.0027
iter=320    weight=0.21    bias=-1.0909    cost=0.0027
iter=330    weight=0.21    bias=-1.0909    cost=0.0027
iter=340    weight=0.21    bias=-1.0909    cost=0.0027
iter=350    weight=0.21    bias=-1.0909    cost=0.0027
iter=360    weight=0.21    bias=-1.0909    cost=0.0027
iter=370    weight=0.21    bias=-1.0909    cost=0.0027
iter=380    weight=0.21    bias=-1.0909    cost=0.0027
```

که همان طور که معلوم است در آخر کار هزینه به 0.0027 رسیده است و در این حالت مقدار teta0 و teta1 را بدست آورده ایم.

سپس با این مقدار ها نمودار و خط پیدا شده را می کشیم

```
plt.xlabel('CGPA')
plt.ylabel('Chance of Admit')
plt.plot(df_New["CGPA"], df_New["Chance of Admit"], 'ko')
x_plot = np.linspace(6.7, 10, 1000)
plt.plot(x_plot, x_plot * teta0 + teta1)
plt.show()
```



حالا با استفاده از این خط مقادیر NaN برای شنس پذیرش را پر می کنیم:

```
def fill_na(x):
    return 0.21 * x - 1.0909


df_withNaN["Chance of Admit"] = df_withNaN["Chance of Admit"].fillna(fill_na(df_withNaN['CGPA']))
print(df_withNaN)
export_csv = df_withNaN.to_csv(
    '/home/sspc/Desktop/AI_Projects/Indroduction_To_Python/dataframe_Final.csv')
```

دیتا فریم کامل را بدست می آوریم