PROBLEM STATEMENTS:

```
1.Classify students based on rank
2.Find the number of students who got admission with rank 1,2 ,3 and 4 seperately.
3.Print the total number of admission secured by the students
4.Print the number of students with highest rank[1] and number of students with lowest rank[4].
5.Print the number of students with highest gpa[4] with their details
6.Filter students with gpa score above 3
7.Check whether the dataset has nan values
8.Find the maximum and minimum gpa score obtained by students
9.Find the average gre scores obtained by the students
10.Draw a boxplot for the gre score obtained by the students
```

IMPORTING REQUIRED LIBRARIES

In [199]:

```python
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [200]:

```python
college_admission=pd.read_csv("6.Team6_college_admission_dataset.csv")
```

In [201]:

```python
college_admission.head()
```

Out[201]:

	admit	gre	gpa	ses	Gender_Male	Race	rank
0	0	380	3.61	1	0	3	3
1	1	660	3.67	2	0	2	3
2	1	800	4.00	2	0	2	1
3	1	640	3.19	1	1	2	4
4	0	520	2.93	3	1	2	4

BASIC ANALYSIS OF THE DATASET

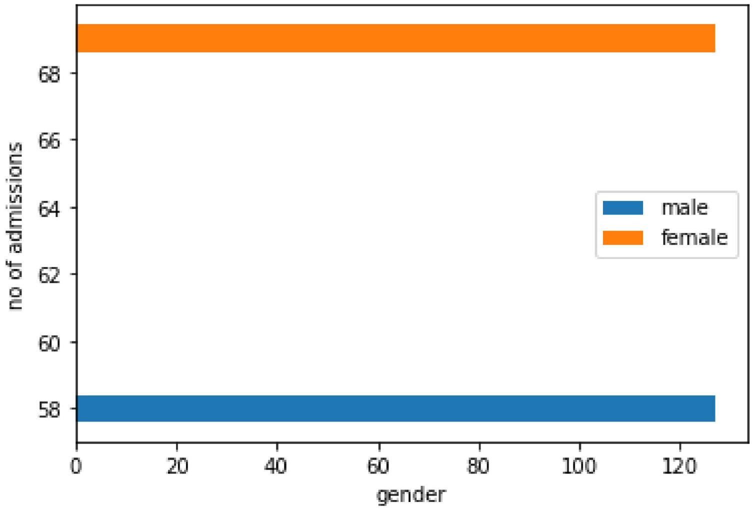
In [202]:

```
b=college_admission.groupby('admit')
b1=b.get_group(1)
no_admission=b1['admit'].count()
print("NO OF ADMISSION GIVEN: ",no_admission)
gm=college_admission.groupby('Gender_Male')
gmg=gm.get_group(1)
result=gmg.groupby('admit').get_group(1)
countm=result['admit'].count()
print("NUMBER OF MALES WHO GOT ADMISSION:",countm)
gf=college_admission.groupby('Gender_Male')
gfg=gf.get_group(0)
resultf=gfg.groupby('admit').get_group(1)
countf=resultf['admit'].count()
print("NUMBER OF FEMALES WHO GOT ADMISSION:",resultf['admit'].count())
```

```
NO OF ADMISSION GIVEN:  127
NUMBER OF MALES WHO GOT ADMISSION: 58
NUMBER OF FEMALES WHO GOT ADMISSION: 69
```

In [203]:

```
plt.barh(countm,no_admission,label="male")
plt.barh(countf,no_admission,label="female")
plt.xlabel("gender")
plt.ylabel("no of admissions")
plt.legend()
plt.show()
```



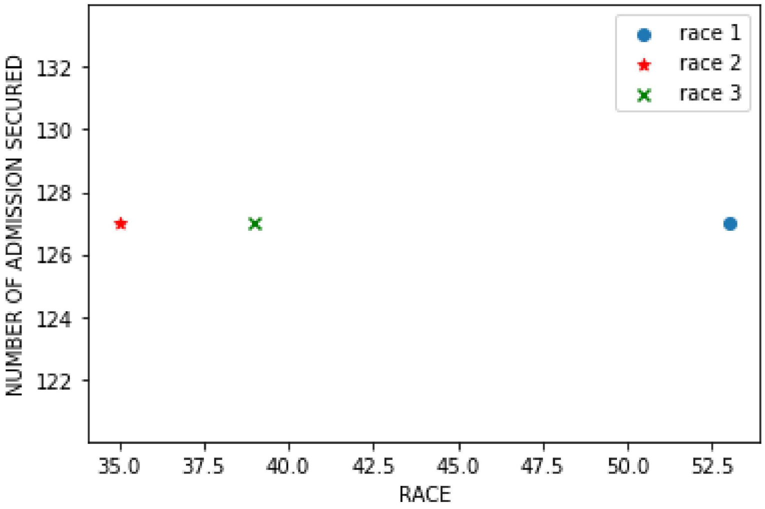
In [204]:

```
race=college_admission.groupby('Race')
```

In [205]:
```
race1=race.get_group(1)
race2=race.get_group(2)
race3=race.get_group(3)
race1_admit=race1.groupby('admit')
race2_admit=race2.groupby('admit')
race3_admit=race3.groupby('admit')
race1_admit1=race1_admit.get_group(1)
race2_admit1=race2_admit.get_group(1)
race3_admit1=race3_admit.get_group(1)
race1_count=race1_admit1['admit'].count()
race2_count=race2_admit1['admit'].count()
race3_count=race3_admit1['admit'].count()
print("NO OF ADMISSION FROM RACE 1: ",race1_count)
print("NO OF ADMISSION FROM RACE 2: ",race2_count)
print("NO OF ADMISSION FROM RACE 3: ",race3_count)
```

```
NO OF ADMISSION FROM RACE 1:  53
NO OF ADMISSION FROM RACE 2:  35
NO OF ADMISSION FROM RACE 3:  39
```

In [206]:
```
plt.scatter(race1_count,no_admission,label="race 1",marker="o")
plt.scatter(race2_count,no_admission,label="race 2",marker="*",c="r")
plt.scatter(race3_count,no_admission,label="race 3",marker="x",c="green")
plt.xlabel("RACE")
plt.ylabel("NUMBER OF ADMISSION SECURED")
plt.legend()
plt.show()
```



PROBLEM STATEMENTS

Classify students based on rank

In [207]:
```
college_admission['rank'].unique()
```

Out[207]:
```
array([3, 1, 4, 2], dtype=int64)
```

In [208]:

```
a=college_admission.groupby('rank')
r1=a.get_group(1)
r2=a.get_group(2)
r3=a.get_group(3)
r4=a.get_group(4)
print("1st rank:\n",r1)
print("2nd rank:\n",r2)
print("3rd rank:\n",r3)
print("4th rank:\n",r4)
```

```
1st rank:
      admit  gre   gpa  ses  Gender_Male  Race  rank
2        1  800  4.00    2            0     2     1
6        1  560  2.98    2            1     2     1
11       0  440  3.22    3            0     2     1
12       1  760  4.00    3            1     2     1
14       1  700  4.00    2            1     1     1
..     ...  ...   ...  ...          ...   ...   ...
368      0  580  4.00    1            0     2     1
372      1  680  2.42    1            1     1     1
373      1  620  3.37    3            1     1     1
383      0  660  4.00    1            1     3     1
385      0  420  3.02    1            1     3     1

[61 rows x 7 columns]
2nd rank:
      admit  gre   gpa  ses  Gender_Male  Race  rank
5        1  760  3.00    2            1     1     2
7        0  400  3.08    2            0     2     2
9        0  700  3.92    1            0     2     2
13       0  700  3.08    2            0     2     2
18       0  800  3.75    1            1     3     2
..     ...  ...   ...  ...          ...   ...   ...
391      1  660  3.88    1            0     1     2
393      1  620  3.75    2            0     2     2
395      0  620  4.00    2            0     2     2
397      0  460  2.63    3            0     2     2
398      0  700  3.65    1            1     1     2

[151 rows x 7 columns]
3rd rank:
      admit  gre   gpa  ses  Gender_Male  Race  rank
0        0  380  3.61    1            0     3     3
1        1  660  3.67    2            0     2     3
8        1  540  3.39    1            1     1     3
15       0  480  3.44    3            0     1     3
17       0  360  2.56    3            1     3     3
..     ...  ...   ...  ...          ...   ...   ...
378      0  640  3.12    2            1     1     3
392      1  600  3.38    3            0     3     3
394      1  460  3.99    3            1     3     3
396      0  560  3.04    2            0     1     3
399      0  600  3.89    2            1     3     3

[121 rows x 7 columns]
4th rank:
     admit  gre   gpa  ses  Gender_Male  Race  rank
3        1  640  3.19    1            1     2     4
4        0  520  2.93    3            1     2     4
10       0  800  4.00    1            1     1     4
16       0  780  3.87    2            0     3     4
22       0  600  2.82    1            0     3     4
..     ...  ...   ...  ...          ...   ...   ...
329      0  500  2.93    2            0     2     4
337      0  620  3.09    3            0     2     4
340      0  500  3.23    3            0     1     4
342      0  500  3.95    2            0     1     4
375      0  560  3.49    3            0     1     4

[67 rows x 7 columns]
```

Find the number of students who got admission with rank 1,2 ,3 and 4 seperately.

In [209]:

```
college_admission[['admit','rank']]
ad1=r1.groupby('admit')
ga1=ad1.get_group(1)
gac1=ga1['admit'].count()
print("NO OF STUDENTS WITH RANK 1 WHO GOT ADMISSION: ",gac1)
ad2=r2.groupby('admit')
ga2=ad2.get_group(1)
gac2=ga2['admit'].count()
print("NO OF STUDENTS WITH RANK 2 WHO GOT ADMISSION: ",gac2)
ad3=r3.groupby('admit')
ga3=ad3.get_group(1)
gac3=ga3['admit'].count()
print("NO OF STUDENTS WITH RANK 3 WHO GOT ADMISSION: ",gac3)
ad4=r4.groupby('admit')
ga4=ad4.get_group(1)
gac4=ga4['admit'].count()
print("NO OF STUDENTS WITH RANK 4 WHO GOT ADMISSION: ",gac4)
```

```
NO OF STUDENTS WITH RANK 1 WHO GOT ADMISSION:  33
NO OF STUDENTS WITH RANK 2 WHO GOT ADMISSION:  54
NO OF STUDENTS WITH RANK 3 WHO GOT ADMISSION:  28
NO OF STUDENTS WITH RANK 4 WHO GOT ADMISSION:  12
```

Print the total number of admission secured by the students

In [210]:

```
b=college_admission.groupby('admit')
b1=b.get_group(1)
no_admission=b1['admit'].count()
print("NO OF ADMISSION SECURED: ",no_admission)
```

```
NO OF ADMISSION SECURED:  127
```

Print the number of students with highest rank[1] and number of students with lowest rank[4].

In [211]:

```
print("number of students who obtained 1st rank: ",r1['rank'].count())
print("number of students who obtained 4th rank: ",r4['rank'].count())
```

```
number of students who obtained 1st rank:  61
number of students who obtained 4th rank:  67
```

Print the number of students with highest gpa[4] with their details

In [212]:

```
c=college_admission[college_admission['gpa']==4]
c1=c['gpa'].count()
print("NO OF STUDENTS WITH HIGHEST GPA: ",c1)
print("\nDETAILS OF STUDENTS WITH HIGHEST GPA:\n",c)
```

```
NO OF STUDENTS WITH HIGHEST GPA:  28

DETAILS OF STUDENTS WITH HIGHEST GPA:
      admit  gre  gpa  ses  Gender_Male  Race  rank
2        1  800  4.0    2            0     2     1
10       0  800  4.0    1            1     1     4
12       1  760  4.0    3            1     2     1
14       1  700  4.0    2            1     1     1
33       1  800  4.0    3            0     1     3
55       1  740  4.0    1            1     2     3
64       0  580  4.0    2            1     3     3
70       0  640  4.0    1            1     1     3
73       0  580  4.0    3            0     3     2
75       0  720  4.0    2            0     3     3
77       1  800  4.0    3            0     3     3
79       1  620  4.0    2            0     2     1
89       1  660  4.0    1            1     1     2
137      0  700  4.0    3            1     1     3
165      0  700  4.0    2            1     3     1
168      0  500  4.0    3            0     2     3
182      0  700  4.0    1            1     3     2
202      1  700  4.0    3            0     3     1
237      0  480  4.0    2            1     2     2
252      1  520  4.0    2            1     1     2
310      0  560  4.0    1            0     3     3
330      0  740  4.0    2            0     1     3
350      1  780  4.0    3            1     3     2
360      1  520  4.0    1            0     1     1
368      0  580  4.0    1            0     2     1
377      1  800  4.0    2            0     3     2
383      0  660  4.0    1            1     3     1
395      0  620  4.0    2            0     2     2
```

Filter students with gpa score above 3

In [213]:
```
college_admission[college_admission['gpa']>3]
```

Out[213]:

	admit	gre	gpa	ses	Gender_Male	Race	rank
0	0	380	3.61	1	0	3	3
1	1	660	3.67	2	0	2	3
2	1	800	4.00	2	0	2	1
3	1	640	3.19	1	1	2	4
7	0	400	3.08	2	0	2	2
...	...	...	...	...	...	...	...
394	1	460	3.99	3	1	3	3
395	0	620	4.00	2	0	2	2
396	0	560	3.04	2	0	1	3
398	0	700	3.65	1	1	1	2
399	0	600	3.89	2	1	3	3

329 rows × 7 columns

Check whether the dataset has nan values

In [214]:
```
np.isnan(college_admission)
```

Out[214]:

	admit	gre	gpa	ses	Gender_Male	Race	rank
0	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...
395	False	False	False	False	False	False	False
396	False	False	False	False	False	False	False
397	False	False	False	False	False	False	False
398	False	False	False	False	False	False	False
399	False	False	False	False	False	False	False

400 rows × 7 columns

Find the maximum and minimum gpa score obtained by students

```
In [215]: college_admission['gpa'].max()

Out[215]: 4.0
```

```
In [216]: college_admission['gpa'].min()

Out[216]: 2.26
```

Find the average gre scores obtained by the students

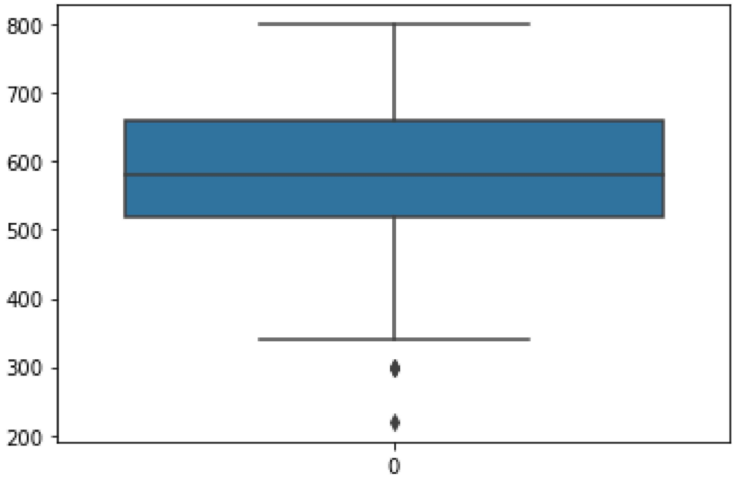
```
In [217]: college_admission['gre'].mean()

Out[217]: 587.7
```

Draw a boxplot for the gre score obtained by the students

```
In [218]: sns.boxplot(data=college_admission['gre'])

Out[218]: <AxesSubplot:>
```



MACHINE LEARNING MODEL - LOGISTIC REGRESSION

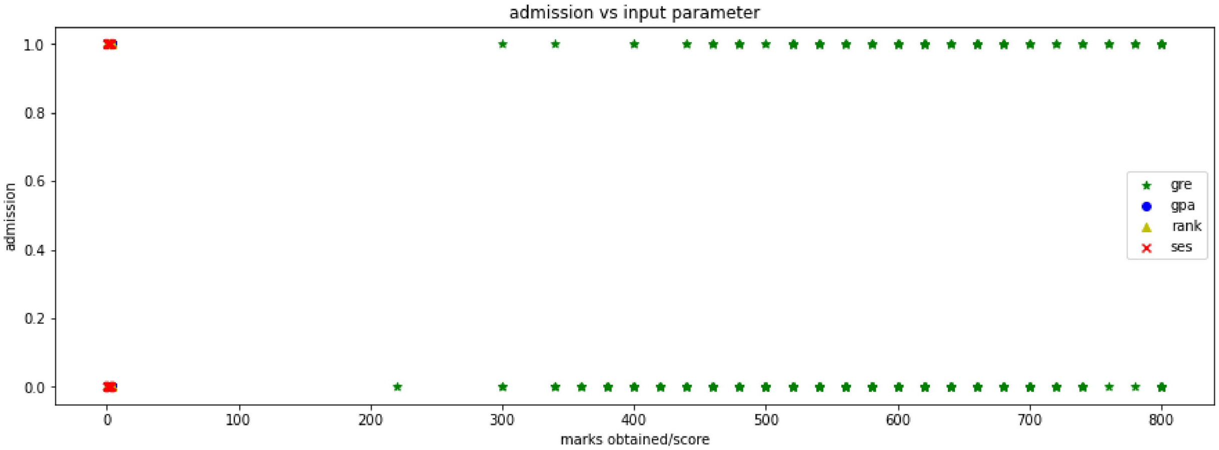
In [219]:
```
college_admission[['gre','gpa','rank','ses']]
```

Out[219]:

	gre	gpa	rank	ses
0	380	3.61	3	1
1	660	3.67	3	2
2	800	4.00	1	2
3	640	3.19	4	1
4	520	2.93	4	3
...	...	...	...	...
395	620	4.00	2	2
396	560	3.04	3	2
397	460	2.63	2	3
398	700	3.65	2	1
399	600	3.89	3	2

400 rows × 4 columns

In [220]:
```
plt.figure(figsize=[15,5])
plt.scatter(college_admission['gre'],college_admission['admit'],c="g",marker="*",
plt.scatter(college_admission['gpa'],college_admission['admit'],c="b",marker="o",
plt.scatter(college_admission['rank'],college_admission['admit'],c="y",marker="^'
plt.scatter(college_admission['ses'],college_admission['admit'],c="r",marker="x",
plt.xlabel("marks obtained/score")
plt.ylabel("admission")
plt.title("admission vs input parameter")
plt.legend()
plt.show()
```



In [221]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
model=LogisticRegression()
```

In [222]:

```
college_admission['admit'].unique()
```

Out[222]:

```
array([0, 1], dtype=int64)
```

In [223]:

```
college_admission.dtypes
```

Out[223]:

```
admit            int64
gre              int64
gpa            float64
ses              int64
Gender_Male      int64
Race             int64
rank             int64
dtype: object
```

In [224]:

```
x=college_admission[['gre','gpa','rank','ses']]
y=college_admission['admit']
```

In [225]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.8)
```

In [226]:

```
x_train.shape
```

Out[226]:

```
(320, 4)
```

In [227]:

```
y_train.shape
```

Out[227]:

```
(320,)
```

In [228]:

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

Out[228]:

```
LogisticRegression()
```

In [229]:

```
predicted_y=model.predict(x_test)
predicted_y
```

Out[229]:

```
array([0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0], dtype=int64)
```

```
In [230]: test=y_test.values
          test
```

```
Out[230]: array([0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0,
                 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
                 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0,
                 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0], dtype=int64)
```

```
In [231]: accuracy_score(predicted_y,test)
```

```
Out[231]: 0.725
```

```
In [232]: print("slope",model.coef_)
```

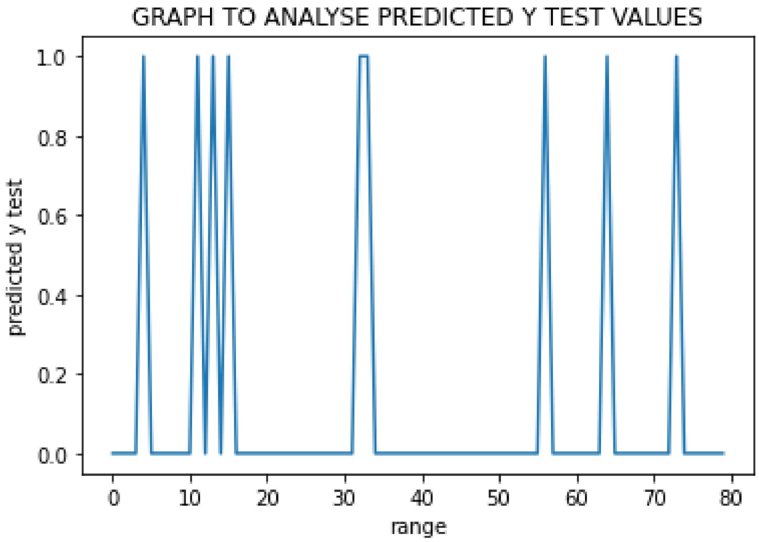
```
slope [[ 0.00295926  0.53425111 -0.53149845 -0.02616324]]
```

```
In [233]: model.intercept_
```

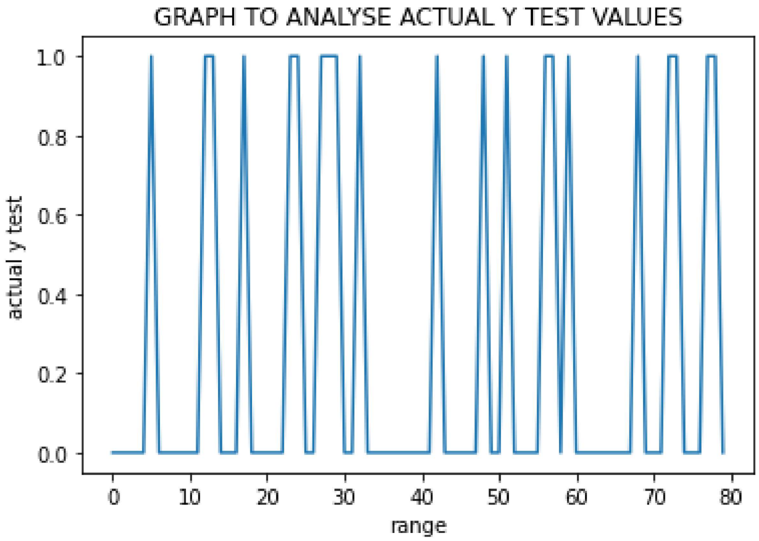
```
Out[233]: array([-2.9839975])
```

COMPARISON BETWEEN PREDICTED Y TEST VALUES AND ACTUAL Y TEST VALUES

```
In [234]: plt.plot(predicted_y)
          plt.xlabel("range")
          plt.ylabel("predicted y test")
          plt.title("GRAPH TO ANALYSE PREDICTED Y TEST VALUES")
          plt.show()
```



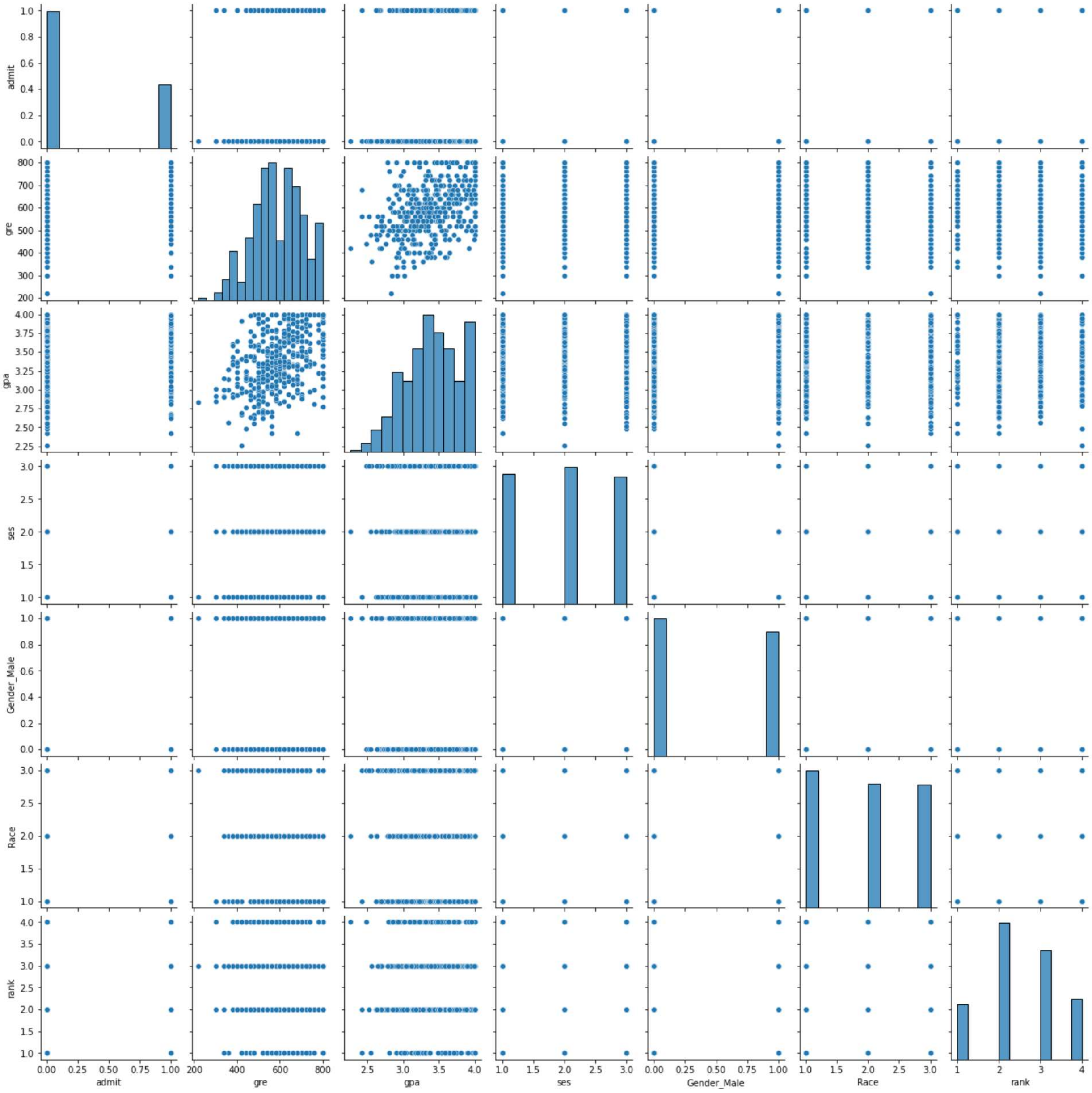
```
In [235]: plt.plot(test)
          plt.xlabel("range")
          plt.ylabel("actual y test")
          plt.title("GRAPH TO ANALYSE ACTUAL Y TEST VALUES")
          plt.show()
```



ANALYSING VARIOUS RELATIONS IN THE DATASET THROUGH PAIRPLOT

```
In [240]: sns.pairplot(data=college_admission)
```

```
Out[240]: <seaborn.axisgrid.PairGrid at 0x25f3e596160>
```



In [ ]: