VAIBHAV RAJPAL

2020146

Ans 1:

a. For simple linear regression model we can write the expression as y=w0 + w1*x where w1 is the coefficient for feature x and w0 is the noise. The least square tends to become w0 and w1 when we minimize the residual sum of squares and while minimizing it we get the value of coefficients w0 and w1 as

$$W1 = \frac{\Sigma (Xi-X^-)*(Yi-Y^-)}{\Sigma (Xi-X^-)^2}$$

$$W0 = Y^- - W1 * X^-$$

Now from the original expression y=w0 + w1*x let the line passes through point X^- so now substituting x with X^- and the value of w0 we get

$$y = Y^- - W1 * X^- + W1 * X^-$$

which reduces to $y = Y^-$. Thus for simple linear regression the least square fit line always passes through the point (X^-, Y^-).

b. If two variables have a high corelation with the third variable then it **Does Not** imply that they will also be highly corelated. An example to prove my argument would be suppose we want to calculate BMI of a person then to compute it we know that BMI depends upon both weight and height but weight and height are not dependent on each other.

c. According to the weak law of large numbers for e>0

P(|Sample Mean – Actual Mean| >= e) ->0 as n->infinity

Now in order to prove it let us assume X1,X2,….Xn. As I.I.D variables with mean ‘m’ and variance ‘v^2’
Now finding the sample mean we get
Sample mean = (x1+x2+….Xn)/n
Now finding variance of sample mean we get

Variance(Sample mean) = variance(x1 + x2 + … xn)/n^2= n*v^2/n^2=v^2/n (this is because all x1,x2,…xn are independent variables thus variance is sum of variance of all of these independent random variables)

Now applying the Chebyshev's inequality we get
P(|sample mean – actual mean | >= e) <= variance of sample mean/e^2 → v^2/n*e^2 →0 as n→0. And which proves our law of large numbers.

Pseudo-Code :
Suppose we want to compute average of dice rolls and if we find the theoretical value it comes out be 3.5

Note: Took the idea from the one showed in the class by sir.

```
Function ans(number of times experiment have to perform) :
        Helper=numpy array with cumulative sum obtained from the dice roll
        For i in range (len(helper)):
                Avg.append(helper[i]/(i+1))

        Return avg
```

Now here we observed that as the number of times experiment have to be performed increases the difference between the sample mean and actual mean tends to zero and which is what weak law of large number states.

d. As follows:

d) As it is a gaussian prior distribution of the weights

$$w \sim N(0, \sigma^2)$$

Which means it have a normal distribution with mean = 0 & standard deviation = σ, variance = σ^2

$$\frac{D}{w} \sim N(w^T x_i, \sigma^2)$$

Mean = $w^T x_i$, variance = σ^2

PDF of gaussian distribution is given by

$$P(y_i | \vec{x_i}, w) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}\left[\frac{w^T x_i - y}{\sigma}\right]^2}$$

Now finding this value for w, & $\frac{D}{w}$

For w

$$\Rightarrow \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}\left[(w-0)^T \cdot (w-0)\right]}$$

Assume Variance (σ^2) = 1

$$\Rightarrow \frac{\sqrt{\lambda}}{\sqrt{2\pi}} e^{-\frac{1}{2}\left(w^T \cdot w \frac{\lambda}{2}\right)}$$

For D/w

We know $\Rightarrow \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(w^T x_i - y_i)^2}$

Now for MAP, we know that

$$P\left(\frac{w}{D}\right) = P\left(\frac{D}{w}\right) \cdot \frac{P(w)}{P(D)}$$

Which means we need to maximize the probability

$$\underset{w}{\operatorname{argmax}} \prod_{i=1}^{n} P(w; y_i | \vec{x_i}) = \underset{w}{\operatorname{argmax}} \sum_{i=1}^{n} \log P(w, y_i | \vec{x_i})$$

∵ Our aim is to get 'w' though the value of both expression is different but the 'w' that we would get from both of them will be the same.

$$\log P\left(\frac{w}{D}\right) = \log\left[P\left(\frac{D}{w}\right) \cdot \frac{P(w)}{P(D)}\right] = \log\left(P\left(\frac{D}{w}\right)\right) + \log[P(w)] - \log(P(D))$$

$$\Rightarrow \underset{w}{\operatorname{argmax}} \log\left(P\left(\frac{D}{w}\right)\right) + \log(P(w)) - \log(P(D))$$

$$\Rightarrow \underset{w}{\operatorname{argmax}} \log P\left(\frac{D}{w}\right) + \log P(w)$$

{ ∵ $\log P(D))$ is a constant value & will not effect the value of w.

$$\log\left(\frac{D}{w}\right) = \sum_{i=1}^{n} \log \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(w^T x_i - y_i)^2}$$

$$= \sum_{i=1}^{n} \log \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (w^T x_i - y_i)^2$$

Removing Constant term from expression we get

$$\log\left(\frac{D}{w}\right) = -\frac{1}{2\sigma^2}\sum_{i=1}^{n}(w^T x_i - y_i)^2$$

Now finding $\log P(w)$

$$\log P(w) = -\frac{\lambda}{2} w^T w$$

Now Putting all these values

$$\Rightarrow \arg\max_w \left[-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(w^T x_i - y_i)^2 - \frac{\lambda}{2} w^T w\right]$$

$$\Rightarrow \arg\min_w \left[\frac{1}{2\sigma^2}\sum_{i=1}^{n}(w^T x_i - y_i)^2 + \frac{\lambda}{2} w^T w\right]$$

Ans 2.

a.

For doing the k-fold cross validation we divided the dataset into k groups from where we took all the k set's one by one for testing and the remaining k-1 for training. There after separating data set into training and testing. Now we passed the training data to gradient descent then from there we obtained the value of weights (theta) for the linear regression model and then computed the RSME value for each of the folds and then finally took the average of all the RSME values obtained for each of the fold's and the result is shown below:

MEAN RSME VALUES FOR DIFFERENT K VALUES

K=	2	3	4	5
RSME=	4.43	3.61	3.11	2.73

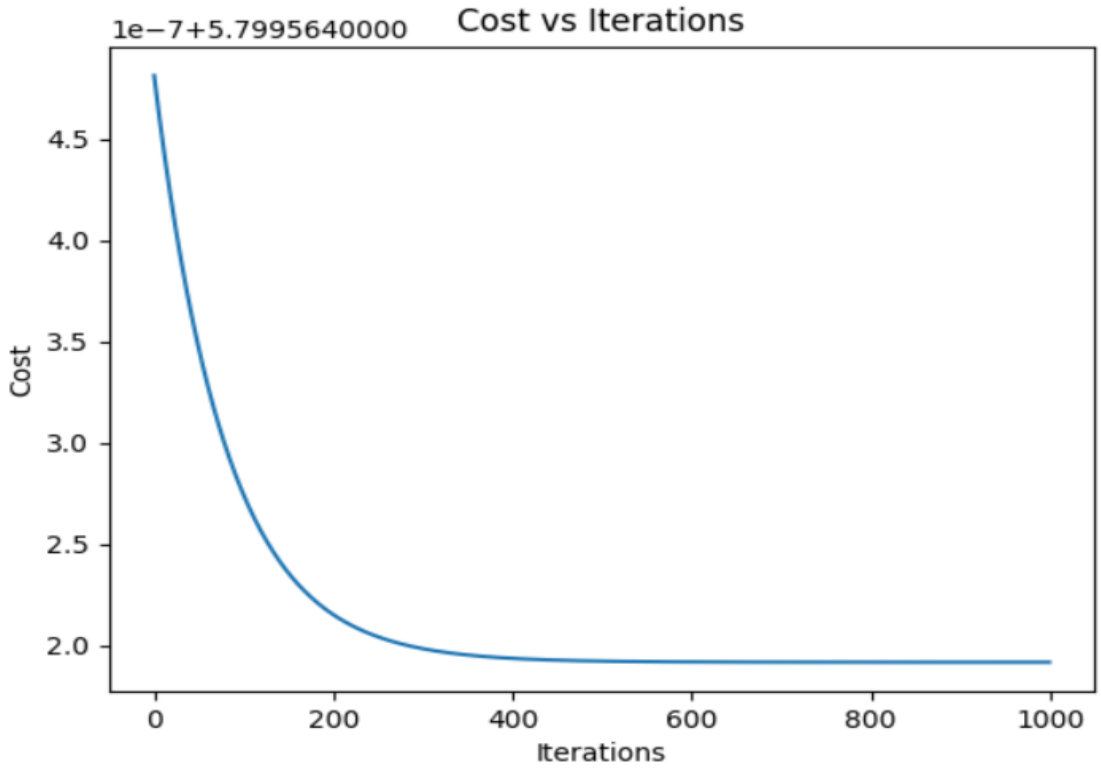
From the above table values we can see that we are getting least rsme value for K=5 . Thus, k=5 is the optimal value of 5.

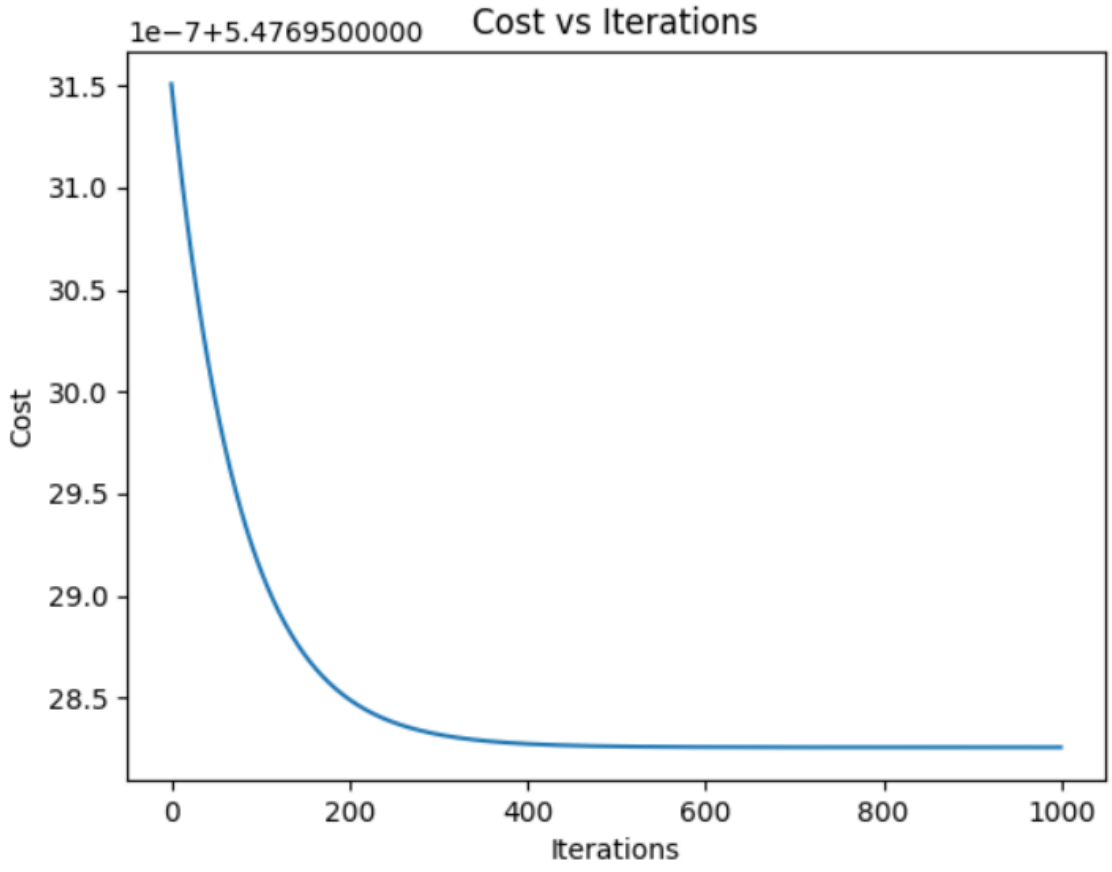
b. The graphs are as follows:

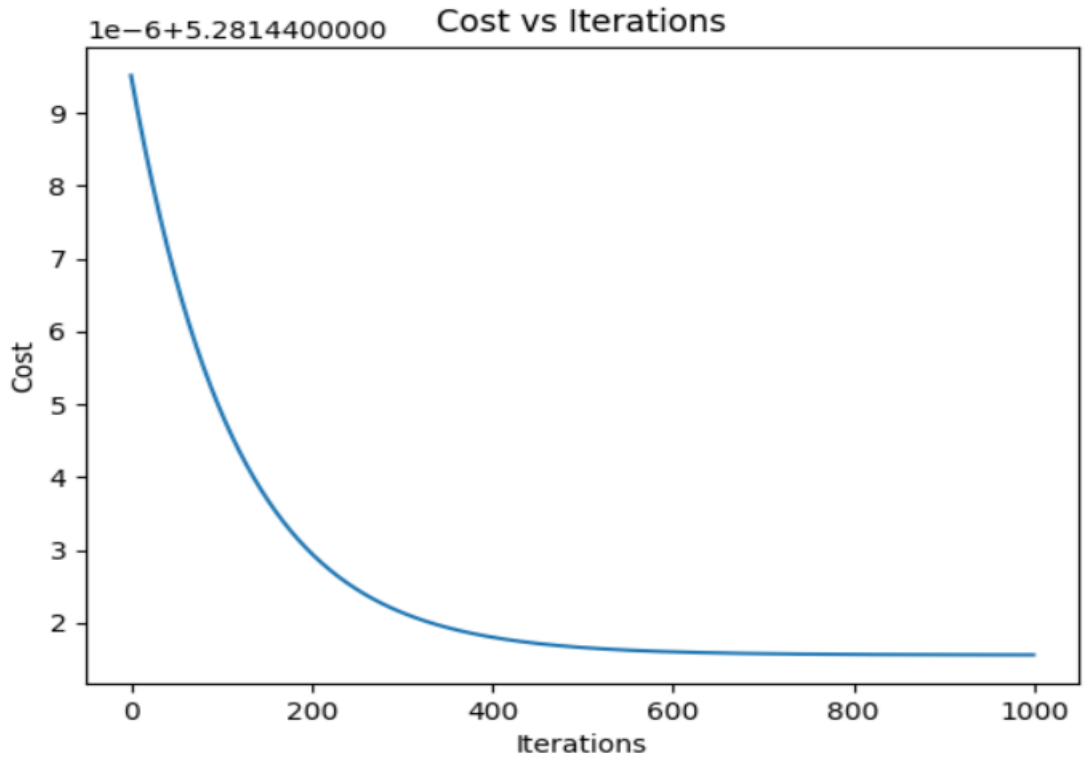
Mostly starting approach is similar to the one used above but there is a new function in it which plot's the RSME vs iteration graph where for each of the training and testing set's we pass the value of weights obtained from gradient descent and pass it to our new function where for every iteration we store the RSME for that particular iteration and update our weight's. Once the iteration get's over we finally plot the graph from the values stored.

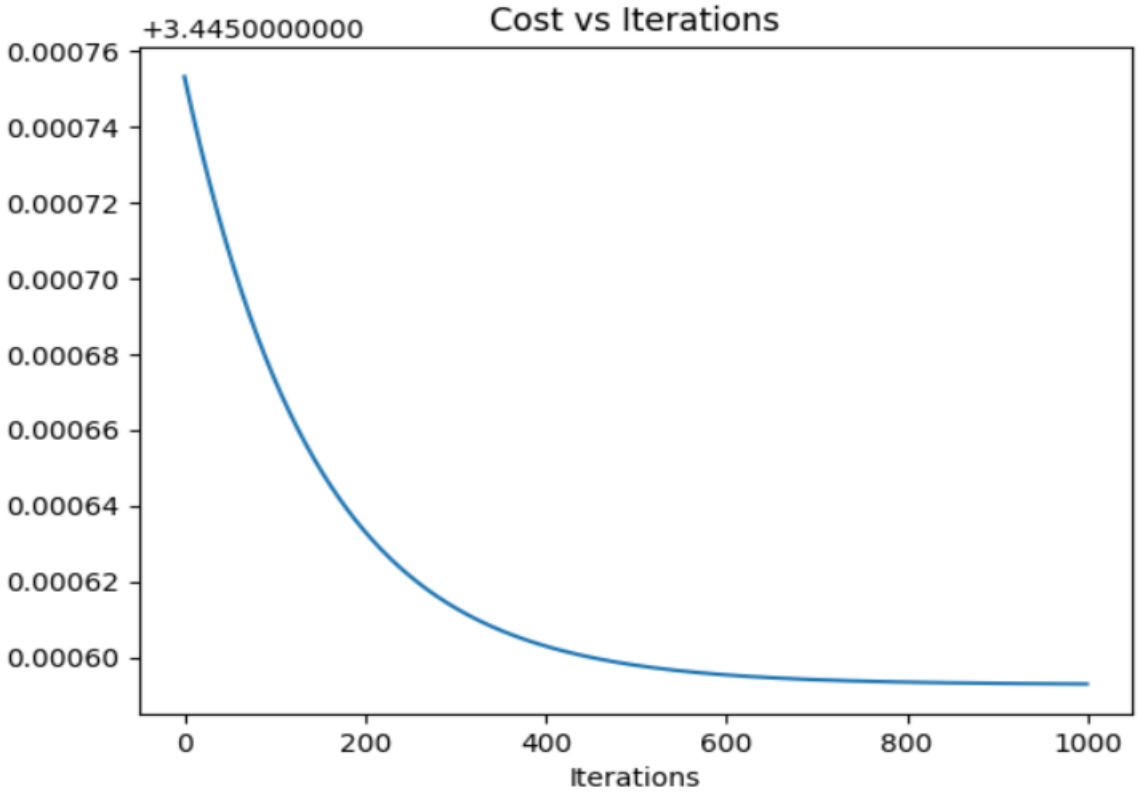
Below are the graphs for all training and testing sets for k=5.

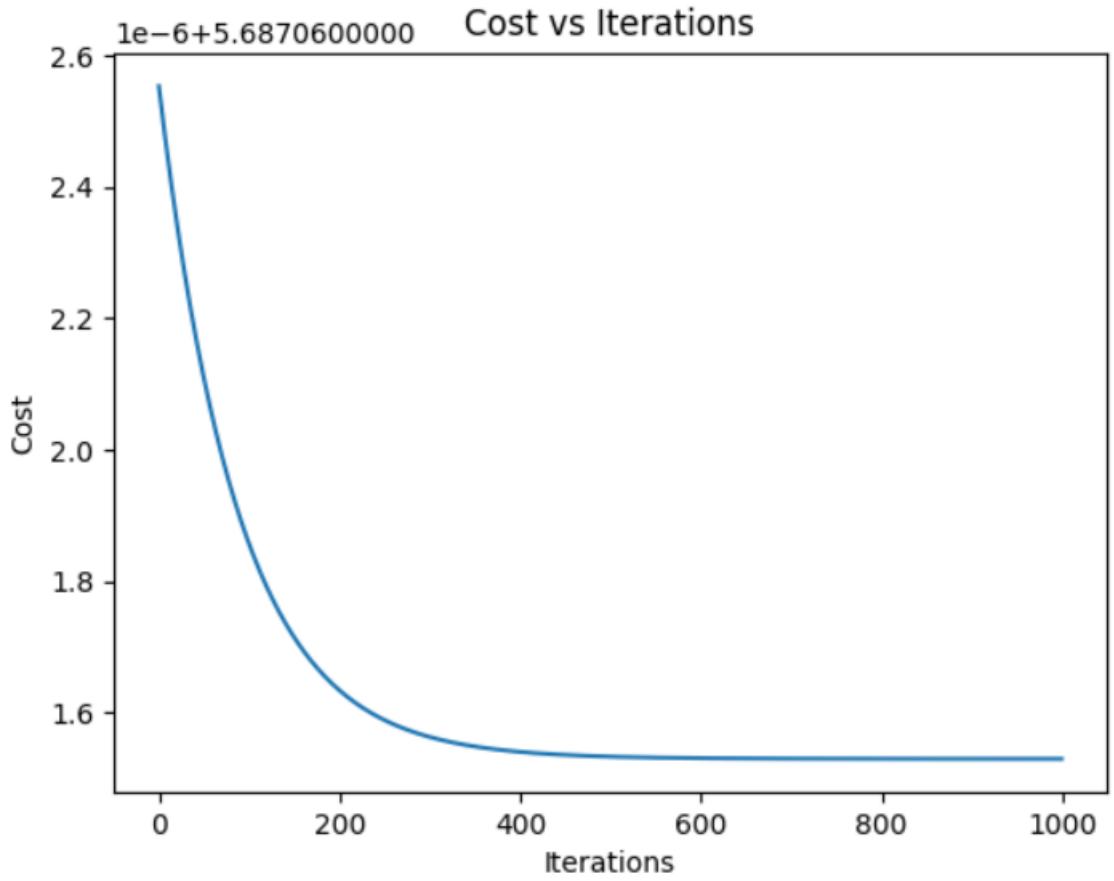
Training Set RSME Graphs


Cost vs Iterations
1e−7+5.7995640000
4.5
4.0
3.5
3.0
2.5
2.0
Cost
0
200
400
600
800
1000
Iterations

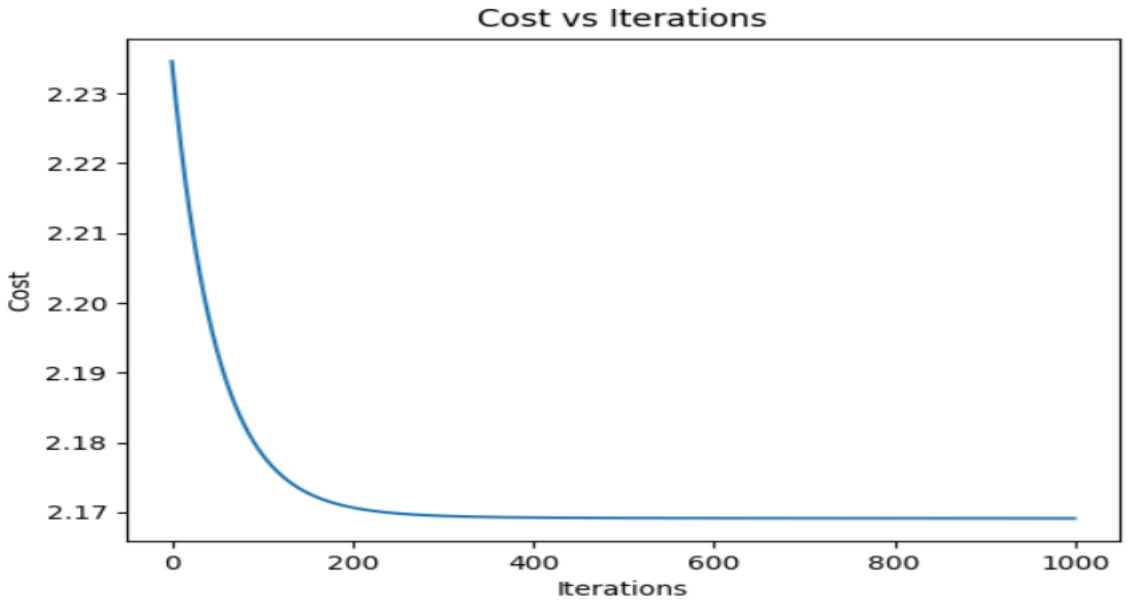

Cost vs Iterations
1e−7+5.4769500000
31.5
31.0
30.5
30.0
29.5
29.0
28.5
Cost
0
200
400
600
800
1000
Iterations

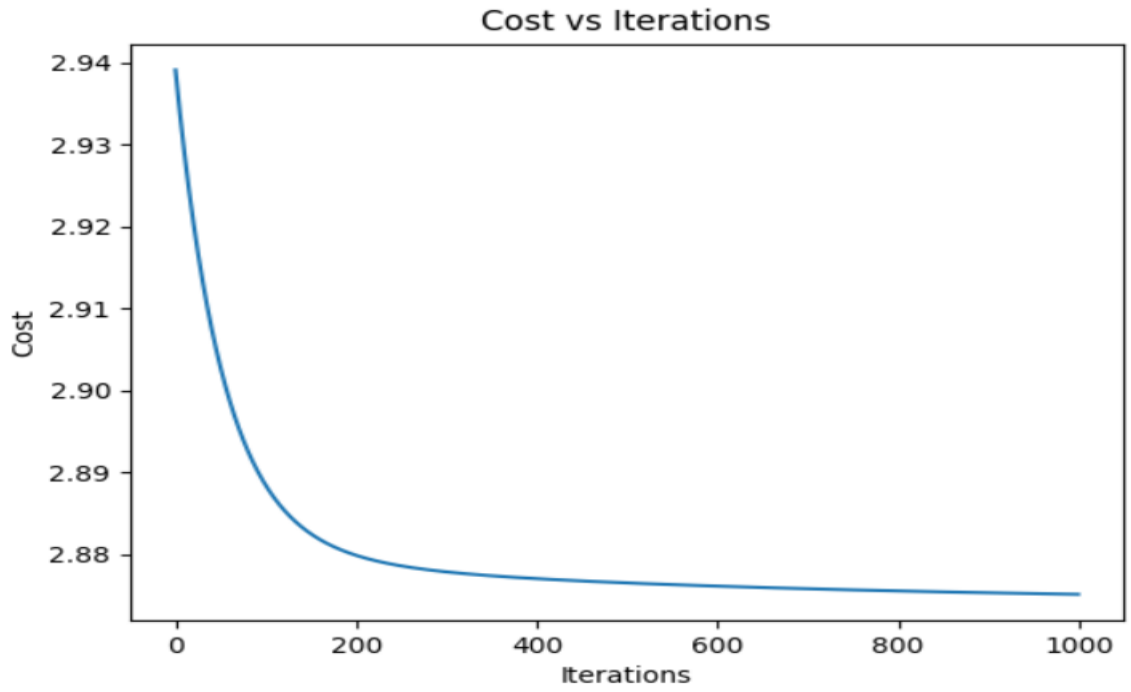

Cost vs Iterations
1e−6+5.2814400000
Cost
9
8
7
6
5
4
3
2
0
200
400
600
800
1000
Iterations

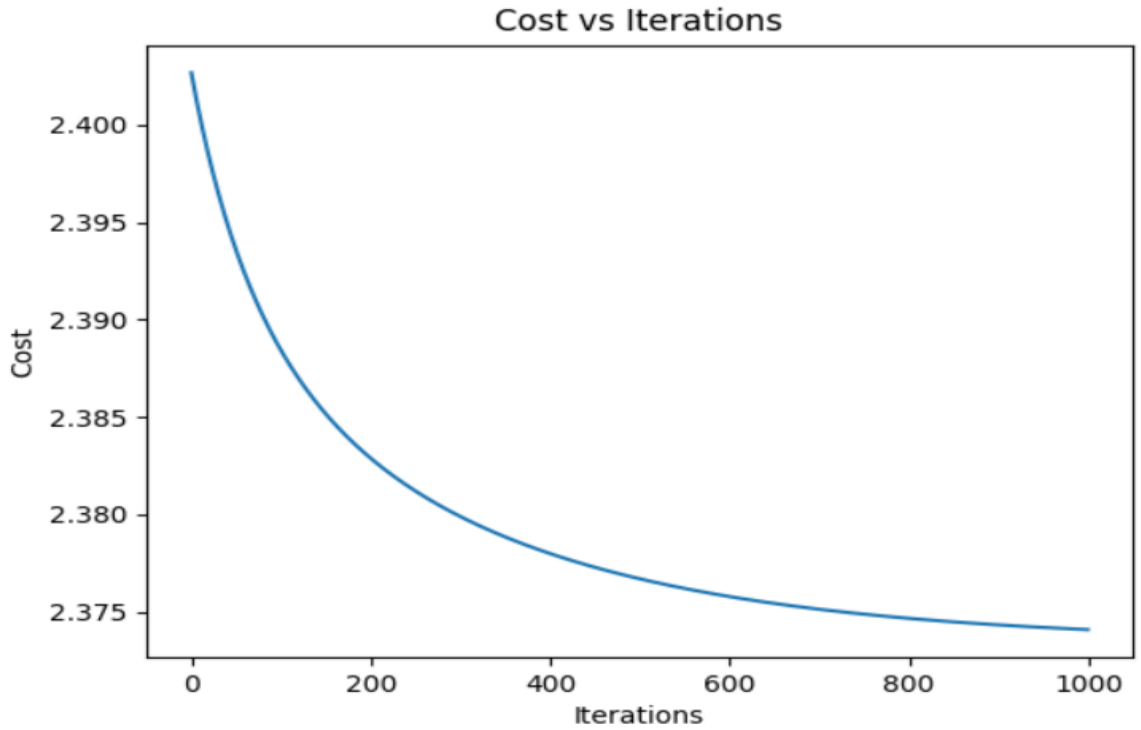

Cost vs Iterations
+3.4450000000
0.00076
0.00074
0.00072
0.00070
0.00068
0.00066
0.00064
0.00062
0.00060
0
200
400
600
800
1000
Iterations

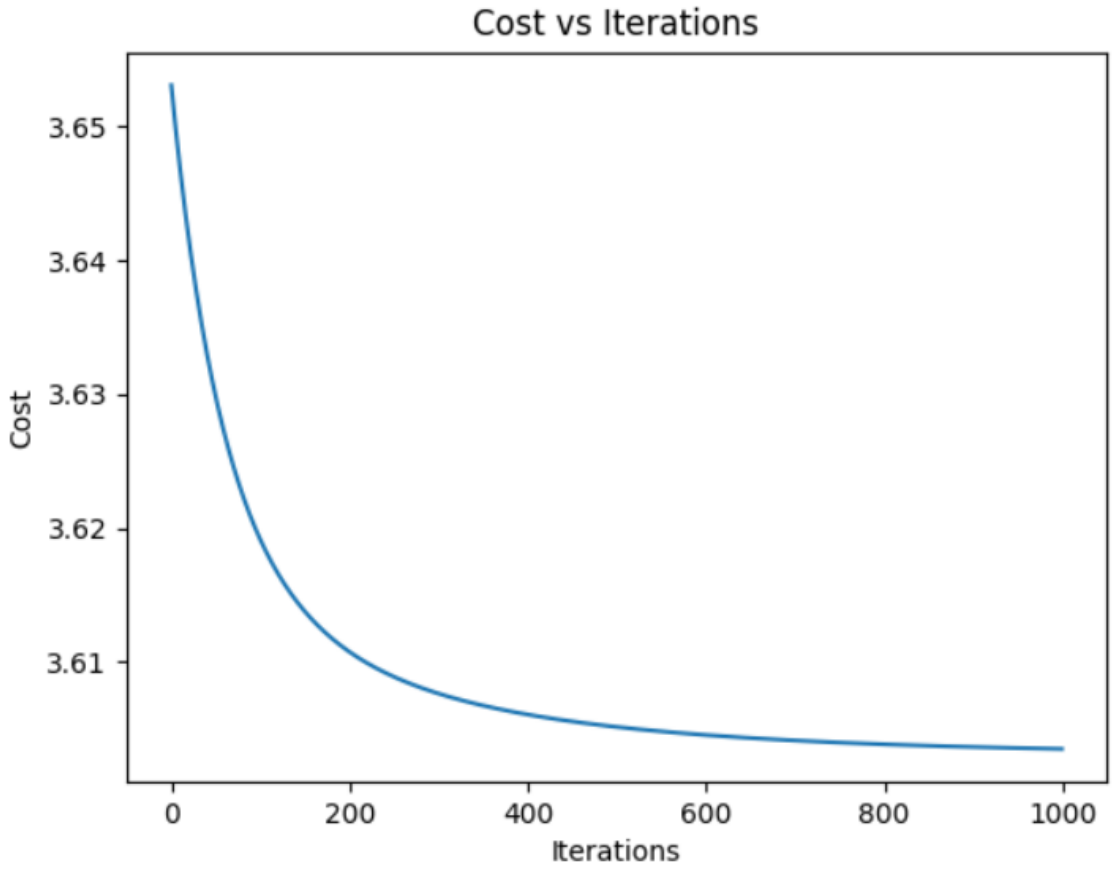


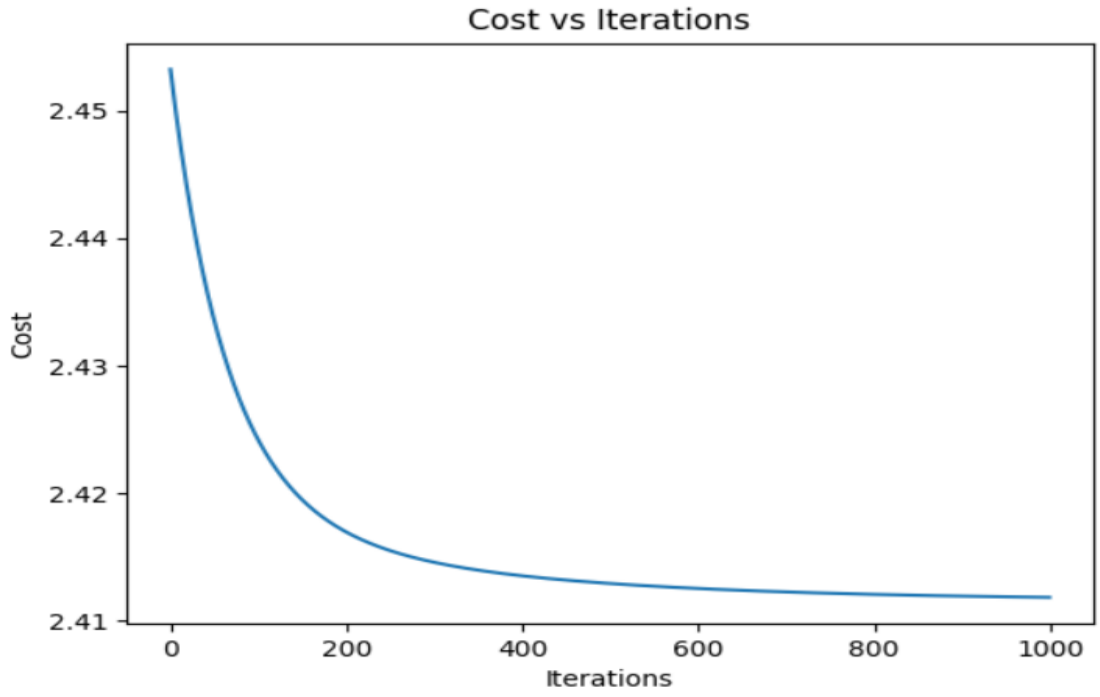
Testing/Validation Set RSME Graphs




Cost vs Iterations
2.94
2.93
2.92
2.91
2.90
2.89
2.88
Cost
0
200
400
600
800
1000
Iterations


Cost vs Iterations
2.400
2.395
2.390
2.385
2.380
2.375
Cost
0
200
400
600
800
1000
Iterations


Cost vs Iterations
3.65
3.64
3.63
3.62
3.61
Cost
0
200
400
600
800
1000
Iterations


Cost vs Iterations
2.45
2.44
2.43
2.42
2.41
Cost
0
200
400
600
800
1000
Iterations

c. The idea followed here is the combination of what I did in part a and b where the slight modification that I made was in the error term and while updating weights. For ridge regression I added (lambda/2)*theta^2 while for lasso I did (lambda/2)*|theta|.

Trying different values of lambda for **Ridge Regression**:

Lambda	RSME
0.01	4.9906
0.1	9.7522
0.5	19.1014
1	19.7873

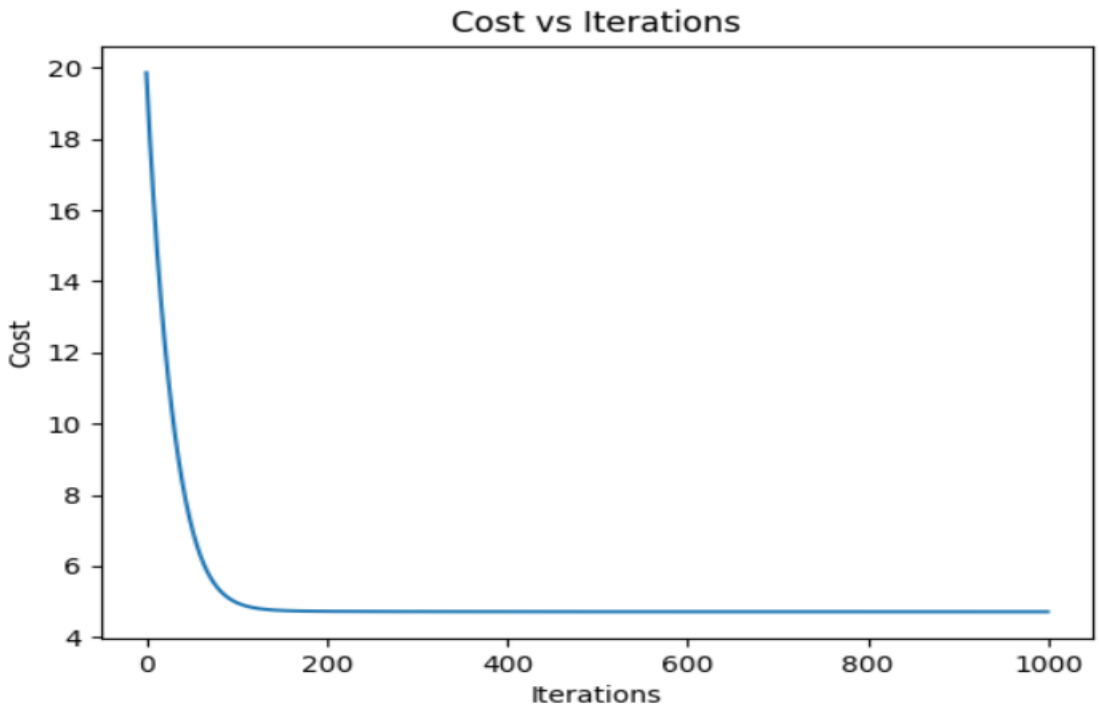
Trying different values of lambda for **Lasso Regression**

Lambda	RSME
0.01	5.311
0.1	7.5217
0.5	17.7650
1	29.8456

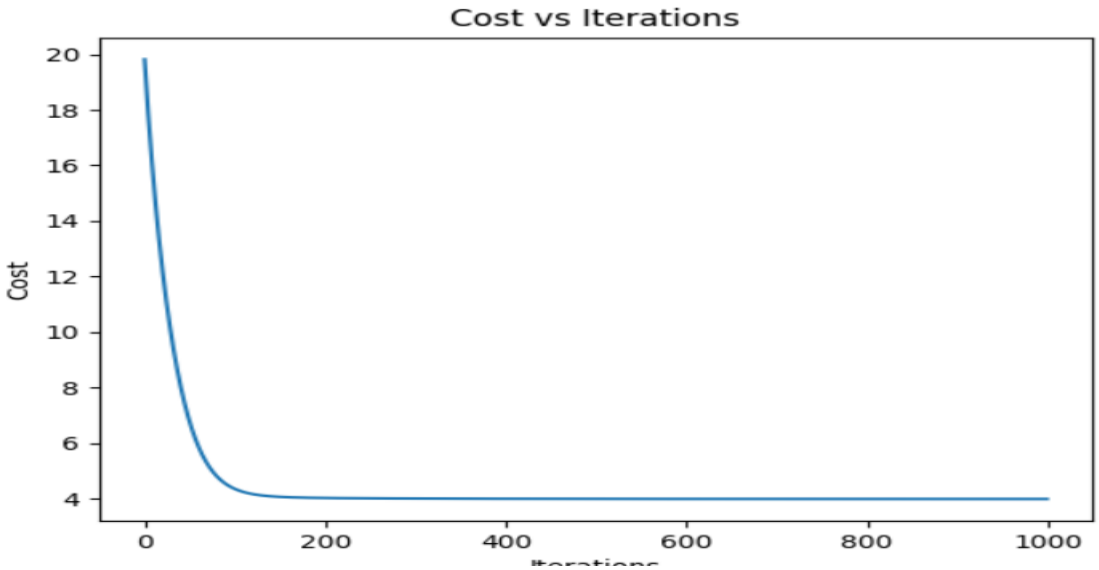
The optimal value of lambda in both cases is 0.01 and thus would use it only.

Graph is being plotted for k=2 folds in this case

Training Set Graph for Ridge Regression



Testing/Validation Set Graph for Ridge Regression



d. To obtain the weights using normal form expression is (X'X)^(-1)*(X'Y) so the only change done here is instead of using gradient descent we would use the above expression to get the result directly and the weights obtained are for k=5:

[38.03900372 1.41792309 -3.22147664 -5.4386273   3.38332826 3.06060878

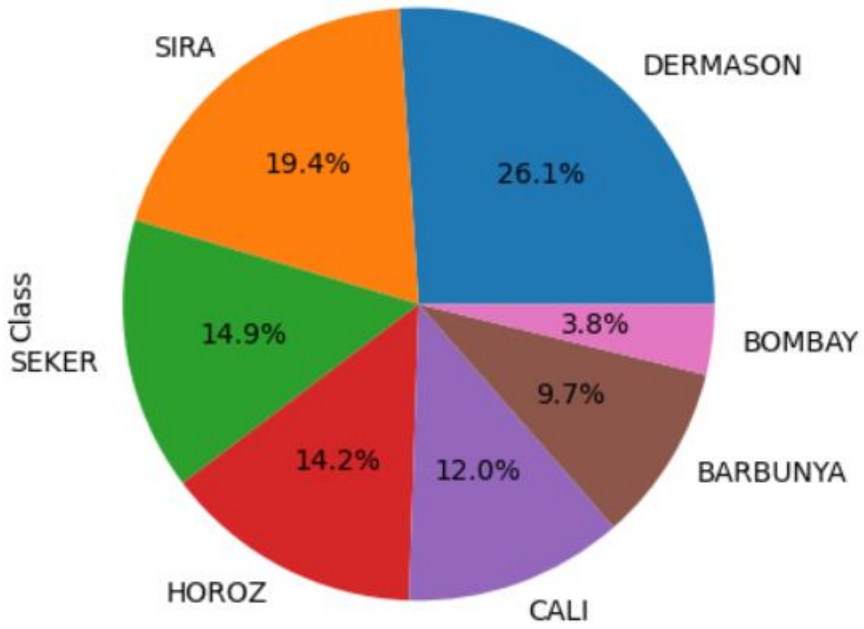
-0.23034832]

RSME values for each folds would be:

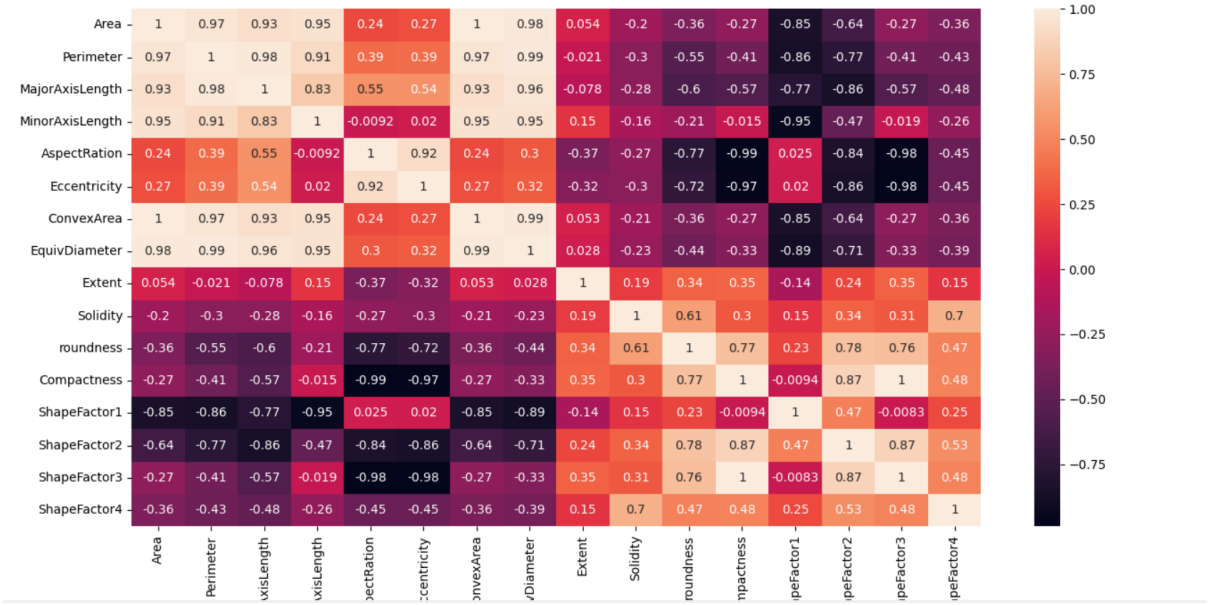
[2.2346785104201623, 2.9391598188023984, 2.4032491031713534, 3.651605347029729, 2.4531984753326914]

ANS 3:

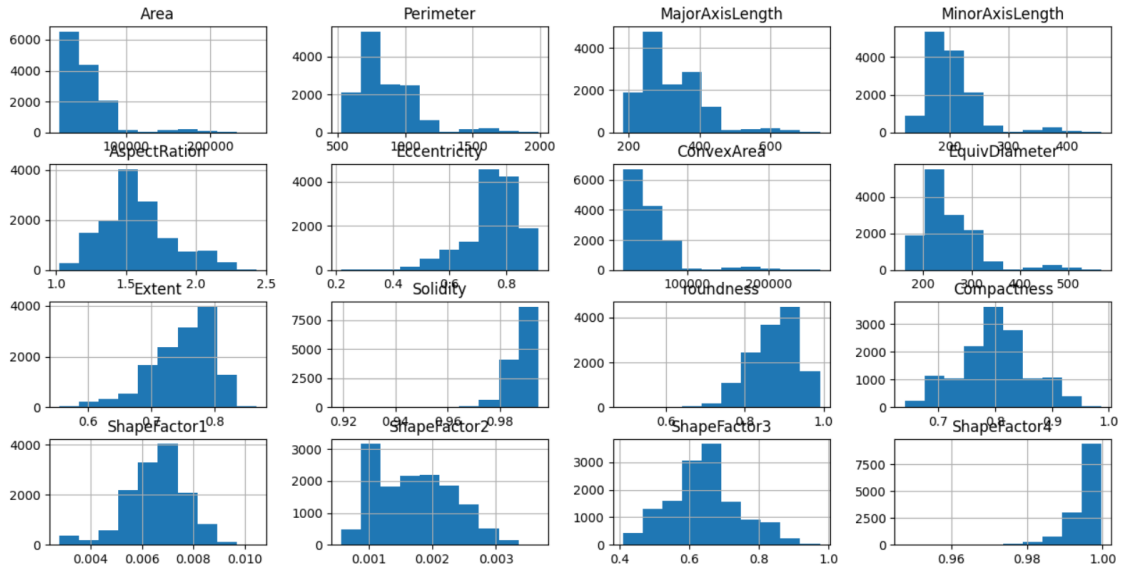
a. The class distribution of the data set is as follows:



b. Heat Map:

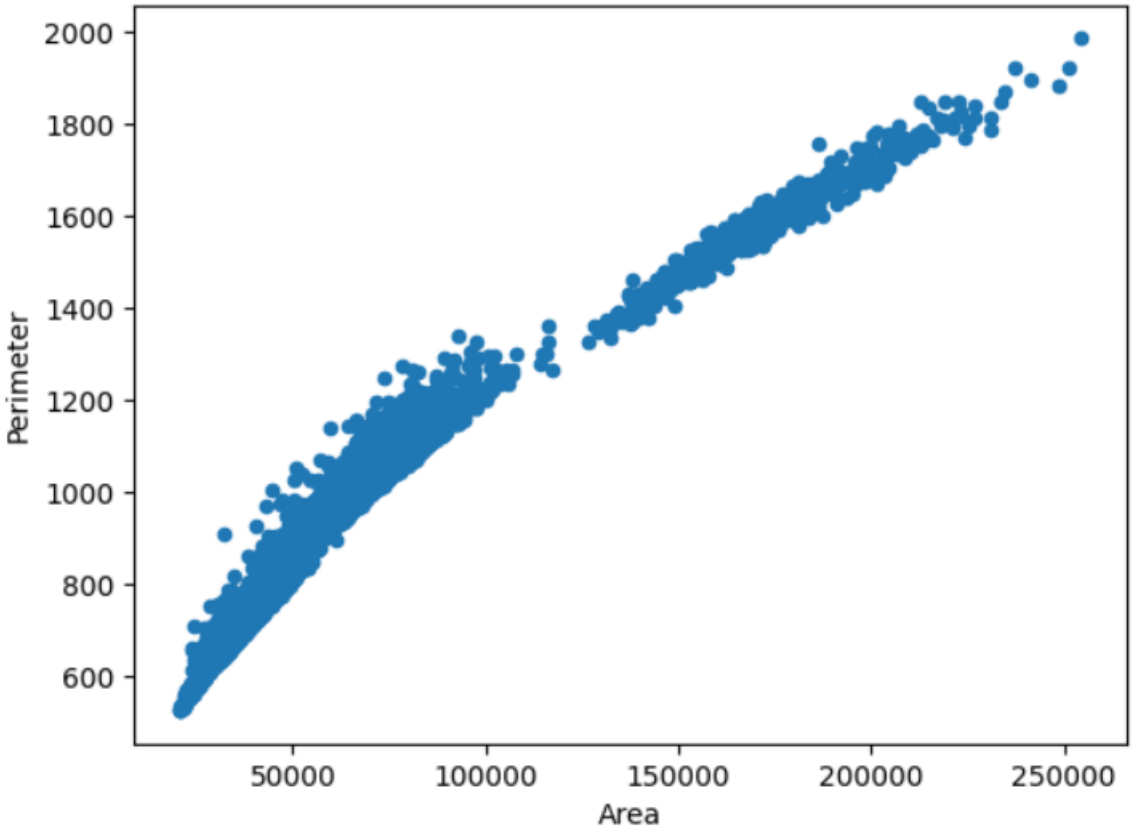


Histogram :

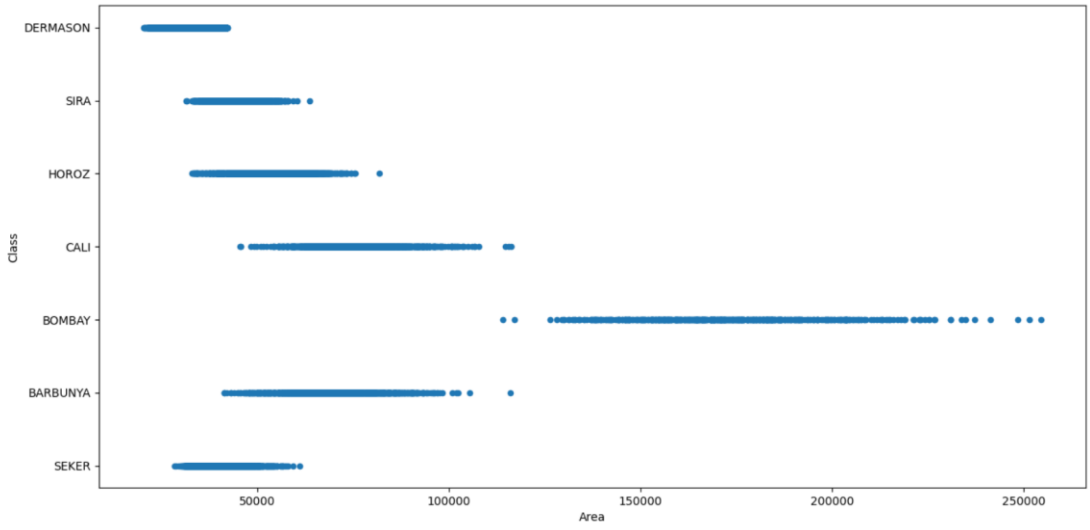


Scatter Plot :

Perimeter and Area:



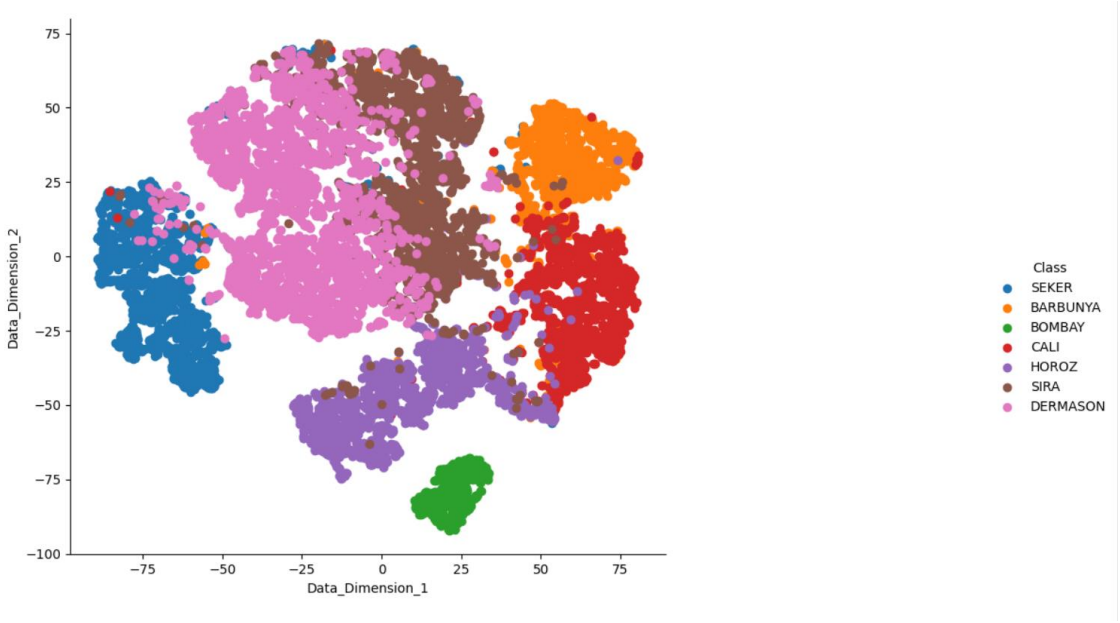
Area and Class:



Insights on the data set are:

1. For almost all the features most of the values of these features are within a fixed range.
2. Thus it is a type of continuous distribution and would give better results for those methods which involve continuous distribution.
3. There are 7 unique classes
4. Maximum area value is covered by BOMBAY whereas minimum is for DERMSON.
5. Perimeter and area can be related using linear regression as the line would cover most of the points as can be seen from the scatter plot.

c. The t-sne plot of the dataset when reduced to 2 dimensions have the following scatter plot:



The data classification for each of the class is mostly confined within a particular region for both the dimensions but there are also some values in it that are quite away from those classes region indicting the discretness nature of the data set.

d. The comparison in them can be made as follows:

Type Of Naïve Bayes	**Accuracy**	**Recall**	**Precision**
Gaussian	0.8968049944913699	0.8968049944913699	0.8974148189793245
Bernoulli	0.7264047006977599	0.7264047006977599	0.7365456835982626

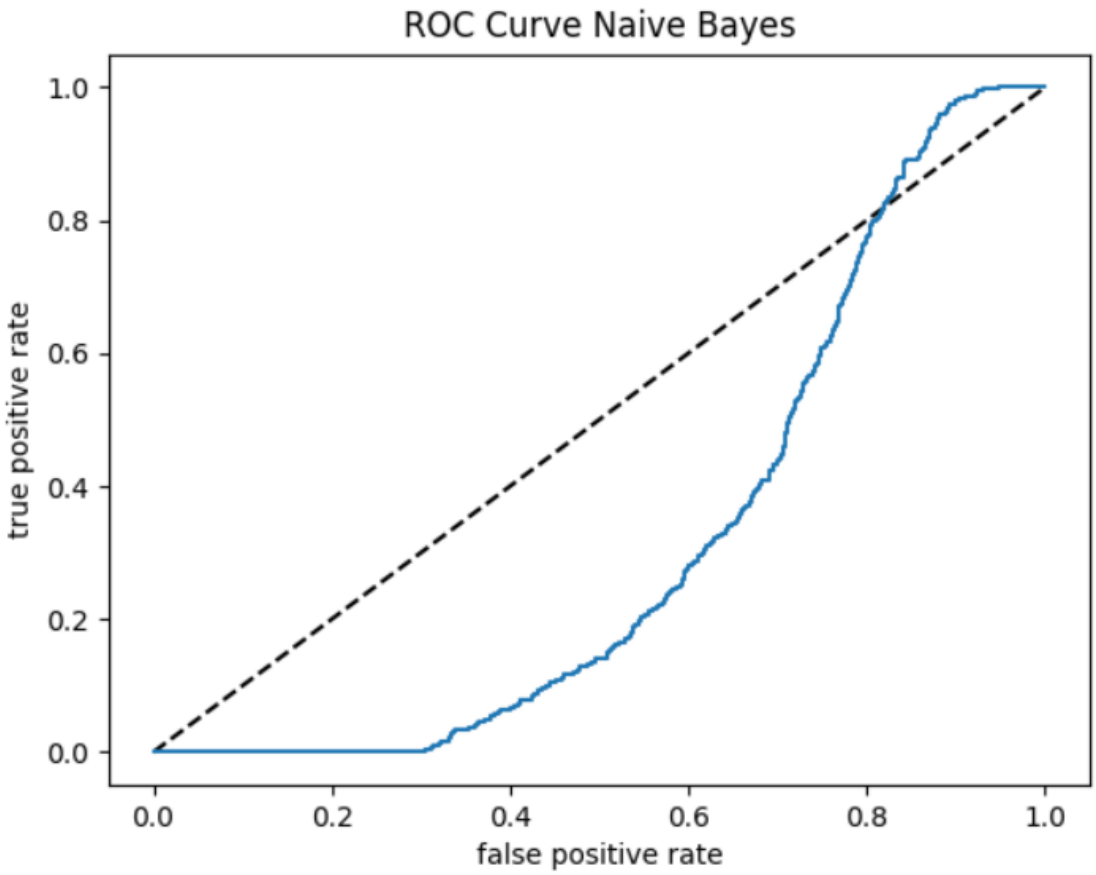
As from the above table we can see that there is difference in accuracy, recall and precision of gaussian and Bernoulli and for all of them gaussian have higher value than bernolli and there is

greater accuracy for gaussian than Bernoulli because Bernoulli assumes only two types of output however there are more than that.

e. The comparison between various components for PCA is as follows (up to 4 decimal places can be extended further also to get more precise result):

Number Of Components	Accuracy	Precision	Recall	F1-Score
4	0.8898	0.8891	0.8898	0.8893
6	0.9247	0.9253	0.9247	0.9247
8	0.9287	0.9293	0.9287	0.9288
10	0.9283	0.9290	0.9283	0.9284
12	0.9283	0.9290	0.9283	0.9284

f. The ROC-AUC curve using Naïve Bayes :



For a roc curve more the value closer to the left corner more better is the performance and more closer it is to the dotted line less accurate it starts becoming and from the above graph

it can be seen that when false positivity rate and true positivity rate is 0.8 then the model is least accurate.

g. When applied logistic regression we got the following result:
Accuracy: 0.9225119353654058

Recall: 0.9225119353654058

Precision: 0.9226384506773714

F1-Score: 0.9224744990982867

Training Method	Accuracy	Recall	Precision
Gaussian Naïve Bayes	0.8968049944913699	0.8968049944913699	0.8974148189793245
Bernoulli Naïve Bayes	0.7264047006977599	0.7264047006977599	0.7365456835982626
Logistic Regression	0.9192067572530297	0.9192067572530297	0.919390067903044

Comparing each of them we can see that
For all the three methods being compared logistic Regression has highest accuracy, recall and precision than gaussian followed by Bernoulli.