MACHINE LEARNING (CSE343/ECE343)
ASSIGNMENT-1
Sourav Goyal 2020341
SECTION-A

ANS - (a)

Assignment-1 (ML)

Section-A

A-① Given :- Model we have is linear regression,

To prove :- Least square fit line always passes through the point $(\bar{x}, \bar{y})$, where $\bar{x}$ and $\bar{y}$ represent the arithmetic mean of the independent variables and dependent variables respectively.

⇒ So, cost function for simple linear regression is :-

$$J(\omega) = \underset{\omega}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} (\omega_1 x_i - y_i + \omega_0)^2$$, where ω_0 is bias and ω_1 is feature weight

⇒ Now in least square fit our cost function will be minimum So, we will derivate our $J(\omega)$ and put 0 because we have to find minima, i.e.,

⇒ $$\frac{dJ(\omega)}{d\omega} = \frac{dJ(\omega)}{d\omega_1} + \frac{dJ(\omega)}{d\omega_0}$$

$\frac{dJ(\omega)}{d\omega_1} = 0$, $\frac{dJ(\omega)}{d\omega_0} = 0$

⇒ $$\frac{dJ(\omega)}{d\omega_1} = \frac{1}{n}\left(\sum_{i=1}^{n} (\omega_1 x_i - y_i + \omega_0) x_i\right) \times 2 = 0 \quad -①$$

⇒ $$\frac{d(J(\omega))}{d\omega_0} = \frac{1}{n} \times 2 \sum_{i=1}^{n} (\omega_1 x_i - y_i + \omega_0) = 0$$

⇒ $$\frac{dJ(\omega)}{d\omega_0} = \sum_{i=1}^{n} \omega_1 x_i - y_i + \sum_{i=1}^{n} \omega_0 = \sum_{i=1}^{n} \omega_1 x_i - y_i + n\omega_0 = 0$$

∴ $$\omega_0 = -\frac{1}{n} \sum_{i=1}^{n} \omega_1 x_i - y_i$$ and also from eq ① we again get

$$\omega_0 = -\frac{1}{n} \sum_{i=1}^{n} \omega_1 x_i - y_i$$

⇒ So, expanding above eqⁿ :-

⇒ $w_0 = -\frac{1}{n}(w_1(x_1+x_2---+x_n) + (-y_1-y_2---y_n))$

⇒ $w_0 = -(w_1\frac{(x_1+x_2---+x_n)}{n} + \frac{(-y_1-y_2---y_2)}{n})$

⇒ $w_0 = -(w_1\bar{X} - \bar{Y})$, where $\bar{X} = \frac{x_1+x_2+---+x_n}{n}$ and

$\bar{Y} = \frac{y_1+y_2+---+y_n}{n}$

⇒ $w_1\bar{X} + w_0 = \bar{Y}$.

⇒ $\bar{Y} = w_1\bar{X} + w_0$ — (2)

⇒ Now from hypothesis of linear regression we know,

$y_i = wx_i + w_0$ — (3)

⇒ Comparing eqⁿ (3) and (2), we get,

$\bar{X} = x_i$ and $\bar{Y} = y_i$.

⇒ Hence, eqⁿ (2) shows that least square fit line passes through co-ordinate $(\bar{X}, \bar{Y})$ and this is our desired result. (Hence Proved).

ANS - (b)

A-② Given:- We have 3 variables let's call it as x, y, z,
so from question we know that 2 variables are highly correlated with 3rd variable i.e., x and y are correlated with z.

⇒ So, now we have to check whether x and y also highly correlated or not.

⇒ We will denote correlation factor as ρ_{xy}, it tells the correlation between x and y.

⇒ Now, we start with a claim that x and y will not always highly correlated.

⇒ Formula of Partial correlation is $\rho_{xy \cdot z} = \frac{\rho_{xy} - \rho_{xz} \cdot \rho_{yz}}{\sqrt{1-\rho_{xz}^2}\sqrt{1-\rho_{yz}^2}}$

⇒ Simplify above formula,

⇒ $\rho_{xy} = \rho_{xy \cdot z}\left(\sqrt{1-\rho_{xz}^2} \times \sqrt{1-\rho_{yz}^2}\right) + \rho_{xz} \cdot \rho_{yz}$

⇒ Now the range of a correlation factor is -1 to 1,

⇒ So, $\rho_{xy} = \rho_{xz} \cdot \rho_{yz} \pm \left(\sqrt{1-\rho_{xz}^2}\right)\left(\sqrt{1-\rho_{yz}^2}\right)$

∴ We have a range for ρ_{xy}, ie,,

$$\rho_{xy} \in \left[\rho_{xz} \cdot \rho_{yz} - \left(\sqrt{1-\rho_{xz}^2}\right)\left(\sqrt{1-\rho_{yz}^2}\right),\ \rho_{xz} \cdot \rho_{yz} + \left(\sqrt{1-\rho_{xz}^2}\right)\left(\sqrt{1-\rho_{yz}^2}\right)\right]$$

⇒ Now we take an example,

⇒ x and z are highly correlated so we take $\rho_{xz} = 0.9$ and similarly for y and z we take $\rho_{yz} = 0.85$

⇒ $\rho_{xy} = [0.765 - (0.435)(0.5267),\ 0.765 + (0.435)(0.5267)]$

$$\Rightarrow \rho_{xy} = [0.535,\ 0.994]$$

∴ Now we can see that we have a lower bound of 0.535 which is moderate correlation.

∴ Our claim is correct that they will not always highly correlated

[Justified]

Ans - (c)

A-③ Given :- $X_1, X_2 \dots X_n$ are iid with finite mean μ and variance σ^2, i.e., $E(X_1) = E(X_2) = \dots = E(X_n) = \mu$ and $Var(X_1) = Var(X_2) = \dots = Var(X_n) = \sigma^2$.

To prove :- $\lim_{n\to\infty} M_n \to \mu$, where M_n is the sample mean and μ is our population mean, so $\lim_{n\to\infty} P(|M_n - \mu| \geq \varepsilon) \to 0$, for any $\varepsilon > 0$

so,

$$\Rightarrow E(M_n) = E\left(\frac{X_1 + X_2 \dots X_n}{n}\right) = \frac{E(X_1) + \dots E(X_n)}{n} \quad [\because X_1 \dots X_n \text{ are iid}]$$

$$\bar{M}_n = \frac{n\mu}{n} = \mu$$

$$\Rightarrow Var(M_n) = Var\left(\frac{X_1 + X_2 \dots X_n}{n}\right) = \frac{Var(X_1) + Var(X_2) \dots Var(X_n)}{n^2}$$

$$\Rightarrow Var(M_n) = \frac{\sigma^2 + \dots + \sigma^2}{n^2} = \frac{n\sigma^2}{n^2} = \frac{\sigma^2}{n}. \quad [\because \text{by iid's}]$$

$\Rightarrow$ Now by Chebyshev's inequality,

$$\Rightarrow P(|\bar{M}_n - \mu| \geq \varepsilon) \leq \frac{Var(M_n)}{\varepsilon^2}$$

$$\Rightarrow \text{So, } P(|\bar{M}_n - \mu| \geq \varepsilon) \leq \frac{\sigma^2}{n\varepsilon^2}$$

$\Rightarrow$ Now at $\lim_{n\to\infty}$,

$$\Rightarrow \lim_{n\to\infty} P(|\bar{M}_n - \mu| \geq \varepsilon) \leq \lim_{n\to\infty} \frac{\sigma^2}{n\varepsilon^2}$$

$\Rightarrow$ Now $\lim_{n\to\infty} \frac{1}{n} \to 0$

$\therefore P(|\bar{M}_n - \mu| \geq \varepsilon) \leq 0$, $\varepsilon \to 0$

$\Rightarrow$ So, this proves that at $n \to \infty$ the sample mean will equal to population mean because probability of > 0 will goes 0.

[Hence Proved]

Pseudo Code :-

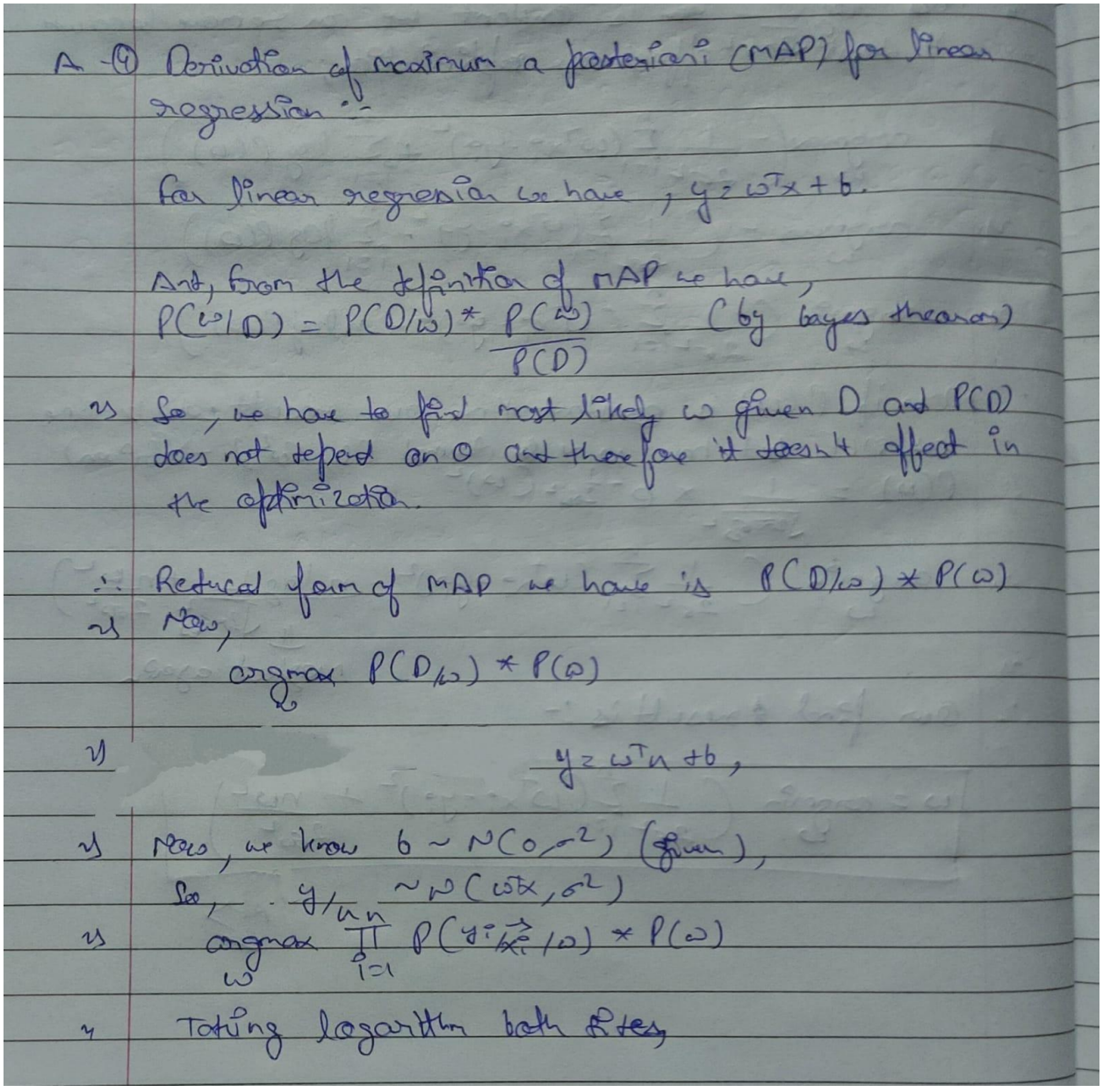
We will take a range of numbers from 0 to 1000 so our population mean will be 500;

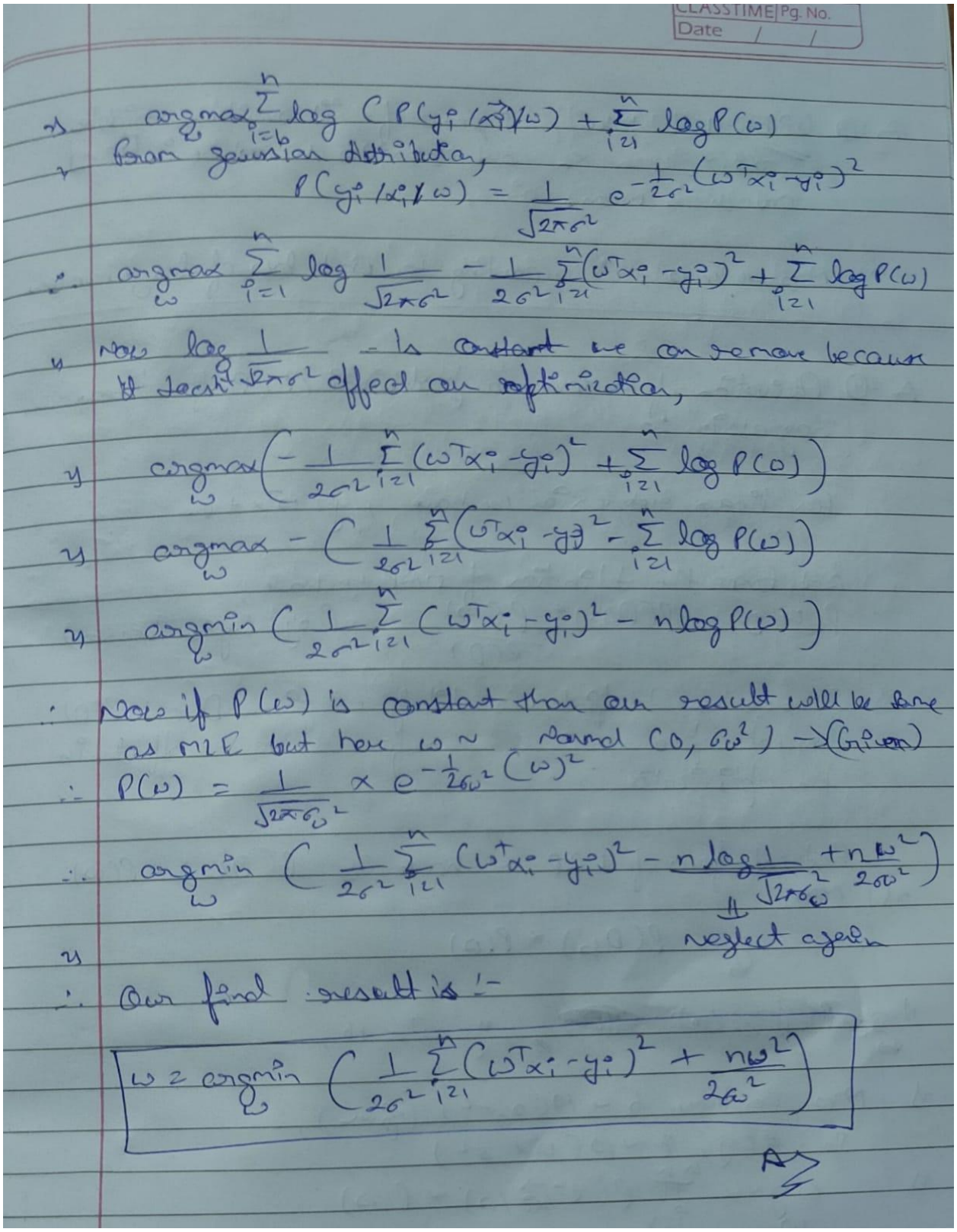
Arr = [] // this array will stores all the samples

For i in range 0 to 1000:
    For j in range (i,1000,3):
        if(array contains j): // if we already add a sample then we will not add it again
                               // so we will jump to next sample
            continue
        arr.append(j)
    A = Sample_mean(arr) // Sample_mean will be a function which gives the mean of samples currently we have in Arr.

If we compare A with population mean in 1 st iteration it will have a high difference but as our function works then we can see A will get closer to population mean that is 500 and after the program is over then our sample mean will be 500 exactly.

Ans - (d)





SECTION-B

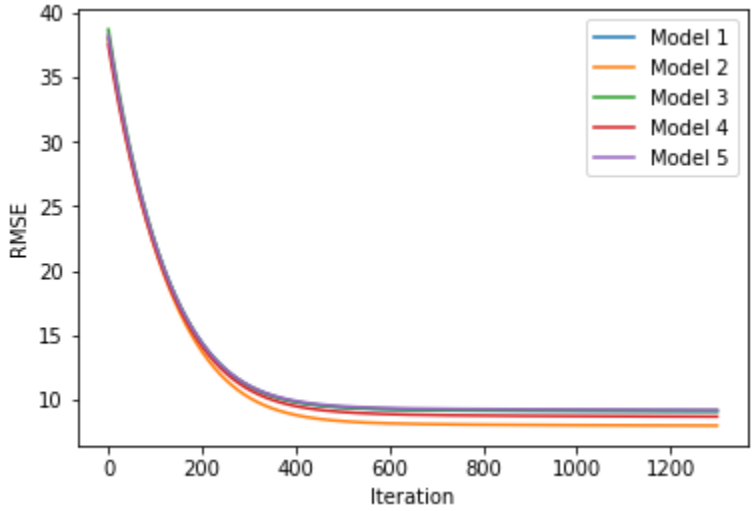
Ans - (a) Table of K value with their Mean RMSE :-

K value	Mean RMSE
2	8.909685034963294
3	8.882254992021346

4	8.84270311014231
5	8.729345090256059

Optimal value of K will be 5 because the mean rmse is lowest among all the mean rmse we get for other values of K.

(b) RMSE Vs Iteration plots for each model we get from Optimal value of K which is 5 :-



For training the models we use linear regression with learning rate = 0.006 and epoch = 1300.

Model Number	RMSE on Train Set	RMSE on Val set
1	9.074522911193775	7.835951654646455
2	7.962530168794994	11.747048375179345
3	9.111364677580731	7.506690888508186
4	8.667331486337238	9.461115568391167
5	9.188713471841982	7.095918964555147

Mean RMSE on Train Set = 8.800892543149743
Mean RMSE on Val set = 8.729345090256059

(c) For Lasso Regularization :-

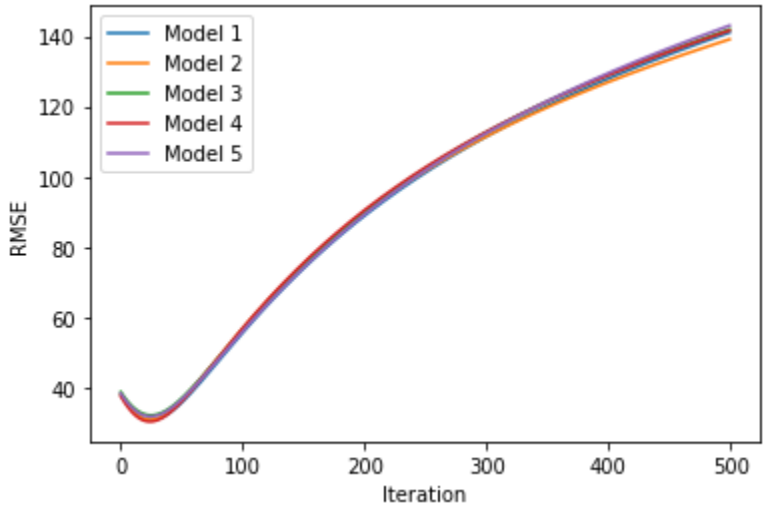
We Fix our learning rate = 0.006 and changes only epochs and lambda (Regularization Parameter)

S NO.	Lambda (Regularization Parameter)	Epochs	Mean RMSE On Train Set	Mean RMSE On Val Set
1	0.05	500	141.453963931	143.523323653

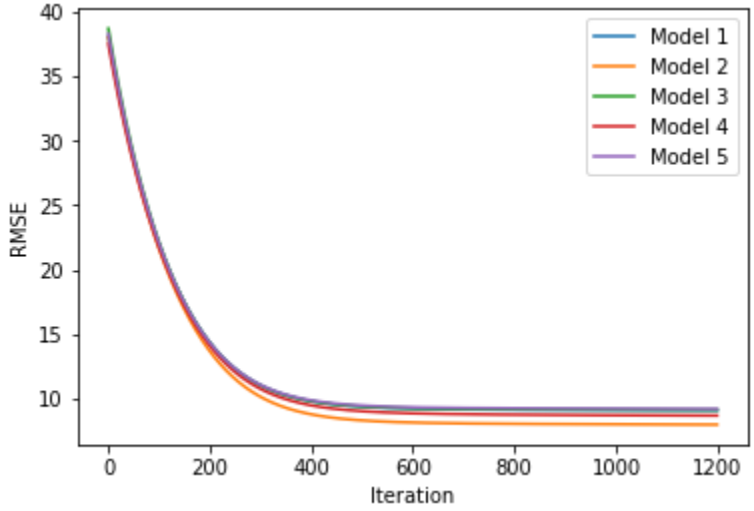
			5704	37116
2	0.0001	1200	8.825305076447638	8.754996554763232
3	1	200	1834.2628596901934	1843.3486039672 6
4	0.2	600	606.6086712571429	616.4692531121339
5	0.009	700	31.120614391436 2	31.763949388878838

Plot for Each Experiment :-

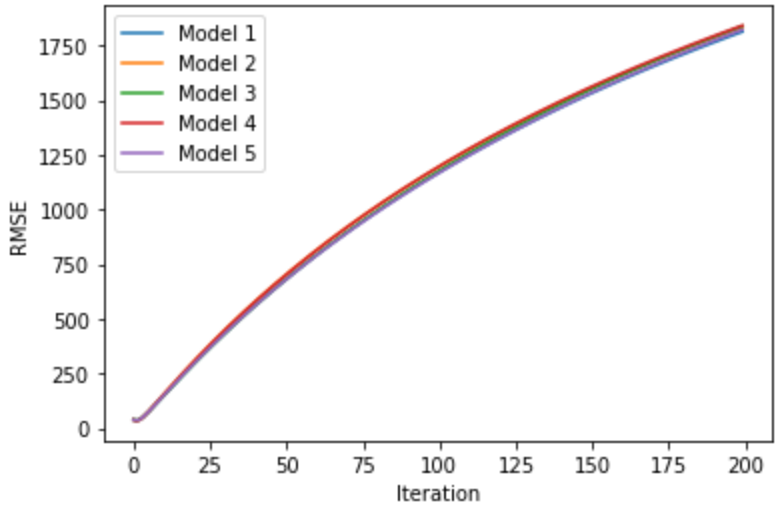
1.



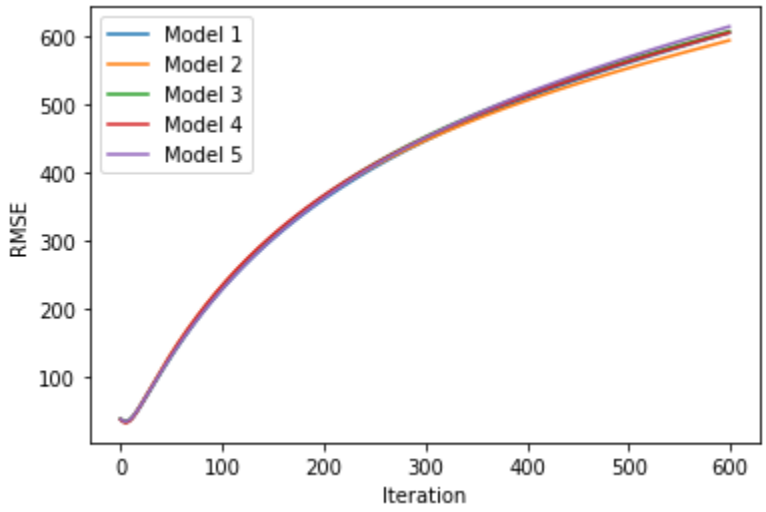
2.



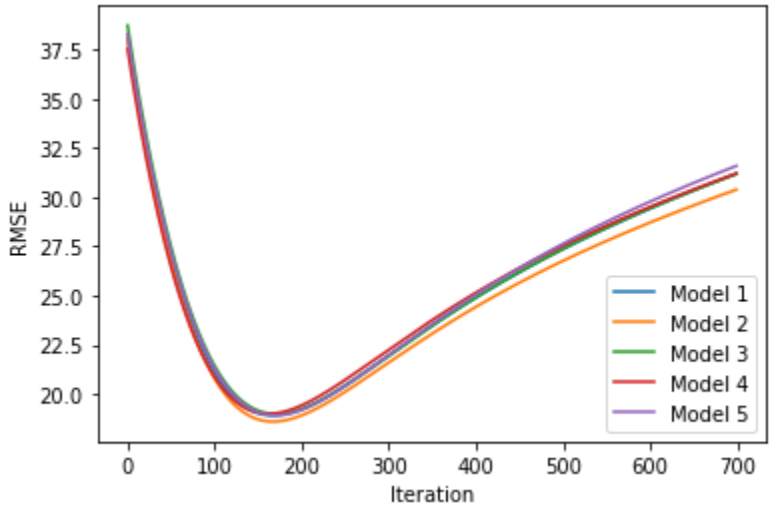
3.



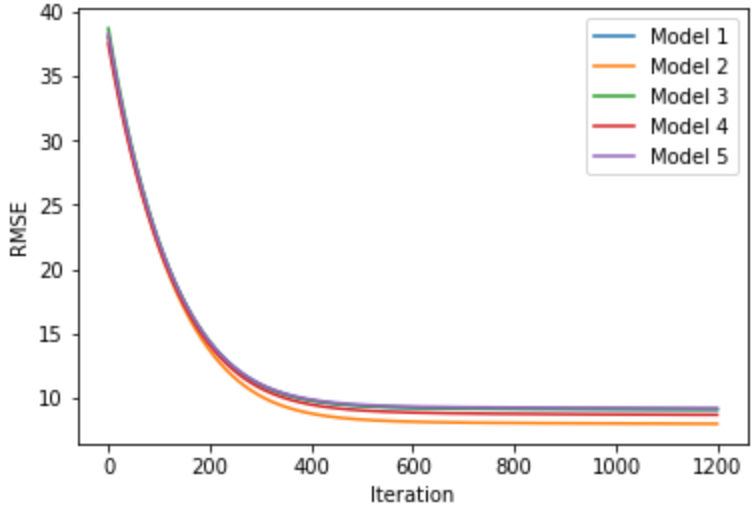
4.



5.



So we have an optimal solution for lambda = 0.0001 and RMSE on train set = 8.825305076447638 and RMSE on Val Set = 8.7549965547632
Plot for lambda = 0.0001 :-

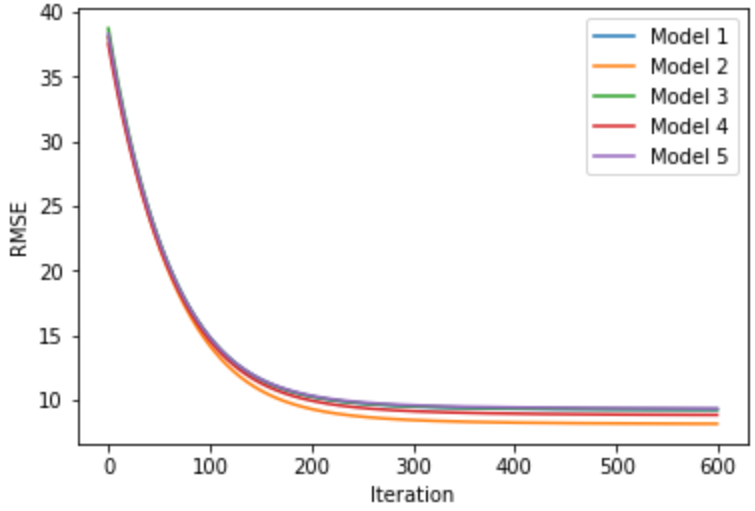


For Ridge Regularization :-

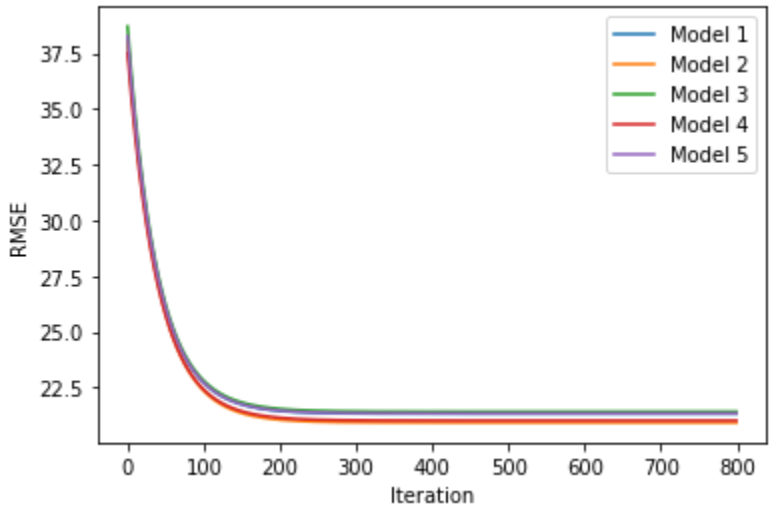
We Fix our learning rate = 0.006 and changes only epochs and lambda (Regularization Parameter)

S.No	Lambda (Regularization Parameter)	Epochs	Mean RMSE On Train Set	Mean RMSE On Val Set
1	0.05	600	8.99991898949606	8.934713062831266
2	1	800	21.200218711505393	21.200218711505393
3	0.0055	1350	8.768364520982155	8.695999747455391
4	10	500	36.596174083243611	36.624125777706649
5	5	1000	33.596786967723191	33.648765481203924

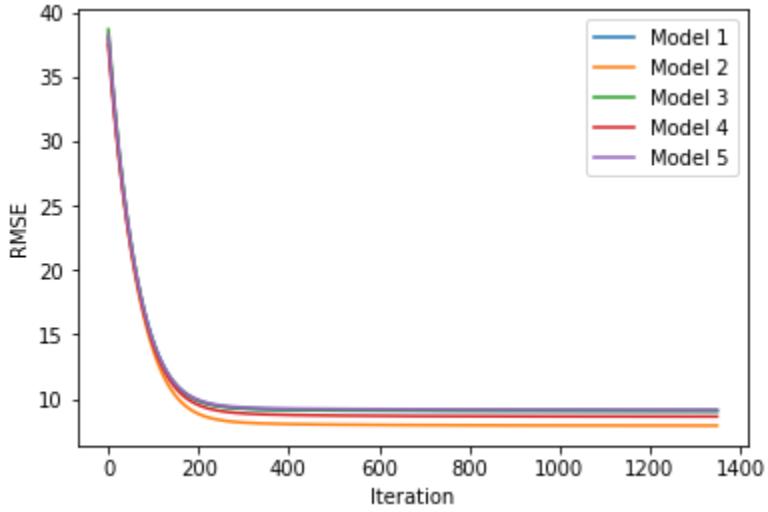
Plot for Each Experiment :-
1.


Model 1
Model 2
Model 3
Model 4
Model 5
RMSE
Iteration
40
35
30
25
20
15
10
0
100
200
300
400
500
600

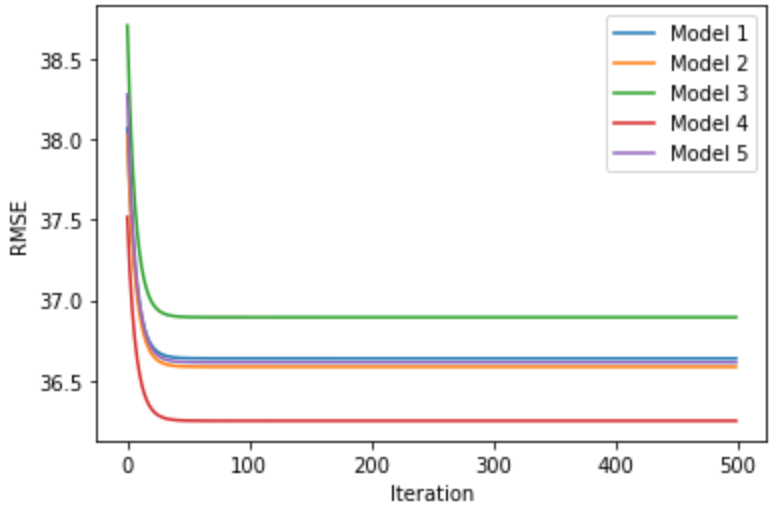
2.


Model 1
Model 2
Model 3
Model 4
Model 5
RMSE
Iteration
37.5
35.0
32.5
30.0
27.5
25.0
22.5
0
100
200
300
400
500
600
700
800

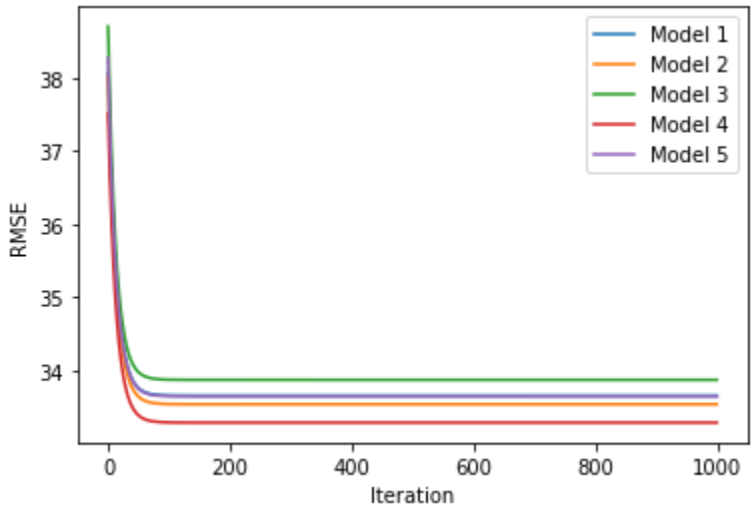
3.


Model 1
Model 2
Model 3
Model 4
Model 5
RMSE
Iteration
40
35
30
25
20
15
10
0
200
400
600
800
1000
1200
1400

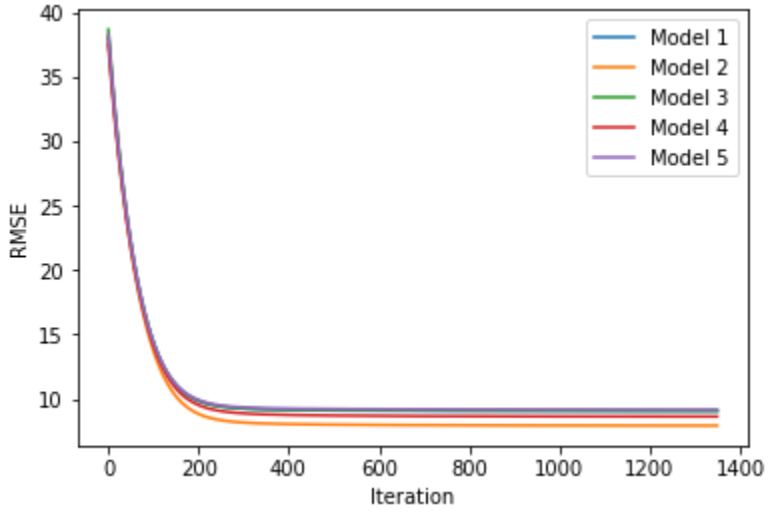
4.



5.



So we have an optimal solution for lambda = 0.0055 and RMSE on train set = 8.768364520982155 and RMSE on Val Set = 8.69599974745391
Plot for lambda = 0.0055 :-



(d) RMSE for each fold by normal equation :-

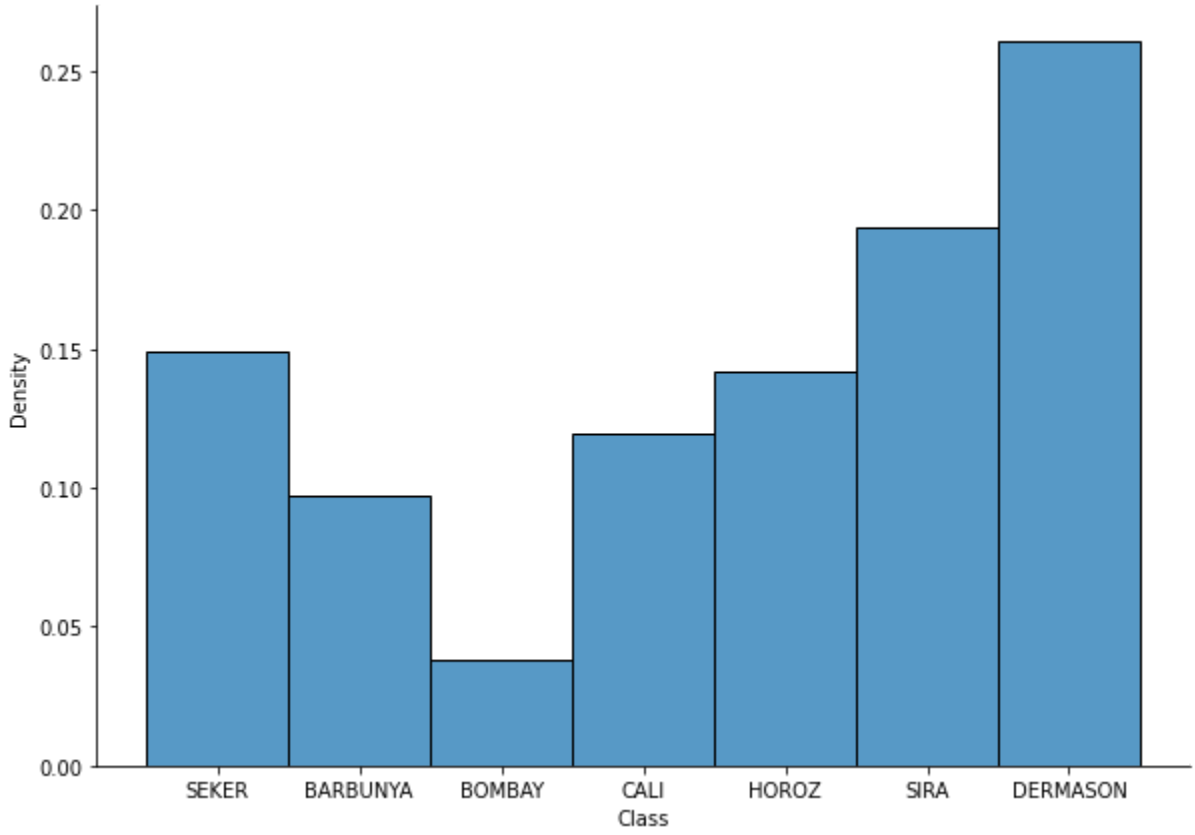
Model Number	RMSE

1	7.707739342790347
2	11.65785008339692
3	7.549243481223661
4	9.434235638731744
5	7.101061841935251

Mean RMSE From Normal Equation = 8.690026077615583

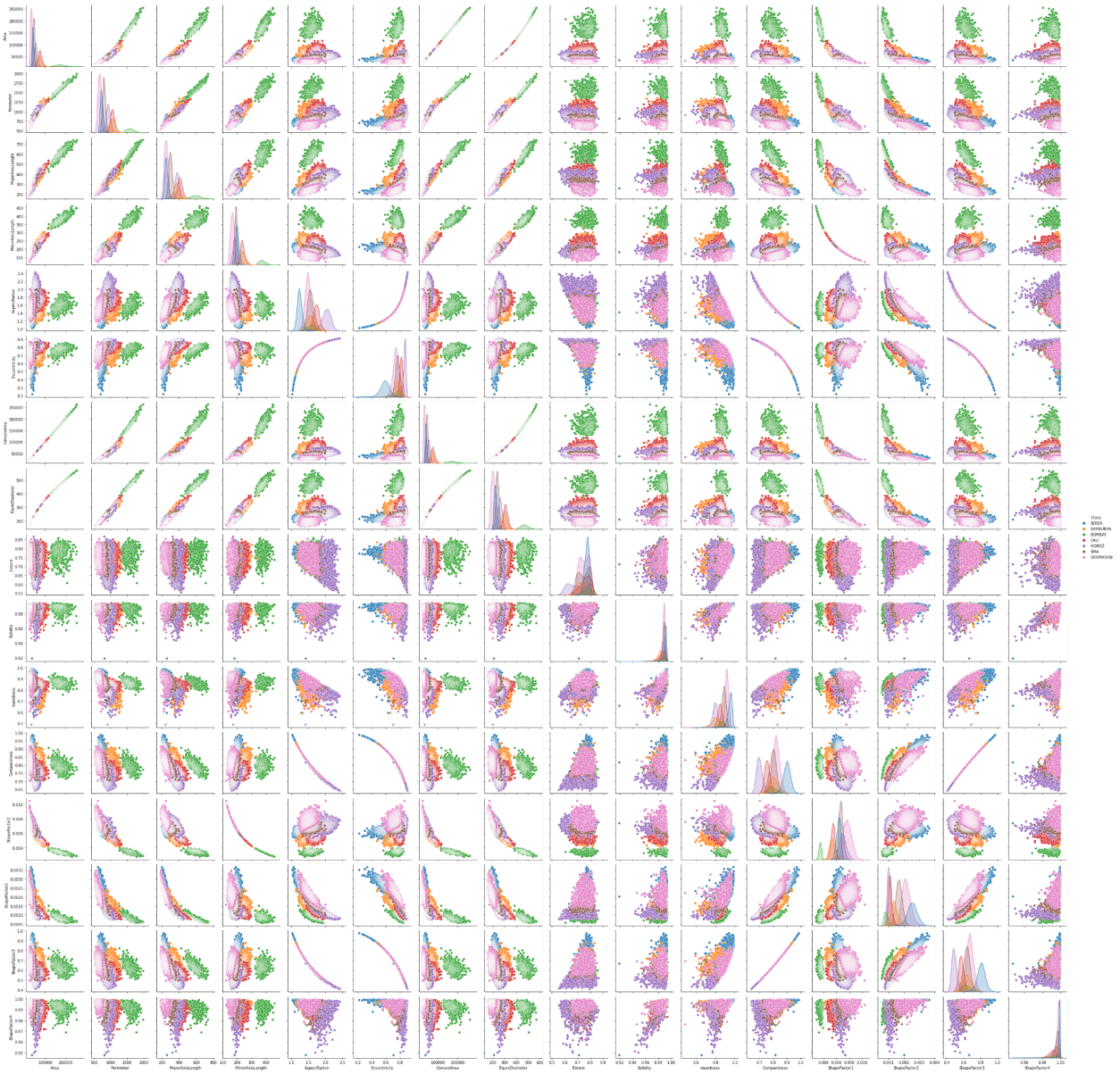
SECTION-C

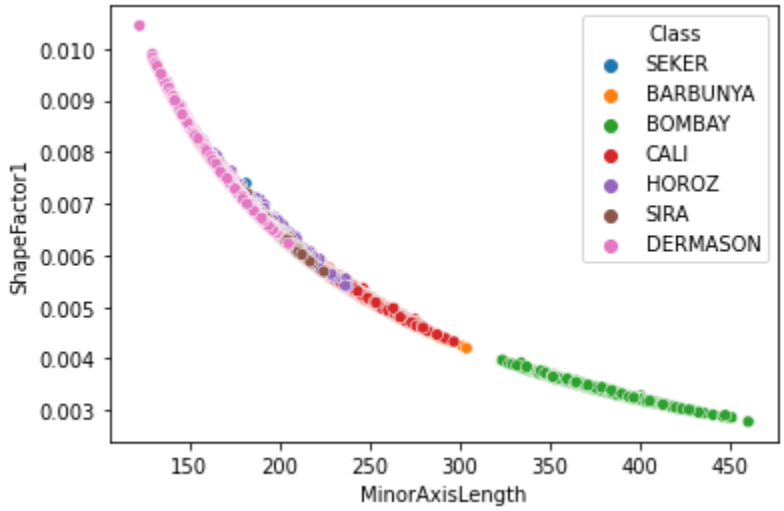
(a) Plot for Distribution which we get :-

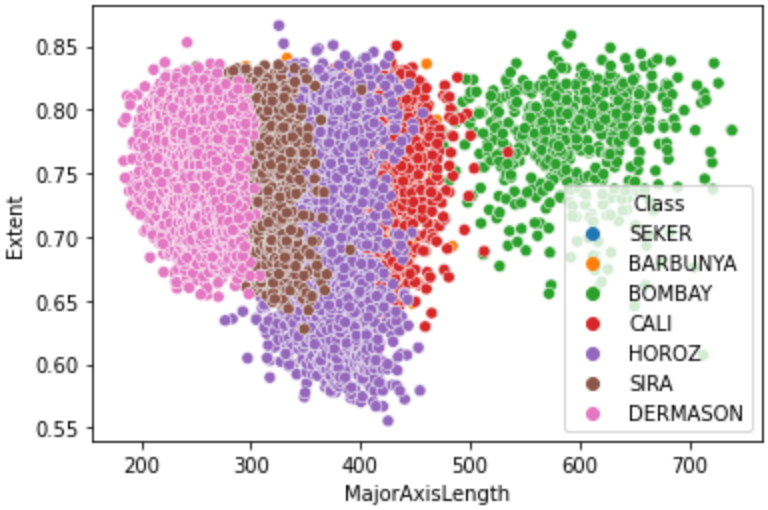


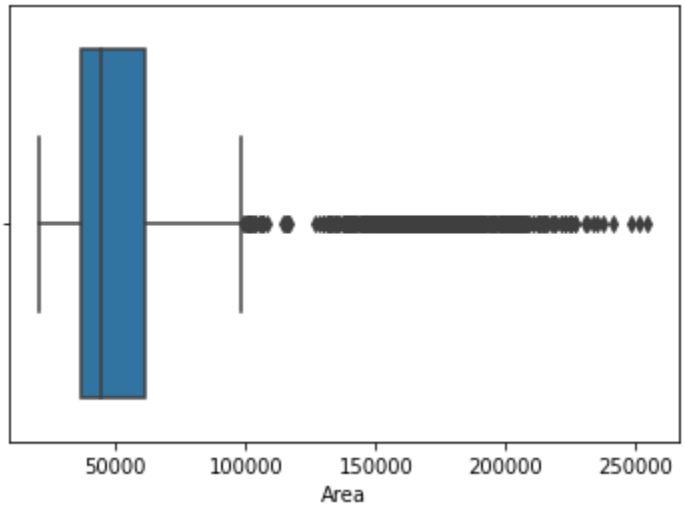
Analysis :- Demarson Class has the highest density then all other classes and Bombay class has the lowest density among all classes.

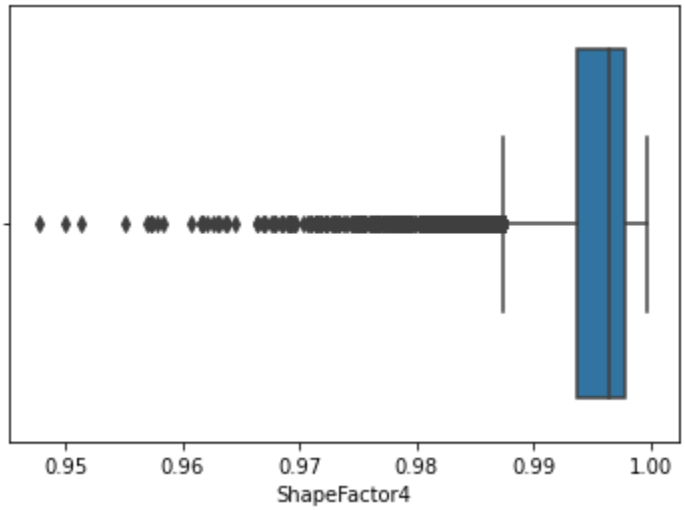
(b) After Plotting different types of graph like :-

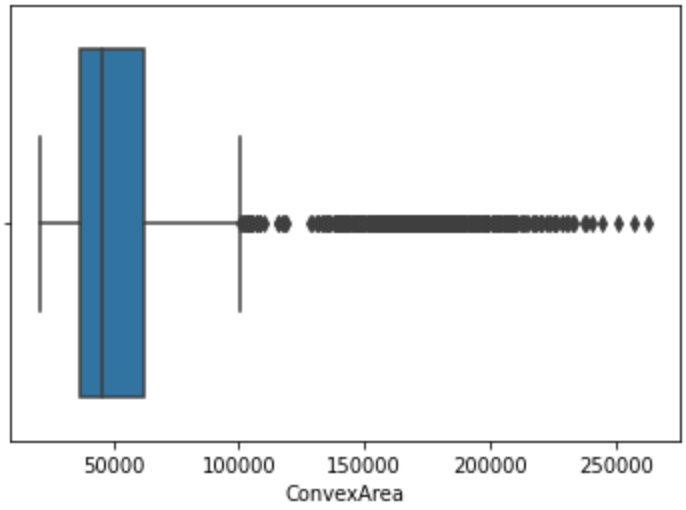


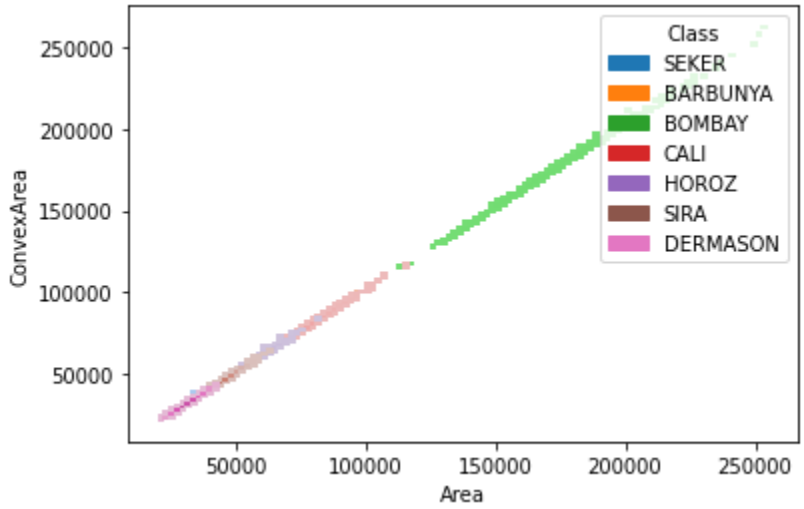

Class
SEKER
BARBUNYA
BOMBAY
CALI
HOROZ
SIRA
DERMASON
ShapeFactor1
MinorAxisLength
0.010
0.009
0.008
0.007
0.006
0.005
0.004
0.003
150
200
250
300
350
400
450


Extent
MajorAxisLength
Class
SEKER
BARBUNYA
BOMBAY
CALI
HOROZ
SIRA
DERMASON


Area


ShapeFactor4



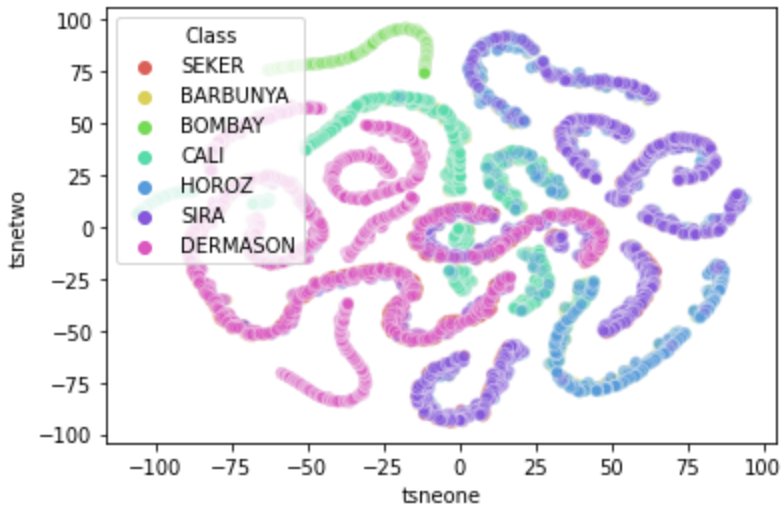


Five insights on the data :-

1. Bombay Class parameters are mostly non overlapping with other classes which means it is easy to classify from other classes.
2. There are many attributes which have a relation as we can see in the graph that some parameters make a linear relation some make exponential and etc.
3. There are many attributes which have many outliers which can affect our prediction.
4. Highest standard deviation seen in Convex Area.
5. ConvexArea and Area are the two parameters which have the bulky data in terms of their value.

No missing value has been reported when we check for missing values in data.

(c) After using TSNE algorithm on our data we have a plot like this :-



Analysis on Separability of the data:-
Only Bombay Class has an non overlapping distribution with other classes otherwise all other classes are overlapped with each other.

(d) We chooses the GaussianNB and BernoulliNB for implementing Naive Bayes :-

Recall for GaussianNB = 0.8968049944913699
Precision for GaussianNB = 0.8968049944913699
Accuracy for GaussianNB = 0.8968049944913699

Recall for BernoulliNB = 0.7245684906353287
Precision for BernoulliNB = 0.7245684906353287
Accuracy for BernoulliNB = 0.7245684906353287

GaussianNB has a higher score than BernoulliNB in all three types of score which shows that GaussianNB naive bayes model will perform better than BernoulliNB naive bayes model for a similar type of data.

(e) We uses GaussianNB naive bayes model for calculating the scores after applying PCA with number of components = 4,6,8,10,12

Accuracy for PCA with number of components 4 = 0.8670583914799853
Recall for PCA with number of components 4 = 0.8670583914799853
Precision for PCA with number of components 4 = 0.8670583914799853
F1 score for PCA with number of components 4 = 0.8670583914799853

Accuracy for PCA with number of components 6 = 0.9008446566287184
Recall for PCA with number of components 6 = 0.9008446566287184
Precision for PCA with number of components 6 = 0.9008446566287184
F1 score for PCA with number of components 6 = 0.9008446566287184

Accuracy for PCA with number of components 8 = 0.9030481087036357
Recall for PCA with number of components 8 = 0.9030481087036357
Precision for PCA with number of components 8 = 0.9030481087036357
F1 score for PCA with number of components 8 = 0.9030481087036357

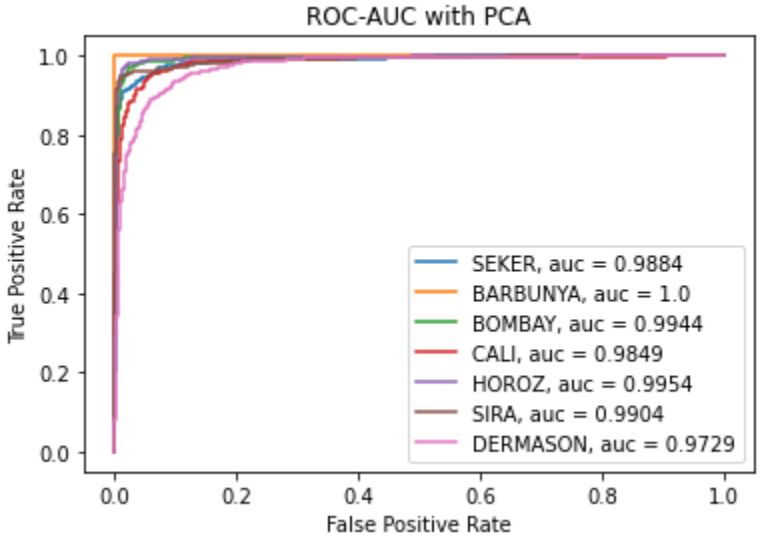
Accuracy for PCA with number of components 10 = 0.8879911861917004
Recall for PCA with number of components 10 = 0.8879911861917004

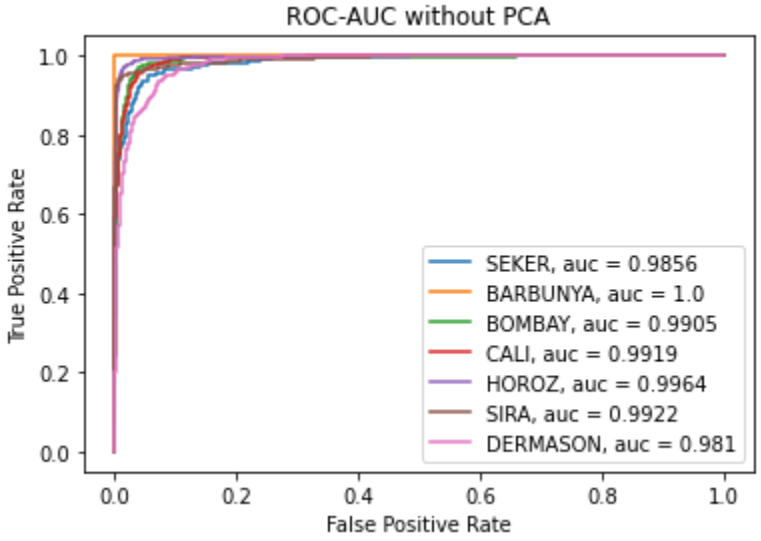
Precision for PCA with number of components 10 = 0.8879911861917004
F1 score for PCA with number of components 10 = 0.8879911861917004

Accuracy for PCA with number of components 12 = 0.8872567021667279
Recall for PCA with number of components 12 = 0.8872567021667279
Precision for PCA with number of components 12 = 0.8872567021667279
F1 score for PCA with number of components 12 = 0.8872567021667279

When we compare the scores which we get from all the models after applying different PCA values, we can see that for number of components = 8 we have the highest score in all four types of score from others which shows that we have better performance if we use PCA with 8 components.

(f)





From both the roc-auc curve (with pca and without pca) we can see that BARBUNYA Class has the highest true positive rate.

(g) We will select only those features which hasn't any direct relationship between them like in (b) part we see that there are many parameters which are dependent linearly or exponentially or other types of relation they have so we will remove all that attributes from the data, while training and testing our Logistic Regression Model.

Logistic Model Score = 0.9280205655526992
Naive Model Score = 0.8968049944913699

We can see that the Logistic Model performs better after removing unnecessary features from the data than the Naive Bayes Model.

Pair plot after removing dependent Features :-