Exploratory Data Analysis

```
In [1]: #Identification of variables and data types
```

```
In [2]: import pandas as pd
        import numpy as np
        import seaborn as sns
        import matplotlib.pyplot as plt
```

```
In [3]: data = pd.read_csv('Jamboree_Admission.csv')
        data
```

Out[3]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	1	337	118	4	4.5	4.5	9.65	1	0.92
1	2	324	107	4	4.0	4.5	8.87	1	0.76
2	3	316	104	3	3.0	3.5	8.00	1	0.72
3	4	322	110	3	3.5	2.5	8.67	1	0.80
4	5	314	103	2	2.0	3.0	8.21	0	0.65
...	...	...	...	...	...	...	...	...	...
495	496	332	108	5	4.5	4.0	9.02	1	0.87
496	497	337	117	5	5.0	5.0	9.87	1	0.96
497	498	330	120	5	4.5	5.0	9.56	1	0.93
498	499	312	103	4	4.0	5.0	8.43	0	0.73
499	500	327	113	4	4.5	4.5	9.04	0	0.84

500 rows × 9 columns

```
In [4]: data.shape
Out[4]: (500, 9)
```

```
In [5]: data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Serial No.         500 non-null    int64  
 1   GRE Score          500 non-null    int64  
 2   TOEFL Score        500 non-null    int64  
 3   University Rating  500 non-null    int64  
 4   SOP                500 non-null    float64
 5   LOR                500 non-null    float64
 6   CGPA               500 non-null    float64
 7   Research           500 non-null    int64  
 8   Chance of Admit    500 non-null    float64
dtypes: float64(4), int64(5)
memory usage: 35.3 KB
```

```
In [6]: #Analysing the basic metrics
```

```
In [7]: data.describe(include="all")
```

Out[7]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
count	500.000000	500.000000	500.000000	500.000000	500.000000	500.00000	500.000000	500.000000	500.00000
mean	250.500000	316.472000	107.192000	3.114000	3.374000	3.48400	8.576440	0.560000	0.72174
std	144.481833	11.295148	6.081868	1.143512	0.991004	0.92545	0.604813	0.496884	0.14114
min	1.000000	290.000000	92.000000	1.000000	1.000000	1.00000	6.800000	0.000000	0.34000
25%	125.750000	308.000000	103.000000	2.000000	2.500000	3.00000	8.127500	0.000000	0.63000
50%	250.500000	317.000000	107.000000	3.000000	3.500000	3.50000	8.560000	1.000000	0.72000
75%	375.250000	325.000000	112.000000	4.000000	4.000000	4.00000	9.040000	1.000000	0.82000
max	500.000000	340.000000	120.000000	5.000000	5.000000	5.00000	9.920000	1.000000	0.97000

```
In [8]: data.describe(include=[np.number])
```

Out[8]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
count	500.000000	500.000000	500.000000	500.000000	500.000000	500.00000	500.000000	500.000000	500.00000
mean	250.500000	316.472000	107.192000	3.114000	3.374000	3.48400	8.576440	0.560000	0.72174
std	144.481833	11.295148	6.081868	1.143512	0.991004	0.92545	0.604813	0.496884	0.14114
min	1.000000	290.000000	92.000000	1.000000	1.000000	1.00000	6.800000	0.000000	0.34000
25%	125.750000	308.000000	103.000000	2.000000	2.500000	3.00000	8.127500	0.000000	0.63000
50%	250.500000	317.000000	107.000000	3.000000	3.500000	3.50000	8.560000	1.000000	0.72000
75%	375.250000	325.000000	112.000000	4.000000	4.000000	4.00000	9.040000	1.000000	0.82000
max	500.000000	340.000000	120.000000	5.000000	5.000000	5.00000	9.920000	1.000000	0.97000

```
In [9]: data.describe(include=[np.float64])
```

Out[9]:

	SOP	LOR	CGPA	Chance of Admit
count	500.000000	500.00000	500.000000	500.00000
mean	3.374000	3.48400	8.576440	0.72174
std	0.991004	0.92545	0.604813	0.14114
min	1.000000	1.00000	6.800000	0.34000
25%	2.500000	3.00000	8.127500	0.63000
50%	3.500000	3.50000	8.560000	0.72000
75%	4.000000	4.00000	9.040000	0.82000
max	5.000000	5.00000	9.920000	0.97000

```
In [10]: data.isnull().sum()/len(data)*100
Out[10]: Serial No.           0.0
         GRE Score            0.0
         TOEFL Score          0.0
         University Rating    0.0
         SOP                  0.0
         LOR                  0.0
         CGPA                 0.0
         Research             0.0
         Chance of Admit      0.0
         dtype: float64
```

```
In [11]: #Non-Graphical Analysis
```

```
In [12]: data['Research'].value_counts()/len(data)*100
Out[12]: 1    56.0
         0    44.0
         Name: Research, dtype: float64
```

```
In [13]: groupby_research = data.groupby(['Research'])
```

```
In [14]: groupby_research.max()
```

Out[14]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Chance of Admit
Research								
0	500	339	120	5	5.0	5.0	9.70	0.89
1	498	340	120	5	5.0	5.0	9.92	0.97

```
In [15]:  data.groupby('GRE Score').agg({'Chance of Admit ':['count', 'mean', 'max', 'min']})
```

Out[15]:

	Chance of Admit			
	count	mean	max	min
GRE Score				
290	2	0.460000	0.47	0.45
293	1	0.640000	0.64	0.64
294	2	0.475000	0.49	0.46
295	5	0.512000	0.69	0.37
296	5	0.522000	0.61	0.44
297	6	0.498333	0.59	0.34
298	10	0.507000	0.69	0.34
299	10	0.537000	0.68	0.38
300	12	0.595833	0.71	0.36
301	11	0.624545	0.68	0.44
302	7	0.558571	0.65	0.46
303	5	0.590000	0.77	0.36
304	12	0.570833	0.71	0.38
305	11	0.624545	0.71	0.53
306	7	0.642857	0.73	0.48
307	10	0.627000	0.79	0.47
308	13	0.655385	0.77	0.46
309	9	0.637778	0.76	0.48
310	11	0.667273	0.76	0.54
311	16	0.665000	0.79	0.42
312	24	0.685417	0.81	0.50
313	12	0.684167	0.79	0.53
314	16	0.696250	0.84	0.54
315	13	0.645385	0.79	0.39
316	18	0.661667	0.77	0.49
317	15	0.690000	0.84	0.49
318	12	0.702500	0.78	0.63
319	12	0.729167	0.82	0.65
320	16	0.790000	0.92	0.64
321	17	0.805294	0.93	0.69
322	17	0.784706	0.92	0.61
323	13	0.785385	0.92	0.45
324	23	0.813913	0.92	0.70
325	15	0.742667	0.91	0.52
326	12	0.822500	0.91	0.74
327	17	0.801176	0.93	0.61
328	9	0.848889	0.94	0.78
329	10	0.853000	0.90	0.72
330	8	0.906250	0.93	0.86
331	9	0.918889	0.94	0.86
332	8	0.893750	0.94	0.79
333	4	0.930000	0.96	0.89
334	8	0.916250	0.97	0.78
335	4	0.940000	0.96	0.93
336	5	0.948000	0.97	0.92
337	2	0.940000	0.96	0.92
338	4	0.920000	0.95	0.91
339	3	0.936667	0.96	0.89
340	9	0.947778	0.97	0.90

```
In [16]:  data.groupby('TOEFL Score').agg({'Chance of Admit ':['count', 'mean', 'max', 'min']})
```

Out[16]:

	Chance of Admit			
	count	mean	max	min
TOEFL Score				
92	1	0.510000	0.51	0.51
93	2	0.460000	0.46	0.46
94	2	0.490000	0.56	0.42
95	3	0.516667	0.62	0.44
96	6	0.476667	0.54	0.34
97	7	0.502857	0.65	0.38
98	10	0.567000	0.76	0.34
99	23	0.566957	0.76	0.36
100	24	0.597083	0.78	0.42
101	20	0.610500	0.79	0.38
102	24	0.651667	0.80	0.50
103	25	0.678000	0.76	0.52
104	29	0.678966	0.82	0.42
105	37	0.648108	0.79	0.39
106	28	0.676429	0.81	0.52
107	28	0.708214	0.84	0.53
108	19	0.724211	0.87	0.45
109	19	0.752632	0.85	0.62
110	44	0.767045	0.92	0.46
111	20	0.804000	0.93	0.52
112	28	0.800000	0.96	0.55
113	19	0.860526	0.96	0.76
114	18	0.840556	0.96	0.61
115	11	0.879091	0.95	0.70
116	16	0.901875	0.96	0.78
117	8	0.927500	0.96	0.87
118	10	0.925000	0.94	0.90
119	10	0.930000	0.97	0.89
120	9	0.934444	0.97	0.86

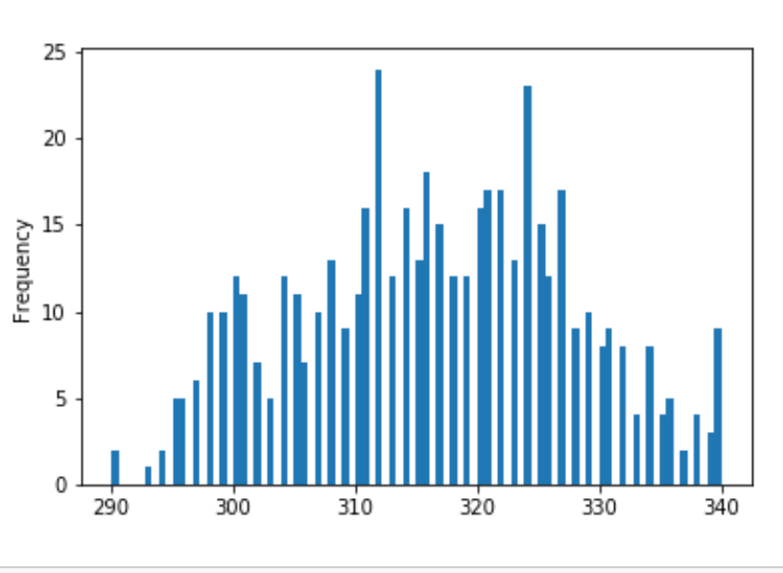
```
In [17]:  data.groupby('Research').agg({'Chance of Admit ':['count', 'mean', 'max', 'min']})
```

Out[17]:

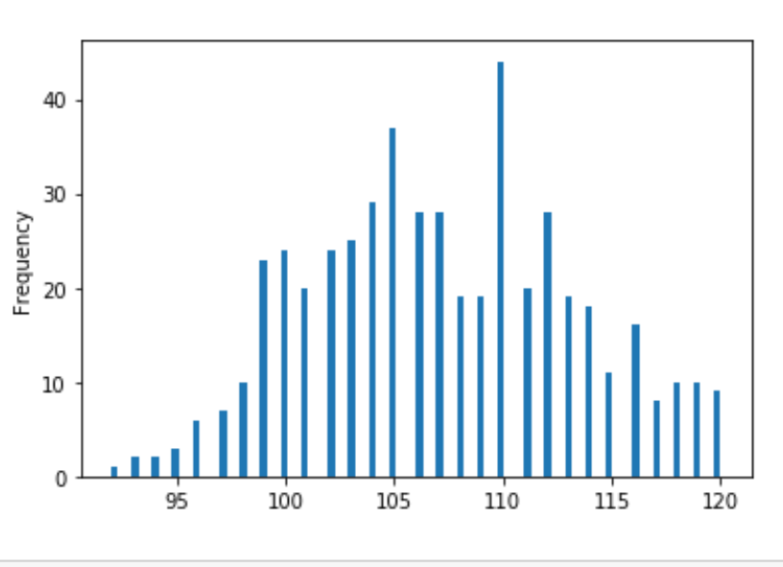
	Chance of Admit			
	count	mean	max	min
Research				
0	220	0.634909	0.89	0.34
1	280	0.789964	0.97	0.36

```
In [18]: #Visual Analysis
```

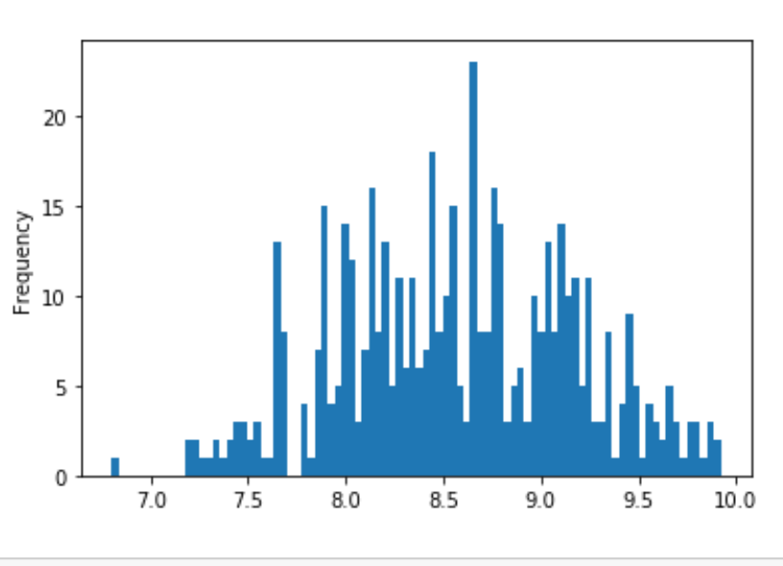
```
In [19]: data['GRE Score'].plot(kind='hist', bins=90)
Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x7f1de1771a58>
```



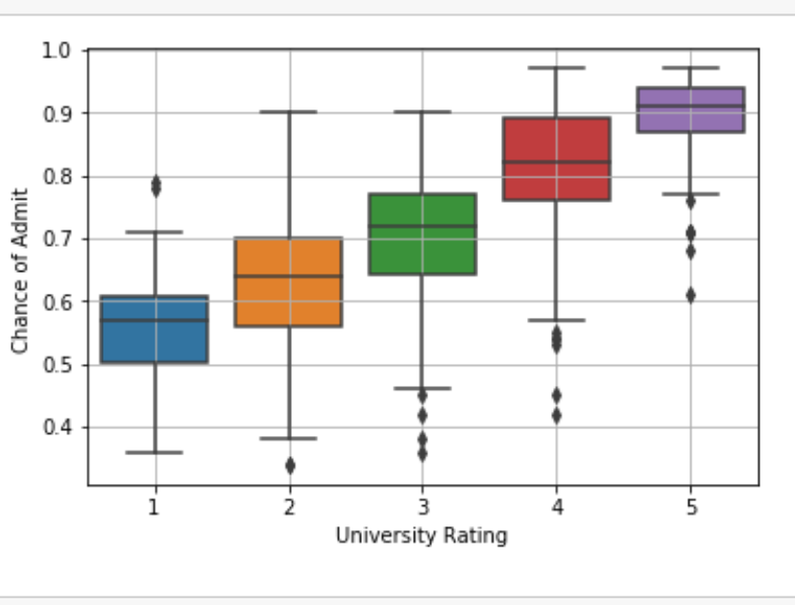
```
In [20]: data['TOEFL Score'].plot(kind='hist', bins=90)
Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x7f1ddef4ec50>
```



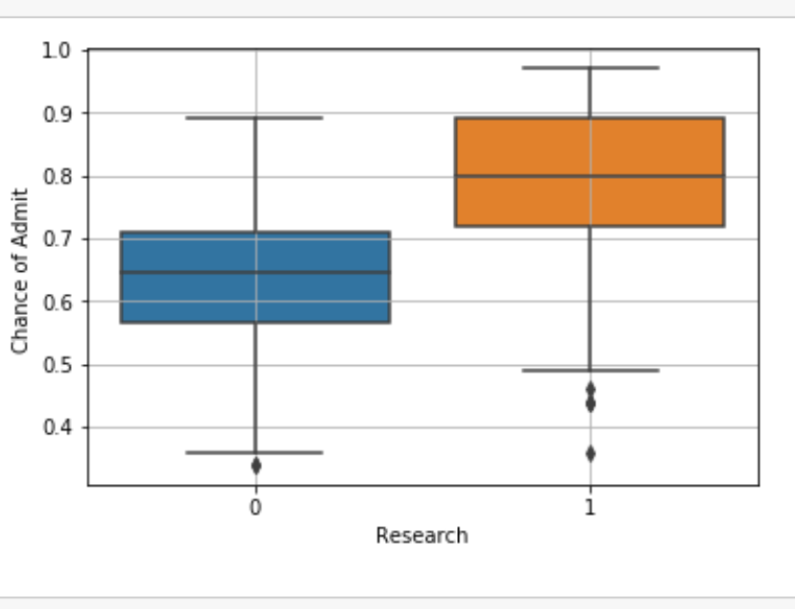
```
In [21]: data['CGPA'].plot(kind='hist', bins=90)
Out[21]: <matplotlib.axes._subplots.AxesSubplot at 0x7f1ddef4e080>
```



```
In [22]: sns.boxplot(x='University Rating', y='Chance of Admit ', data=data)
         plt.grid()
         plt.show()
```

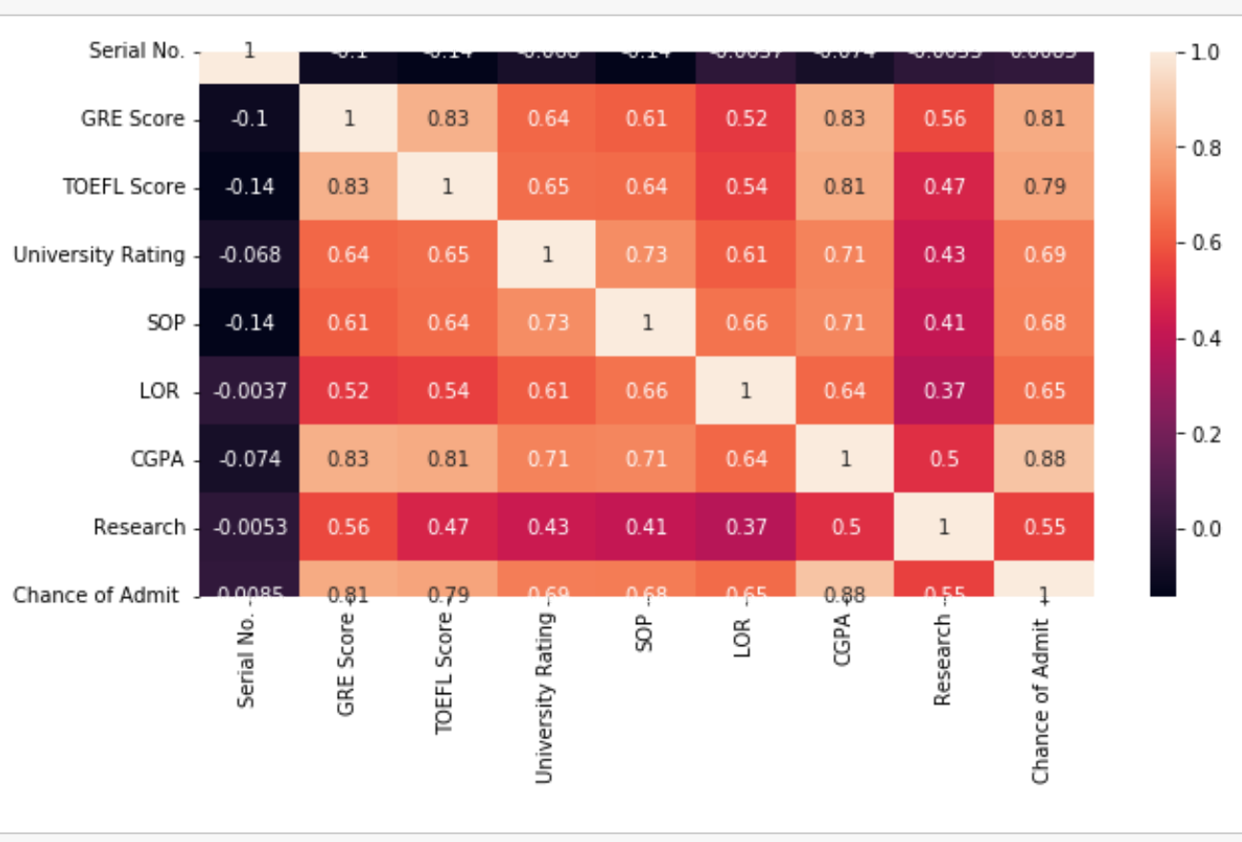


```
In [23]: sns.boxplot(x='Research', y='Chance of Admit ', data=data)
         plt.grid()
         plt.show()
```

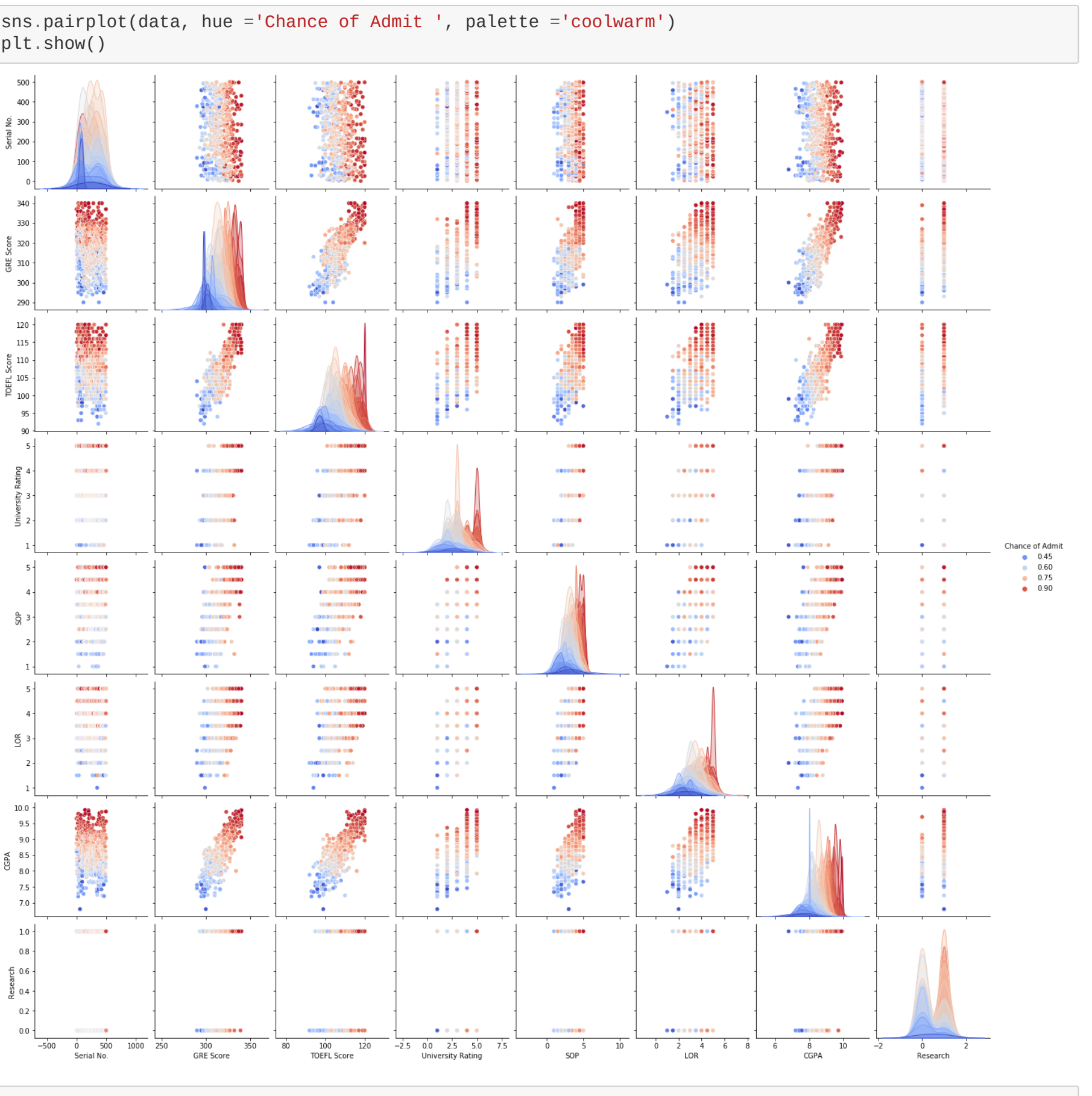


```
In [24]: #Bivariate Analysis
```

```
In [25]: sns.heatmap(data.corr(),annot=True)
         plt.gcf().set_size_inches(10, 5)
         plt.show()
```



```
In [26]: sns.pairplot(data, hue ='Chance of Admit ', palette ='coolwarm')
         plt.show()
```



```
In [27]: #Correlation and Covariance
```

```
In [28]: #For sample
         Covariance_gre= np.cov(data['GRE Score'], data['Chance of Admit '])[0][1]
         Covariance_gre
Out[28]: 1.29186244488978
```

```
In [29]: Covariance_toefl= np.cov(data['TOEFL Score'], data['Chance of Admit '])[0][1]
         Covariance_toefl
Out[29]: 0.6800460120240481
```

```
In [30]: Covariance_ur= np.cov(data['University Rating'], data['Chance of Admit '])[0][1]
         Covariance_ur
Out[30]: 0.11138440881763538
```

```
In [31]: Covariance_sop= np.cov(data['SOP'], data['Chance of Admit '])[0][1]
         Covariance_sop
Out[31]: 0.09569062124248494
```

```
In [32]: Covariance_lor= np.cov(data['LOR '], data['Chance of Admit '])[0][1]
         Covariance_lor
Out[32]: 0.08429643286573148
```

```
In [33]: Covariance_cgpa= np.cov(data['CGPA'], data['Chance of Admit '])[0][1]
         Covariance_cgpa
Out[33]: 0.07532584609218435
```

```
In [34]: Covariance_r= np.cov(data['Research'], data['Chance of Admit '])[0][1]
         Covariance_r
Out[34]: 0.038282164328657324
```

```
In [35]: #For population
         np.cov(data['GRE Score'], data['Chance of Admit '], bias=True)[0][1]
Out[35]: 1.2892787200000004
```

```
In [36]: np.cov(data['TOEFL Score'], data['Chance of Admit '], bias=True)[0][1]
Out[36]: 0.6786859200000002
```

```
In [37]: np.cov(data['University Rating'], data['Chance of Admit '], bias=True)[0][1]
Out[37]: 0.11116164000000012
```

```
In [38]: np.cov(data['SOP'], data['Chance of Admit '], bias=True)[0][1]
Out[38]: 0.09549923999999997
```

```
In [39]: np.cov(data['LOR '], data['Chance of Admit '], bias=True)[0][1]
Out[39]: 0.08412784000000002
```

```
In [40]: np.cov(data['CGPA'], data['Chance of Admit '], bias=True)[0][1]
Out[40]: 0.07517519439999998
```

```
In [41]: np.cov(data['Research'], data['Chance of Admit '], bias=True)[0][1]
Out[41]: 0.03820560000000001
```

```
In [42]: #Correlation Coefficient
         np.corrcoef(data['GRE Score'], data['Chance of Admit '])[0][1]
Out[42]: 0.8103506354632607
```

```
In [43]: np.corrcoef(data['TOEFL Score'], data['Chance of Admit '])[0][1]
Out[43]: 0.7922276143050841
```

```
In [44]: np.corrcoef(data['University Rating'], data['Chance of Admit '])[0][1]
Out[44]: 0.6901323687886906
```

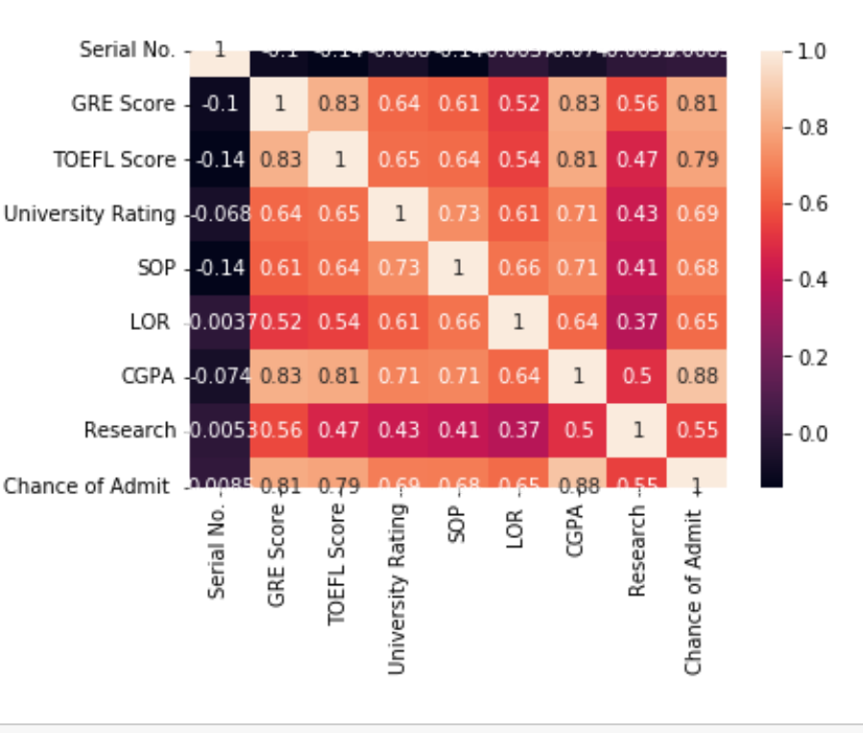
```
In [45]: np.corrcoef(data['SOP'], data['Chance of Admit '])[0][1]
Out[45]: 0.6841365241316723
```

```
In [46]: np.corrcoef(data['LOR '], data['Chance of Admit '])[0][1]
Out[46]: 0.6453645135280114
```

```
In [47]: np.corrcoef(data['CGPA'], data['Chance of Admit '])[0][1]
Out[47]: 0.8824125749045737
```

```
In [48]: np.corrcoef(data['Research'], data['Chance of Admit '])[0][1]
Out[48]: 0.5458710294711387
```

```
In [49]: #Visualization
         sns.heatmap(data.corr(), annot=True)
Out[49]: <matplotlib.axes._subplots.AxesSubplot at 0x7f1dda6dac18>
```



```
In [50]: #Outlier treatment using Interquartile Range(IQR)
```

```
In [51]: q1 = data['Chance of Admit '].quantile(0.25)
         q2 = data['Chance of Admit '].quantile(0.75)
         iqr = q2 - q1
```

```
In [52]: df = data.loc[(data['Chance of Admit ']>q1-1.5*iqr) & (data['Chance of Admit ']<q2+1.5*iqr)]
```

```
In [53]: df
```

Out[53]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	1	337	118	4	4.5	4.5	9.65	1	0.92
1	2	324	107	4	4.0	4.5	8.87	1	0.76
2	3	316	104	3	3.0	3.5	8.00	1	0.72
3	4	322	110	3	3.5	2.5	8.67	1	0.80
4	5	314	103	2	2.0	3.0	8.21	0	0.65
...	...	...	...	...	...	...	...	...	...
495	496	332	108	5	4.5	4.0	9.02	1	0.87
496	497	337	117	5	5.0	5.0	9.87	1	0.96
497	498	330	120	5	4.5	5.0	9.56	1	0.93
498	499	312	103	4	4.0	5.0	8.43	0	0.73
499	500	327	113	4	4.5	4.5	9.04	0	0.84

498 rows × 9 columns

Hypothesis Testing

```
In [54]: #2- Sample T-Test to check if Research has an effect on the Chance of Admit
         #H0: Chance of Admit for researcher and non researcher is equal.
         #H1:Chance of Admit for researcher is more that non researcher
```

```
In [55]: from scipy.stats import ttest_ind
```

```
In [56]: df.groupby('Research')['Chance of Admit '].describe()
```

Out[56]:

	count	mean	std	min	25%	50%	75%	max
Research								
0	218.0	0.637615	0.108775	0.36	0.57	0.65	0.7100	0.89
1	280.0	0.789964	0.123208	0.36	0.72	0.80	0.8925	0.97

```
In [57]: researcher = df[df['Research']==1]['Chance of Admit '].sample(218)
```

```
In [58]: not_researcher = df[df['Research']==0]['Chance of Admit '].sample(218)
```

```
In [59]: def researcher_chanceOfAdmit(researcher, not_researcher, alpha):
             stat, p = ttest_ind(researcher, not_researcher,  equal_var=False)
             print("Statistics=%.3f, p=%.3f" % (stat, p))
             if p > alpha:
                 print("fail to reject H0")
             else:
                 print("reject H0")
```

```
In [60]: researcher_chanceOfAdmit(researcher, not_researcher, 0.05)

         Statistics=13.812, p=0.000
         reject H0
```

```
In [61]: #ANNOVA to check if Chance of Admit is similar or different in different 1. SOP 2. LOR
```

```
In [62]: from scipy.stats import levene
         from scipy.stats import shapiro
```

```
In [63]: df.groupby('SOP')['Chance of Admit '].describe()
```

Out[63]:

	count	mean	std	min	25%	50%	75%	max
SOP								
1.0	6.0	0.538333	0.062102	0.42	0.530	0.565	0.5775	0.58
1.5	25.0	0.546400	0.084749	0.42	0.490	0.540	0.5800	0.79
2.0	43.0	0.589535	0.118622	0.36	0.505	0.620	0.6600	0.81
2.5	63.0	0.650159	0.081051	0.43	0.595	0.670	0.7100	0.80
3.0	80.0	0.678500	0.106309	0.36	0.635	0.700	0.7500	0.91
3.5	88.0	0.712045	0.092104	0.45	0.650	0.730	0.7800	0.91
4.0	88.0	0.787841	0.114836	0.46	0.720	0.800	0.8700	0.97
4.5	63.0	0.850000	0.087584	0.57	0.790	0.870	0.9150	0.97
5.0	42.0	0.885000	0.106181	0.38	0.880	0.925	0.9400	0.96

```
In [64]: sop1=df[df['SOP']==2.0]['Chance of Admit '].sample(42)
         sop2=df[df['SOP']==2.5]['Chance of Admit '].sample(42)
         sop3=df[df['SOP']==3.0]['Chance of Admit '].sample(42)
         sop4=df[df['SOP']==3.5]['Chance of Admit '].sample(42)
         sop5=df[df['SOP']==4.0]['Chance of Admit '].sample(42)
         sop6=df[df['SOP']==4.5]['Chance of Admit '].sample(42)
         sop7=df[df['SOP']==5.0]['Chance of Admit '].sample(42)
```

```
In [65]: test_stats, p_val= levene (sop1, sop2, sop3, sop4, sop5, sop6, sop7)
         print(test_stats)
         print(p_val)

         2.7793009396081887
         0.01212856705765533
```

```
In [66]: test_stats, p_val= shapiro (df['Chance of Admit '].sample(42))
         print(test_stats)
         print(p_val)

         0.9613025188446045
         0.16439814865589142
```

```
In [67]: df.groupby('LOR ')['Chance of Admit '].describe()
```

Out[67]:

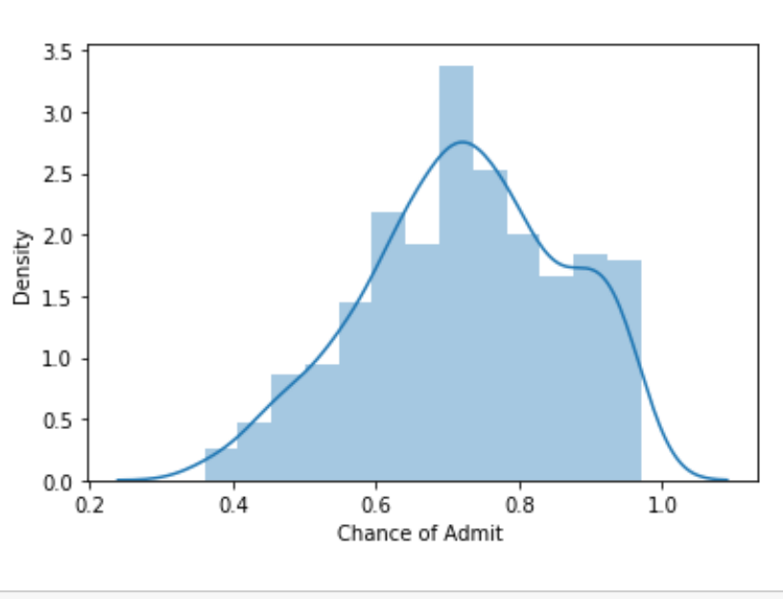
	count	mean	std	min	25%	50%	75%	max
LOR								
1.0	1.0	0.420000	NaN	0.42	0.4200	0.42	0.4200	0.42
1.5	11.0	0.550000	0.122556	0.37	0.4950	0.52	0.5900	0.81
2.0	45.0	0.573333	0.097817	0.36	0.4900	0.57	0.6400	0.79
2.5	50.0	0.640600	0.116293	0.36	0.5825	0.65	0.7275	0.83
3.0	98.0	0.671837	0.106600	0.42	0.6200	0.66	0.7400	0.90
3.5	86.0	0.723023	0.121511	0.38	0.6625	0.72	0.7975	0.97
4.0	94.0	0.764149	0.103533	0.52	0.6900	0.77	0.8400	0.97
4.5	63.0	0.831905	0.104076	0.53	0.7650	0.87	0.9100	0.97
5.0	50.0	0.872600	0.083025	0.65	0.8225	0.91	0.9400	0.97

```
In [68]: lor1=df[df['LOR ']==2.0]['Chance of Admit '].sample(45)
         lor2=df[df['LOR ']==2.5]['Chance of Admit '].sample(45)
         lor3=df[df['LOR ']==3.0]['Chance of Admit '].sample(45)
         lor4=df[df['LOR ']==3.5]['Chance of Admit '].sample(45)
         lor5=df[df['LOR ']==4.0]['Chance of Admit '].sample(45)
         lor6=df[df['LOR ']==4.5]['Chance of Admit '].sample(45)
         lor7=df[df['LOR ']==5.0]['Chance of Admit '].sample(45)
```

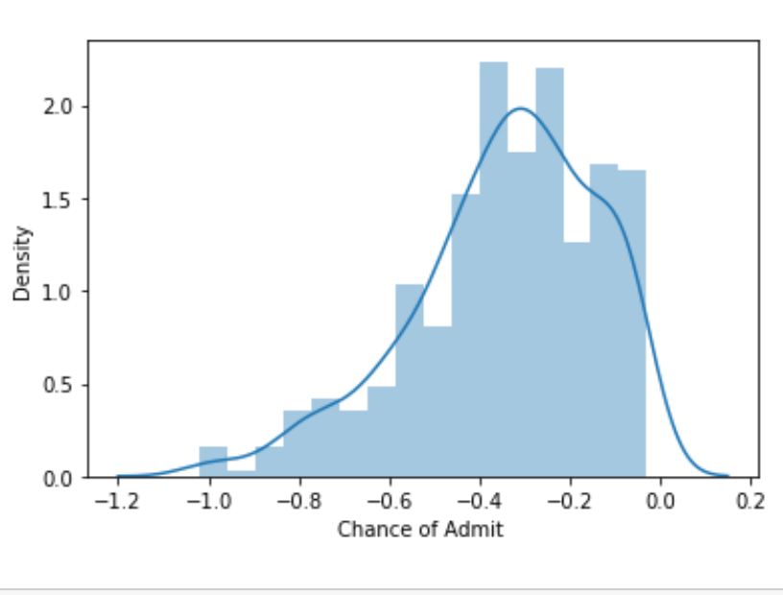
```
In [69]: test_stats, p_val= levene (lor1, lor2, lor3, lor4, lor5, lor6, lor7)
         print(test_stats)
         print(p_val)

         0.7722156364311455
         0.5922568033226403
```

```
In [70]: test_stats, p_val= shapiro (df['Chance of Admit '].sample(300))
         print(test_stats)
         print(p_val)

         0.9738160371780396
         2.7315752959111705e-05
```

```
In [71]: sns.distplot(df['Chance of Admit '])

         /home/mohan/anaconda3/lib/python3.7/site-packages/seaborn/distributions.py:2619: FutureWarning: `dist
         plot` is a deprecated function and will be removed in a future version. Please adapt your code to use
         either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level func
         tion for histograms).
           warnings.warn(msg, FutureWarning)
Out[71]: <matplotlib.axes._subplots.AxesSubplot at 0x7f1dd84bc518>
```



```
In [72]: sns.distplot(np.log(df['Chance of Admit ']))

         /home/mohan/anaconda3/lib/python3.7/site-packages/seaborn/distributions.py:2619: FutureWarning: `dist
         plot` is a deprecated function and will be removed in a future version. Please adapt your code to use
         either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level func
         tion for histograms).
           warnings.warn(msg, FutureWarning)
Out[72]: <matplotlib.axes._subplots.AxesSubplot at 0x7f1dd83f8e80>
```



```
In [73]: #Chi-square test to check if CGPA is dependent on the Chance of Admit
```

```
In [74]: gre = df['GRE Score']
         tofel = df['TOEFL Score']
         chance_of_admit = df['Chance of Admit ']
         gre_coa = [[gre], [chance_of_admit]]
         tofel_coa = [[tofel], [chance_of_admit]]
```

In [75]:
```
from scipy.stats import chi2_contingency

stat, p, dof, expected = chi2_contingency(gre_coa)

# interpret p-value
alpha = 0.05
print("p value is " + str(p))
if p <= alpha:
    print('Dependent (reject H0)')
else:
    print('Independent (H0 holds true)')
```
```
p value is 1.0
Independent (H0 holds true)
```

In [76]:
```
from scipy.stats import chi2_contingency

stat, p, dof, expected = chi2_contingency(tofel_coa)

# interpret p-value
alpha = 0.05
print("p value is " + str(p))
if p <= alpha:
    print('Dependent (reject H0)')
else:
    print('Independent (H0 holds true)')
```
```
p value is 1.0
Independent (H0 holds true)
```

Linear Regression

In [77]:
```
X = df.drop(['Chance of Admit '], axis=1)
Y = df['Chance of Admit ']
```

In [78]:
```
from sklearn.model_selection import train_test_split
```

In [79]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2)
```

In [80]:
```
from sklearn import linear_model
from sklearn.metrics import mean_squared_error, r2_score
```

In [81]:
```
model = linear_model.LinearRegression()
model.fit(X_train, Y_train)
```

Out[81]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

In [82]:
```
Y_pred_train = model.predict(X_train)
```

In [83]:
```
print('Coefficients:', model.coef_)
print('Intercept:', model.intercept_)
print('Mean squared error (MSE): %.2f'% mean_squared_error(Y_train, Y_pred_train))
print('Coefficient of determination (R^2): %.2f' % r2_score(Y_train, Y_pred_train))
```
```
Coefficients: [9.46356118e-05 1.78528194e-03 2.89878249e-03 2.97587888e-03
 7.86984587e-03 1.31683648e-02 1.20375548e-01 2.44657402e-02]
Intercept: -1.3047007473693473
Mean squared error (MSE): 0.00
Coefficient of determination (R^2): 0.84
```

In [84]:
```
Y_pred_test = model.predict(X_test)
```

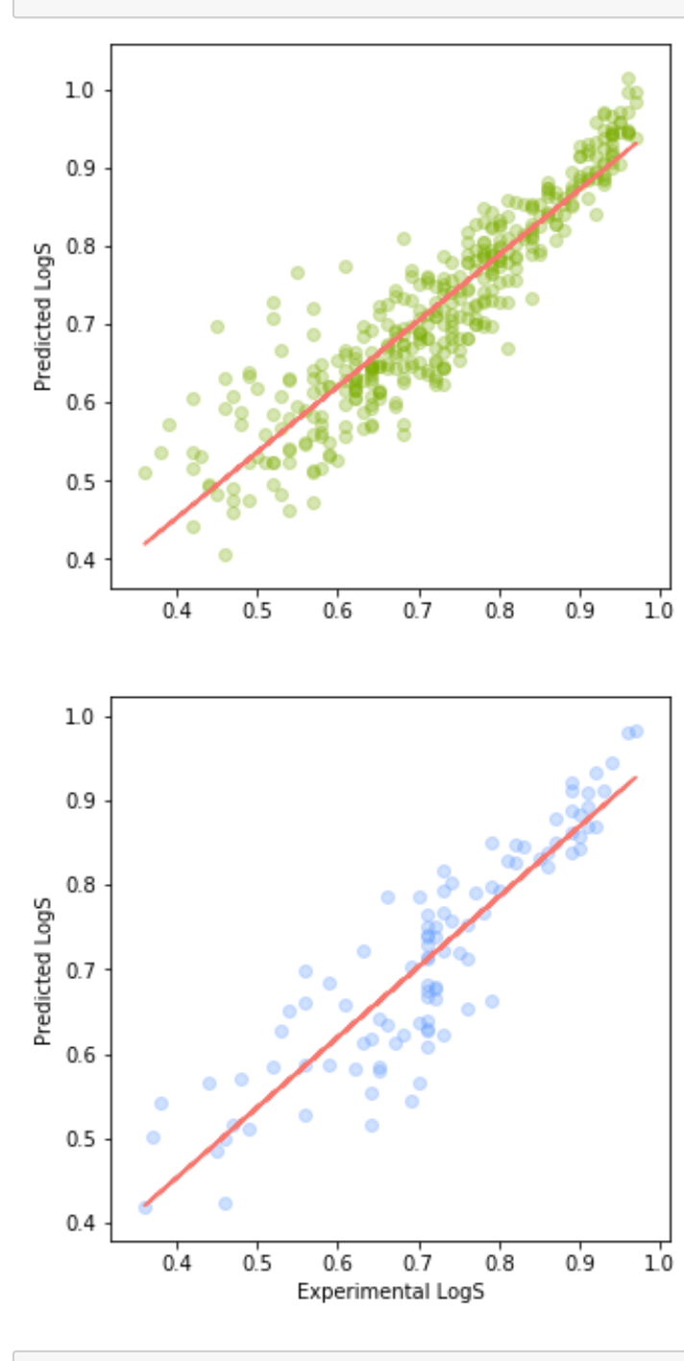
In [85]:
```
print('Coefficients:', model.coef_)
print('Intercept:', model.intercept_)
print('Mean squared error (MSE): %.2f'
      % mean_squared_error(Y_test, Y_pred_test))
print('Coefficient of determination (R^2): %.2f'
      % r2_score(Y_test, Y_pred_test))
```
```
Coefficients: [9.46356118e-05 1.78528194e-03 2.89878249e-03 2.97587888e-03
 7.86984587e-03 1.31683648e-02 1.20375548e-01 2.44657402e-02]
Intercept: -1.3047007473693473
Mean squared error (MSE): 0.00
Coefficient of determination (R^2): 0.80
```

In [86]:
```
yintercept = '%.2f' % model.intercept_
LogP = '%.2f LogP' % model.coef_[0]
MW = '%.4f MW' % model.coef_[1]
RB = '%.4f RB' % model.coef_[2]
AP = '%.2f AP' % model.coef_[3]
```

In [87]:
```
print('LogS = ' +' ' +yintercept + ' ' +LogP + ' ' + MW +' + ' +RB + ' ' + AP)
```
```
LogS =  -1.30 0.00 LogP 0.0018 MW + 0.0029 RB 0.00 AP
```

In [88]:
```
#Vertical plot
```

In [89]:
```
plt.figure(figsize=(5,11))

plt.subplot(2, 1, 1)
plt.scatter(x=Y_train, y=Y_pred_train, c="#7CAE00", alpha=0.3)

z = np.polyfit(Y_train, Y_pred_train, 1)
p = np.poly1d(z)
plt.plot(Y_test,p(Y_test),"#F8766D")

plt.ylabel('Predicted LogS')


# 2 row, 1 column, plot 2
plt.subplot(2, 1, 2)
plt.scatter(x=Y_test, y=Y_pred_test, c="#619CFF", alpha=0.3)

z = np.polyfit(Y_test, Y_pred_test, 1)
p = np.poly1d(z)
plt.plot(Y_test,p(Y_test),"#F8766D")

plt.ylabel('Predicted LogS')
plt.xlabel('Experimental LogS')

plt.savefig('plot_vertical_logS.png')
plt.savefig('plot_vertical_logS.pdf')
plt.show()
```



In [90]:
```
#Horizontal plot
```

In [91]:
```
plt.figure(figsize=(11,5))

plt.subplot(1, 2, 1)
plt.scatter(x=Y_train, y=Y_pred_train, c="#7CAE00", alpha=0.3)

z = np.polyfit(Y_train, Y_pred_train, 1)
p = np.poly1d(z)
plt.plot(Y_test,p(Y_test),"#F8766D")

plt.ylabel('Predicted LogS')
plt.xlabel('Experimental LogS')

# 1 row, 2 column, plot 2
plt.subplot(1, 2, 2)
plt.scatter(x=Y_test, y=Y_pred_test, c="#619CFF", alpha=0.3)

z = np.polyfit(Y_test, Y_pred_test, 1)
p = np.poly1d(z)
plt.plot(Y_test,p(Y_test),"#F8766D")

plt.xlabel('Experimental LogS')

plt.savefig('plot_horizontal_logS.png')
plt.savefig('plot_horizontal_logS.pdf')
plt.show()
```