College Admission.

Q1) Find the missing values. (if any, perform missing value treatment)

Ans) The below attached is the code in R,which gives an overview of the missing values

```
college_data<-
read.csv("https://raw.githubusercontent.com/shivanipriya89/College/main/College.csv")

print(college_data)

View(college_data)

missing<-is.na(college_data)

missing

which(is.na(missing))

sum(is.na(missing))
```

Here,is the output of code mention above. The first screenshot represents the tabular format of college admission details

```
admit gre  gpa ses Gender_Male Race rank
1       0 380 3.61   1           0    3    3
2       1 660 3.67   2           0    2    3
3       1 800 4.00   2           0    2    1
4       1 640 3.19   1           1    2    4
5       0 520 2.93   3           1    2    4
6       1 760 3.00   2           1    1    2
7       1 560 2.98   2           1    2    1
8       0 400 3.08   2           0    2    2
9       1 540 3.39   1           1    1    3
10      0 700 3.92   1           0    2    2
11      0 800 4.00   1           1    1    4
12      0 440 3.22   3           0    2    1
13      1 760 4.00   3           1    2    1
14      0 700 3.08   2           0    2    2
15      1 700 4.00   2           1    1    1
16      0 480 3.44   3           0    1    3
17      0 780 3.87   2           0    3    4
18      0 360 2.56   3           1    3    3
19      0 800 3.75   1           1    3    2
20      1 540 3.81   1           0    3    1
21      0 500 3.17   3           0    2    3
22      1 660 3.63   1           0    1    2
23      0 600 2.82   1           0    3    4
24      0 680 3.19   1           0    1    4
25      1 760 3.35   2           0    2    2
26      1 800 3.66   2           1    1    1
27      1 620 3.61   2           0    1    1
28      1 520 3.74   2           0    3    4
29      1 780 3.22   1           0    1    2
30      0 520 3.29   1           0    1    1
```

31	0	540	3.78	1	1	1	4
32	0	760	3.35	2	1	1	3
33	0	600	3.40	3	0	1	3
34	1	800	4.00	3	0	1	3
35	0	360	3.14	1	1	2	1
36	0	400	3.05	3	0	2	2
37	0	580	3.25	1	0	2	1
38	0	520	2.90	2	0	2	3
39	1	500	3.13	2	0	2	2
40	1	520	2.68	2	0	1	3
41	0	560	2.42	1	1	3	2
42	1	580	3.32	1	0	1	2
43	1	600	3.15	2	1	1	2
44	0	500	3.31	2	0	2	3
45	0	700	2.94	1	0	3	2
46	1	460	3.45	2	1	3	3
47	1	580	3.46	3	1	1	2
48	0	500	2.97	3	0	2	4
49	0	440	2.48	3	0	3	4
50	0	400	3.35	3	0	1	3
51	0	640	3.86	2	1	3	3
52	0	440	3.13	2	0	2	4
53	0	740	3.37	2	1	3	4
54	1	680	3.27	2	0	2	2
55	0	660	3.34	1	0	1	3
56	1	740	4.00	1	1	2	3
57	0	560	3.19	3	1	1	3
58	0	380	2.94	3	0	2	3
59	0	400	3.65	3	1	2	2
60	0	600	2.82	3	1	1	4
61	1	620	3.18	2	1	1	2
62	0	560	3.32	1	0	3	4
63	0	640	3.67	1	1	2	3
64	1	680	3.85	1	1	3	3
65	0	580	4.00	2	1	3	3
66	0	600	3.59	1	0	1	2
67	0	740	3.62	3	1	2	4
68	0	620	3.30	2	1	3	1
69	0	580	3.69	3	0	3	1
70	0	800	3.73	1	1	1	1
71	0	640	4.00	1	1	1	3
72	0	300	2.92	1	1	1	4
73	0	480	3.39	2	0	2	4
74	0	580	4.00	3	0	3	2
75	0	720	3.45	2	1	2	4
76	0	720	4.00	2	0	3	3
77	0	560	3.36	1	1	2	3
78	1	800	4.00	3	0	3	3
79	0	540	3.12	3	1	2	1
80	1	620	4.00	2	0	2	1
81	0	700	2.90	2	0	2	4
82	0	620	3.07	3	1	2	2
83	0	500	2.71	2	0	3	2
84	0	380	2.91	3	1	2	4
85	1	500	3.60	1	1	1	3
86	0	520	2.98	2	0	2	2

87	0	600	3.32	1	0	3	2
88	0	600	3.48	1	0	1	2
89	0	700	3.28	3	0	3	1
90	1	660	4.00	1	1	1	2
91	0	700	3.83	2	0	2	2
92	1	720	3.64	2	0	2	1
93	0	800	3.90	3	1	1	2
94	0	580	2.93	3	1	1	2
95	1	660	3.44	2	0	3	2
96	0	660	3.33	2	1	3	2
97	0	640	3.52	2	1	3	4
98	0	480	3.57	3	1	2	2
99	0	700	2.88	2	1	3	2
100	0	400	3.31	3	1	2	3
101	0	340	3.15	2	0	1	3
102	0	580	3.57	1	1	2	3
103	0	380	3.33	3	0	3	4
104	0	540	3.94	3	0	1	3
105	1	660	3.95	2	1	1	2
106	1	740	2.97	1	1	1	2
107	1	700	3.56	1	1	2	1
108	0	480	3.13	2	0	1	2
109	0	400	2.93	1	1	3	3
110	0	480	3.45	3	0	1	2
111	0	680	3.08	3	0	3	4
112	0	420	3.41	2	1	3	4
113	0	360	3.00	1	0	1	3
114	0	600	3.22	3	1	2	1
115	0	720	3.84	1	1	2	3
116	0	620	3.99	2	1	2	3
117	1	440	3.45	1	1	3	2
118	0	700	3.72	2	1	2	2
119	1	800	3.70	1	0	2	1
120	0	340	2.92	3	1	2	3
121	1	520	3.74	2	0	2	2
122	1	480	2.67	1	0	1	2
123	0	520	2.85	3	0	1	3
124	0	500	2.98	3	0	2	3
125	0	720	3.88	2	0	3	3
126	0	540	3.38	3	0	3	4
127	1	600	3.54	3	0	3	1
128	0	740	3.74	1	0	3	4
129	0	540	3.19	1	1	3	2
130	0	460	3.15	3	0	2	4
131	1	620	3.17	1	0	3	2
132	0	640	2.79	3	1	1	2
133	0	580	3.40	3	0	1	2
134	0	500	3.08	2	1	2	3
135	0	560	2.95	3	1	1	2
136	0	500	3.57	2	1	3	3
137	0	560	3.33	3	1	2	4
138	0	700	4.00	3	1	1	3
139	0	620	3.40	3	0	1	2
140	1	600	3.58	3	0	3	1
141	0	640	3.93	2	1	2	2
142	1	700	3.52	2	0	1	4

```
> missing<-is.na(college_data)
> missing
       admit   gre   gpa   ses Gender_Male  Race  rank
  [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
  [5,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
  [6,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
  [7,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
  [8,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
  [9,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [10,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [11,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [12,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [13,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [14,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [15,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [16,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [17,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [18,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [19,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [20,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [21,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [22,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [23,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [24,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [25,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [26,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [27,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [28,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [29,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [30,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [31,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [32,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [33,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [34,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [35,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [36,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [37,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [38,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [39,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [40,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [41,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [42,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [43,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [44,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [45,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [46,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [47,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [48,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [49,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [50,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [51,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [52,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [53,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
```

```
 [54,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [55,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [56,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [57,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [58,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [59,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [60,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [61,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [62,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [63,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [64,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [65,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [66,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [67,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [68,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [69,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [70,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [71,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [72,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [73,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [74,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [75,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [76,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [77,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [78,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [79,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [80,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [81,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [82,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [83,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [84,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [85,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [86,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [87,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [88,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [89,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [90,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [91,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [92,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [93,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [94,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [95,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [96,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [97,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [98,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [99,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[100,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[101,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[102,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[103,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[104,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[105,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[106,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[107,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[108,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[109,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
```

```
[110,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[111,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[112,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[113,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[114,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[115,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[116,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[117,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[118,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[119,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[120,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[121,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[122,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[123,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[124,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[125,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[126,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[127,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[128,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[129,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[130,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[131,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[132,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[133,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[134,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[135,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[136,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[137,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[138,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[139,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[140,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[141,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[142,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [ reached getOption("max.print") -- omitted 258 rows ]
```

This screenshot gives the overview of the missing values entries of all columns

```
sum(is.na(missing))
[1] 0
```

The sum of the missing values is 0. It clearly indicates that there are no missing values in abo ve mention columns of the dataframe

Q2) Find outliers (if any, then perform outlier treatment)

Ans) For the analysis of outliers, I am using the concept of boxplot and histogram on all 7 columns of the dataset. The below attached is the code in R

college_data<-
read.csv("https://raw.githubusercontent.com/shivanipriya89/College/main/College.csv")

print(college_data)

View(college_data)

```
summary(college_data)

# Converting all columns to Numberic
college_data$admit<- as.numeric(college_data$admit)
college_data$gre<- as.numeric(college_data$gre)
college_data$gpa<- as.numeric(college_data$gpa)
college_data$ses<- as.numeric(college_data$ses)
college_data$Gender_Male<- as.numeric(college_data$Gender_Male)
college_data$Race<- as.numeric(college_data$Race)
college_data$rank<-as.numeric(college_data$rank)

head(college_data)

# Analysis of outlier through Boxplot
boxplot(college_data$admit,horizontal = TRUE)
boxplot.stats(college_data$admit)$out
boxplot.stats(college_data$gre)$out
boxplot.stats(college_data$gpa)$out
boxplot.stats(college_data$ses)$out
boxplot.stats(college_data$Gender_Male)$out
boxplot.stats(college_data$Race)$out
boxplot.stats(college_data$rank)$out

# Histogram
png(file="college1.png")
hist(college_data$admit,xlab = "Admission of Students",col = "yellow",border =
"blue",main="Histogram of Admission of Students")

dev.off()
```

```
png(file="college2.png")

hist(college_data$gre,xlab = "Graduate Record Exam Scores",col = "red",border =
"green",main="Histogram of GRE Scores")

dev.off()

png(file="college3.png")

hist(college_data$gpa,xlab = "Grade Point Average",col = "blue",border =
"green",main="Histogram of GPA")

dev.off()

png(file="college4.png")

hist(college_data$ses,xlab = "Socioeconomic Status",col = "red",border =
"yellow",main="Histogram of Socioeconomic Status")

dev.off()

png(file="college5.png")

hist(college_data$Gender_Male,xlab = "Gender",col = "blue",border =
"green",main="Histogram of Gender")

dev.off()

png(file="college6.png")

hist(college_data$Gender_Male,xlab = "Race",col = "red",border =
"green",main="Histogram of Race")
```

dev.off()

png(file="college7.png")

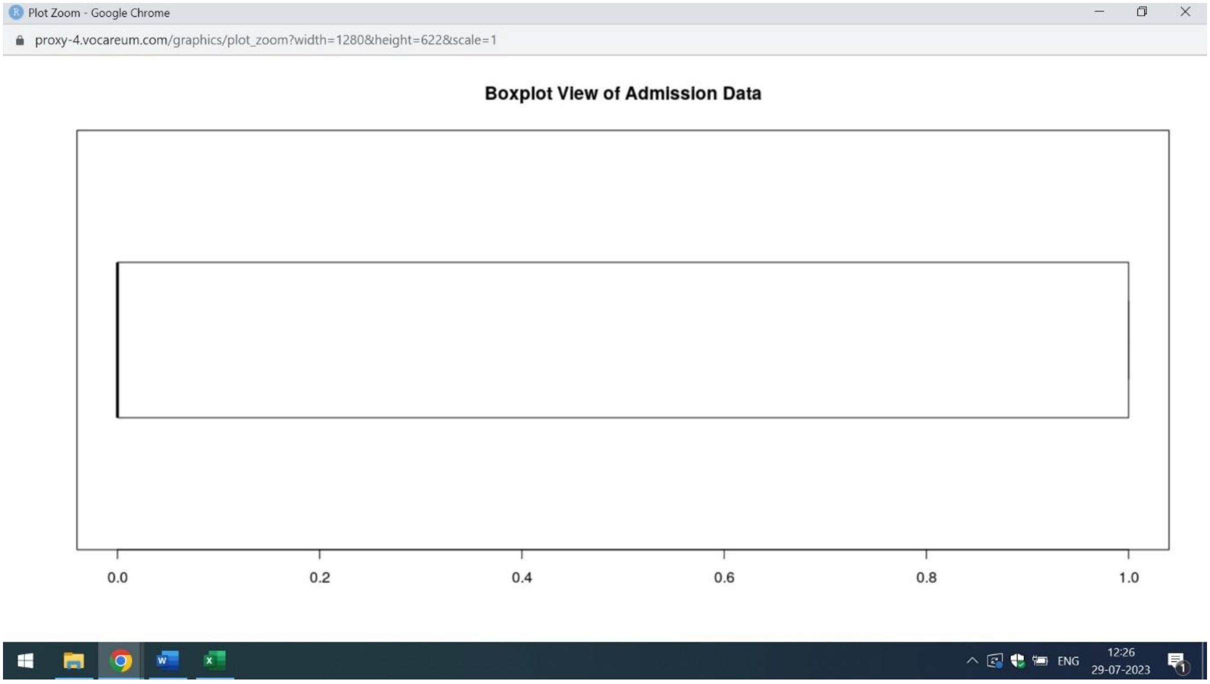
hist(college_data$rank,xlab = "Ranking of Institutions",col = "orange",border = "green",main="Histogram of Ranking of Institutions")

dev.off()

Below attached are the screenshots of the output

```
summary(college_data)
     admit             gre             gpa             ses       
 Min.   :0.0000   Min.   :220.0   Min.   :2.260   Min.   :1.000  
 1st Qu.:0.0000   1st Qu.:520.0   1st Qu.:3.130   1st Qu.:1.000  
 Median :0.0000   Median :580.0   Median :3.395   Median :2.000  
 Mean   :0.3175   Mean   :587.7   Mean   :3.390   Mean   :1.992  
 3rd Qu.:1.0000   3rd Qu.:660.0   3rd Qu.:3.670   3rd Qu.:3.000  
 Max.   :1.0000   Max.   :800.0   Max.   :4.000   Max.   :3.000  
  Gender_Male        Race            rank      
 Min.   :0.000   Min.   :1.000   Min.   :1.000  
 1st Qu.:0.000   1st Qu.:1.000   1st Qu.:2.000  
 Median :0.000   Median :2.000   Median :2.000  
 Mean   :0.475   Mean   :1.962   Mean   :2.485  
 3rd Qu.:1.000   3rd Qu.:3.000   3rd Qu.:3.000  
 Max.   :1.000   Max.   :3.000   Max.   :4.000  


>
```

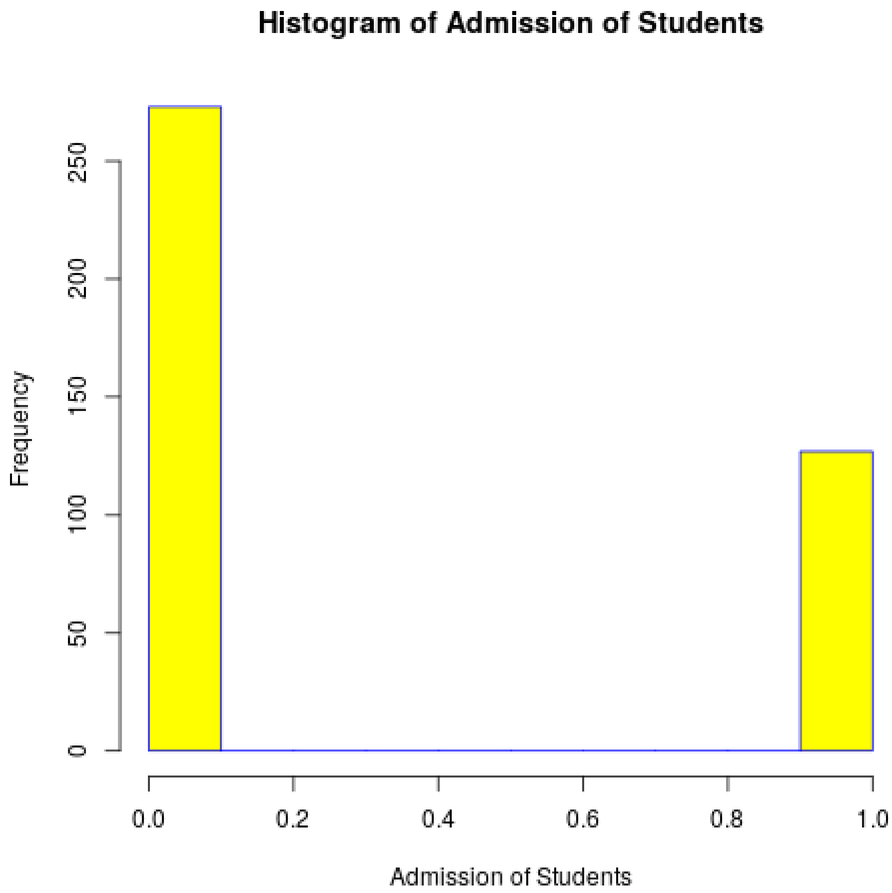
The first screenshot represents the minimum,1st Quantile,Median,3rd Quantile and Maximum Values of all 7 columns of the dataset

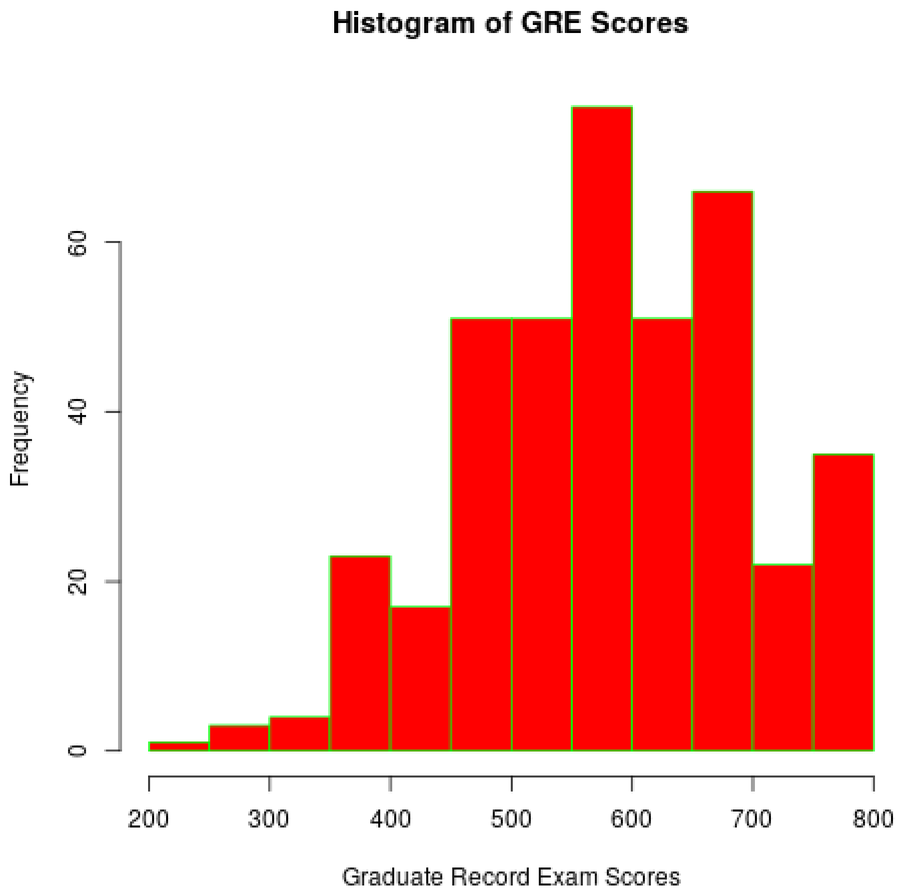


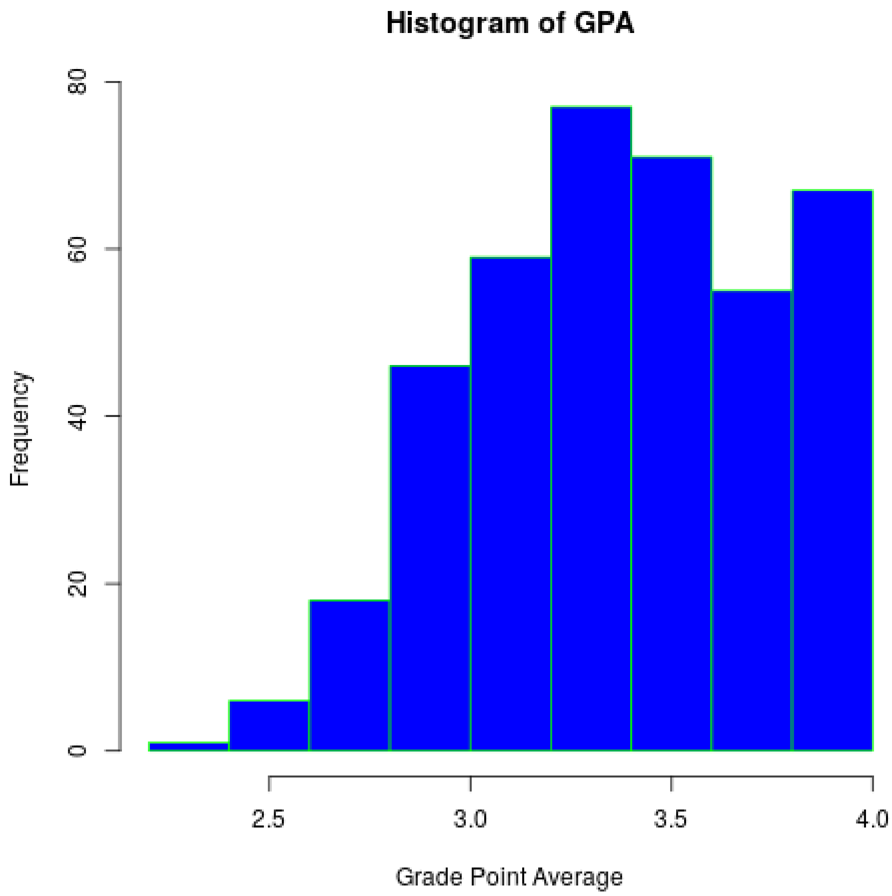

```
boxplot.stats(college_data$admit)$out
numeric(0)
> boxplot.stats(college_data$gre)$out
[1] 300 300 220 300
> boxplot.stats(college_data$gpa)$out
[1] 2.26
> boxplot.stats(college_data$ses)$out
numeric(0)
> boxplot.stats(college_data$Gender_Male)$out
numeric(0)
> boxplot.stats(college_data$Race)$out
numeric(0)
> boxplot.stats(college_data$rank)$out
numeric(0)


>
```

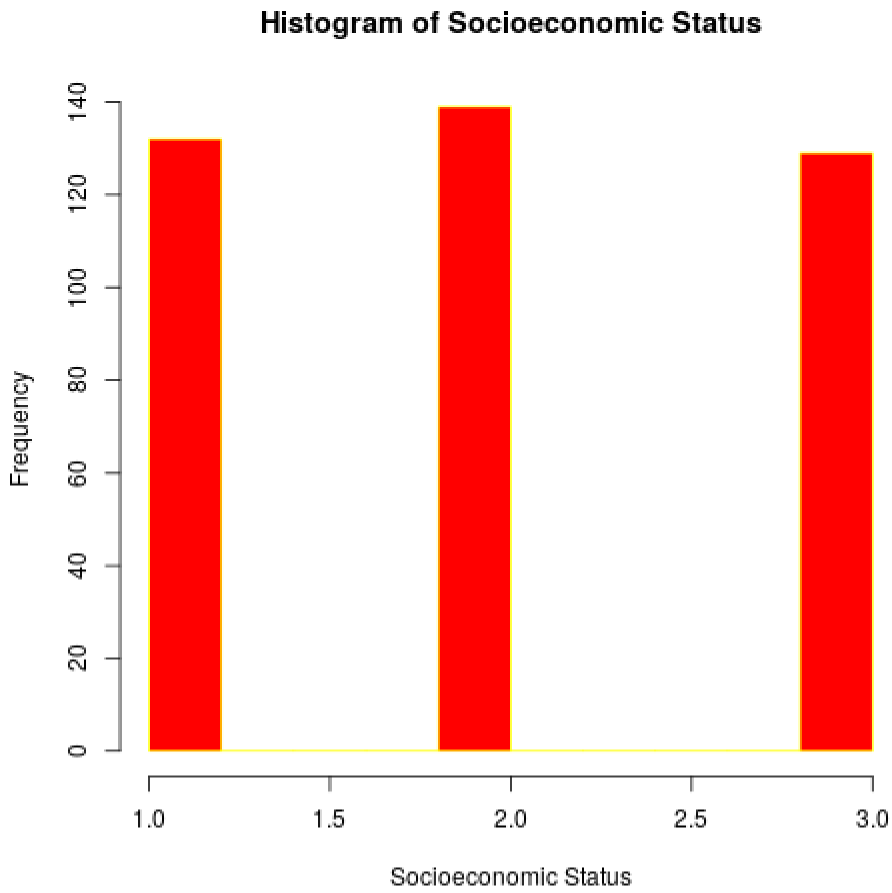
From,the above mention screenshot one can analyse that admission column has no outliers. However,the gre and gpa column has outlier values. The outlier values of the gre column are 300 and 220 and the gpa column is 2.26.

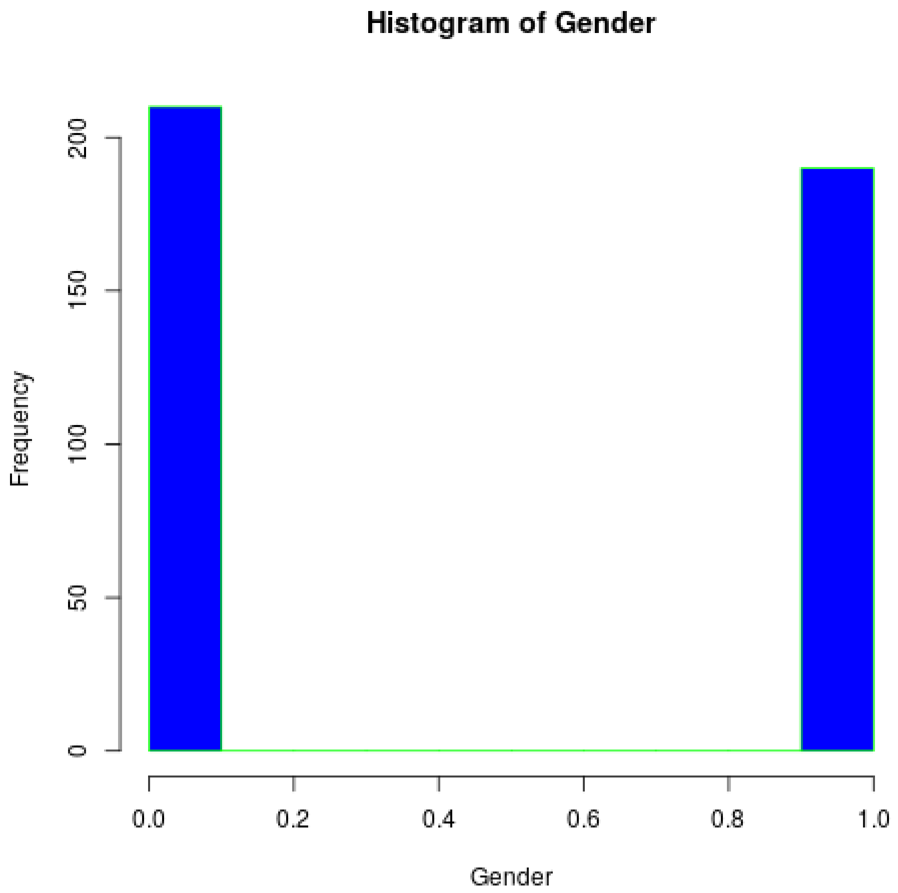
Below attached are the histogram of various columns

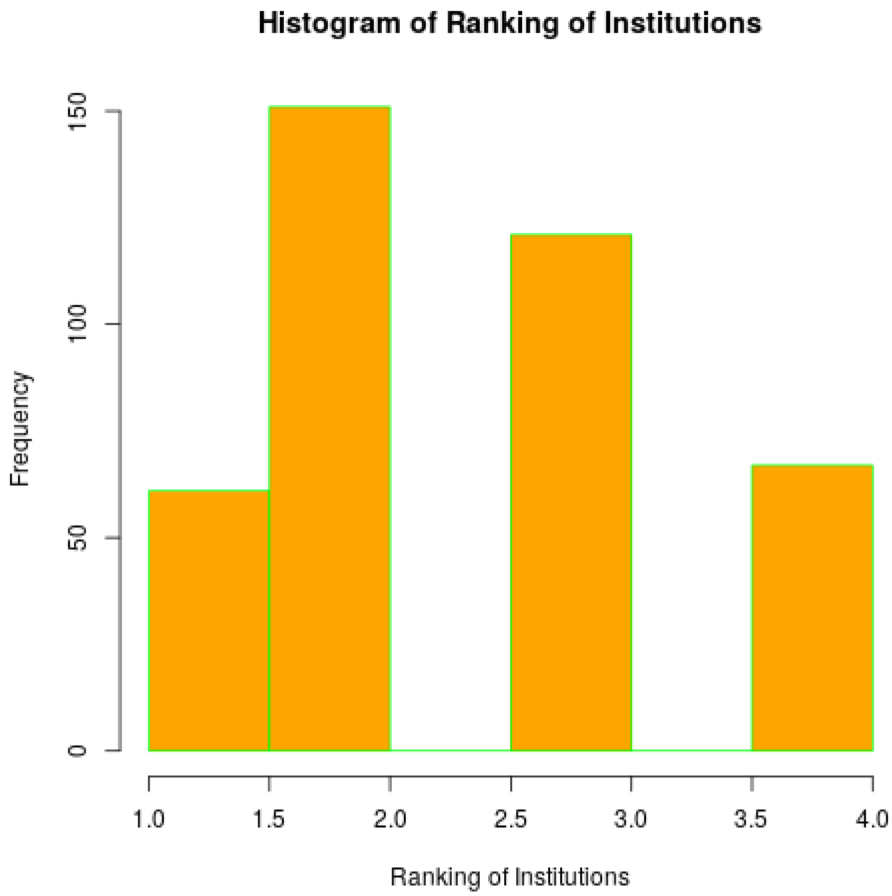

Histogram of Admission of Students
Frequency
0
50
100
150
200
250
0.0
0.2
0.4
0.6
0.8
1.0
Admission of Students

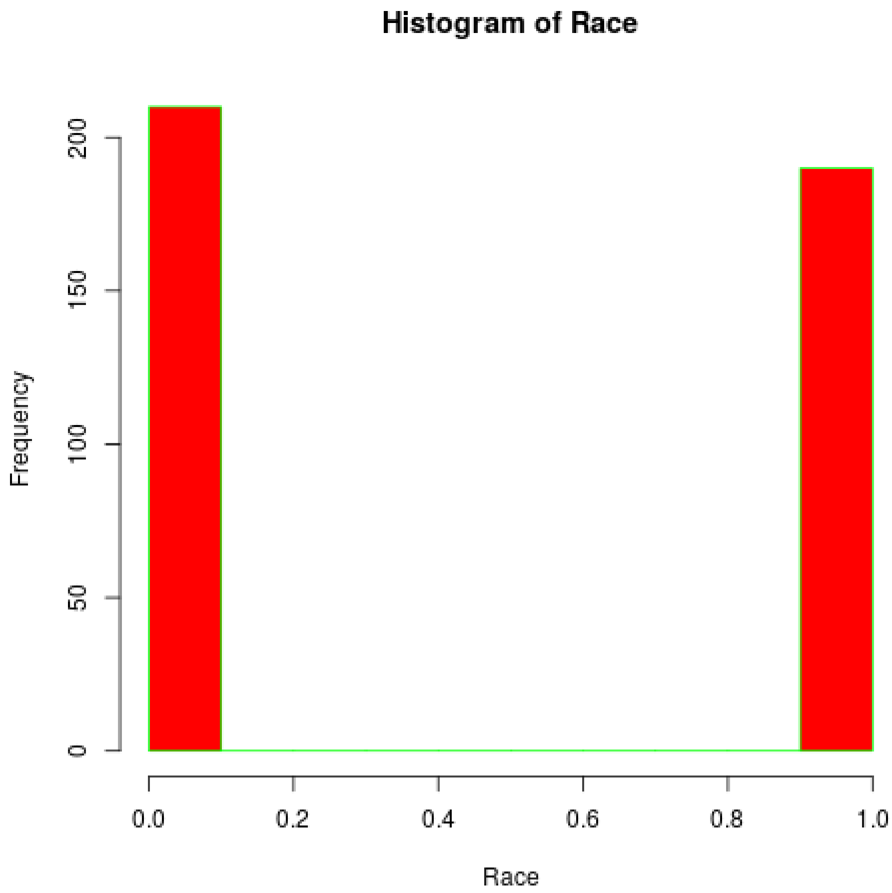

Histogram of GRE Scores
Frequency
0
20
40
60
200
300
400
500
600
700
800
Graduate Record Exam Scores


Histogram of GPA
Frequency
0
20
40
60
80
2.5
3.0
3.5
4.0
Grade Point Average


Histogram of Socioeconomic Status
Frequency
0
20
40
60
80
100
120
140
1.0
1.5
2.0
2.5
3.0
Socioeconomic Status


Histogram of Gender
Frequency
0
50
100
150
200
0.0
0.2
0.4
0.6
0.8
1.0
Gender


Histogram of Ranking of Institutions
Frequency
0
50
100
150
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Ranking of Institutions



Q3) Find the structure of the data set and if required, transform the numeric data type to factor and vice-versa.

Ans) The below attached is the code in R which represents the structure of the dataset and conversion of all datatypes to factor and vice-versa

college_data<-
read.csv("https://raw.githubusercontent.com/shivanipriya89/College/main/College.csv")

print(college_data)

View(college_data)

str(college_data)

Converting Numeric Columns to Factor

college_data$rank<-sapply(college_data$rank,factor)

str(college_data$rank)

```
college_data$admit<-sapply(college_data$admit,factor)
str(college_data$admit)

college_data$ses<-sapply(college_data$ses,factor)
str(college_data$ses)

college_data$Gender_Male<-sapply(college_data$Gender_Male,factor)
str(college_data$Gender_Male)

college_data$Race<-sapply(college_data$Race,factor)
str(college_data$Race)

# Conversion of factors to Numeric
vec1<-as.numeric(college_data$Race)
vec2<-as.numeric(college_data$Rank)
vec3<-as.numeric(college_data$Gender_Male)
vec4<-as.numeric(college_data$ses)
vec5<-as.numeric(college_data$admit)

str(vec1)
str(vec2)
str(vec3)
str(vec4)
str(vec5)
```

```
str(college_data)
'data.frame':   400 obs. of  7 variables:
 $ admit      : int  0 1 1 1 0 1 1 0 1 0 ...
 $ gre        : int  380 660 800 640 520 760 560 400 540 700 ...
 $ gpa        : num  3.61 3.67 4 3.19 2.93 3 2.98 3.08 3.39 3.92 ...
 $ ses        : int  1 2 2 1 3 2 2 2 1 1 ...
 $ Gender_Male: int  0 0 0 1 1 1 1 0 1 0 ...
 $ Race       : int  3 2 2 2 2 1 2 2 1 2 ...
 $ rank       : int  3 3 1 4 4 2 1 2 3 2 ...
```

The above mention screenshot represents the structure of college data set and datatypes of various columns. From the above mention screenshot,it is clear that except gpa rest of the columns has the integer datatypes

```
college_data$rank<-sapply(college_data$rank,factor)
> str(college_data$rank)
 Factor w/ 4 levels "3","1","4","2": 1 1 2 3 3 4 2 4 1 4 ...
>
> college_data$admit<-sapply(college_data$admit,factor)
> str(college_data$admit)
 Factor w/ 2 levels "0","1": 1 2 2 2 1 2 2 1 2 1 ...
>
> college_data$ses<-sapply(college_data$ses,factor)
> str(college_data$ses)
 Factor w/ 3 levels "1","2","3": 1 2 2 1 3 2 2 2 1 1 ...
>
> college_data$Gender_Male<-sapply(college_data$Gender_Male,factor)
> str(college_data$Gender_Male)
 Factor w/ 2 levels "0","1": 1 1 1 2 2 2 2 1 2 1 ...
>
> college_data$Race<-sapply(college_data$Race,factor)
> str(college_data$Race)
 Factor w/ 3 levels "3","2","1": 1 2 2 2 2 3 2 2 3 2 ...
```

The above screenshot represents the factor conversion of all 7 columns.

```
str(vec1)
 num [1:400] 1 2 2 2 2 3 2 2 3 2 ...
> str(vec1)
 num [1:400] 1 2 2 2 2 3 2 2 3 2 ...
> str(vec2)
 num(0)
> str(vec3)
 num [1:400] 1 1 1 2 2 2 2 1 2 1 ...
> str(vec4)
 num [1:400] 1 2 2 1 3 2 2 2 1 1 ...
> str(vec5)
 num [1:400] 1 2 2 2 1 2 2 1 2 1 ...
```

The above mention screenshot represents that datatype of all 7 columns has been changed to numeric

Q4) Find whether the data is normally distributed or not. Use the plot to determine the same.

Ans) The below attached is the code in R which gives an overview of the Normal Distribution of data

college_data<-read.csv("https://raw.githubusercontent.com/shivanipriya89/College/main/College.csv")

print(college_data)

```
View(college_data)
str(college_data)
summary(college_data)

# Analysis of Normal Distribution

t1<-sd(college_data$admit)
t2<-sd(college_data$gre)
t3<-sd(college_data$gpa)
t4<-sd(college_data$ses)
t5<-sd(college_data$Gender_Male)
t6<-sd(college_data$Race)
t7<-sd(college_data$Rank)



print(t1)
print(t2)
print(t3) #0.38
print(t4) #0.81
print(t5) #0.5
print(t6) #0.82
print(t7) #Na
y1<- dnorm(college_data$admit, mean = 0.32, sd = 0.466)

plot(college_data$admit,y1)

y2<- dnorm(college_data$gre, mean = 588, sd = 116)

plot(college_data$gre,y2)
```

y3<- dnorm(college_data$gpa, mean = 3.4, sd = 0.38)

plot(college_data$gpa,y3)

y4<- dnorm(college_data$ses, mean = 0.2, sd = 0.81)

plot(college_data$ses,y4)

y5<- dnorm(college_data$Gender_Male, mean = 0.5, sd = 0.5)

plot(college_data$Gender_Male,y5)

y6<- dnorm(college_data$Race, mean = 2, sd = 0.82)

plot(college_data$Race,y6)

y7<- dnorm(college_data$rank, mean = 2.5, sd = 0)

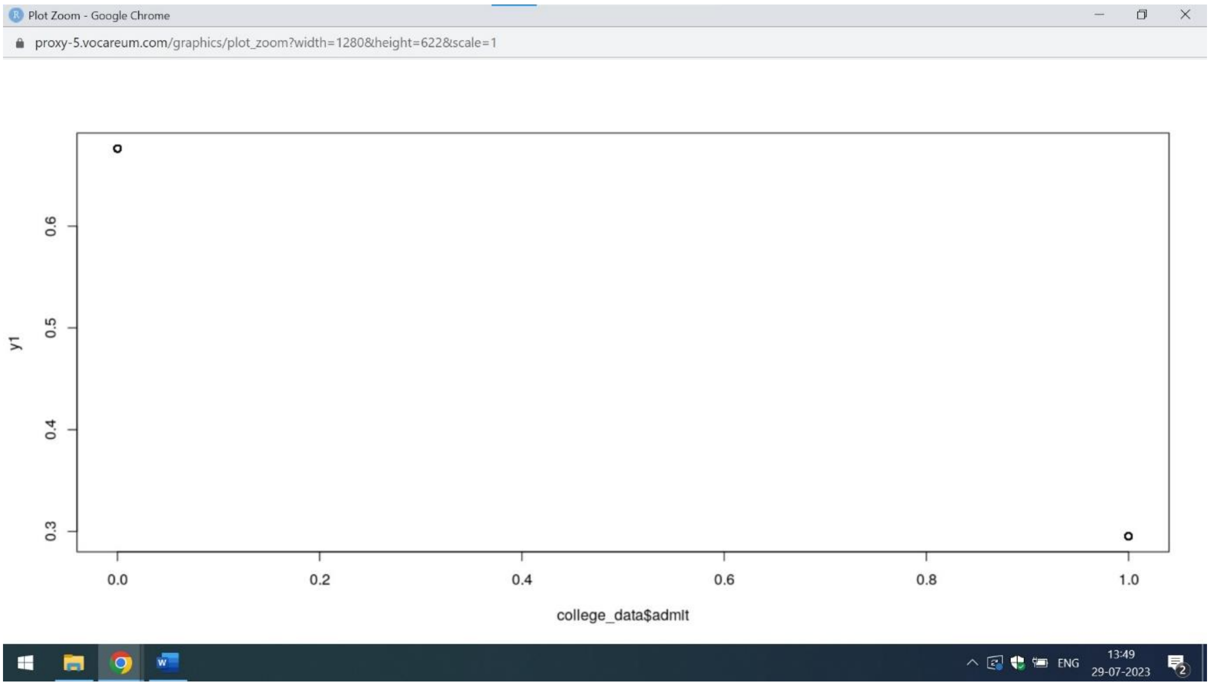
plot(college_data$rank,y7)

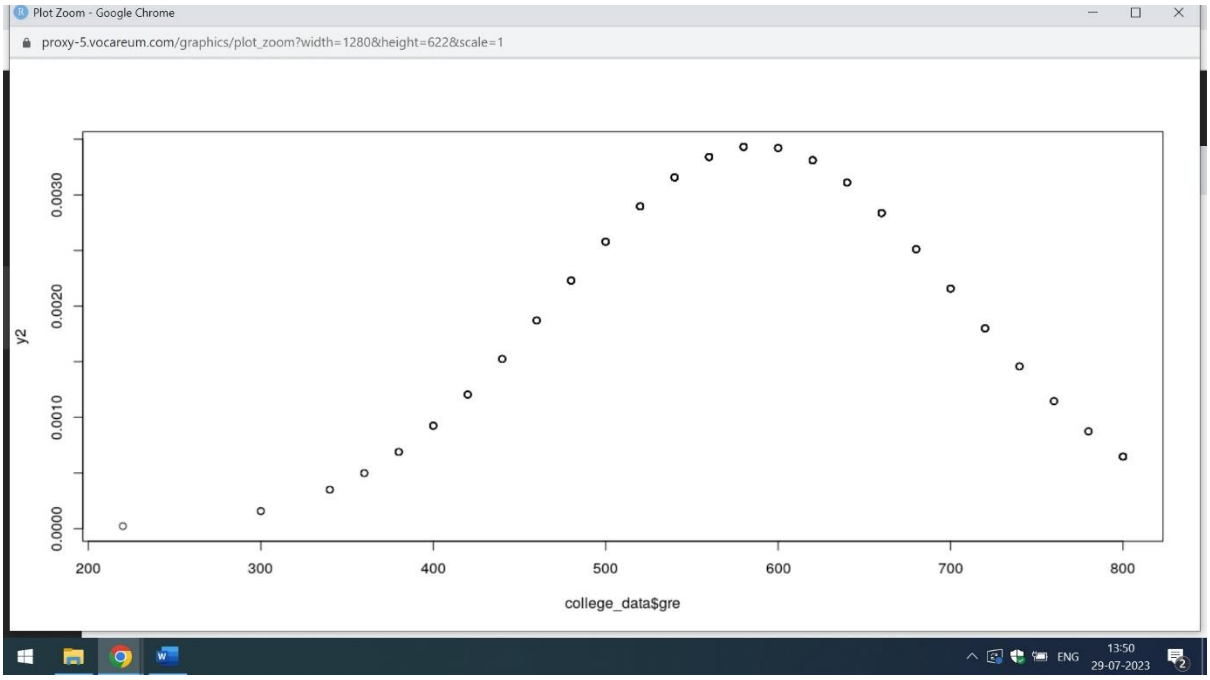
Below attached are the screenshots.

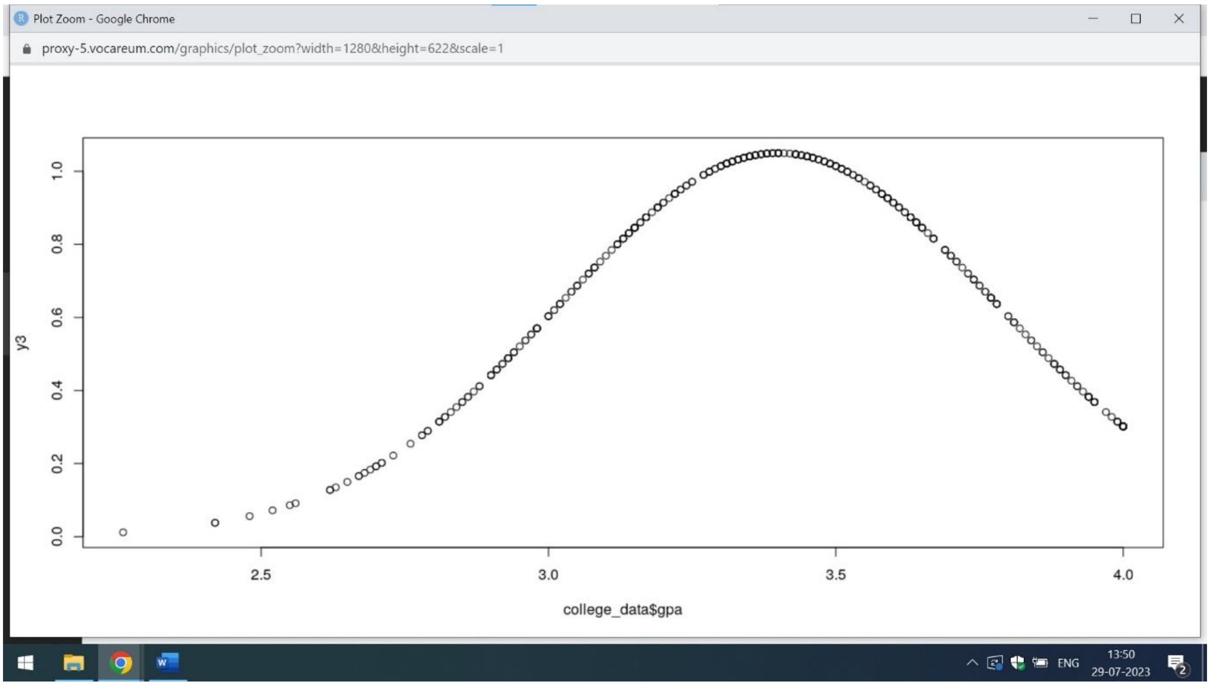
```
t1<-sd(college_data$admit)
> t2<-sd(college_data$gre)
> t3<-sd(college_data$gpa)
> t4<-sd(college_data$ses)
> t5<-sd(college_data$Gender_Male)
> t6<-sd(college_data$Race)
> t7<-sd(college_data$Rank)
>
>
>
> print(t1)
```

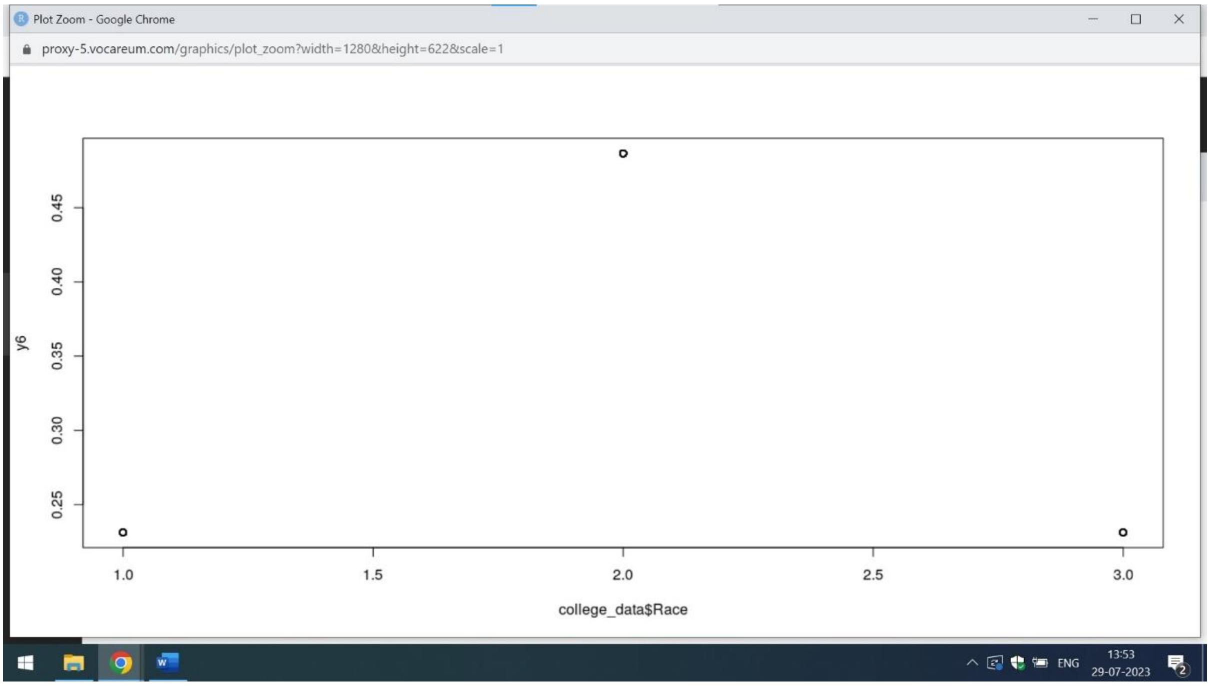
```
[1] 0.4660867
> print(t2)
[1] 115.5165
> print(t3) #0.38
[1] 0.3805668
> print(t4) #0.81
[1] 0.8087515
> print(t5) #0.5
[1] 0.5
> print(t6) #0.82
[1] 0.8232789
> print(t7) #Na
[1] NA


>
```

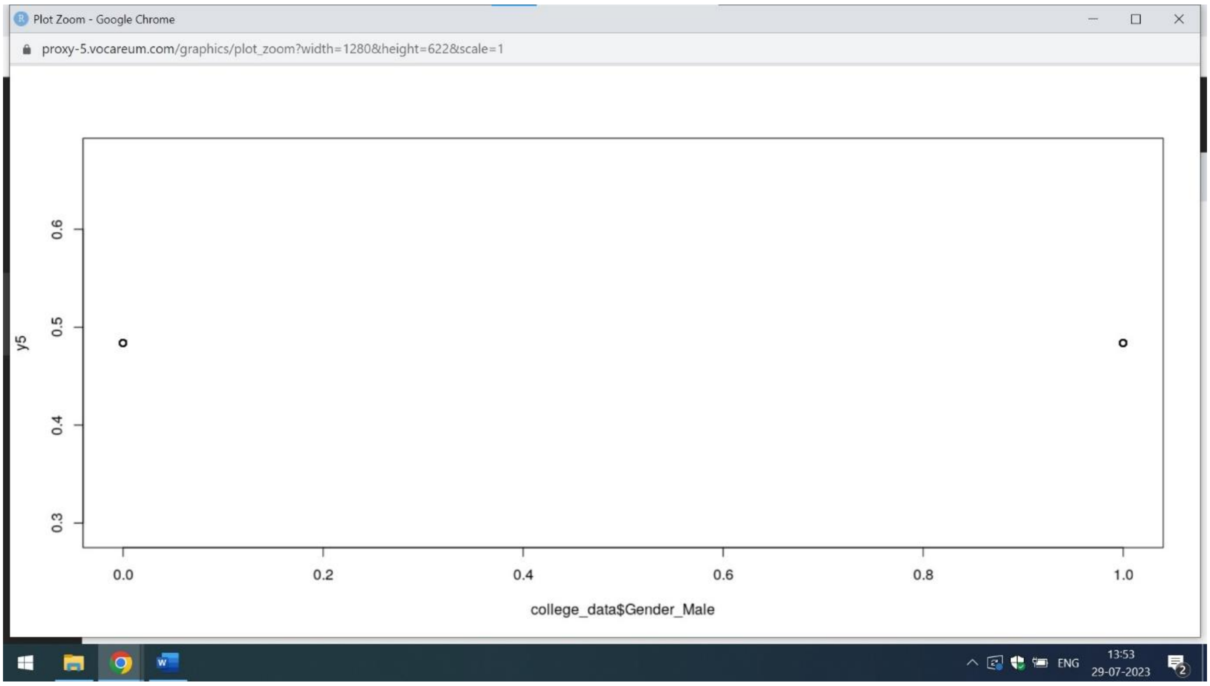
The above mention screenshot represent the standard deviation of various columns

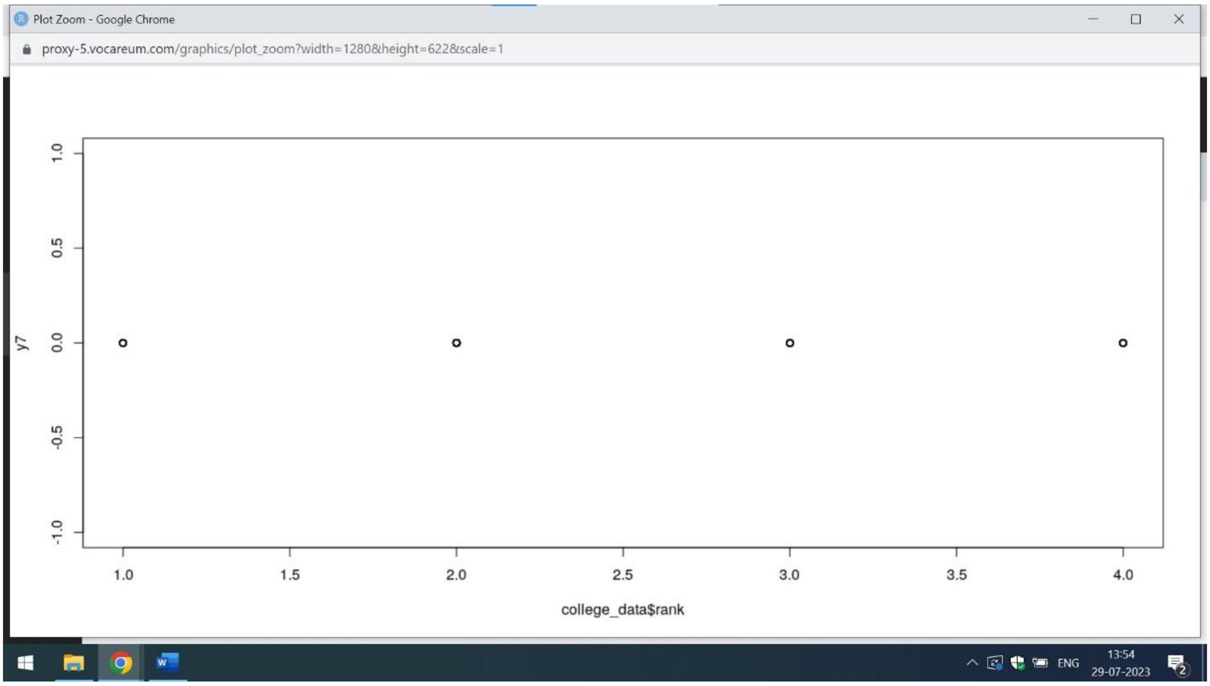



Plot Zoom - Google Chrome
proxy-5.vocareum.com/graphics/plot_zoom?width=1280&height=622&scale=1
y2
0.0000
0.0010
0.0020
0.0030
200
300
400
500
600
700
800
college_data$gre
ENG
13:50
29-07-2023


Plot Zoom - Google Chrome
proxy-5.vocareum.com/graphics/plot_zoom?width=1280&height=622&scale=1
y3
0.0
0.2
0.4
0.6
0.8
1.0
2.5
3.0
3.5
4.0
college_data$gpa
ENG
13:50
29-07-2023


Plot Zoom - Google Chrome
proxy-5.vocareum.com/graphics/plot_zoom?width=1280&height=622&scale=1
y6
0.25
0.30
0.35
0.40
0.45
1.0
1.5
2.0
2.5
3.0
college_data$Race
ENG
13:53
29-07-2023


Plot Zoom - Google Chrome
proxy-5.vocareum.com/graphics/plot_zoom?width=1280&height=622&scale=1
y5
0.3
0.4
0.5
0.6
0.0
0.2
0.4
0.6
0.8
1.0
college_data$Gender_Male
ENG
13:53
29-07-2023



Normal distribution of data represents the bell shaped curve. From the above mention screenshot it is clear that only gre(Graduate Record Exam Scores) and gpa(Grade Point Average) column has normal distribution

Q5) Normalize the data if not normally distributed.

Ans) The below attached is the code in R which shows the normal distribution of data

```
college_data<-
read.csv("https://raw.githubusercontent.com/shivanipriya89/College/main/College.csv")

print(college_data)

View(college_data)

summary(college_data)

log_scale = log(as.data.frame(college_data)) # Normalizing the data

print(log_scale)


scale_data<-as.data.frame(scale(college_data))

print(scale_data)


summary(college_data)
```

```
log_scale = log(as.data.frame(college_data)) # Normalizing the data
> print(log_scale)
   admit      gre       gpa       ses Gender_Male      Race      rank
1   -Inf 5.940171 1.2837078 0.0000000        -Inf 1.0986123 1.0986123
2      0 6.492240 1.3001917 0.6931472        -Inf 0.6931472 1.0986123
3      0 6.684612 1.3862944 0.6931472        -Inf 0.6931472 0.0000000
4      0 6.461468 1.1600209 0.0000000           0 0.6931472 1.3862944
5   -Inf 6.253829 1.0750024 1.0986123           0 0.6931472 1.3862944
6      0 6.633318 1.0986123 0.6931472           0 0.0000000 0.6931472
7      0 6.327937 1.0919233 0.6931472           0 0.6931472 0.0000000
8   -Inf 5.991465 1.1249296 0.6931472        -Inf 0.6931472 0.6931472
9      0 6.291569 1.2208299 0.0000000           0 0.0000000 1.0986123
10  -Inf 6.551080 1.3660917 0.0000000        -Inf 0.6931472 0.6931472
11  -Inf 6.684612 1.3862944 0.0000000           0 0.0000000 1.3862944
12  -Inf 6.086775 1.1693814 1.0986123        -Inf 0.6931472 0.0000000
13     0 6.633318 1.3862944 1.0986123           0 0.6931472 0.0000000
14  -Inf 6.551080 1.1249296 0.6931472        -Inf 0.6931472 0.6931472
15     0 6.551080 1.3862944 0.6931472           0 0.0000000 0.0000000
16  -Inf 6.173786 1.2354715 1.0986123        -Inf 0.0000000 1.0986123
17  -Inf 6.659294 1.3532545 0.6931472        -Inf 1.0986123 1.3862944
18  -Inf 5.886104 0.9400073 1.0986123           0 1.0986123 1.0986123
19  -Inf 6.684612 1.3217558 0.0000000           0 1.0986123 0.6931472
20     0 6.291569 1.3376292 0.0000000        -Inf 1.0986123 0.0000000
21  -Inf 6.214608 1.1537316 1.0986123        -Inf 0.6931472 1.0986123
22     0 6.492240 1.2892326 0.0000000        -Inf 0.0000000 0.6931472
23  -Inf 6.396930 1.0367369 0.0000000        -Inf 1.0986123 1.3862944
24  -Inf 6.522093 1.1600209 0.0000000        -Inf 0.0000000 1.3862944
25     0 6.633318 1.2089603 0.6931472        -Inf 0.6931472 0.6931472
26     0 6.684612 1.2974631 0.6931472           0 0.0000000 0.0000000
27     0 6.429719 1.2837078 0.6931472        -Inf 0.0000000 0.0000000
28     0 6.253829 1.3190856 0.6931472        -Inf 1.0986123 1.3862944
29     0 6.659294 1.1693814 0.0000000        -Inf 0.0000000 0.6931472
30  -Inf 6.253829 1.1908876 0.0000000        -Inf 0.0000000 0.0000000
31  -Inf 6.291569 1.3297240 0.0000000           0 0.0000000 1.3862944
32  -Inf 6.633318 1.2089603 0.6931472           0 0.0000000 1.0986123
33  -Inf 6.396930 1.2237754 1.0986123        -Inf 0.0000000 1.0986123
34     0 6.684612 1.3862944 1.0986123        -Inf 0.0000000 1.0986123
35  -Inf 5.886104 1.1442228 0.0000000           0 0.6931472 0.0000000
36  -Inf 5.991465 1.1151416 1.0986123        -Inf 0.6931472 0.6931472
37  -Inf 6.363028 1.1786550 0.0000000        -Inf 0.6931472 0.0000000
38  -Inf 6.253829 1.0647107 0.6931472        -Inf 0.6931472 1.0986123
39     0 6.214608 1.1410330 0.6931472        -Inf 0.6931472 0.6931472
40     0 6.253829 0.9858168 0.6931472        -Inf 0.0000000 1.0986123
41  -Inf 6.327937 0.8837675 0.0000000           0 1.0986123 0.6931472
42     0 6.363028 1.1999648 0.0000000        -Inf 0.0000000 0.6931472
43     0 6.396930 1.1474025 0.6931472           0 0.0000000 0.6931472
44  -Inf 6.214608 1.1969482 0.6931472        -Inf 0.6931472 1.0986123
45  -Inf 6.551080 1.0784096 0.0000000        -Inf 1.0986123 0.6931472
46     0 6.131226 1.2383742 0.6931472           0 1.0986123 1.0986123
47     0 6.363028 1.2412686 1.0986123           0 0.0000000 0.6931472
48  -Inf 6.214608 1.0885620 1.0986123        -Inf 0.6931472 1.3862944
49  -Inf 6.086775 0.9082586 1.0986123        -Inf 1.0986123 1.3862944
50  -Inf 5.991465 1.2089603 1.0986123        -Inf 0.0000000 1.0986123
51  -Inf 6.461468 1.3506672 0.6931472           0 1.0986123 1.0986123
52  -Inf 6.086775 1.1410330 0.6931472        -Inf 0.6931472 1.3862944
53  -Inf 6.606650 1.2149127 0.6931472           0 1.0986123 1.3862944
```

54	0	6.522093	1.1847900	0.6931472	-Inf	0.6931472	0.6931472
55	-Inf	6.492240	1.2059708	0.0000000	-Inf	0.0000000	1.0986123
56	0	6.606650	1.3862944	0.0000000	0	0.6931472	1.0986123
57	-Inf	6.327937	1.1600209	1.0986123	0	0.0000000	1.0986123
58	-Inf	5.940171	1.0784096	1.0986123	-Inf	0.6931472	1.0986123
59	-Inf	5.991465	1.2947272	1.0986123	0	0.6931472	0.6931472
60	-Inf	6.396930	1.0367369	1.0986123	0	0.0000000	1.3862944
61	0	6.429719	1.1568812	0.6931472	0	0.0000000	0.6931472
62	-Inf	6.327937	1.1999648	0.0000000	-Inf	1.0986123	1.3862944
63	-Inf	6.461468	1.3001917	0.0000000	0	0.6931472	1.0986123
64	0	6.522093	1.3480731	0.0000000	0	1.0986123	1.0986123
65	-Inf	6.363028	1.3862944	0.6931472	0	1.0986123	1.0986123
66	-Inf	6.396930	1.2781522	0.0000000	-Inf	0.0000000	0.6931472
67	-Inf	6.606650	1.2864740	1.0986123	0	0.6931472	1.3862944
68	-Inf	6.429719	1.1939225	0.6931472	0	1.0986123	0.0000000
69	-Inf	6.363028	1.3056265	1.0986123	-Inf	1.0986123	0.0000000
70	-Inf	6.684612	1.3164082	0.0000000	0	0.0000000	0.0000000
71	-Inf	6.461468	1.3862944	0.0000000	0	0.0000000	1.0986123
72	-Inf	5.703782	1.0715836	0.0000000	0	0.0000000	1.3862944
73	-Inf	6.173786	1.2208299	0.6931472	-Inf	0.6931472	1.3862944
74	-Inf	6.363028	1.3862944	1.0986123	-Inf	1.0986123	0.6931472
75	-Inf	6.579251	1.2383742	0.6931472	0	0.6931472	1.3862944
76	-Inf	6.579251	1.3862944	0.6931472	-Inf	1.0986123	1.0986123
77	-Inf	6.327937	1.2119410	0.0000000	0	0.6931472	1.0986123
78	0	6.684612	1.3862944	1.0986123	-Inf	1.0986123	1.0986123
79	-Inf	6.291569	1.1378330	1.0986123	0	0.6931472	0.0000000
80	0	6.429719	1.3862944	0.6931472	-Inf	0.6931472	0.0000000
81	-Inf	6.551080	1.0647107	0.6931472	-Inf	0.6931472	1.3862944
82	-Inf	6.429719	1.1216776	1.0986123	0	0.6931472	0.6931472
83	-Inf	6.214608	0.9969486	0.6931472	-Inf	1.0986123	0.6931472
84	-Inf	5.940171	1.0681531	1.0986123	0	0.6931472	1.3862944
85	0	6.214608	1.2809338	0.0000000	0	0.0000000	1.0986123
86	-Inf	6.253829	1.0919233	0.6931472	-Inf	0.6931472	0.6931472
87	-Inf	6.396930	1.1999648	0.0000000	-Inf	1.0986123	0.6931472
88	-Inf	6.396930	1.2470323	0.0000000	-Inf	0.0000000	0.6931472
89	-Inf	6.551080	1.1878434	1.0986123	-Inf	1.0986123	0.0000000
90	0	6.492240	1.3862944	0.0000000	0	0.0000000	0.6931472
91	-Inf	6.551080	1.3428648	0.6931472	-Inf	0.6931472	0.6931472
92	0	6.579251	1.2919837	0.6931472	-Inf	0.6931472	0.0000000
93	-Inf	6.684612	1.3609766	1.0986123	0	0.0000000	0.6931472
94	-Inf	6.363028	1.0750024	1.0986123	0	0.0000000	0.6931472
95	0	6.492240	1.2354715	0.6931472	-Inf	1.0986123	0.6931472
96	-Inf	6.492240	1.2029723	0.6931472	0	1.0986123	0.6931472
97	-Inf	6.461468	1.2584610	0.6931472	0	1.0986123	1.3862944
98	-Inf	6.173786	1.2725656	1.0986123	0	0.6931472	0.6931472
99	-Inf	6.551080	1.0577903	0.6931472	0	1.0986123	0.6931472
100	-Inf	5.991465	1.1969482	1.0986123	0	0.6931472	1.0986123
101	-Inf	5.828946	1.1474025	0.6931472	-Inf	0.0000000	1.0986123
102	-Inf	6.363028	1.2725656	0.0000000	0	0.6931472	1.0986123
103	-Inf	5.940171	1.2029723	1.0986123	-Inf	1.0986123	1.3862944
104	-Inf	6.291569	1.3711807	1.0986123	-Inf	0.0000000	1.0986123
105	0	6.492240	1.3737156	0.6931472	0	0.0000000	0.6931472
106	0	6.606650	1.0885620	0.0000000	0	0.0000000	0.6931472
107	0	6.551080	1.2697605	0.0000000	0	0.6931472	0.0000000
108	-Inf	6.173786	1.1410330	0.6931472	-Inf	0.0000000	0.6931472
109	-Inf	5.991465	1.0750024	0.0000000	0	1.0986123	1.0986123

```
110  -Inf 6.173786 1.2383742 1.0986123        -Inf 0.0000000 0.6931472
111  -Inf 6.522093 1.1249296 1.0986123        -Inf 1.0986123 1.3862944
112  -Inf 6.040255 1.2267123 0.6931472           0 1.0986123 1.3862944
113  -Inf 5.886104 1.0986123 0.0000000        -Inf 0.0000000 1.0986123
114  -Inf 6.396930 1.1693814 1.0986123           0 0.6931472 0.0000000
115  -Inf 6.579251 1.3454724 0.0000000           0 0.6931472 1.0986123
116  -Inf 6.429719 1.3837912 0.6931472           0 0.6931472 1.0986123
117     0 6.086775 1.2383742 0.0000000           0 1.0986123 0.6931472
118  -Inf 6.551080 1.3137237 0.6931472           0 0.6931472 0.6931472
119     0 6.684612 1.3083328 0.0000000        -Inf 0.6931472 0.0000000
120  -Inf 5.828946 1.0715836 1.0986123           0 0.6931472 1.0986123
121     0 6.253829 1.3190856 0.6931472        -Inf 0.6931472 0.6931472
122     0 6.173786 0.9820785 0.0000000        -Inf 0.0000000 0.6931472
123  -Inf 6.253829 1.0473190 1.0986123        -Inf 0.0000000 1.0986123
124  -Inf 6.214608 1.0919233 1.0986123        -Inf 0.6931472 1.0986123
125  -Inf 6.579251 1.3558352 0.6931472        -Inf 1.0986123 1.0986123
126  -Inf 6.291569 1.2178757 1.0986123        -Inf 1.0986123 1.3862944
127     0 6.396930 1.2641267 1.0986123        -Inf 1.0986123 0.0000000
128  -Inf 6.606650 1.3190856 0.0000000        -Inf 1.0986123 1.3862944
129  -Inf 6.291569 1.1600209 0.0000000           0 1.0986123 0.6931472
130  -Inf 6.131226 1.1474025 1.0986123        -Inf 0.6931472 1.3862944
131     0 6.429719 1.1537316 0.0000000        -Inf 1.0986123 0.6931472
132  -Inf 6.461468 1.0260416 1.0986123           0 0.0000000 0.6931472
133  -Inf 6.363028 1.2237754 1.0986123        -Inf 0.0000000 0.6931472
134  -Inf 6.214608 1.1249296 0.6931472           0 0.6931472 1.0986123
135  -Inf 6.327937 1.0818052 1.0986123           0 0.0000000 0.6931472
136  -Inf 6.214608 1.2725656 0.6931472           0 1.0986123 1.0986123
137  -Inf 6.327937 1.2029723 1.0986123           0 0.6931472 1.3862944
138  -Inf 6.551080 1.3862944 1.0986123           0 0.0000000 1.0986123
139  -Inf 6.429719 1.2237754 1.0986123        -Inf 0.0000000 0.6931472
140     0 6.396930 1.2753628 1.0986123        -Inf 1.0986123 0.0000000
141  -Inf 6.461468 1.3686394 0.6931472           0 0.6931472 0.6931472
142     0 6.551080 1.2584610 0.6931472        -Inf 0.0000000 1.3862944
 [ reached 'max' / getOption("max.print") -- omitted 258 rows ]
```

```
>
scale_data<-as.data.frame(scale(college_data))
> print(scale_data)
        admit         gre          gpa          ses Gender_Male
1   -0.6812037 -1.79801097  0.578347918 -1.227200236       -0.95
2    1.4643197  0.62588442  0.736007505  0.009273553       -0.95
3    1.4643197  1.83783211  1.603135233  0.009273553       -0.95
4    1.4643197  0.45274903 -0.525269190 -1.227200236        1.05
5   -0.6812037 -0.58606328 -1.208460734  1.245747343        1.05
6    1.4643197  1.49156134 -1.024524549  0.009273553        1.05
7    1.4643197 -0.23979251 -1.077077744  0.009273553        1.05
8   -0.6812037 -1.62487559 -0.814311766  0.009273553       -0.95
9    1.4643197 -0.41292789  0.000262766 -1.227200236        1.05
10  -0.6812037  0.97215519  1.392922450 -1.227200236       -0.95
11  -0.6812037  1.83783211  1.603135233 -1.227200236        1.05
12  -0.6812037 -1.27860482 -0.446439397  1.245747343       -0.95
13   1.4643197  1.49156134  1.603135233  1.245747343        1.05
14  -0.6812037  0.97215519 -0.814311766  0.009273553       -0.95
15   1.4643197  0.97215519  1.603135233  0.009273553        1.05
16  -0.6812037 -0.93233405  0.131645755  1.245747343       -0.95
```

17	-0.6812037	1.66469673	1.261539461	0.009273553	-0.95
18	-0.6812037	-1.97114636	-2.180694853	1.245747343	1.05
19	-0.6812037	1.83783211	0.946220287	-1.227200236	1.05
20	1.4643197	-0.41292789	1.103879874	-1.227200236	-0.95
21	-0.6812037	-0.75919866	-0.577822386	1.245747343	-0.95
22	1.4643197	0.62588442	0.630901114	-1.227200236	-0.95
23	-0.6812037	0.10647826	-1.497503309	-1.227200236	-0.95
24	-0.6812037	0.79901980	-0.525269190	-1.227200236	-0.95
25	1.4643197	1.49156134	-0.104843625	0.009273553	-0.95
26	1.4643197	1.83783211	0.709730907	0.009273553	1.05
27	1.4643197	0.27961365	0.578347918	0.009273553	-0.95
28	1.4643197	-0.58606328	0.919943690	0.009273553	-0.95
29	1.4643197	1.66469673	-0.446439397	-1.227200236	-0.95
30	-0.6812037	-0.58606328	-0.262503212	-1.227200236	-0.95
31	-0.6812037	-0.41292789	1.025050081	-1.227200236	1.05
32	-0.6812037	1.49156134	-0.104843625	0.009273553	1.05
33	-0.6812037	0.10647826	0.026539364	1.245747343	-0.95
34	1.4643197	1.83783211	1.603135233	1.245747343	-0.95
35	-0.6812037	-1.97114636	-0.656652179	-1.227200236	1.05
36	-0.6812037	-1.62487559	-0.893141560	1.245747343	-0.95
37	-0.6812037	-0.06665712	-0.367609603	-1.227200236	-0.95
38	-0.6812037	-0.58606328	-1.287290527	0.009273553	-0.95
39	1.4643197	-0.75919866	-0.682928777	0.009273553	-0.95
40	1.4643197	-0.58606328	-1.865375679	0.009273553	-0.95
41	-0.6812037	-0.23979251	-2.548567222	-1.227200236	1.05
42	1.4643197	-0.06665712	-0.183673419	-1.227200236	-0.95
43	1.4643197	0.10647826	-0.630375582	0.009273553	1.05
44	-0.6812037	-0.75919866	-0.209950017	0.009273553	-0.95
45	-0.6812037	0.97215519	-1.182184136	-1.227200236	-0.95
46	1.4643197	-1.10546943	0.157922353	0.009273553	1.05
47	1.4643197	-0.06665712	0.184198951	1.245747343	1.05
48	-0.6812037	-0.75919866	-1.103354342	1.245747343	-0.95
49	-0.6812037	-1.27860482	-2.390907635	1.245747343	-0.95
50	-0.6812037	-1.62487559	-0.104843625	1.245747343	-0.95
51	-0.6812037	0.45274903	1.235262863	0.009273553	1.05
52	-0.6812037	-1.27860482	-0.682928777	0.009273553	-0.95
53	-0.6812037	1.31842596	-0.052290430	0.009273553	1.05
54	1.4643197	0.79901980	-0.315056408	0.009273553	-0.95
55	-0.6812037	0.62588442	-0.131120223	-1.227200236	-0.95
56	1.4643197	1.31842596	1.603135233	-1.227200236	1.05
57	-0.6812037	-0.23979251	-0.525269190	1.245747343	1.05
58	-0.6812037	-1.79801097	-1.182184136	1.245747343	-0.95
59	-0.6812037	-1.62487559	0.683454309	1.245747343	1.05
60	-0.6812037	0.10647826	-1.497503309	1.245747343	1.05
61	1.4643197	0.27961365	-0.551545788	0.009273553	1.05
62	-0.6812037	-0.23979251	-0.183673419	-1.227200236	-0.95
63	-0.6812037	0.45274903	0.736007505	-1.227200236	1.05
64	1.4643197	0.79901980	1.208986265	-1.227200236	1.05
65	-0.6812037	-0.06665712	1.603135233	0.009273553	1.05
66	-0.6812037	0.10647826	0.525794722	-1.227200236	-0.95
67	-0.6812037	1.31842596	0.604624516	1.245747343	1.05
68	-0.6812037	0.27961365	-0.236226614	0.009273553	1.05
69	-0.6812037	-0.06665712	0.788560700	1.245747343	-0.95
70	-0.6812037	1.83783211	0.893667092	-1.227200236	1.05
71	-0.6812037	0.45274903	1.603135233	-1.227200236	1.05
72	-0.6812037	-2.49055251	-1.234737331	-1.227200236	1.05

73	-0.6812037	-0.93233405	0.000262766	0.009273553	-0.95
74	-0.6812037	-0.06665712	1.603135233	1.245747343	-0.95
75	-0.6812037	1.14529057	0.157922353	0.009273553	1.05
76	-0.6812037	1.14529057	1.603135233	0.009273553	-0.95
77	-0.6812037	-0.23979251	-0.078567027	-1.227200236	1.05
78	1.4643197	1.83783211	1.603135233	1.245747343	-0.95
79	-0.6812037	-0.41292789	-0.709205375	1.245747343	1.05
80	1.4643197	0.27961365	1.603135233	0.009273553	-0.95
81	-0.6812037	0.97215519	-1.287290527	0.009273553	-0.95
82	-0.6812037	0.27961365	-0.840588364	1.245747343	1.05
83	-0.6812037	-0.75919866	-1.786545885	0.009273553	-0.95
84	-0.6812037	-1.79801097	-1.261013929	1.245747343	1.05
85	1.4643197	-0.75919866	0.552071320	-1.227200236	1.05
86	-0.6812037	-0.58606328	-1.077077744	0.009273553	-0.95
87	-0.6812037	0.10647826	-0.183673419	-1.227200236	-0.95
88	-0.6812037	0.10647826	0.236752146	-1.227200236	-0.95
89	-0.6812037	0.97215519	-0.288779810	1.245747343	-0.95
90	1.4643197	0.62588442	1.603135233	-1.227200236	1.05
91	-0.6812037	0.97215519	1.156433070	0.009273553	-0.95
92	1.4643197	1.14529057	0.657177711	0.009273553	-0.95
93	-0.6812037	1.83783211	1.340369255	1.245747343	1.05
94	-0.6812037	-0.06665712	-1.208460734	1.245747343	1.05
95	1.4643197	0.62588442	0.131645755	0.009273553	-0.95
96	-0.6812037	0.62588442	-0.157396821	0.009273553	1.05
97	-0.6812037	0.45274903	0.341858538	0.009273553	1.05
98	-0.6812037	-0.93233405	0.473241527	1.245747343	1.05
99	-0.6812037	0.97215519	-1.339843723	0.009273553	1.05
100	-0.6812037	-1.62487559	-0.209950017	1.245747343	1.05
101	-0.6812037	-2.14428174	-0.630375582	0.009273553	-0.95
102	-0.6812037	-0.06665712	0.473241527	-1.227200236	1.05
103	-0.6812037	-1.79801097	-0.157396821	1.245747343	-0.95
104	-0.6812037	-0.41292789	1.445475646	1.245747343	-0.95
105	1.4643197	0.62588442	1.471752244	0.009273553	1.05
106	1.4643197	1.31842596	-1.103354342	-1.227200236	1.05
107	1.4643197	0.97215519	0.446964929	-1.227200236	1.05
108	-0.6812037	-0.93233405	-0.682928777	0.009273553	-0.95
109	-0.6812037	-1.62487559	-1.208460734	-1.227200236	1.05
110	-0.6812037	-0.93233405	0.157922353	1.245747343	-0.95
111	-0.6812037	0.79901980	-0.814311766	1.245747343	-0.95
112	-0.6812037	-1.45174020	0.052815962	0.009273553	1.05
113	-0.6812037	-1.97114636	-1.024524549	-1.227200236	-0.95
114	-0.6812037	0.10647826	-0.446439397	1.245747343	1.05
115	-0.6812037	1.14529057	1.182709668	-1.227200236	1.05
116	-0.6812037	0.27961365	1.576858635	0.009273553	1.05
117	1.4643197	-1.27860482	0.157922353	-1.227200236	1.05
118	-0.6812037	0.97215519	0.867390494	0.009273553	1.05
119	1.4643197	1.83783211	0.814837298	-1.227200236	-0.95
120	-0.6812037	-2.14428174	-1.234737331	1.245747343	1.05
121	1.4643197	-0.58606328	0.919943690	0.009273553	-0.95
122	1.4643197	-0.93233405	-1.891652277	-1.227200236	-0.95
123	-0.6812037	-0.58606328	-1.418673516	1.245747343	-0.95
124	-0.6812037	-0.75919866	-1.077077744	1.245747343	-0.95
125	-0.6812037	1.14529057	1.287816059	0.009273553	-0.95
126	-0.6812037	-0.41292789	-0.026013832	1.245747343	-0.95
127	1.4643197	0.10647826	0.394411733	1.245747343	-0.95
128	-0.6812037	1.31842596	0.919943690	-1.227200236	-0.95

```
129 -0.6812037 -0.41292789 -0.525269190 -1.227200236        1.05
130 -0.6812037 -1.10546943 -0.630375582  1.245747343       -0.95
131  1.4643197  0.27961365 -0.577822386 -1.227200236       -0.95
132 -0.6812037  0.45274903 -1.576333103  1.245747343        1.05
133 -0.6812037 -0.06665712  0.026539364  1.245747343       -0.95
134 -0.6812037 -0.75919866 -0.814311766  0.009273553        1.05
135 -0.6812037 -0.23979251 -1.155907538  1.245747343        1.05
136 -0.6812037 -0.75919866  0.473241527  0.009273553        1.05
137 -0.6812037 -0.23979251 -0.157396821  1.245747343        1.05
138 -0.6812037  0.97215519  1.603135233  1.245747343        1.05
139 -0.6812037  0.27961365  0.026539364  1.245747343       -0.95
140  1.4643197  0.10647826  0.499518124  1.245747343       -0.95
141 -0.6812037  0.45274903  1.419199048  0.009273553        1.05
142  1.4643197  0.97215519  0.341858538  0.009273553       -0.95
           Race       rank
1    1.26020471  0.5452850
2    0.04554957  0.5452850
3    0.04554957 -1.5723268
4    0.04554957  1.6040909
5    0.04554957  1.6040909
6   -1.16910557 -0.5135209
7    0.04554957 -1.5723268
8    0.04554957 -0.5135209
9   -1.16910557  0.5452850
10   0.04554957 -0.5135209
11  -1.16910557  1.6040909
12   0.04554957 -1.5723268
13   0.04554957 -1.5723268
14   0.04554957 -0.5135209
15  -1.16910557 -1.5723268
16  -1.16910557  0.5452850
17   1.26020471  1.6040909
18   1.26020471  0.5452850
19   1.26020471 -0.5135209
20   1.26020471 -1.5723268
21   0.04554957  0.5452850
22  -1.16910557 -0.5135209
23   1.26020471  1.6040909
24  -1.16910557  1.6040909
25   0.04554957 -0.5135209
26  -1.16910557 -1.5723268
27  -1.16910557 -1.5723268
28   1.26020471  1.6040909
29  -1.16910557 -0.5135209
30  -1.16910557 -1.5723268
31  -1.16910557  1.6040909
32  -1.16910557  0.5452850
33  -1.16910557  0.5452850
34  -1.16910557  0.5452850
35   0.04554957 -1.5723268
36   0.04554957 -0.5135209
37   0.04554957 -1.5723268
38   0.04554957  0.5452850
39   0.04554957 -0.5135209
40  -1.16910557  0.5452850
41   1.26020471 -0.5135209
```

42	-1.16910557	-0.5135209
43	-1.16910557	-0.5135209
44	0.04554957	0.5452850
45	1.26020471	-0.5135209
46	1.26020471	0.5452850
47	-1.16910557	-0.5135209
48	0.04554957	1.6040909
49	1.26020471	1.6040909
50	-1.16910557	0.5452850
51	1.26020471	0.5452850
52	0.04554957	1.6040909
53	1.26020471	1.6040909
54	0.04554957	-0.5135209
55	-1.16910557	0.5452850
56	0.04554957	0.5452850
57	-1.16910557	0.5452850
58	0.04554957	0.5452850
59	0.04554957	-0.5135209
60	-1.16910557	1.6040909
61	-1.16910557	-0.5135209
62	1.26020471	1.6040909
63	0.04554957	0.5452850
64	1.26020471	0.5452850
65	1.26020471	0.5452850
66	-1.16910557	-0.5135209
67	0.04554957	1.6040909
68	1.26020471	-1.5723268
69	1.26020471	-1.5723268
70	-1.16910557	-1.5723268
71	-1.16910557	0.5452850
72	-1.16910557	1.6040909
73	0.04554957	1.6040909
74	1.26020471	-0.5135209
75	0.04554957	1.6040909
76	1.26020471	0.5452850
77	0.04554957	0.5452850
78	1.26020471	0.5452850
79	0.04554957	-1.5723268
80	0.04554957	-1.5723268
81	0.04554957	1.6040909
82	0.04554957	-0.5135209
83	1.26020471	-0.5135209
84	0.04554957	1.6040909
85	-1.16910557	0.5452850
86	0.04554957	-0.5135209
87	1.26020471	-0.5135209
88	-1.16910557	-0.5135209
89	1.26020471	-1.5723268
90	-1.16910557	-0.5135209
91	0.04554957	-0.5135209
92	0.04554957	-1.5723268
93	-1.16910557	-0.5135209
94	-1.16910557	-0.5135209
95	1.26020471	-0.5135209
96	1.26020471	-0.5135209
97	1.26020471	1.6040909

```
98   0.04554957 -0.5135209
99   1.26020471 -0.5135209
100  0.04554957  0.5452850
101 -1.16910557  0.5452850
102  0.04554957  0.5452850
103  1.26020471  1.6040909
104 -1.16910557  0.5452850
105 -1.16910557 -0.5135209
106 -1.16910557 -0.5135209
107  0.04554957 -1.5723268
108 -1.16910557 -0.5135209
109  1.26020471  0.5452850
110 -1.16910557 -0.5135209
111  1.26020471  1.6040909
112  1.26020471  1.6040909
113 -1.16910557  0.5452850
114  0.04554957 -1.5723268
115  0.04554957  0.5452850
116  0.04554957  0.5452850
117  1.26020471 -0.5135209
118  0.04554957 -0.5135209
119  0.04554957 -1.5723268
120  0.04554957  0.5452850
121  0.04554957 -0.5135209
122 -1.16910557 -0.5135209
123 -1.16910557  0.5452850
124  0.04554957  0.5452850
125  1.26020471  0.5452850
126  1.26020471  1.6040909
127  1.26020471 -1.5723268
128  1.26020471  1.6040909
129  1.26020471 -0.5135209
130  0.04554957  1.6040909
131  1.26020471 -0.5135209
132 -1.16910557 -0.5135209
133 -1.16910557 -0.5135209
134  0.04554957  0.5452850
135 -1.16910557 -0.5135209
136  1.26020471  0.5452850
137  0.04554957  1.6040909
138 -1.16910557  0.5452850
139 -1.16910557 -0.5135209
140  1.26020471 -1.5723268
141  0.04554957 -0.5135209
142 -1.16910557  1.6040909
 [ reached 'max' / getOption("max.print") -- omitted 258 rows ]
```

The above mention screenshot represent the normalize view of data. Data's are normalized via log and scale functions. The output of scale and log values are attached in the above mentioned screenshots

Q6) Use variable reduction techniques to identify significant variables.

Ans) Principal Component Analysis(PCA) is an important variable reduction techniques to identify the significant variables in R

Below attached is the code in R

college_data<-
read.csv("https://raw.githubusercontent.com/shivanipriya89/College/main/College.csv")

print(college_data)

View(college_data)

data_normalized <- scale(college_data)

head(data_normalized)

corr_matrix <- cor(data_normalized)

data.pca <- princomp(corr_matrix)

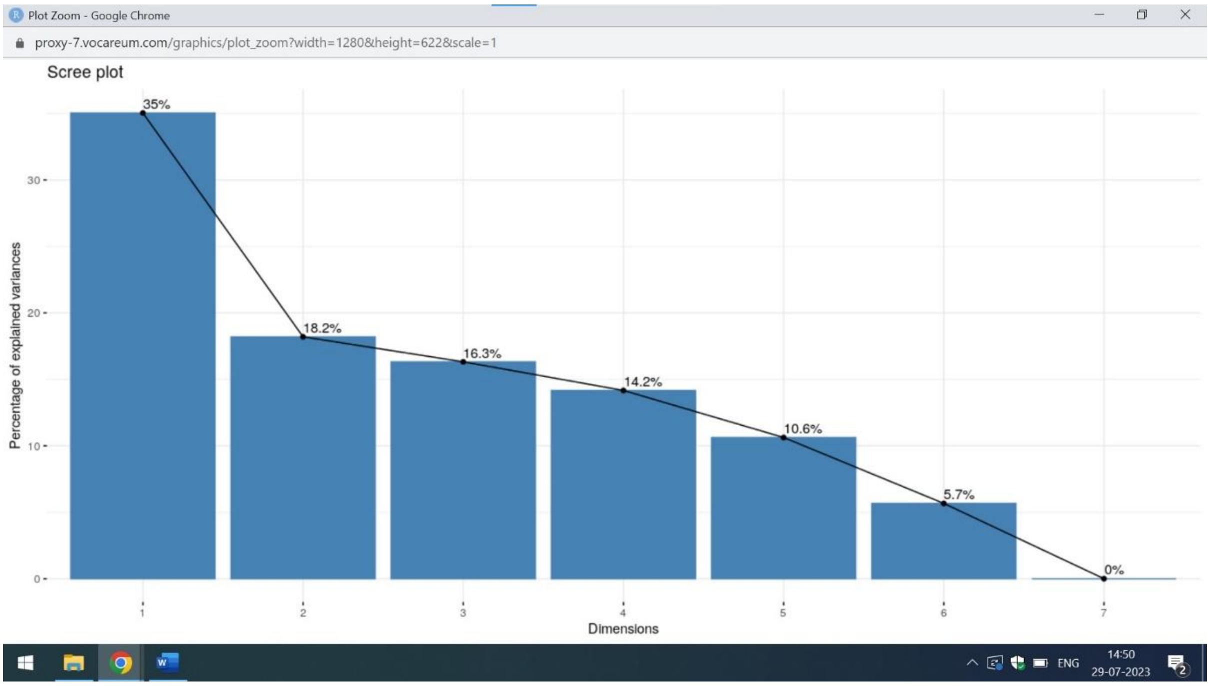
summary(data.pca)

fviz_eig(data.pca, addlabels = TRUE)

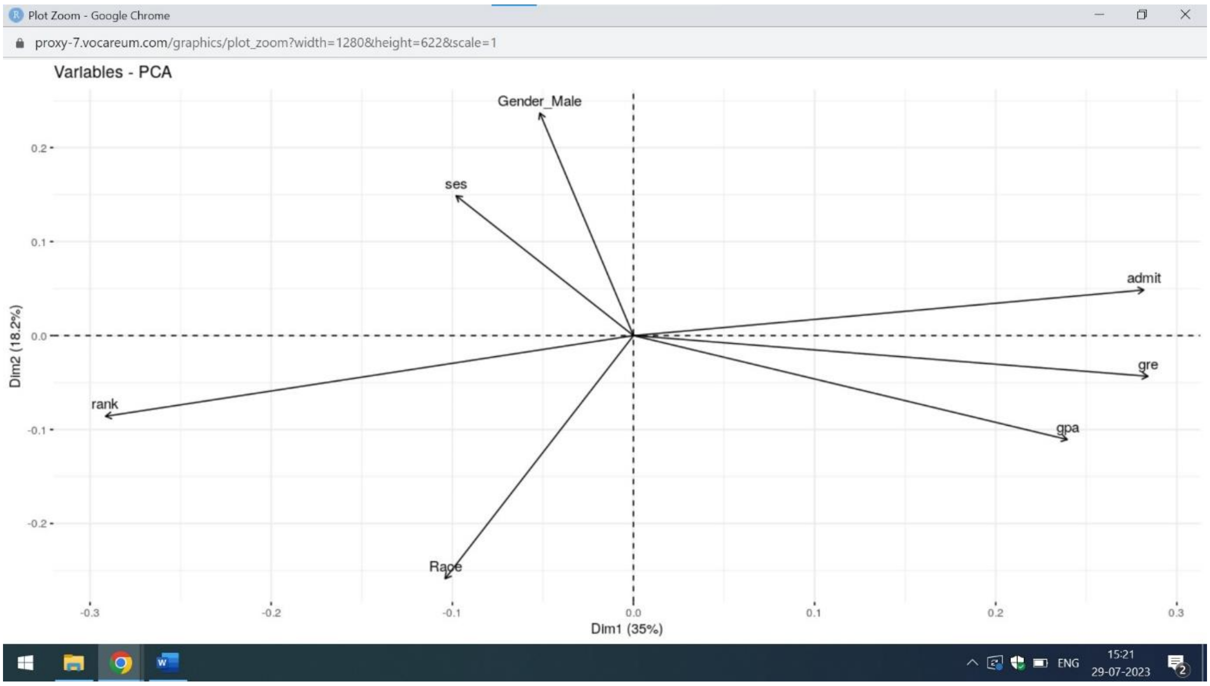
fviz_pca_var(data.pca, col.var = "black")

```
corr_matrix <- cor(data_normalized)
> data.pca <- princomp(corr_matrix)
> summary(data.pca)
Importance of components:
                          Comp.1    Comp.2    Comp.3    Comp.4    Comp.5
Standard deviation     0.5705470 0.4111876 0.3893538 0.3627437 0.3141902
Proportion of Variance 0.3503618 0.1819759 0.1631634 0.1416229 0.1062476
Cumulative Proportion  0.3503618 0.5323377 0.6955011 0.8371240 0.9433716
                           Comp.6       Comp.7
Standard deviation     0.22937709 2.528936e-09
Proportion of Variance 0.05662836 6.883506e-18
Cumulative Proportion  1.00000000 1.000000e+00


>
```

The above mention screenshot represents the standard deviation,proportion of variance and Cumulative proportion of various components



The above attached is the screeplot which explains the percentage of explained variances over dimensions. The first two components ie component 1 and component 2 holds the 53% of overall information of data. Hence the two most important variables of the dataset are admit and gre



The above attached is the biplot of the attributes which gives the information of the variables of the dataset.

From the above mention screenshot,it is clear that admit is the most important variable of the college data set and have the highest value in the loading matrix followed by gre and gpa.

Q7) Run logistic model to determine the factors that influence the admission process of a student (Drop insignificant variables)

Ans) For analysing the factors that influence the admission process of a student,I am using the concept of Simple Linear Regression with Multiple variables. Below attached is the code in R

print("College Admission")

college_data<-
read.csv("https://raw.githubusercontent.com/shivanipriya89/College/main/College.csv")

print(college_data)

View(college_data)

admission<-lm(formula=admit~gre+gpa+ses+Gender_Male+Race+rank,data=college_data)

print(admission)

Dropping the columns

select(college_data,-c("ses","Gender_Male","Race"))

Below attached are the output of simple Linear Regression

```
admission<-lm(formula=admit~gre+gpa+ses+Gender_Male+Race+rank,data=college_dat
a)
> print(admission)

Call:
lm(formula = admit ~ gre + gpa + ses + Gender_Male + Race + rank,
    data = college_data)

Coefficients:
(Intercept)          gre          gpa          ses  Gender_Male
 -0.0581285    0.0004178    0.1579084   -0.0268790   -0.0317487
       Race         rank
 -0.0335600   -0.1089262
```

From the above mention output of the simple Linear Regression,it is clear that major factors affecting the admission process of student are gre and gpa.As gre and gpa are the major factors affecting the admission process of the student,so I am dropping the rest of the column. Below attached is the screenshot

```
> select(college_data,-c("ses","Gender_Male","Race"))
    admit gre  gpa rank
1       0 380 3.61    3
2       1 660 3.67    3
```

3	1	800	4.00	1
4	1	640	3.19	4
5	0	520	2.93	4
6	1	760	3.00	2
7	1	560	2.98	1
8	0	400	3.08	2
9	1	540	3.39	3
10	0	700	3.92	2
11	0	800	4.00	4
12	0	440	3.22	1
13	1	760	4.00	1
14	0	700	3.08	2
15	1	700	4.00	1
16	0	480	3.44	3
17	0	780	3.87	4
18	0	360	2.56	3
19	0	800	3.75	2
20	1	540	3.81	1
21	0	500	3.17	3
22	1	660	3.63	2
23	0	600	2.82	4
24	0	680	3.19	4
25	1	760	3.35	2
26	1	800	3.66	1
27	1	620	3.61	1
28	1	520	3.74	4
29	1	780	3.22	2
30	0	520	3.29	1
31	0	540	3.78	4
32	0	760	3.35	3
33	0	600	3.40	3
34	1	800	4.00	3
35	0	360	3.14	1
36	0	400	3.05	2
37	0	580	3.25	1
38	0	520	2.90	3
39	1	500	3.13	2
40	1	520	2.68	3
41	0	560	2.42	2
42	1	580	3.32	2
43	1	600	3.15	2
44	0	500	3.31	3
45	0	700	2.94	2
46	1	460	3.45	3
47	1	580	3.46	2
48	0	500	2.97	4
49	0	440	2.48	4
50	0	400	3.35	3
51	0	640	3.86	3
52	0	440	3.13	4
53	0	740	3.37	4
54	1	680	3.27	2
55	0	660	3.34	3
56	1	740	4.00	3
57	0	560	3.19	3
58	0	380	2.94	3

59	0	400	3.65	2
60	0	600	2.82	4
61	1	620	3.18	2
62	0	560	3.32	4
63	0	640	3.67	3
64	1	680	3.85	3
65	0	580	4.00	3
66	0	600	3.59	2
67	0	740	3.62	4
68	0	620	3.30	1
69	0	580	3.69	1
70	0	800	3.73	1
71	0	640	4.00	3
72	0	300	2.92	4
73	0	480	3.39	4
74	0	580	4.00	2
75	0	720	3.45	4
76	0	720	4.00	3
77	0	560	3.36	3
78	1	800	4.00	3
79	0	540	3.12	1
80	1	620	4.00	1
81	0	700	2.90	4
82	0	620	3.07	2
83	0	500	2.71	2
84	0	380	2.91	4
85	1	500	3.60	3
86	0	520	2.98	2
87	0	600	3.32	2
88	0	600	3.48	2
89	0	700	3.28	1
90	1	660	4.00	2
91	0	700	3.83	2
92	1	720	3.64	1
93	0	800	3.90	2
94	0	580	2.93	2
95	1	660	3.44	2
96	0	660	3.33	2
97	0	640	3.52	4
98	0	480	3.57	2
99	0	700	2.88	2
100	0	400	3.31	3
101	0	340	3.15	3
102	0	580	3.57	3
103	0	380	3.33	4
104	0	540	3.94	3
105	1	660	3.95	2
106	1	740	2.97	2
107	1	700	3.56	1
108	0	480	3.13	2
109	0	400	2.93	3
110	0	480	3.45	2
111	0	680	3.08	4
112	0	420	3.41	4
113	0	360	3.00	3
114	0	600	3.22	1

115	0	720	3.84	3
116	0	620	3.99	3
117	1	440	3.45	2
118	0	700	3.72	2
119	1	800	3.70	1
120	0	340	2.92	3
121	1	520	3.74	2
122	1	480	2.67	2
123	0	520	2.85	3
124	0	500	2.98	3
125	0	720	3.88	3
126	0	540	3.38	4
127	1	600	3.54	1
128	0	740	3.74	4
129	0	540	3.19	2
130	0	460	3.15	4
131	1	620	3.17	2
132	0	640	2.79	2
133	0	580	3.40	2
134	0	500	3.08	3
135	0	560	2.95	2
136	0	500	3.57	3
137	0	560	3.33	4
138	0	700	4.00	3
139	0	620	3.40	2
140	1	600	3.58	1
141	0	640	3.93	2
142	1	700	3.52	4
143	0	620	3.94	4
144	0	580	3.40	3
145	0	580	3.40	4
146	0	380	3.43	3
147	0	480	3.40	2
148	0	560	2.71	3
149	1	480	2.91	1
150	0	740	3.31	1
151	1	800	3.74	1
152	0	400	3.38	2
153	1	640	3.94	2
154	0	580	3.46	3
155	0	620	3.69	3
156	1	580	2.86	4
157	0	560	2.52	2
158	1	480	3.58	1
159	0	660	3.49	2
160	0	700	3.82	3
161	0	600	3.13	2
162	0	640	3.50	2
163	1	700	3.56	2
164	0	520	2.73	2
165	0	580	3.30	2
166	0	700	4.00	1
167	0	440	3.24	4
168	0	720	3.77	3
169	0	500	4.00	3
170	0	600	3.62	3

171	0	400	3.51	3
172	0	540	2.81	3
173	0	680	3.48	3
174	1	800	3.43	2
175	0	500	3.53	4
176	1	620	3.37	2
177	0	520	2.62	2
178	1	620	3.23	3
179	0	620	3.33	3
180	0	300	3.01	3
181	0	620	3.78	3
182	0	500	3.88	4
183	0	700	4.00	2
184	1	540	3.84	2
185	0	500	2.79	4
186	0	800	3.60	2
187	0	560	3.61	3
188	0	580	2.88	2
189	0	560	3.07	2
190	0	500	3.35	2
191	1	640	2.94	2
192	0	800	3.54	3
193	0	640	3.76	3
194	0	380	3.59	4
195	1	600	3.47	2
196	0	560	3.59	2
197	0	660	3.07	3
198	1	400	3.23	4
199	0	600	3.63	3
200	0	580	3.77	4
201	0	800	3.31	3
202	1	580	3.20	2
203	1	700	4.00	1
204	0	420	3.92	4
205	1	600	3.89	1
206	1	780	3.80	3
207	0	740	3.54	1
208	1	640	3.63	1
209	0	540	3.16	3
210	0	580	3.50	2
211	0	740	3.34	4
212	0	580	3.02	2
213	0	460	2.87	2
214	0	640	3.38	3
215	1	600	3.56	2
216	1	660	2.91	3
217	0	340	2.90	1
218	1	460	3.64	1
219	0	460	2.98	1
220	1	560	3.59	2
221	0	540	3.28	3
222	0	680	3.99	3
223	1	480	3.02	1
224	0	800	3.47	3
225	0	800	2.90	2
226	1	720	3.50	3

```
227      0 620 3.58    2
228      0 540 3.02    4
229      0 480 3.43    2
230      1 720 3.42    2
231      0 580 3.29    4
232      0 600 3.28    3
233      0 380 3.38    2
234      0 420 2.67    3
235      1 800 3.53    1
236      0 620 3.05    2
237      1 660 3.49    2
238      0 480 4.00    2
239      0 500 2.86    4
240      0 700 3.45    3
241      0 440 2.76    2
242      1 520 3.81    1
243      1 680 2.96    3
244      0 620 3.22    2
245      0 540 3.04    1
246      0 800 3.91    3
247      0 680 3.34    2
248      0 440 3.17    2
249      0 680 3.64    3
250      0 640 3.73    3
 [ reached 'max' / getOption("max.print") -- omitted 150 rows ]
```

Q8) Calculate the accuracy of the model and run validation techniques.

Ans) For determining the accuracy of the model,I am using the concept of K-Fold Cross Vali dation. Below attached is the code in R

```
# K-Fold Cross Validation
college_data<-read.csv("https://raw.githubusercontent.com/shivanipriya89/College/main/Coll
ege.csv")
print(college_data)
View(college_data)
str(college_data)

# Convert admit to factor

college_data$admit<-sapply(college_data$admit,factor)

# Define Training Control
mytraining<-trainControl(method="cv",number=10)

# Fix the Parameters of the Algorithm
grid<-expand.grid(.fL = c(0), .usekernel = c(FALSE),.adjust = 0.5 )

# train the model
model<-train(admit ~., data = college_data, trControl = mytraining, method = 'nb', tuneGrid =
grid)
```

print(model)

```
> print(model)
Naive Bayes

400 samples
  6 predictor
  2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 360, 360, 360, 359, 360, 360, ...
Resampling results:

  Accuracy   Kappa
  0.6751517  0.13692

Tuning parameter 'fL' was held constant at a value of 0
Tuning
 parameter 'usekernel' was held constant at a value of FALSE

Tuning parameter 'adjust' was held constant at a value of 0.5


>
```

The above mention is the output of the K-Fold Cross Validation. The accuracy of this model is around 68%(approx.)

Q9) Try other modelling techniques like decision tree and SVM and select a champion model

Ans) Below attached is the code in R which gives an overview of the decision tree and SVM

Decision Tree

college_data<-read.csv("https://raw.githubusercontent.com/shivanipriya89/College/main/College.csv")

print(college_data)

View(college_data)

str(college_data)

Convert admit to factor

college_data$admit<-sapply(college_data$admit,factor)

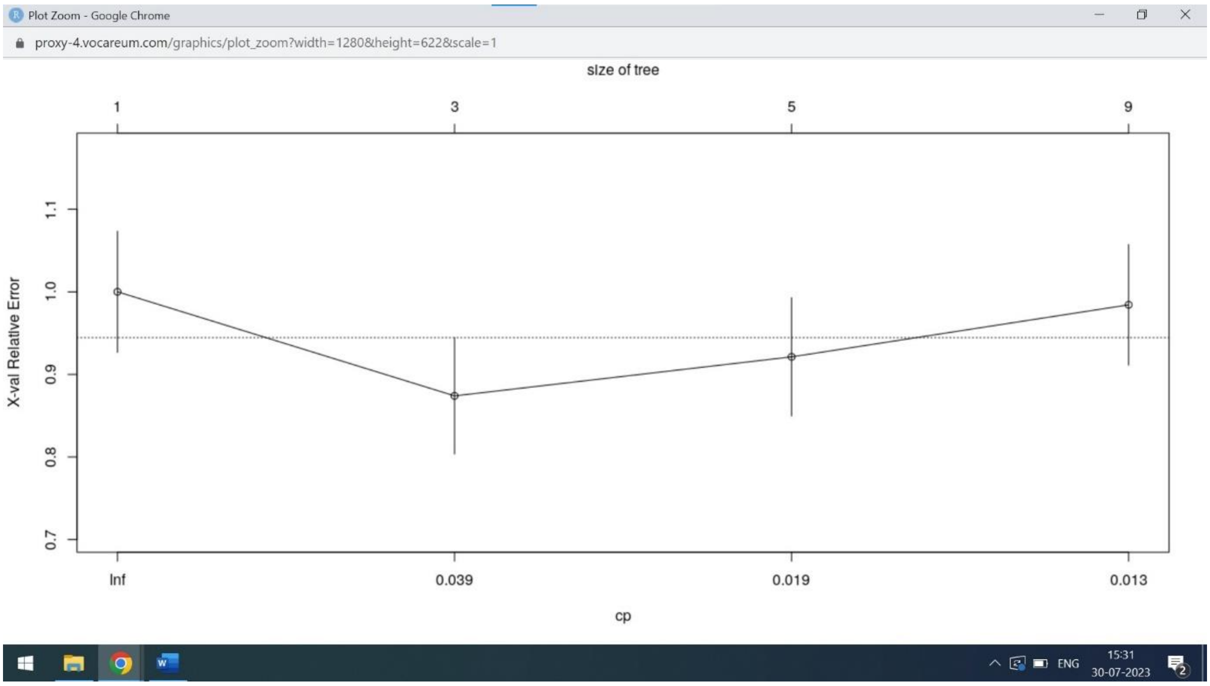
str(college_data)

Building the mode

```
college_model<-rpart(admit~.,data=college_data,method="class")

college_model

printcp(college_model)

plotcp(college_model)

summary(college_model)


# Support Vector Machine

college_data<-
read.csv("https://raw.githubusercontent.com/shivanipriya89/College/main/College.csv")

print(college_data)

View(college_data)

str(college_data)


# Convert admit to factor

college_data$admit<-sapply(college_data$admit,factor)


str(college_data)


# Spliting the data

sample_split<-floor(0.7*nrow(college_data))

set.seed(1)

training<-sample(seq_len(nrow(college_data)),size=sample_split)


# Training and testing Data


mytraining<-college_data[training,]

mytesting<-college_data[-training,]


# Support Vector Machine

support_vector<-svm(admit~.,mytraining)

confusionMatrix(mytraining$admit,predict(support_vector),positive='1')
```

The output of the Decision Tree and SVM are mention below:

```
library(rpart)
> college_model<-rpart(admit~.,data=college_data,method="class")
> college_model
n= 400

node), split, n, loss, yval, (yprob)
      * denotes terminal node

  1) root 400 127 0 (0.6825000 0.3175000)
    2) gpa< 3.415 208  45 0 (0.7836538 0.2163462)
      4) rank>=2.5 99  13 0 (0.8686869 0.1313131) *
      5) rank< 2.5 109  32 0 (0.7064220 0.2935780)
       10) gre< 730 99  25 0 (0.7474747 0.2525253) *
       11) gre>=730 10   3 1 (0.3000000 0.7000000) *
    3) gpa>=3.415 192  82 0 (0.5729167 0.4270833)
      6) rank>=1.5 160  58 0 (0.6375000 0.3625000)
       12) rank>=2.5 89  27 0 (0.6966292 0.3033708) *
       13) rank< 2.5 71  31 0 (0.5633803 0.4366197)
         26) gpa>=3.495 55  20 0 (0.6363636 0.3636364)
           52) gpa< 3.73 26   5 0 (0.8076923 0.1923077) *
           53) gpa>=3.73 29  14 1 (0.4827586 0.5172414)
            106) Race>=1.5 19   8 0 (0.5789474 0.4210526) *
            107) Race< 1.5 10   3 1 (0.3000000 0.7000000) *
         27) gpa< 3.495 16   5 1 (0.3125000 0.6875000) *
      7) rank< 1.5 32   8 1 (0.2500000 0.7500000) *


rpart(formula = admit ~ ., data = college_data, method = "class")

Variables actually used in tree construction:
[1] gpa  gre  Race rank

Root node error: 127/400 = 0.3175

n= 400

        CP nsplit rel error  xerror     xstd
1 0.062992      0   1.00000 1.00000 0.073308
2 0.023622      2   0.87402 0.87402 0.070514
3 0.015748      4   0.82677 0.92126 0.071639
4 0.010000      8   0.76378 0.98425 0.072994


>
```



It is clear from the decision tree of College Admissions ie the relative error first decreases and then increases when the size of the tree increases.

```
summary(college_model)
Call:
rpart(formula = admit ~ ., data = college_data, method = "class")
  n= 400

          CP nsplit rel error    xerror       xstd
1 0.06299213      0 1.0000000 1.0000000 0.07330768
2 0.02362205      2 0.8740157 0.8740157 0.07051422
3 0.01574803      4 0.8267717 0.9212598 0.07163948
4 0.01000000      8 0.7637795 0.9842520 0.07299408

Variable importance
 gpa rank  gre Race  ses
  41   34   19    5    1

Node number 1: 400 observations,    complexity param=0.06299213
  predicted class=0  expected loss=0.3175  P(node) =1
    class counts:   273   127
   probabilities: 0.682 0.318
  left son=2 (208 obs) right son=3 (192 obs)
  Primary splits:
      gpa  < 3.415 to the left,  improve=8.8678210, (0 missing)
      rank < 2.5   to the right, improve=7.7819370, (0 missing)
      gre  < 510   to the left,  improve=4.8439760, (0 missing)
      Race < 1.5   to the right, improve=1.2564990, (0 missing)
      ses  < 1.5   to the right, improve=0.3782926, (0 missing)
  Surrogate splits:
```

```
      gre  < 610   to the left,  agree=0.645, adj=0.260, (0 split)
      Race < 2.5   to the left,  agree=0.545, adj=0.052, (0 split)
      rank < 1.5   to the right, agree=0.528, adj=0.016, (0 split)

Node number 2: 208 observations,    complexity param=0.01574803
  predicted class=0  expected loss=0.2163462  P(node) =0.52
    class counts:   163    45
   probabilities: 0.784 0.216
  left son=4 (99 obs) right son=5 (109 obs)
  Primary splits:
      rank < 2.5   to the right, improve=2.731978, (0 missing)
      gre  < 750   to the left,  improve=2.515925, (0 missing)
      ses  < 2.5   to the right, improve=1.524154, (0 missing)
      Race < 1.5   to the right, improve=1.390400, (0 missing)
      gpa  < 3.235 to the right, improve=1.053296, (0 missing)
  Surrogate splits:
      gre         < 530   to the left,  agree=0.582, adj=0.121, (0 split)
      gpa         < 3.225 to the right, agree=0.553, adj=0.061, (0 split)
      ses         < 2.5   to the right, agree=0.548, adj=0.051, (0 split)
      Gender_Male < 0.5   to the left,  agree=0.534, adj=0.020, (0 split)

Node number 3: 192 observations,    complexity param=0.06299213
  predicted class=0  expected loss=0.4270833  P(node) =0.48
    class counts:   110    82
   probabilities: 0.573 0.427
  left son=6 (160 obs) right son=7 (32 obs)
  Primary splits:
      rank        < 1.5   to the right, improve=8.0083330, (0 missing)
      gre         < 450   to the left,  improve=1.1737770, (0 missing)
      ses         < 2.5   to the left,  improve=0.5522727, (0 missing)
      gpa         < 3.945 to the left,  improve=0.5037879, (0 missing)
      Gender_Male < 0.5   to the right, improve=0.4942810, (0 missing)

Node number 4: 99 observations
  predicted class=0  expected loss=0.1313131  P(node) =0.2475
    class counts:    86    13
   probabilities: 0.869 0.131

Node number 5: 109 observations,    complexity param=0.01574803
  predicted class=0  expected loss=0.293578  P(node) =0.2725
    class counts:    77    32
   probabilities: 0.706 0.294
  left son=10 (99 obs) right son=11 (10 obs)
  Primary splits:
      gre  < 730   to the left,  improve=3.63727200, (0 missing)
      Race < 1.5   to the right, improve=1.22822600, (0 missing)
      ses  < 2.5   to the right, improve=1.01945100, (0 missing)
      gpa  < 2.905 to the left,  improve=0.33841060, (0 missing)
      rank < 1.5   to the right, improve=0.02221607, (0 missing)

Node number 6: 160 observations,    complexity param=0.02362205
  predicted class=0  expected loss=0.3625  P(node) =0.4
    class counts:   102    58
   probabilities: 0.637 0.362
  left son=12 (89 obs) right son=13 (71 obs)
  Primary splits:
```

```
      rank        < 2.5   to the right, improve=1.4024450, (0 missing)
      gre         < 650   to the left,  improve=1.1404990, (0 missing)
      gpa         < 3.945 to the left,  improve=0.7032468, (0 missing)
      Gender_Male < 0.5   to the right, improve=0.1831404, (0 missing)
      Race        < 1.5   to the right, improve=0.1587912, (0 missing)
  Surrogate splits:
      gpa < 3.515 to the right, agree=0.594, adj=0.085, (0 split)

Node number 7: 32 observations
  predicted class=1  expected loss=0.25  P(node) =0.08
    class counts:     8    24
   probabilities: 0.250 0.750

Node number 10: 99 observations
  predicted class=0  expected loss=0.2525253  P(node) =0.2475
    class counts:    74    25
   probabilities: 0.747 0.253

Node number 11: 10 observations
  predicted class=1  expected loss=0.3  P(node) =0.025
    class counts:     3     7
   probabilities: 0.300 0.700

Node number 12: 89 observations
  predicted class=0  expected loss=0.3033708  P(node) =0.2225
    class counts:    62    27
   probabilities: 0.697 0.303

Node number 13: 71 observations,    complexity param=0.02362205
  predicted class=0  expected loss=0.4366197  P(node) =0.1775
    class counts:    40    31
   probabilities: 0.563 0.437
  left son=26 (55 obs) right son=27 (16 obs)
  Primary splits:
      gpa         < 3.495 to the right, improve=2.6000320, (0 missing)
      gre         < 500   to the left,  improve=1.7510060, (0 missing)
      Gender_Male < 0.5   to the right, improve=0.8327521, (0 missing)
      ses         < 2.5   to the left,  improve=0.8001448, (0 missing)
      Race        < 2.5   to the left,  improve=0.1452296, (0 missing)

Node number 26: 55 observations,    complexity param=0.01574803
  predicted class=0  expected loss=0.3636364  P(node) =0.1375
    class counts:    35    20
   probabilities: 0.636 0.364
  left son=52 (26 obs) right son=53 (29 obs)
  Primary splits:
      gpa         < 3.73  to the left,  improve=2.8948640, (0 missing)
      Gender_Male < 0.5   to the right, improve=1.2412120, (0 missing)
      ses         < 2.5   to the left,  improve=0.7030835, (0 missing)
      gre         < 690   to the right, improve=0.3878788, (0 missing)
      Race        < 1.5   to the right, improve=0.1515152, (0 missing)
  Surrogate splits:
      gre  < 610   to the left,  agree=0.582, adj=0.115, (0 split)
      Race < 1.5   to the left,  agree=0.564, adj=0.077, (0 split)
      ses  < 2.5   to the right, agree=0.545, adj=0.038, (0 split)
```

```
Node number 27: 16 observations
  predicted class=1  expected loss=0.3125  P(node) =0.04
    class counts:     5    11
   probabilities: 0.312 0.688

Node number 52: 26 observations
  predicted class=0  expected loss=0.1923077  P(node) =0.065
    class counts:    21     5
   probabilities: 0.808 0.192

Node number 53: 29 observations,    complexity param=0.01574803
  predicted class=1  expected loss=0.4827586  P(node) =0.0725
    class counts:    14    15
   probabilities: 0.483 0.517
  left son=106 (19 obs) right son=107 (10 obs)
  Primary splits:
      Race        < 1.5   to the right, improve=1.0196010, (0 missing)
      gre         < 690   to the right, improve=0.8827586, (0 missing)
      ses         < 2.5   to the left,  improve=0.5827586, (0 missing)
      gpa         < 3.945 to the left,  improve=0.5029606, (0 missing)
      Gender_Male < 0.5   to the right, improve=0.4539125, (0 missing)
  Surrogate splits:
      ses < 2.5   to the left,  agree=0.69, adj=0.1, (0 split)

Node number 106: 19 observations
  predicted class=0  expected loss=0.4210526  P(node) =0.0475
    class counts:    11     8
   probabilities: 0.579 0.421

Node number 107: 10 observations
  predicted class=1  expected loss=0.3  P(node) =0.025
    class counts:     3     7
   probabilities: 0.300 0.700
```

It is clear from the summary of the college data set ie gpa and rank are the two most important variables of the dataset for decision making

```
support_vector<-svm(admit~.,mytraining)
> confusionMatrix(mytraining$admit,predict(support_vector),positive='1')
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 183   5
         1  71  21
                                          
               Accuracy : 0.7286          
                 95% CI : (0.6725, 0.7798)
    No Information Rate : 0.9071          
    P-Value [Acc > NIR] : 1               
                                          
                  Kappa : 0.2469          
```

```
 Mcnemar's Test P-Value : 8.918e-14

            Sensitivity : 0.80769
            Specificity : 0.72047
         Pos Pred Value : 0.22826
         Neg Pred Value : 0.97340
             Prevalence : 0.09286
         Detection Rate : 0.07500
   Detection Prevalence : 0.32857
      Balanced Accuracy : 0.76408

       'Positive' Class : 1
```

From the Support Vector Machine Algorithm(SVM),it is clear that important variable for tree construction or in decision making are gpa,gre,Race and rank. The accuracy result of SVM is 73%(approx.).Hence,it is suitable to opt for SVM for decision making

Q10) Determine the accuracy rates for each kind of model

Ans) The various modelling techniques which I am using to analyse the admission process of the student are Boostrap,K-Fold Cross Validation and Repeated K-Fold Cross Validation. Below mention is the code in R for modelling techniques

Boostrap

college_data<-
read.csv("https://raw.githubusercontent.com/shivanipriya89/College/main/College.csv")

print(college_data)

View(college_data)

Convert admit to factor

college_data$admit<-sapply(college_data$admit,factor)

Define Training

train_control<-trainControl(method = 'boot', number = 100)

train the model

model<-train(admit ~., data = college_data, trControl = train_control, method = 'nb')

print(model)

```
# K-Fold Cross Validation
college_data<-
read.csv("https://raw.githubusercontent.com/shivanipriya89/College/main/College.csv")
print(college_data)
View(college_data)
str(college_data)

# Convert admit to factor

college_data$admit<-sapply(college_data$admit,factor)

# Define Training Control
mytraining<-trainControl(method="cv",number=10)

# Fix the Parameters of the Algorithm
grid<-expand.grid(.fL = c(0), .usekernel = c(FALSE),.adjust = 0.5 )

# train the model
model<-train(admit ~., data = college_data, trControl = mytraining, method = 'nb', tuneGrid =
grid)

print(model)

# Repeated K-Fold Cross Validation
college_data<-
read.csv("https://raw.githubusercontent.com/shivanipriya89/College/main/College.csv")
print(college_data)
View(college_data)
college_data$admit<-sapply(college_data$admit,factor)

# Define the Training Control
```

train_control<-trainControl(method="repeatedcv",number=10,repeats = 3)

model<-train(admit~.,data=college_data,trControl=train_control,method='nb')

print(model)

The output of Boostrap,K-Fold and Repeated K-Fold Cross Validation techniques are mention below:

```
> model<-train(admit ~., data = college_data, trControl = train_control, metho
d = 'nb')
> print(model)
Naive Bayes

400 samples
  6 predictor
  2 classes: '0', '1'

No pre-processing
Resampling: Bootstrapped (100 reps)
Summary of sample sizes: 400, 400, 400, 400, 400, 400, ...
Resampling results across tuning parameters:

  usekernel  Accuracy   Kappa
  FALSE      0.6817573  0.17467431
   TRUE      0.6883763  0.07976111

Tuning parameter 'fL' was held constant at a value of 0
Tuning
 parameter 'adjust' was held constant at a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were fL = 0, usekernel = TRUE
 and adjust = 1.
```

The accuracy result of Boostrap model is 69% (approx.)

```
print(model)
Naive Bayes

400 samples
  6 predictor
  2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 359, 360, 360, 360, 360, 360, ...
Resampling results:

  Accuracy  Kappa
  0.690358  0.1804949
```

```
Tuning parameter 'fL' was held constant at a value of 0
Tuning
 parameter 'usekernel' was held constant at a value of FALSE

Tuning parameter 'adjust' was held constant at a value of 0.5


>
```

The accuracy result of K-Fold cross validation is 69%(approx.)

```
print(model)
Naive Bayes

400 samples
  6 predictor
  2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 360, 360, 360, 361, 359, 360, ...
Resampling results across tuning parameters:

  usekernel  Accuracy   Kappa
  FALSE      0.6852319  0.16618787
   TRUE      0.6908417  0.05981299

Tuning parameter 'fL' was held constant at a value of 0
Tuning
 parameter 'adjust' was held constant at a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were fL = 0, usekernel = TRUE

 and adjust = 1.
```

The accuracy result of repeated K-Fold cross validation is 69%(approx). From the above mention data,it is clear that probability that student will not take admission is 68.5% and student will take admission is 69%

Q11) Select the most accurate model

Ans) The most accurate model is Support Vector Machine(SVM). It's accuracy result is 73% (approx.). Below mention is the code and output of SVM

Support Vector Machine

college_data<-
read.csv("https://raw.githubusercontent.com/shivanipriya89/College/main/College.csv")

print(college_data)

View(college_data)

str(college_data)

Convert admit to factor

college_data$admit<-sapply(college_data$admit,factor)

str(college_data)

Spliting the data

sample_split<-floor(0.7*nrow(college_data))

set.seed(1)

training<-sample(seq_len(nrow(college_data)),size=sample_split)

Training and testing Data

mytraining<-college_data[training,]

mytesting<-college_data[-training,]

Support Vector Machine

support_vector<-svm(admit~.,mytraining)

confusionMatrix(mytraining$admit,predict(support_vector),positive='1')

```
confusionMatrix(mytraining$admit,predict(support_vector),positive='1')
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 183   5
         1  71  21

               Accuracy : 0.7286
                 95% CI : (0.6725, 0.7798)
    No Information Rate : 0.9071
    P-Value [Acc > NIR] : 1
```

```
                 Kappa : 0.2469

Mcnemar's Test P-Value : 8.918e-14

           Sensitivity : 0.80769
           Specificity : 0.72047
        Pos Pred Value : 0.22826
        Neg Pred Value : 0.97340
            Prevalence : 0.09286
        Detection Rate : 0.07500
  Detection Prevalence : 0.32857
     Balanced Accuracy : 0.76408

      'Positive' Class : 1


>
```

It is clear from the above attached screenshot of SVM ie the accuracy result of this model is 73%(approx.). Hence,from the above mention models, SVM gives the highest accuracy results

Q12) Identify other Machine learning or statistical techniques

Ans) Naïve Bayes and Leave one out Cross Validation are some of the other Machine Learning and Statistical techniques. Below attached is the code in R

```
# Naive Bays Model

college_data<-read.csv("https://raw.githubusercontent.com/shivanipriya89/College/main/College.csv")

print(college_data)

View(college_data)

str(college_data)

# Use the Naive Bays Algorithim


mycollege<-naiveBayes(admit~.,data=college_data)

mycollege

summary(mycollege)

# Leave one out cross validation

college_data<-read.csv("https://raw.githubusercontent.com/shivanipriya89/College/main/College.csv")

print(college_data)

View(college_data)
```

college_data$admit<-sapply(college_data$admit,factor)

Define Train Control

train_control=trainControl(method='LOOCV')

model<-train(admit~.,data=college_data,trControl=train_control,method="nb")

print(model)

The output of both the models are mention below:

```
> mycollege

Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
     0      1 
0.6825 0.3175 

Conditional probabilities:
   gre
Y       [,1]     [,2]
  0 573.1868 115.8302
  1 618.8976 108.8849

   gpa
Y       [,1]      [,2]
  0 3.343700 0.3771330
  1 3.489213 0.3701771

   ses
Y       [,1]      [,2]
  0 2.018315 0.8064730
  1 1.937008 0.8140437

   Gender_Male
Y         [,1]      [,2]
  0 0.4835165 0.5006460
  1 0.4566929 0.5000937

   Race
Y       [,1]      [,2]
  0 1.996337 0.8112181
  1 1.889764 0.8472959

   rank
Y       [,1]      [,2]
  0 2.641026 0.9171978
  1 2.149606 0.9178887
```

```
>
```

The apriori probabilities of naïve Bayes is mention above

```
print(model)
Naive Bayes

400 samples
  6 predictor
  2 classes: '0', '1'

No pre-processing
Resampling: Leave-One-Out Cross-Validation
Summary of sample sizes: 399, 399, 399, 399, 399, 399, ...
Resampling results across tuning parameters:

  usekernel  Accuracy  Kappa
  FALSE      0.69      0.17730967
   TRUE      0.69      0.05354349

Tuning parameter 'fL' was held constant at a value of 0
Tuning
 parameter 'adjust' was held constant at a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were fL = 0, usekernel = FALSE
 and adjust = 1.


>
```

The output of the Leave-one out cross validation is mention above. The accuracy result of Naïve Bayes is 69% approx.

Q13) Categorize the average of grade point into High, Medium, and Low (with admission probability percentages) and plot it on a point chart.
Cross grid for admission variables with GRE Categorization is shown below:

Ans) The below mention is the code in R which analyses the average point of grade in various categories

college_data<-
read.csv("https://raw.githubusercontent.com/shivanipriya89/College/main/College.csv")

print(college_data)

View(college_data)

str(college_data)

print(college_data$gre)

max(college_data$gre)

min(college_data$gre)

college_analysis<-
transform(college_data,Categorized=ifelse(gre<441,"Low",ifelse(gre<581,"Medium","High")
))

print(college_analysis)

print(college_data$gre,college_data$college_analysis)

summary(college_analysis)

```
print(college_analysis)
    admit gre  gpa ses Gender_Male Race rank Categorized
1       0 380 3.61   1           0    3    3         Low
2       1 660 3.67   2           0    2    3        High
3       1 800 4.00   2           0    2    1        High
4       1 640 3.19   1           1    2    4        High
5       0 520 2.93   3           1    2    4      Medium
6       1 760 3.00   2           1    1    2        High
7       1 560 2.98   2           1    2    1      Medium
8       0 400 3.08   2           0    2    2         Low
9       1 540 3.39   1           1    1    3      Medium
10      0 700 3.92   1           0    2    2        High
11      0 800 4.00   1           1    1    4        High
12      0 440 3.22   3           0    2    1         Low
13      1 760 4.00   3           1    2    1        High
14      0 700 3.08   2           0    2    2        High
15      1 700 4.00   2           1    1    1        High
16      0 480 3.44   3           0    1    3      Medium
17      0 780 3.87   2           0    3    4        High
18      0 360 2.56   3           1    3    3         Low
19      0 800 3.75   1           1    3    2        High
20      1 540 3.81   1           0    3    1      Medium
21      0 500 3.17   3           0    2    3      Medium
22      1 660 3.63   1           0    1    2        High
23      0 600 2.82   1           0    3    4        High
24      0 680 3.19   1           0    1    4        High
25      1 760 3.35   2           0    2    2        High
26      1 800 3.66   2           1    1    1        High
27      1 620 3.61   2           0    1    1        High
28      1 520 3.74   2           0    3    4      Medium
29      1 780 3.22   1           0    1    2        High
30      0 520 3.29   1           0    1    1      Medium
31      0 540 3.78   1           1    1    4      Medium
32      0 760 3.35   2           1    1    3        High
33      0 600 3.40   3           0    1    3        High
34      1 800 4.00   3           0    1    3        High
35      0 360 3.14   1           1    2    1         Low
36      0 400 3.05   3           0    2    2         Low
```

37	0	580	3.25	1	0	2	1	Medium
38	0	520	2.90	2	0	2	3	Medium
39	1	500	3.13	2	0	2	2	Medium
40	1	520	2.68	2	0	1	3	Medium
41	0	560	2.42	1	1	3	2	Medium
42	1	580	3.32	1	0	1	2	Medium
43	1	600	3.15	2	1	1	2	High
44	0	500	3.31	2	0	2	3	Medium
45	0	700	2.94	1	0	3	2	High
46	1	460	3.45	2	1	3	3	Medium
47	1	580	3.46	3	1	1	2	Medium
48	0	500	2.97	3	0	2	4	Medium
49	0	440	2.48	3	0	3	4	Low
50	0	400	3.35	3	0	1	3	Low
51	0	640	3.86	2	1	3	3	High
52	0	440	3.13	2	0	2	4	Low
53	0	740	3.37	2	1	3	4	High
54	1	680	3.27	2	0	2	2	High
55	0	660	3.34	1	0	1	3	High
56	1	740	4.00	1	1	2	3	High
57	0	560	3.19	3	1	1	3	Medium
58	0	380	2.94	3	0	2	3	Low
59	0	400	3.65	3	1	2	2	Low
60	0	600	2.82	3	1	1	4	High
61	1	620	3.18	2	1	1	2	High
62	0	560	3.32	1	0	3	4	Medium
63	0	640	3.67	1	1	2	3	High
64	1	680	3.85	1	1	3	3	High
65	0	580	4.00	2	1	3	3	Medium
66	0	600	3.59	1	0	1	2	High
67	0	740	3.62	3	1	2	4	High
68	0	620	3.30	2	1	3	1	High
69	0	580	3.69	3	0	3	1	Medium
70	0	800	3.73	1	1	1	1	High
71	0	640	4.00	1	1	1	3	High
72	0	300	2.92	1	1	1	4	Low
73	0	480	3.39	2	0	2	4	Medium
74	0	580	4.00	3	0	3	2	Medium
75	0	720	3.45	2	1	2	4	High
76	0	720	4.00	2	0	3	3	High
77	0	560	3.36	1	1	2	3	Medium
78	1	800	4.00	3	0	3	3	High
79	0	540	3.12	3	1	2	1	Medium
80	1	620	4.00	2	0	2	1	High
81	0	700	2.90	2	0	2	4	High
82	0	620	3.07	3	1	2	2	High
83	0	500	2.71	2	0	3	2	Medium
84	0	380	2.91	3	1	2	4	Low
85	1	500	3.60	1	1	1	3	Medium
86	0	520	2.98	2	0	2	2	Medium
87	0	600	3.32	1	0	3	2	High
88	0	600	3.48	1	0	1	2	High
89	0	700	3.28	3	0	3	1	High
90	1	660	4.00	1	1	1	2	High
91	0	700	3.83	2	0	2	2	High
92	1	720	3.64	2	0	2	1	High

```
93       0 800 3.90   3           1    1    2        High
94       0 580 2.93   3           1    1    2      Medium
95       1 660 3.44   2           0    3    2        High
96       0 660 3.33   2           1    3    2        High
97       0 640 3.52   2           1    3    4        High
98       0 480 3.57   3           1    2    2      Medium
99       0 700 2.88   2           1    3    2        High
100      0 400 3.31   3           1    2    3         Low
101      0 340 3.15   2           0    1    3         Low
102      0 580 3.57   1           1    2    3      Medium
103      0 380 3.33   3           0    3    4         Low
104      0 540 3.94   3           0    1    3      Medium
105      1 660 3.95   2           1    1    2        High
106      1 740 2.97   1           1    1    2        High
107      1 700 3.56   1           1    2    1        High
108      0 480 3.13   2           0    1    2      Medium
109      0 400 2.93   1           1    3    3         Low
110      0 480 3.45   3           0    1    2      Medium
111      0 680 3.08   3           0    3    4        High
112      0 420 3.41   2           1    3    4         Low
113      0 360 3.00   1           0    1    3         Low
114      0 600 3.22   3           1    2    1        High
115      0 720 3.84   1           1    2    3        High
116      0 620 3.99   2           1    2    3        High
117      1 440 3.45   1           1    3    2         Low
118      0 700 3.72   2           1    2    2        High
119      1 800 3.70   1           0    2    1        High
120      0 340 2.92   3           1    2    3         Low
121      1 520 3.74   2           0    2    2      Medium
122      1 480 2.67   1           0    1    2      Medium
123      0 520 2.85   3           0    1    3      Medium
124      0 500 2.98   3           0    2    3      Medium
125      0 720 3.88   2           0    3    3        High
 [ reached 'max' / getOption("max.print") -- omitted 275 rows ]

>
```

The above mention screenshot has a new column Categorized with values Low,Medium and High in it wrt GRE

Click on this

https://github.com/shivanipriya89/College/blob/main/College.csv

link for viewing the tabular format of College Admission Data

This https://raw.githubusercontent.com/shivanipriya89/College/main/College.csv

Link has the csv file for analysis