Reza Mahroo EE7600: Machine Learning Project 2

Linear Regression, Naive Bayes, and Logistic Regression

The performance of different regression models for the data set “load diabetes” from Scikit learn is compared in this project, and Naive Bayes and Logistic regression to the weather data set is applied.

Goal: To compare the performance of different models for the data set “load diabetes“ from Scikit learn.

1. Examine the data set. Print out a few rows from the dataset.
   - Number of Instances: 442
   - Number of features: 10
   - Target Names: ['target'], It shows the disease progression
   - Feature Names: ['age', 'sex', 'bmi', 'bp', 's1', 's2', 's3', 's4', 's5', 's6']
   - Missing Feature Values: None
   - Targets: integer 25 - 346

Visualize some of the data:

First ten instances of the dataset:

```
In [7]: pdata.head(10)
```

Out[7]:

	age	sex	bmi	bp	s1	s2	s3	s4	s5	s6	target
0	0.038076	0.050680	0.061696	0.021872	-0.044223	-0.034821	-0.043401	-0.002592	0.019908	-0.017646	151.0
1	-0.001882	-0.044642	-0.051474	-0.026328	-0.008449	-0.019163	0.074412	-0.039493	-0.068330	-0.092204	75.0
2	0.085299	0.050680	0.044451	-0.005671	-0.045599	-0.034194	-0.032356	-0.002592	0.002864	-0.025930	141.0
3	-0.089063	-0.044642	-0.011595	-0.036656	0.012191	0.024991	-0.036038	0.034309	0.022692	-0.009362	206.0
4	0.005383	-0.044642	-0.036385	0.021872	0.003935	0.015596	0.008142	-0.002592	-0.031991	-0.046641	135.0
5	-0.092695	-0.044642	-0.040696	-0.019442	-0.068991	-0.079288	0.041277	-0.076395	-0.041180	-0.096346	97.0
6	-0.045472	0.050680	-0.047163	-0.015999	-0.040096	-0.024800	0.000779	-0.039493	-0.062913	-0.038357	138.0
7	0.063504	0.050680	-0.001895	0.066630	0.090620	0.108914	0.022869	0.017703	-0.035817	0.003064	63.0
8	0.041708	0.050680	0.061696	-0.040099	-0.013953	0.006202	-0.028674	-0.002592	-0.014956	0.011349	110.0
9	-0.070900	-0.044642	0.039062	-0.033214	-0.012577	-0.034508	-0.024993	-0.002592	0.067736	-0.013504	310.0

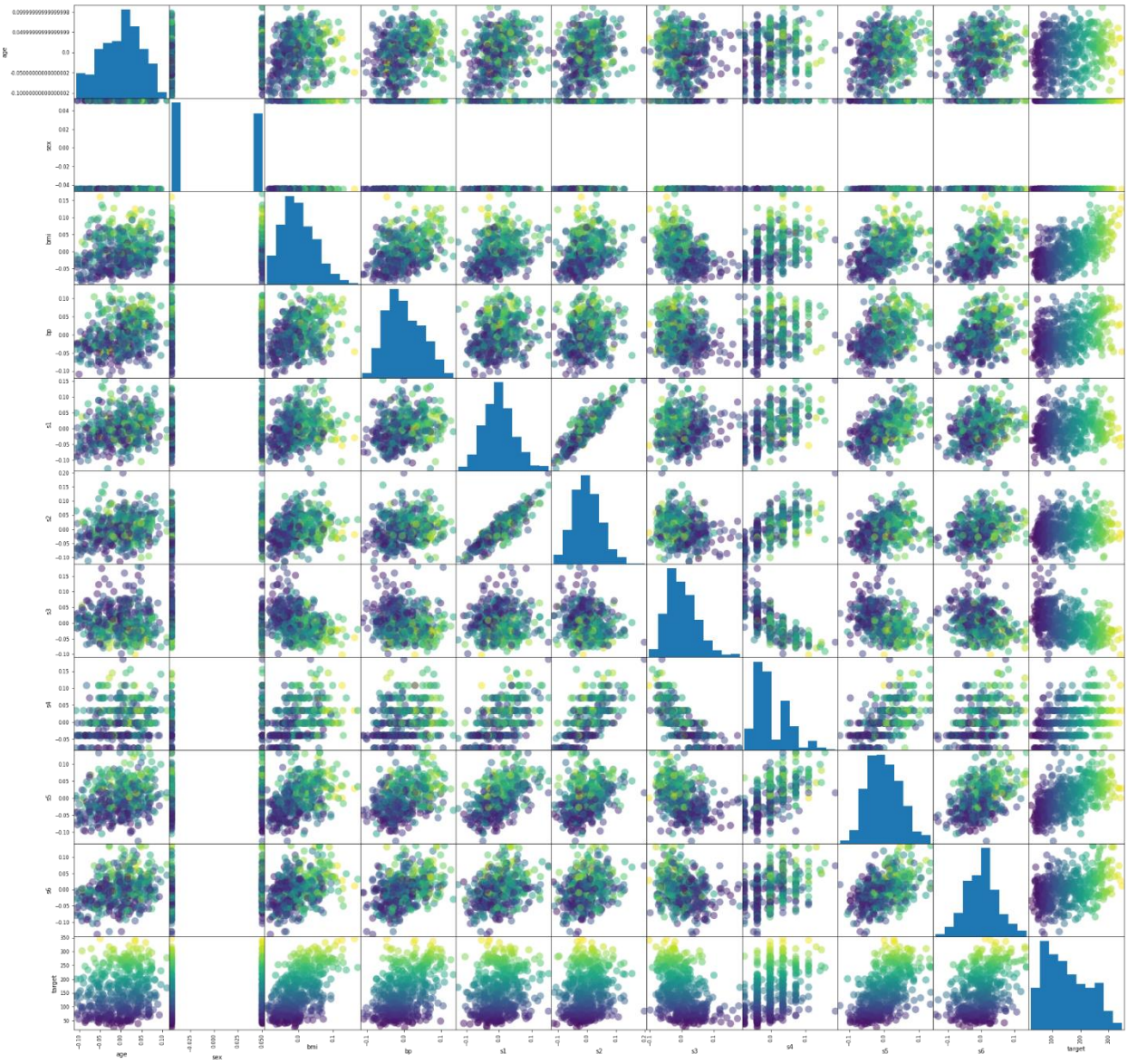
The statistical description of the dataset:

```
In [8]: pdata.describe()
```

Out[8]:

	age	sex	bmi	bp	s1	s2	s3	s4	s5	s6	targe
count	4.420000e+02	4.420000e+02	4.420000e+02	4.420000e+02	4.420000e+02	4.420000e+02	4.420000e+02	4.420000e+02	4.420000e+02	4.420000e+02	442.00000
mean	-3.639623e-16	1.309912e-16	-8.013951e-16	1.289818e-16	-9.042540e-17	1.301121e-16	-4.563971e-16	3.863174e-16	-3.848103e-16	-3.398488e-16	152.13348
std	4.761905e-02	4.761905e-02	4.761905e-02	4.761905e-02	4.761905e-02	4.761905e-02	4.761905e-02	4.761905e-02	4.761905e-02	4.761905e-02	77.09300
min	-1.072256e-01	-4.464164e-02	-9.027530e-02	-1.123996e-01	-1.267807e-01	-1.156131e-01	-1.023071e-01	-7.639450e-02	-1.260974e-01	-1.377672e-01	25.00000
25%	-3.729927e-02	-4.464164e-02	-3.422907e-02	-3.665645e-02	-3.424784e-02	-3.035840e-02	-3.511716e-02	-3.949338e-02	-3.324879e-02	-3.317903e-02	87.00000
50%	5.383060e-03	-4.464164e-02	-7.283766e-03	-5.670611e-03	-4.320866e-03	-3.819065e-03	-6.584468e-03	-2.592262e-03	-1.947634e-03	-1.077698e-03	140.50000
75%	3.807591e-02	5.068012e-02	3.124802e-02	3.564384e-02	2.835801e-02	2.984439e-02	2.931150e-02	3.430886e-02	3.243323e-02	2.791705e-02	211.50000
max	1.107267e-01	5.068012e-02	1.705552e-01	1.320442e-01	1.539137e-01	1.987880e-01	1.811791e-01	1.852344e-01	1.335990e-01	1.356118e-01	346.00000

The overview image of the correlation between each pair of the features is printed out here.



To scrutinize the correlation between each pair of features, we can draw the above pictures on a large scale or calculate the Pearson correlation coefficient. The correlation coefficient (Pearson) between each set of features is calculated as follows.

```
In [9]: pdata.corr(method='pearson')
```

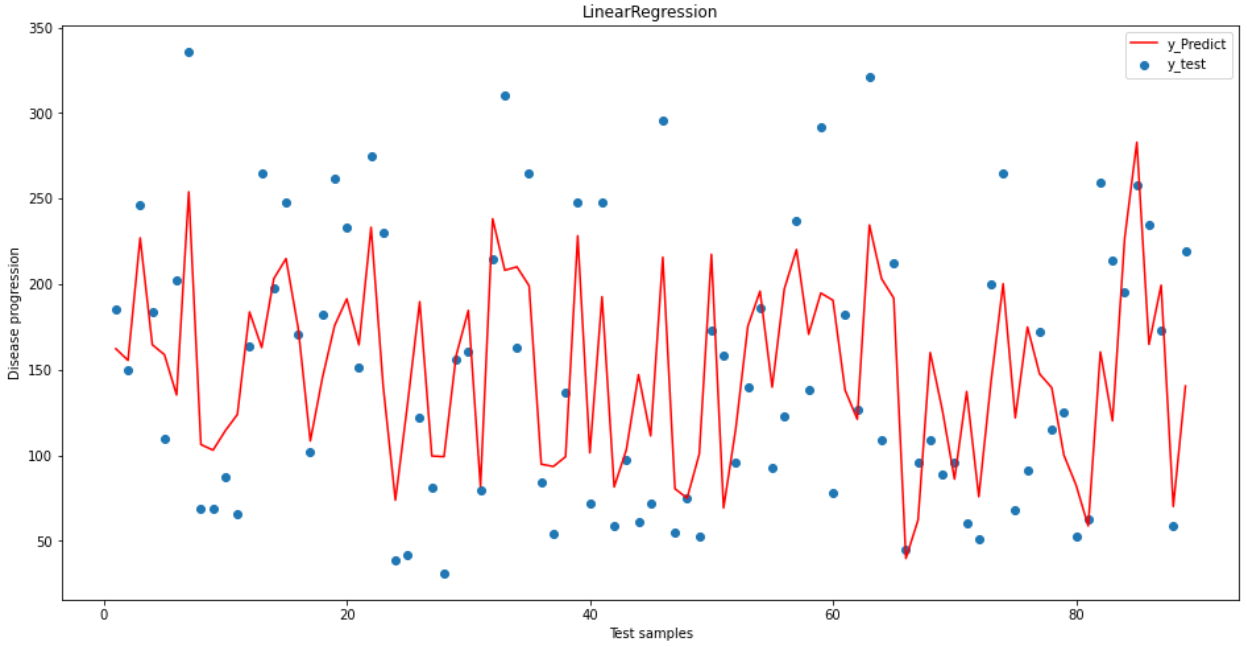
Out[9]:

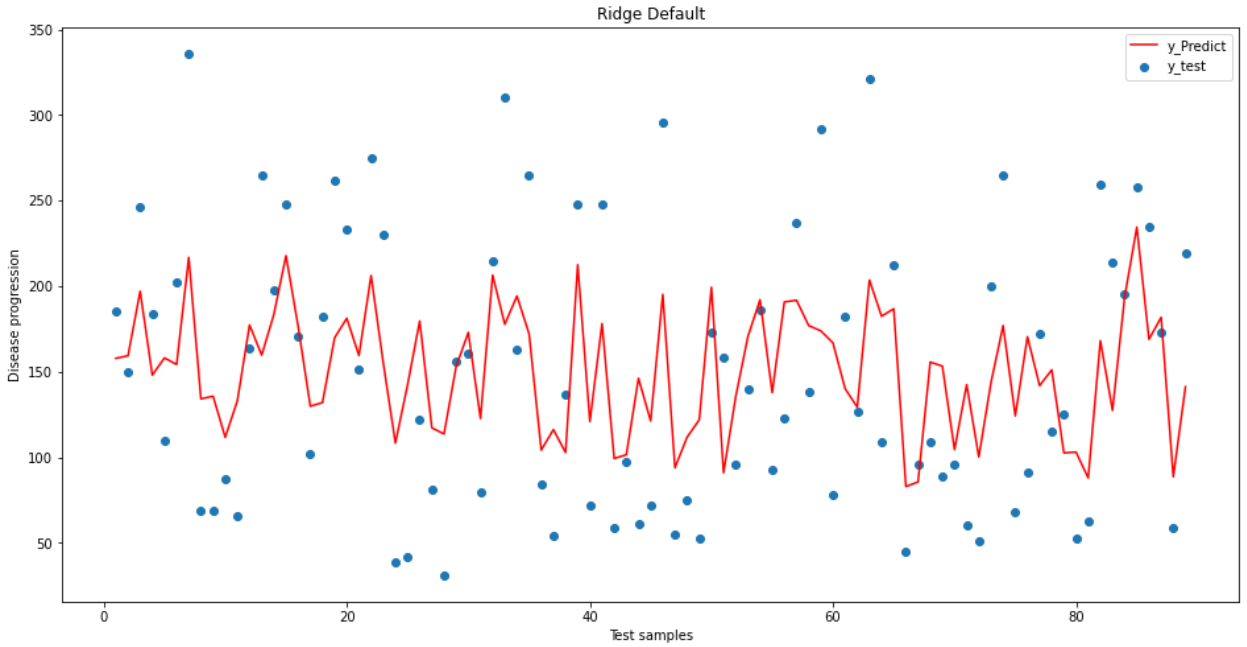
	age	sex	bmi	bp	s1	s2	s3	s4	s5	s6	target
age	1.000000	0.173737	0.185085	0.335427	0.260061	0.219243	-0.075181	0.203841	0.270777	0.301731	0.187889
sex	0.173737	1.000000	0.088161	0.241013	0.035277	0.142637	-0.379090	0.332115	0.149918	0.208133	0.043062
bmi	0.185085	0.088161	1.000000	0.395415	0.249777	0.261170	-0.366811	0.413807	0.446159	0.388680	0.586450
bp	0.335427	0.241013	0.395415	1.000000	0.242470	0.185558	-0.178761	0.257653	0.393478	0.390429	0.441484
s1	0.260061	0.035277	0.249777	0.242470	1.000000	0.896663	0.051519	0.542207	0.515501	0.325717	0.212022
s2	0.219243	0.142637	0.261170	0.185558	0.896663	1.000000	-0.196455	0.659817	0.318353	0.290600	0.174054
s3	-0.075181	-0.379090	-0.366811	-0.178761	0.051519	-0.196455	1.000000	-0.738493	-0.398577	-0.273697	-0.394789
s4	0.203841	0.332115	0.413807	0.257653	0.542207	0.659817	-0.738493	1.000000	0.617857	0.417212	0.430453
s5	0.270777	0.149918	0.446159	0.393478	0.515501	0.318353	-0.398577	0.617857	1.000000	0.464670	0.565883
s6	0.301731	0.208133	0.388680	0.390429	0.325717	0.290600	-0.273697	0.417212	0.464670	1.000000	0.382483
target	0.187889	0.043062	0.586450	0.441484	0.212022	0.174054	-0.394789	0.430453	0.565883	0.382483	1.000000

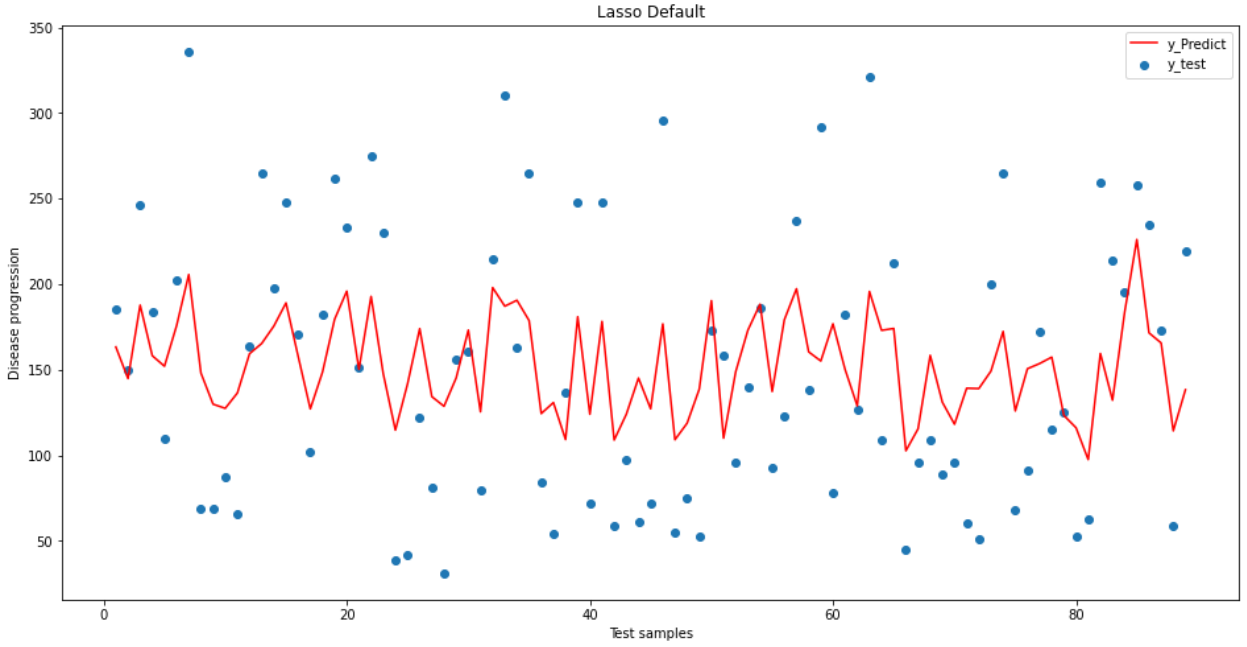
2. Apply linear regression, Ridge regression and Lasso models to this dataset. Obtain the MSE of the models for training and test sets.

Since the size of the dataset is small, I considered 80 percent of it as the training set and 20 percent as the testing set. In this section, the hyper-parameter α is considered a default value of 1.

Model	Linear Regression	Ridge Regression ($\alpha = 1$)	Lasso Regression ($\alpha = 1$)
Train MSE	2908.98	3273.6	3751.9
Test MSE	2724.24	3430.07	3804.02







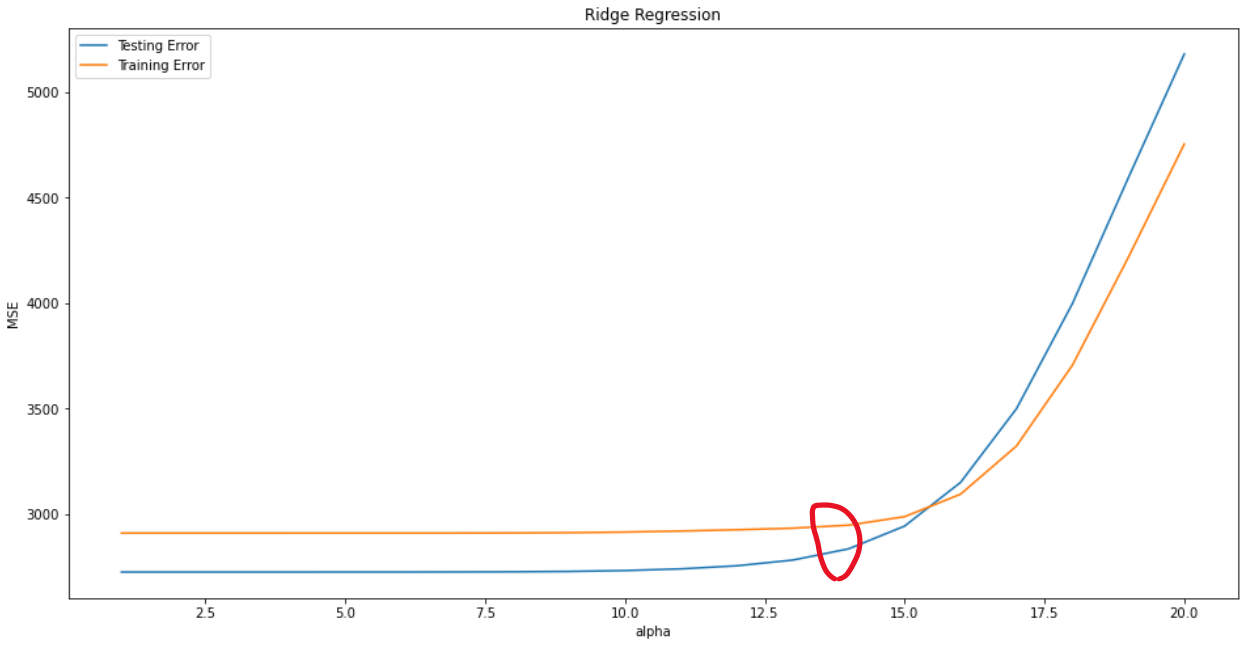
The above graphs show how the shrinkage of the predicted line from complex linear regression to Lasso regression.

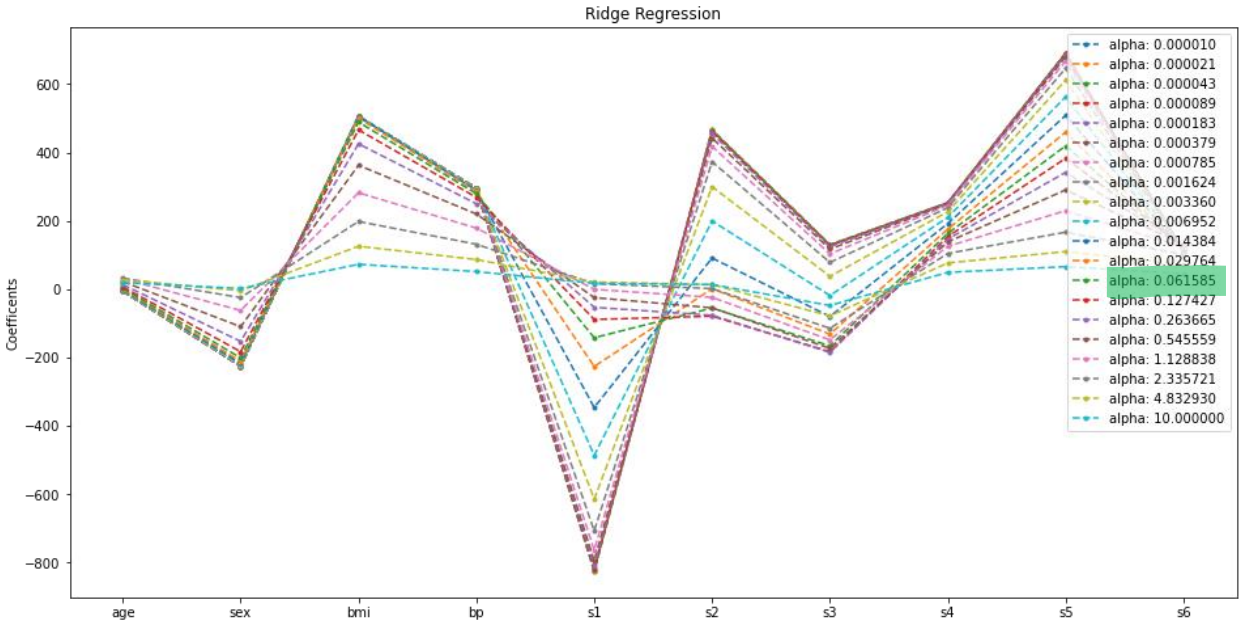
3. For ridge and lasso models plot the error vs. the hyper-parameter α of regularization and find the best of value of α.

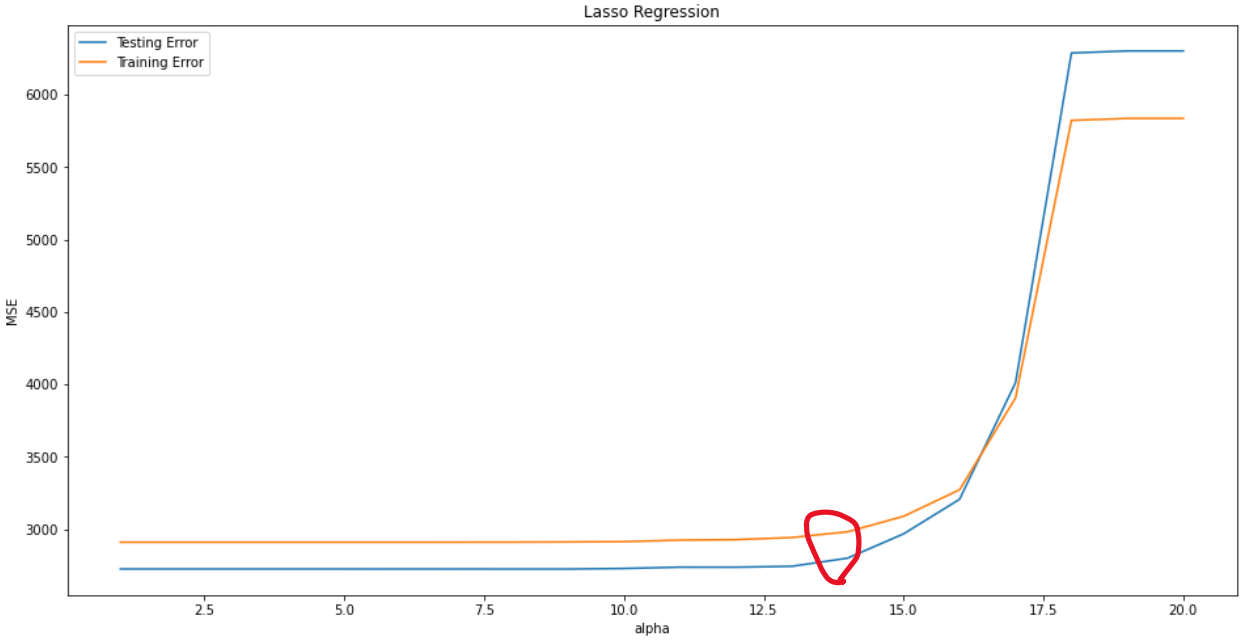
For both the Ridge and Lasso regressions, the MSE vs. α is plotted for 20 numbers of αs spaced evenly on a log scale from -5 to 1. Then for every value of α, feature coefficients are plotted.

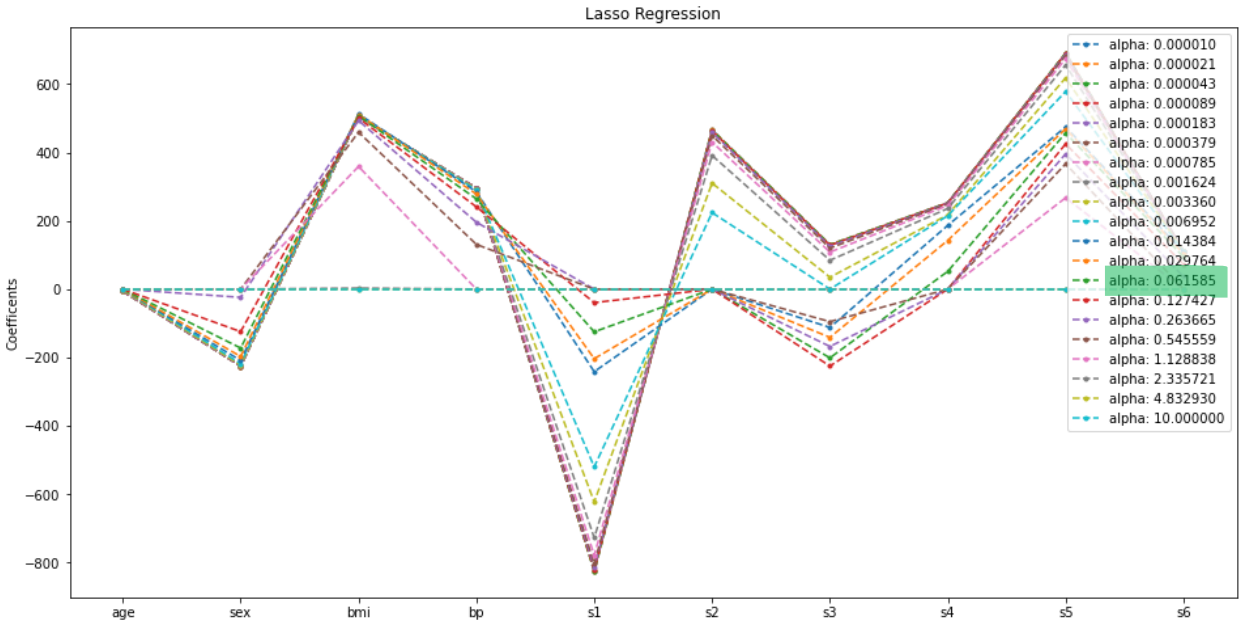
The best α for both ridge and lasso is that α which gives a reasonable error compared to the typical linear regressor (complex regression model) and simplifies the model as much as possible. In this case, it would happen right before the knee area of the plotted graph MSE vs. α. Here the order of α is 13 among these 20 numbers we have considered, and its value is 0.0615. Therefore the best value of α is something around 0.0615 for both cases.

The second graph plotted for each case shows the value of coefficients related to every feature for different values of hyperparameter α. It reveals how the magnitude of the coefficients reduces as the hyperparameter α increases.




Ridge Regression
Coefficents
600
400
200
0
−200
−400
−600
−800
age
sex
bmi
bp
s1
s2
s3
s4
s5
s6
alpha: 0.000010
alpha: 0.000021
alpha: 0.000043
alpha: 0.000089
alpha: 0.000183
alpha: 0.000379
alpha: 0.000785
alpha: 0.001624
alpha: 0.003360
alpha: 0.006952
alpha: 0.014384
alpha: 0.029764
alpha: 0.061585
alpha: 0.127427
alpha: 0.263665
alpha: 0.545559
alpha: 1.128838
alpha: 2.335721
alpha: 4.832930
alpha: 10.000000

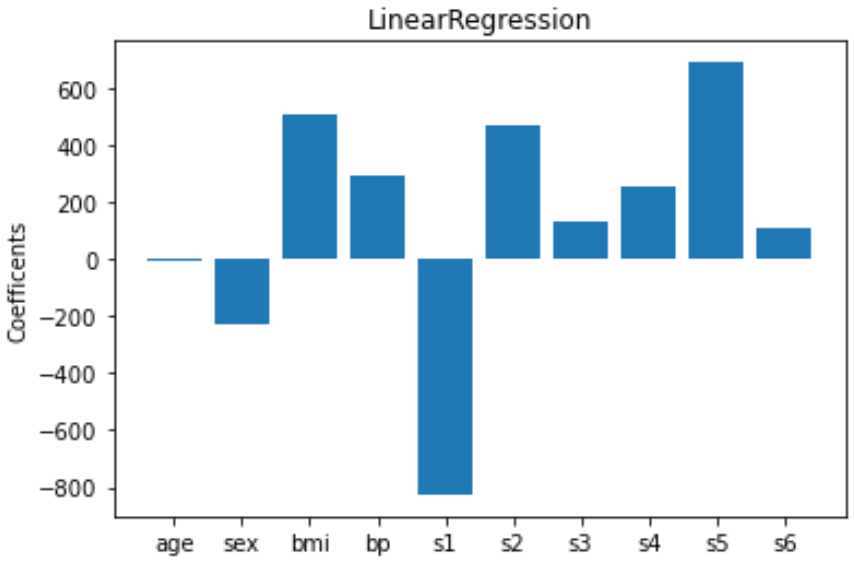

Lasso Regression
Testing Error
Training Error
MSE
6000
5500
5000
4500
4000
3500
3000
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
alpha

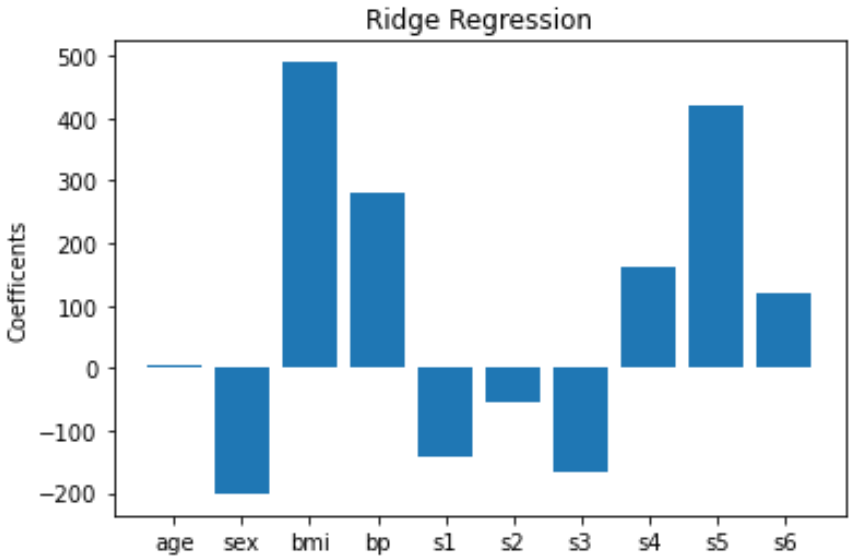


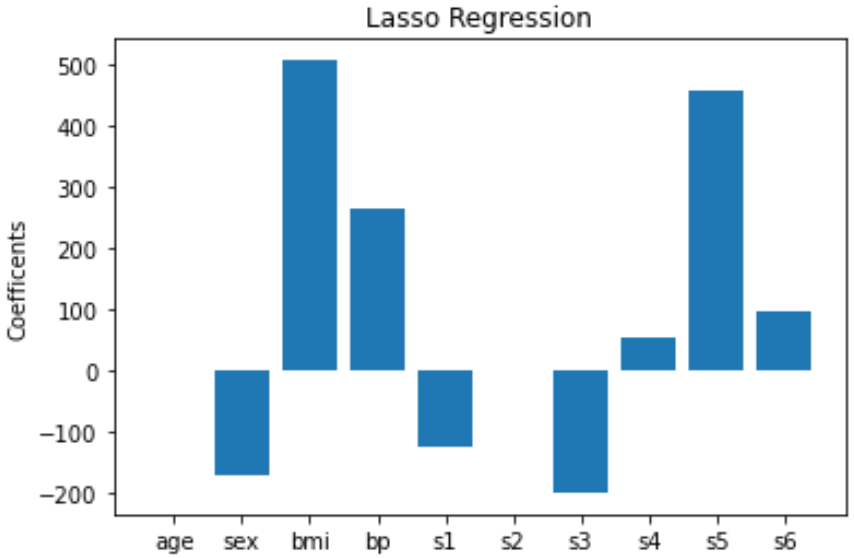
4. Plot the regression coefficients for the three models in bar graphs and rank the features in terms of their significance in prediction.

The significance of a feature depends on the size of the coefficients and the scale/size of the feature. In this case, since the scale of the features is kind of normalized, we can state that those features with a larger magnitude (negative or positive) of coefficients are more important.

The regression coefficients are plotted for the best selected $\alpha = 0.0615$ in Ridge and Lasso cases.







	Linear Regression	Ridge Regression	Lasso Regression
age	-6.18	2.9	0
sex	-225.2	-200.9	-172.3
bmi	505.2	488.9	506.9
bp	295.6	279.0	264.1
S1	-826.0	-142.1	-125.0
S2	466.9	-55.0	0
S3	130.7	-165.8	-200.9
S4	252.9	162.9	52.2
S5	691.0	419.1	457.2
S6	111.2	119.7	94.5

bmi	s5	bp	s3	sex	s1	s6	s4	s2	age

From left to right, the significance of the features decreases.

Based on the calculated coefficients for ridge and lasso regressions, it is obvious to say that "bmi," "s5," "bp," "s3," and "sex" are the most important features. On the other hand, "age," "s2," and "s4" are the less important features. In the complex linear regression model, features "s1" and "s2" might cause overfit in the linear regression model. These features are highly correlated with the Pearson correlation coefficient of 0.89. Lasso arbitrarily selects any feature among the highly correlated ones and reduces the rest's coefficients to zero (in this case, "s2")

Apart from the expected inference of higher MSE for higher alphas, we can see the following:

- For the same alpha values, the lasso regression coefficients are smaller than that of ridge regression. This inference might not always generalize but will hold for this case and many cases.
- For the same alpha, lasso has higher MSE (Weaker fit) than ridge regression. This inference might not always generalize but will hold for this case and many cases.
- Many of the coefficients are zero, even for very small alpha values. The real difference from the ridge comes out in this inference.

- Differences

Ridge: It includes all (or none) of the features in the model. Thus, the major advantage of ridge regression is coefficient shrinkage and reducing model complexity.

Lasso: Along with shrinking coefficients, the lasso performs feature selection as well. As we observed earlier, some of the coefficients become zero, which means a particular feature is excluded from the model.

Ridge: It is majorly used to prevent overfitting. Since it includes all the features, it is not very useful in case of a very high number of features, as it will pose computational challenges.

Lasso: Since it provides sparse solutions, it is generally the model of choice (or some variant of this concept) for modeling cases where the number of features is very high. In such a case, getting a sparse solution is of great computational advantage as the features with zero coefficients can be ignored.

Ridge: It generally works well even in the presence of highly correlated features as it will include all of them in the model, but the coefficients will be distributed among them depending on the correlation.

Lasso: It arbitrarily selects any feature among the highly correlated ones and reduces the rest's coefficients to zero. Also, the chosen variable changes randomly with changes in model parameters. This generally doesn't work that well as compared to ridge regression.

Goal: To apply Naive Bayes and Logistic regression to weather data set for weather prediction. This data set is uploaded to Moodle in .cvs format

1. Examine the data set (it contains missing data).
2. Number of Instances: 145460
3. Number of features: 13
4. Target Names: [‘Rain Tomorrow’], 0 stands for no rain, and 1 stands for rain
5. Feature Names: ['MinTemp', 'MaxTemp', 'Rainfall', 'Wind Gust Speed', 'Wind Speed 9am', 'Wind Speed 3pm', 'Humidity 9am', 'Humidity 3pm', 'Pressure 9am', 'Pressure 3pm', 'Temp 9am', 'Temp 3pm', 'Rain Today']
6. Missing Feature Values: Yes
7. Targets: [‘0’, ‘1’]

First ten instances of the dataset:

```
data = pd.read_csv('C:/Users/rezam/OneDrive - Louisiana State University/PhD Courses/EE 7600 Machine Learning/Project_2/weather_r
data.head(10)
```

	MinTemp	MaxTemp	Rainfall	Wind Gust Speed	Wind Speed 9am	Wind Speed 3pm	Humidity 9am	Humidity 3pm	Pressure 9am	Pressure 3pm	Temp 9am	Temp 3pm	Rain Today	Rain Tomorrow
0	13.4	22.9	0.6	44.0	20.0	24.0	71.0	22.0	1007.7	1007.1	16.9	21.8	0	0.0
1	7.4	25.1	0.0	44.0	4.0	22.0	44.0	25.0	1010.6	1007.8	17.2	24.3	0	0.0
2	12.9	25.7	0.0	46.0	19.0	26.0	38.0	30.0	1007.6	1008.7	21.0	23.2	0	0.0
3	9.2	28.0	0.0	24.0	11.0	9.0	45.0	16.0	1017.6	1012.8	18.1	26.5	0	0.0
4	17.5	32.3	1.0	41.0	7.0	20.0	82.0	33.0	1010.8	1006.0	17.8	29.7	0	0.0
5	14.6	29.7	0.2	56.0	19.0	24.0	55.0	23.0	1009.2	1005.4	20.6	28.9	0	0.0
6	14.3	25.0	0.0	50.0	20.0	24.0	49.0	19.0	1009.6	1008.2	18.1	24.6	0	0.0
7	7.7	26.7	0.0	35.0	6.0	17.0	48.0	19.0	1013.4	1010.1	16.3	25.5	0	0.0
8	9.7	31.9	0.0	80.0	7.0	28.0	42.0	9.0	1008.9	1003.6	18.3	30.2	0	1.0
9	13.1	30.1	1.4	28.0	15.0	11.0	58.0	27.0	1007.0	1005.7	20.1	28.2	1	0.0

A summary of a Dataset:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 145460 entries, 0 to 145459
Data columns (total 14 columns):
 #   Column           Non-Null Count   Dtype  
---  ------           --------------   -----  
 0   MinTemp          143975 non-null  float64
 1   MaxTemp          144199 non-null  float64
 2   Rainfall         142199 non-null  float64
 3   Wind Gust Speed  135197 non-null  float64
 4   Wind Speed 9am   143693 non-null  float64
 5   Wind Speed 3pm   142398 non-null  float64
 6   Humidity 9am     142806 non-null  float64
 7   Humidity 3pm     140953 non-null  float64
 8   Pressure 9am     130395 non-null  float64
 9   Pressure 3pm     130432 non-null  float64
 10  Temp 9am         143693 non-null  float64
 11  Temp 3pm         141851 non-null  float64
 12  Rain Today       145460 non-null  int64  
 13  Rain Tomorrow    142193 non-null  float64
dtypes: float64(13), int64(1)
memory usage: 15.5 MB
```

The statistical description of the dataset before dealing with missing values:

```
data.describe() #Data Describtion
```

	MinTemp	MaxTemp	Rainfall	Wind Gust Speed	Wind Speed 9am	Wind Speed 3pm	Humidity 9am	Humidity 3pm	Pressure 9am	Pressure 3pm
count	143975.000000	144199.000000	142199.000000	135197.000000	143693.000000	142398.000000	142806.000000	140953.000000	130395.00000	130432.000000
mean	12.194034	23.221348	2.360918	40.035230	14.043426	18.662657	68.880831	51.539116	1017.64994	1015.255889
std	6.398495	7.119049	8.478060	13.607062	8.915375	8.809800	19.029164	20.795902	7.10653	7.037414
min	-8.500000	-4.800000	0.000000	6.000000	0.000000	0.000000	0.000000	0.000000	980.50000	977.100000
25%	7.600000	17.900000	0.000000	31.000000	7.000000	13.000000	57.000000	37.000000	1012.90000	1010.400000
50%	12.000000	22.600000	0.000000	39.000000	13.000000	19.000000	70.000000	52.000000	1017.60000	1015.200000
75%	16.900000	28.200000	0.800000	48.000000	19.000000	24.000000	83.000000	66.000000	1022.40000	1020.000000
max	33.900000	48.100000	371.000000	135.000000	130.000000	87.000000	100.000000	100.000000	1041.00000	1039.600000

The overview heatmap image of the correlation between each pair of the features is printed out here.

To scrutinize the correlation between each pair of features, we can draw the above pictures on a large scale or calculate the Pearson correlation coefficient. The correlation coefficient (Pearson) between each set of features is calculated as follows.

```
data.corr()
```

	MinTemp	MaxTemp	Rainfall	Wind Gust Speed	Wind Speed 9am	Wind Speed 3pm	Humidity 9am	Humidity 3pm	Pressure 9am	Pressure 3pm	Temp 9am	Temp 3pm	Rain Today	Rain Tomorrow
MinTemp	1.000000	0.736555	0.103938	0.177415	0.175064	0.175173	-0.232899	0.006089	-0.450970	-0.461292	0.901821	0.708906	0.054702	0.083936
MaxTemp	0.736555	1.000000	-0.074992	0.067615	0.014450	0.050300	-0.504110	-0.508855	-0.332061	-0.427167	0.887210	0.984503	-0.226001	-0.159237
Rainfall	0.103938	-0.074992	1.000000	0.133659	0.087338	0.057887	0.224405	0.255755	-0.168154	-0.126534	0.011192	-0.079657	0.501516	0.239032
Wind Gust Speed	0.177415	0.067615	0.133659	1.000000	0.605303	0.686307	-0.215070	-0.026327	-0.458744	-0.413749	0.150150	0.032748	0.151605	0.234010
Wind Speed 9am	0.175064	0.014450	0.087338	0.605303	1.000000	0.519547	-0.270858	-0.031614	-0.228743	-0.175817	0.128545	0.004569	0.099084	0.090995
Wind Speed 3pm	0.175173	0.050300	0.057887	0.686307	0.519547	1.000000	-0.145525	0.016432	-0.296351	-0.255439	0.163030	0.027778	0.077913	0.087817
Humidity 9am	-0.232899	-0.504110	0.224405	-0.215070	-0.270858	-0.145525	1.000000	0.666949	0.139442	0.186858	-0.471354	-0.498399	0.349752	0.257161
Humidity 3pm	0.006089	-0.508855	0.255755	-0.026327	-0.031614	0.016432	0.666949	1.000000	-0.027544	0.051997	-0.221019	-0.557841	0.373596	0.446160
Pressure 9am	-0.450970	-0.332061	-0.168154	-0.458744	-0.228743	-0.296351	0.139442	-0.027544	1.000000	0.961326	-0.422556	-0.286770	-0.187547	-0.246371
Pressure 3pm	-0.461292	-0.427167	-0.126534	-0.413749	-0.175817	-0.255439	0.186858	0.051997	0.961326	1.000000	-0.470187	-0.389548	-0.104862	-0.226031
Temp 9am	0.901821	0.887210	0.011192	0.150150	0.128545	0.163030	-0.471354	-0.221019	-0.422556	-0.470187	1.000000	0.860591	-0.096357	-0.025691
Temp 3pm	0.708906	0.984503	-0.079657	0.032748	0.004569	0.027778	-0.498399	-0.557841	-0.286770	-0.389548	0.860591	1.000000	-0.232407	-0.192424
Rain Today	0.054702	-0.226001	0.501516	0.151605	0.099084	0.077913	0.349752	0.373596	-0.187547	-0.104862	-0.096357	-0.232407	1.000000	0.306555
Rain Tomorrow	0.083936	-0.159237	0.239032	0.234010	0.090995	0.087817	0.257161	0.446160	-0.246371	-0.226031	-0.025691	-0.192424	0.306555	1.000000

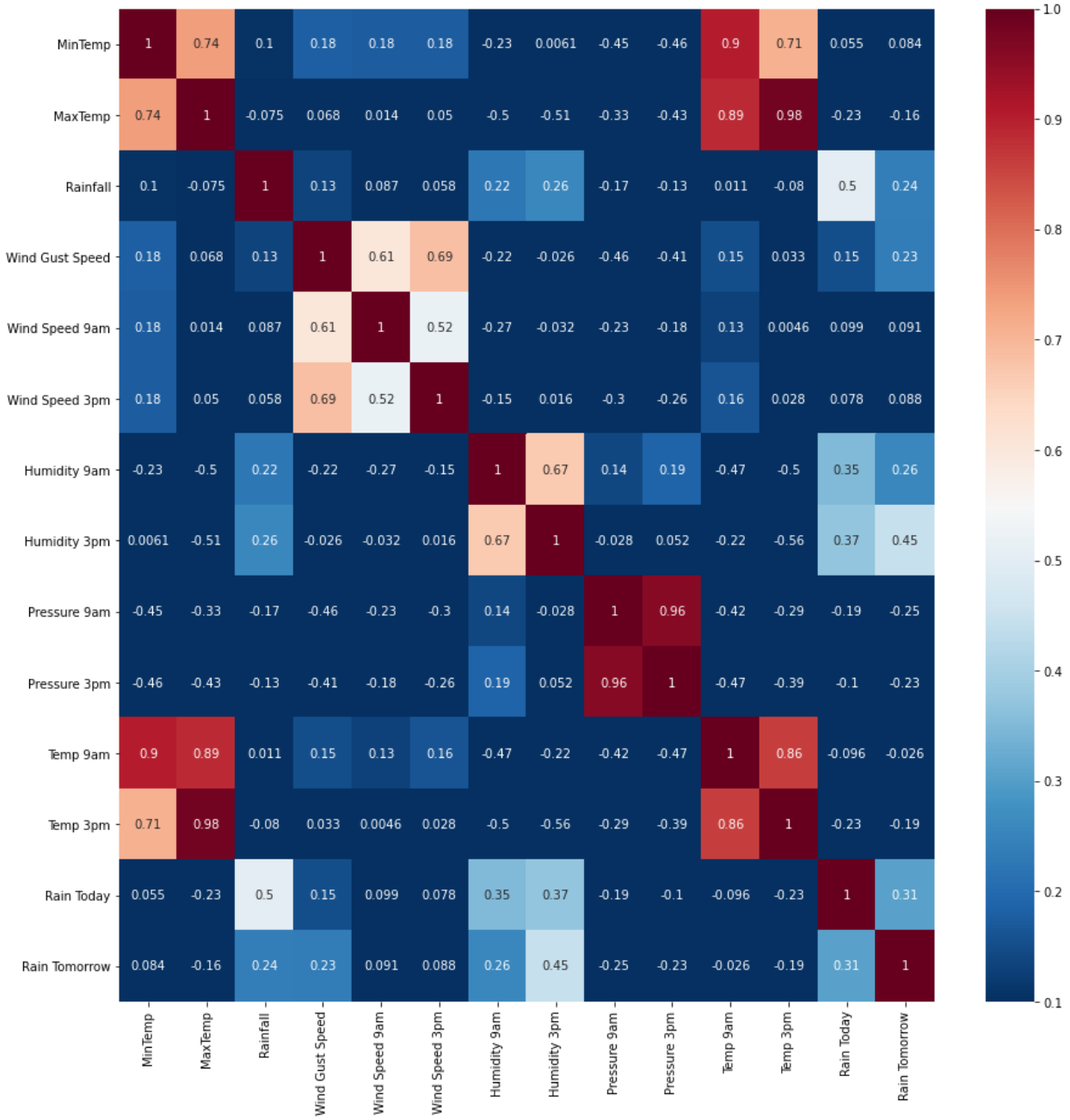
Handling Missing values in Numerical features:

```
numerical_features = [column_name for column_
data[numerical_features].isnull().sum()
```

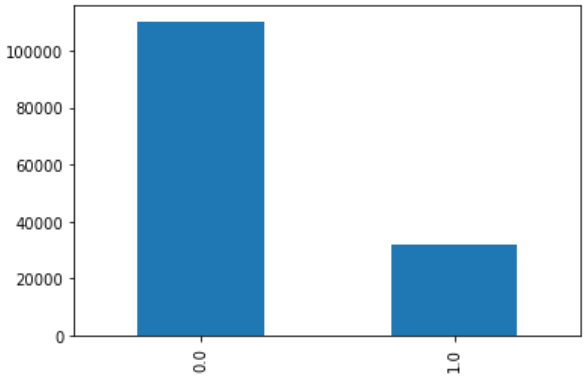
```
MinTemp          1485
MaxTemp          1261
Rainfall         3261
Wind1           10263
Wind2            1767
Wind3            3062
Humidity1        2654
Humidity2        4507
Pressure1       15065
Pressure2       15028
Temp1            1767
Temp2            3609
Rain Today          0
Rain_Tomorrow    3267
dtype: int64
```

All of the features contain missing values but ‘Rain Today.’ I drop all of samples with missing values.

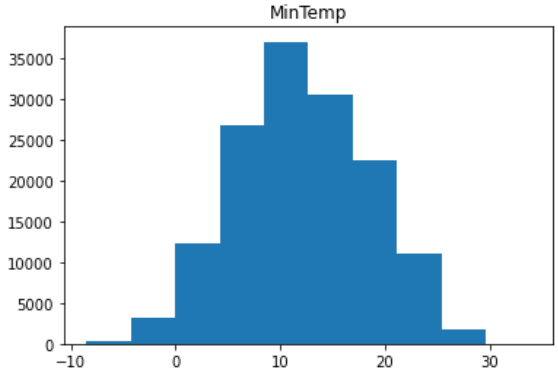
See correlation after cleaning the data using the heatmap: Most of the features have no correlation

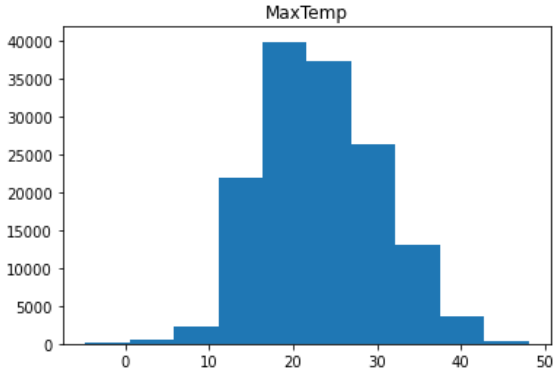


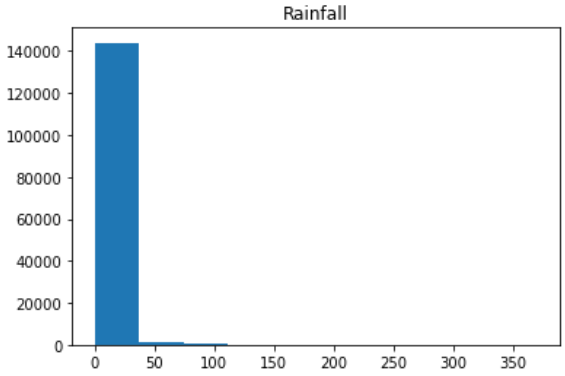
Also target values are not balanced.

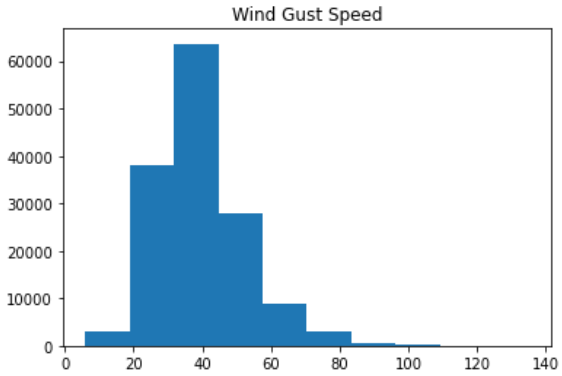


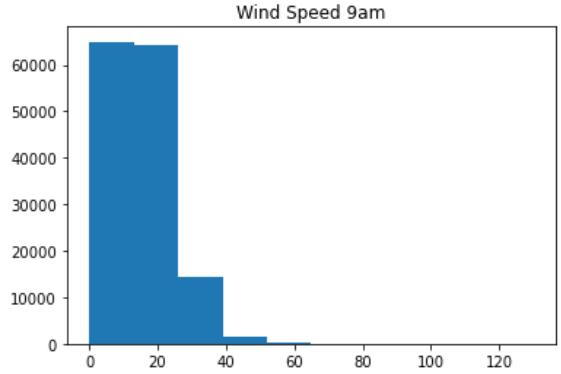
8. Plot the histogram of the features. Is Gaussian a good model for the distribution of the features?

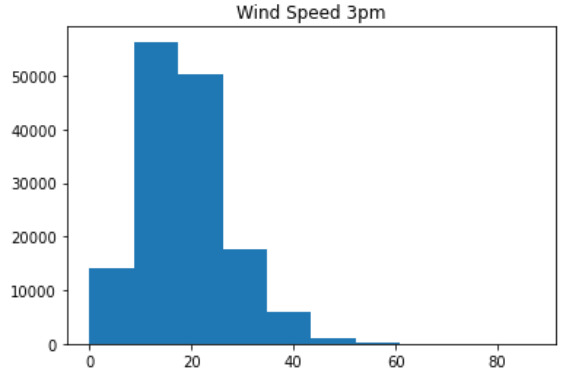


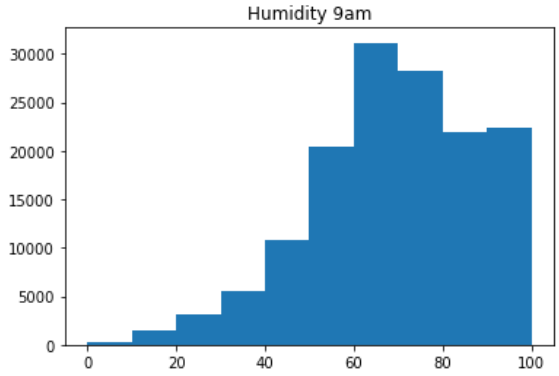


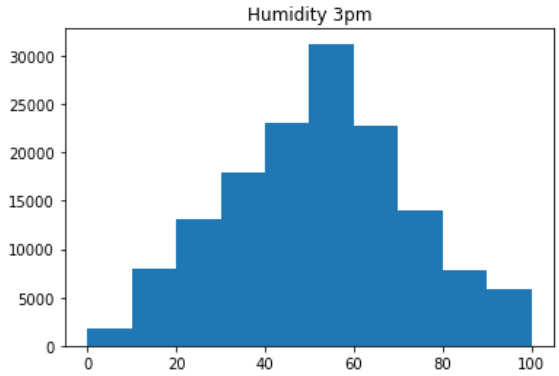

Rainfall
140000
120000
100000
80000
60000
40000
20000
0
0
50
100
150
200
250
300
350

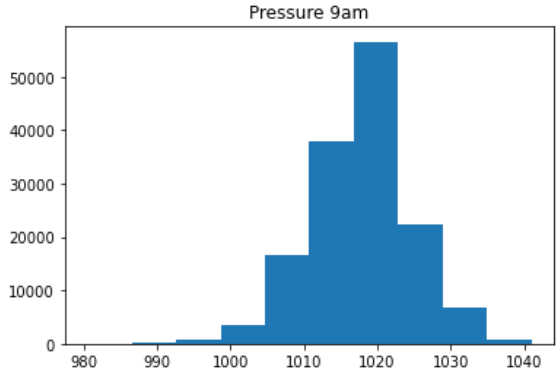

Wind Gust Speed
60000
50000
40000
30000
20000
10000
0
0
20
40
60
80
100
120
140

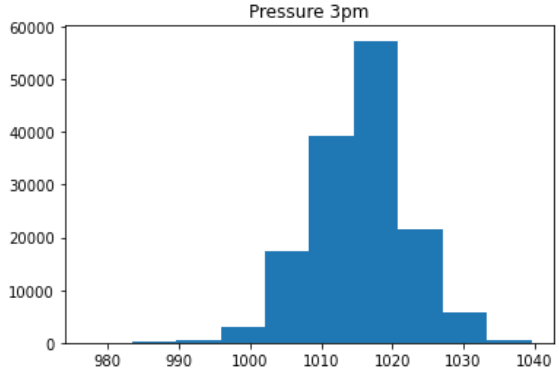

Wind Speed 9am
60000
50000
40000
30000
20000
10000
0
0
20
40
60
80
100
120

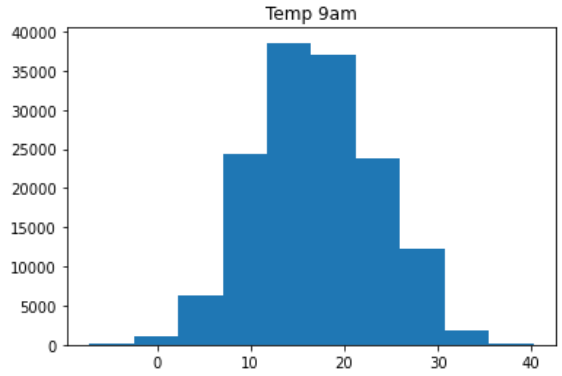

Wind Speed 3pm
50000
40000
30000
20000
10000
0
0
20
40
60
80

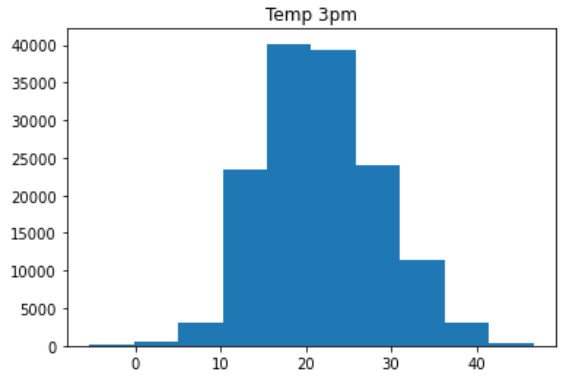

Humidity 9am
30000
25000
20000
15000
10000
5000
0
0
20
40
60
80
100

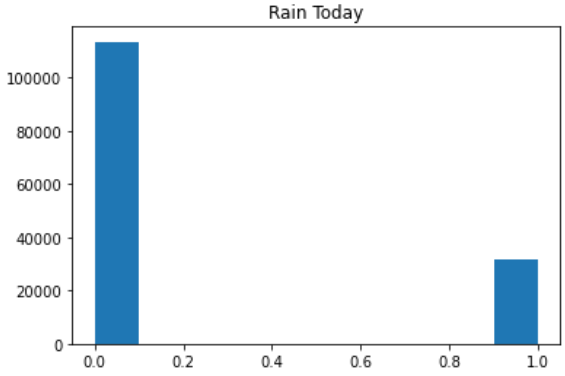

Humidity 3pm
30000
25000
20000
15000
10000
5000
0
0
20
40
60
80
100


Pressure 9am
50000
40000
30000
20000
10000
0
980
990
1000
1010
1020
1030
1040


Pressure 3pm
60000
50000
40000
30000
20000
10000
0
980
990
1000
1010
1020
1030
1040







Since most of the features (not all of them) are independent Gaussian is a good model for these features.

9. Apply a Naive Bayes classification model.

a) Find the misclassification error for both the training set and test set.

Train error is: 18.4 %

Test error is: 18.2 %

b) Show the confusion matrix for both training and test sets.

Confusion matrix for training set:

66979	**7743**
9911	11039

Confusion matrix for testing set:

16754	**1927**
2445	2792

c) Does scaling (use Standards scaler) have any effect on the results?

Naive Bayes does not affect by feature scaling. In fact, any Algorithm which is NOT distance-based is not affected by Feature Scaling. Here are the results after scaling the dataset. In this case scaling improved the testing error by 0.1 persent.

Train error is: 18.4 %

Test error is: 18.3 %

Confusion matrix for scaled training set:

66979	**7743**
9911	11039

Confusion matrix for scaled testing set:

16734	**1947**
2441	2796

10. Apply the logistic regression classifier. Consider both $l2$ and $l1$ regularitzation. The optimization algorithm (solver) needs to be appropriately chosen.

a) Answer the three questions (a)-(c) above.

For $l2$ case I used 'lbfgs' solver and get the following results:

Train error is: 15.2 %

Test error is: 15.2 %

Confusion matrix for training set:

71031	**3691**
10875	10075

Confusion matrix for testing set:

17772	**909**
2729	2508

For $l1$ case I used 'liblinear' (slow convergence but better than saga) solver and get the following results:

Train error is: 15.1 %

Test error is: 15.1 %

Confusion matrix for training set:

71047	3675
10820	10131

Confusion matrix for testing set:

17772	909
2720	2517

I got almost the same accuracy results for both $l1$ and $l2$ regularizations, however solver 'newton-cg' performs the best accuracy results, but the slow convergence might make up for this solver in large scale datasets.

For $l2$ case and scaled data I used 'lbfgs' solver and get the following results:

Train error is: 14.9 %

Test error is: 14.9 %

Confusion matrix for scaled training set:

70919	**3803**
10485	10465

Confusion matrix for scaled testing set:

17732	**949**
2621	2616

For $l1$ case and scaled data I used 'liblinear' (slow convergence but better than saga) solver and get the following results:

Train error is: 14.9 %

Test error is: 14.9 %

Confusion matrix for scaled training set:

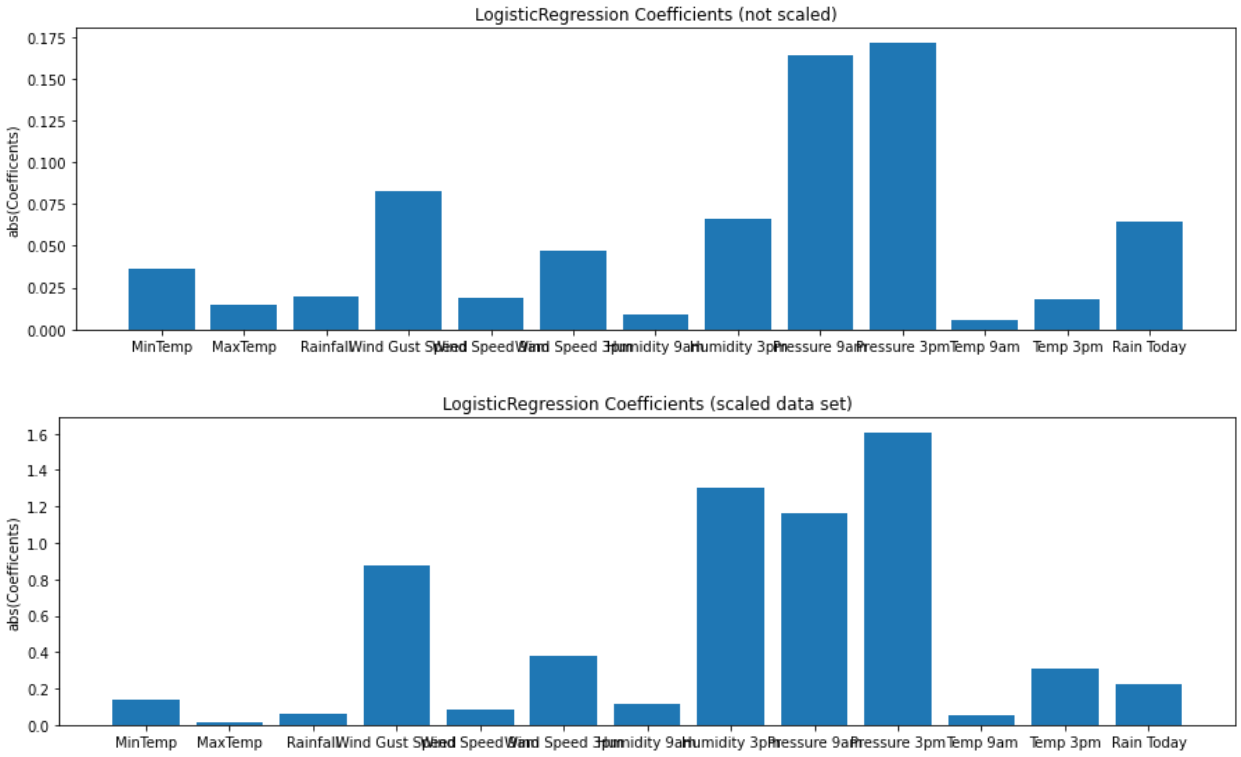
70918	3804
10485	10465

Confusion matrix for scaled testing set:

17731	950
2620	2617

The data scaling did not improve the performance of logistic regression significantly. But in general, we need to perform Feature Scaling when we are dealing with Gradient Descent Based algorithms like logistic regression.

b) Plot the bar graph of the weights and rank the features according to the weights.



Features with higher coefficient weights are moer important.

	Feature name-from important to not important
1	Pressure 3pm
2	Pressure 9am

3	Humidity
4	Wind Gust Speed
5	Wind Speed 3pm
6	Rain Today
7	Humidity 9am
8	Temp 9am
9	Min Temp
10	Wind Speed 9am
11	Rainfall
12	Temp 3pm
13	Max Temp