Title: (Kmean and Clustering)

Purpose: *"This assignment helped us study the application of Kmean and Clustering, Hierarchical Clustering techniques."*

Dataset(s): **Forest type mapping Data Set**

(https://archive.ics.uci.edu/ml/datasets/Forest+type+mapping**)**

Approach:

- *We loaded the* **Forest type mapping Data Set**
- *first,* **we combine two data train and test together,** *and we converted the data into data frame removing the first column of the data.*
- *Thereafter, we removed the quality attribute as we are supposed to do clustering independent of that value. And drop the other columns except b1 to b9*
- *We made two copies of the data : one is scaled and the other is a non scaled version of the same.*
- *We select K=4 to apply k-means clustering and the value of* `between_SS / total_SS is 62.5%`
- If we applied scaling with K=4 the value of between_SS / total_SS = 63.6 %
- We also implement elbow method for K=2 to K=15(See graph 2)
- To see how many dimensions we really need, we apply PCA to scaled data(see graph 2). It seems 5 PCs are enough
- We then select 5 PCs to apply K-means clustering(K=4) and the value of between_SS / total_SS is 65.4% which is a little bit better.

- If we set seed set.seed(200) and just use two column b2 and b3 ,the value of betwee n_SS / total_SS is 89.4 %
  If we set set.seed(200) and just use two column b5 and b6 , the value of between_SS / total_SS is 88.0 %)
  We also applied hierarchical clustering on the data, from the graph, we determined k =4. For two columns 2 and 3.

we select 5 PCs to apply H cluster for complete, average and single

```
> table(cutree(full.complete,4), Data[,1])
```

	d	h	o	s
1	146	86	14	195
2	13	0	55	0
3	0	0	13	0
4	0	0	1	0

> table(cutree(full.average,4), Data[,1])

	d	h	o	s
1	148	86	13	195
2	11	0	68	0
3	0	0	1	0
4	0	0	1	0

> table(cutree(full.single,4), Data[,1])

	d	h	o	s
1	159	86	79	195
2	0	0	1	0
3	0	0	2	0
4	0	0	1	0

We also use original data to implement H cluster (see graph 3)

```
> table(cutree(full.complete,4), Data[,1])

      d   h   o   s
  1 152  86  15 195
  2   7   0  63   0
  3   0   0   4   0
  4   0   0   1   0
> table(cutree(full.average,4), Data[,1])

      d   h   o   s
  1 155  86  14 195
  2   4   0  67   0
  3   0   0   1   0
  4   0   0   1   0
> table(cutree(full.single,4), Data[,1])

      d   h   o   s
  1 159  86  79 195
  2   0   0   1   0
  3   0   0   2   0
  4   0   0   1   0



With fewer vriable we have
> full.average.restr = hclust(dist(full_scale[,2:3]), method="average")
> table(cutree(full.average.restr,3), Data[,1])

```

- `    d   h   o   s`
- `1 142  86  29 195`
- `2  17   0  53   0`
- `3   0   0   1   0`
-
- A little bit better , but not good
-
-
- With some method we can see number of cluster in hcluster, graph 4
-

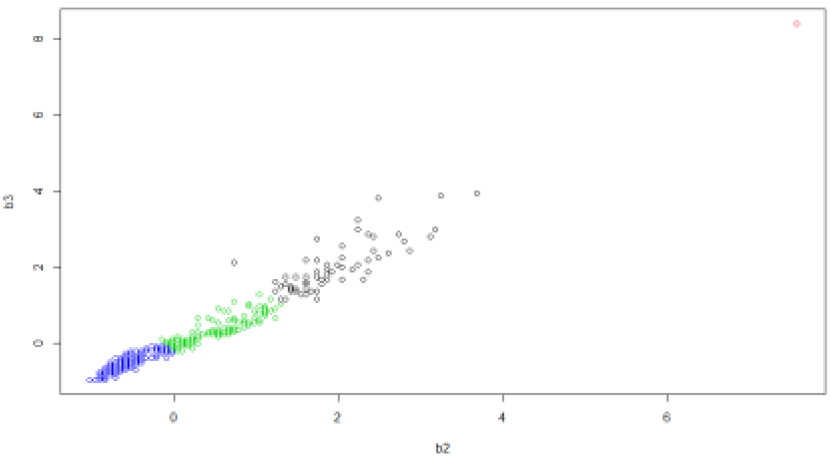
With some other model, we check to see what is good number for clustering, model like Model Based

With this model we check several graphs, graph 5,6,7

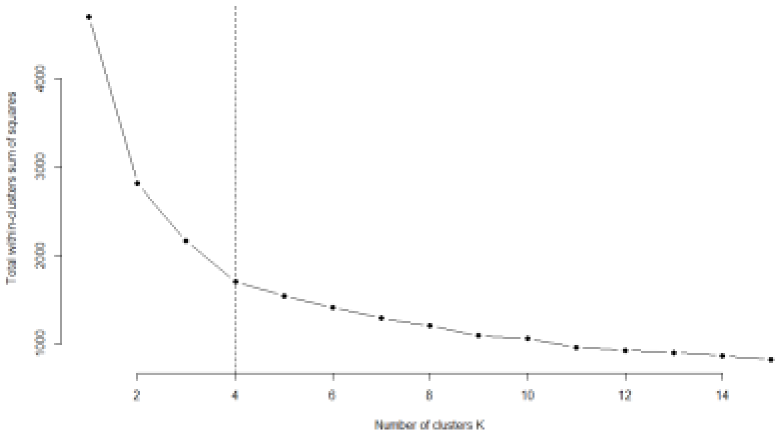
Graphs:

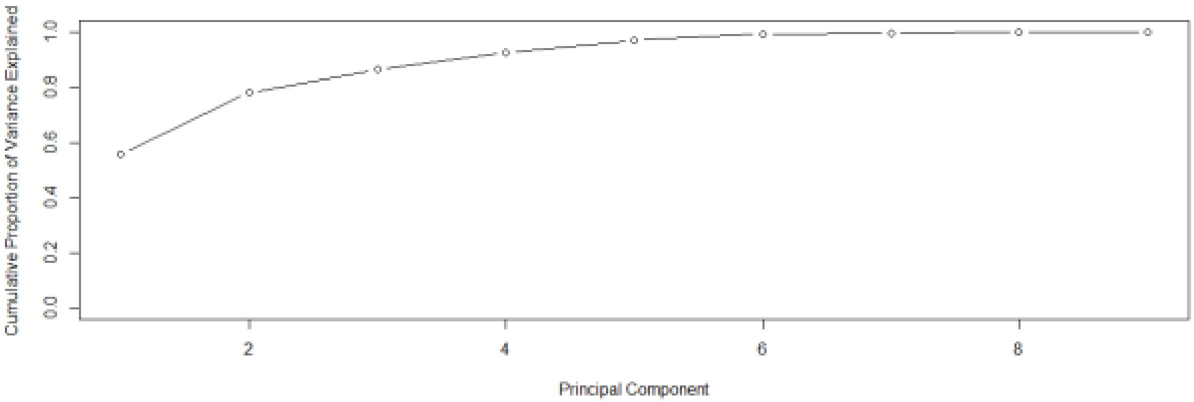
```
> plot(full_scale[,c(2,3)], col=Cluster$cluster)
```

Graph 1

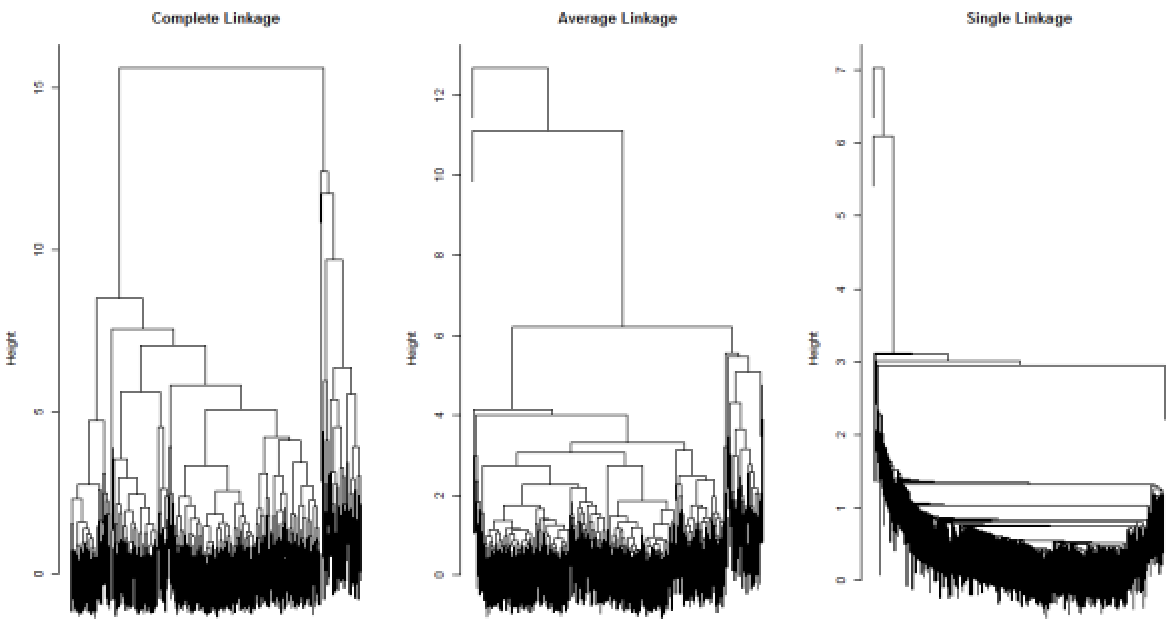


Graph 2


Total within-clusters sum of squares
Number of clusters K

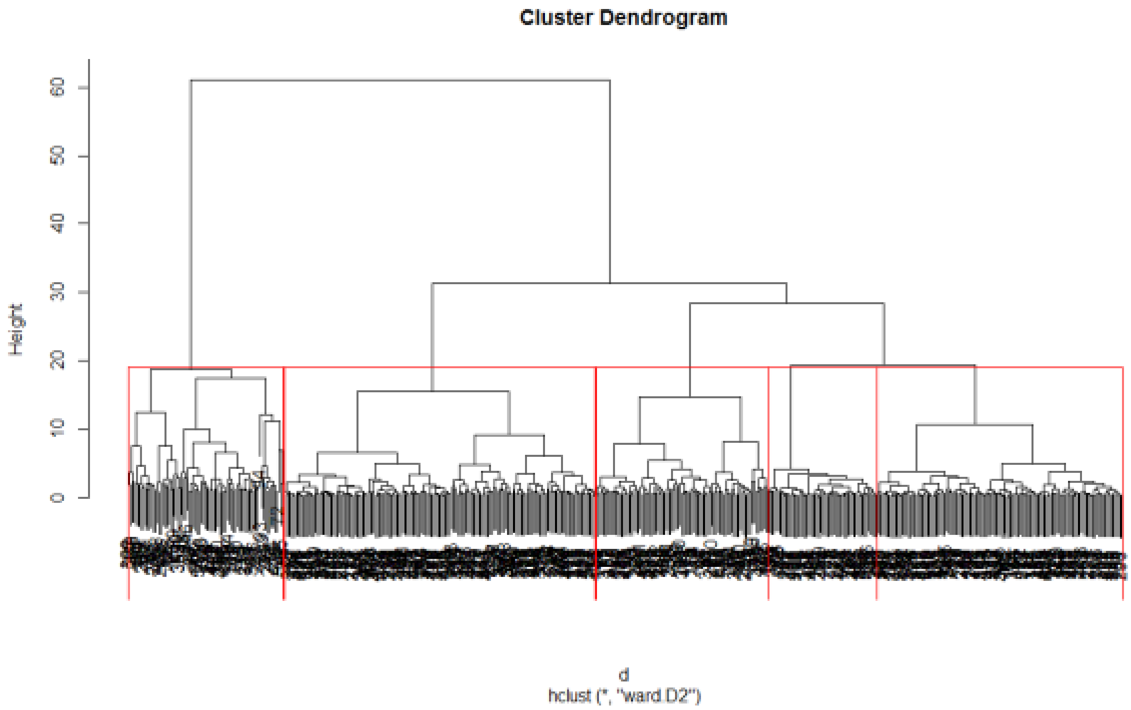

Cumulative Proportion of Variance Explained
Principal Component

Graph 3

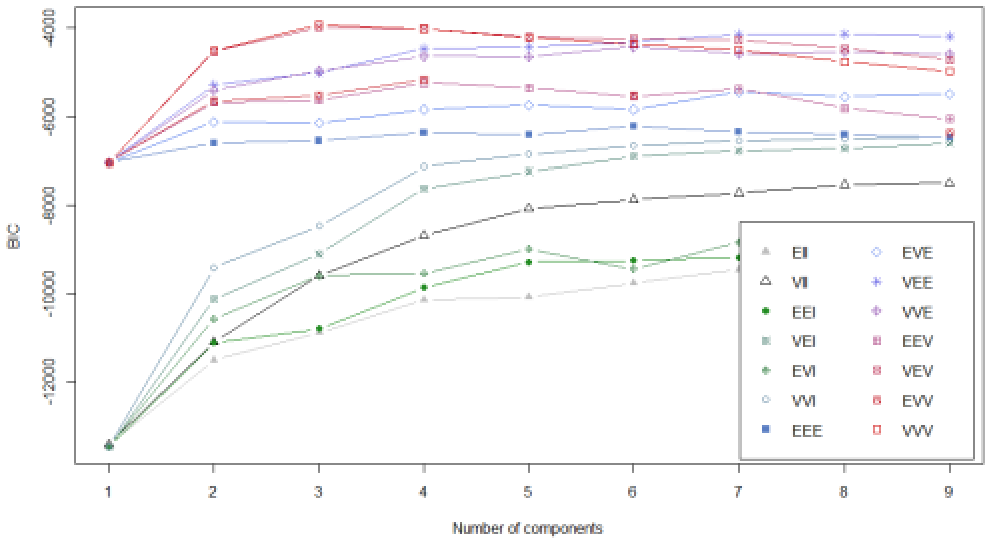


Graph 4

```
# Ward Hierarchical Clustering

d <- dist(full_scale, method = "euclidean") # distance matrix

fit <- hclust(d, method="ward.D2")

plot(fit) # display dendogram

groups <- cutree(fit, k=5) # cut tree into 5 clusters

# draw dendogram with red borders around the 5 clusters

rect.hclust(fit, k=5, border="red")
```



Graph 5, BIC



Summary:

If we use raw data(column b2 to b9) to implement K means, the value of between_SS / total_SS is 62.5%. If we use scaled data or 5 PCs, the value of between_SS / total_SS can be increased to 63.6% and 65.4%. Furthermore, if we use two columns (b2 and b3 or b5 and b6), between_SS / total_SS can be improved a lot to 89.4% or 88%. For H cluster, selecting b2 and b3 can still be a better result.

Part II - Title: (Regression)

Purpose: *"This assignment helped us understand the ridge and lasso regression along with PCR and KNN applied on the dataset. According to our initial hypothesis, as the racial match 'RacialMatchCommPol' between a community's police force and its population decreased, it led to an increase in magnitude of crimes in that society ."*

Dataset(s): **Communities and Crime Unnormalized Data Set**

(http://archive.ics.uci.edu/ml/datasets/communities+and+crime+unnormalized**)**

Approach:

Cleaning

- *First we add the column names to the data set taken from the above link as this helps in analyzing the dataset in a better fashion.*
- *Since the dataset was filled with missing data in form of ?, we clean the dataset by first converting all ? to NAs*
- *The explanatory variable for our dataset was RacialMatchCommPol (racial match between community and police), we removed every row in the data set for which the RacialMatchCommPol value for that observation was NA.*
- *The columns 'countyCode' and 'communityCode' contained mostly NAs and were therefore removed along with 'communityName', 'LemasGangUnitDeploy' and 'state'.*
- *We do an na.omit to remove all NAs from the dataset and convert certain columns to numeric before finally proceeding with the algorithm.*
- *Since our predictor variable is 'ViolentCrimesPerPop' we remove the other 17 predictor variables from the dataset.*

Ridge and Lasso Regression

- *Ridge: We use the grid method to try out various values of lambda and then find the best lambda using cross validation which turns out to be 3305.082.*

```
> bestlam=cv.out$lambda.min
> bestlam
[1] 3305.082
```

- *Setting lambda to 'bestlambda', we calculate the Mean Squared Error which turns out to be 232858.9. Since Ridge Regression takes all coefficients into factor, we see a non-zero value for each coefficient.* We get a high percentage of error around 47%.

```
> ridge.model=glmnet(x,y,alpha=0)
> predict(ridge.model,type="coefficients",s=bestlam)[1:20,]
  (Intercept)          fold    population householdsize  racepctblack
 8.853977e+02 -6.313380e-02 -5.481326e-07 -3.551130e-01 -3.848697e-02
 racePctWhite  racePctAsian   racePctHisp   agePct12t21   agePct12t29
 3.407513e-02  8.153112e-03  1.641083e-02 -7.594082e-02 -6.730975e-03
  agePct16t24    agePct65up     numbUrban      pctUrban     medIncome
-5.008173e-02 -2.896850e-03 -5.465096e-07  3.325066e-02  2.128765e-05
     pctWWage  pctWFarmSelf    pctWInvInc    pctWSocSec    pctWPubAsst
 3.489031e-02  1.808464e+00  3.164001e-02 -1.394426e-02 -6.657406e-02
```

```
> ridge.pred=predict(ridge.mod,s=bestlam,newx=x[test,])
> mean((ridge.pred-y.test)^2)
[1] 232858.9
> sqrt(mean((ridge.pred-y.test)^2))/mean(y.test)
[1] 0.470786
```

- *Lasso: Setting lambda to bestlambda, we calculate the Mean Squared Error which turns out to be 180634.9 (slightly better than Ridge). Lasso takes only finite coefficients into factor and drives the rest of the coefficients to zero. In this case only the 'Racepctblack' and 'pctWFarmSelf' seem to contribute towards the factors with their coefficient values shown below. Ridge regression uses the l2-norm while lasso regression uses the l1-norm*
  We get a percentage error around 41%.

```
> lasso.coef=predict(out,type ="coefficients",s=bestlam )[1:20,]
> lasso.coef
  (Intercept)          fold    population householdsize  racepctblack
   948.058163      0.000000      0.000000      0.000000     -1.066765
 racePctWhite  racePctAsian   racePctHisp   agePct12t21   agePct12t29
     0.000000      0.000000      0.000000      0.000000      0.000000
  agePct16t24    agePct65up     numbUrban      pctUrban     medIncome
     0.000000      0.000000      0.000000      0.000000      0.000000
     pctWWage  pctWFarmSelf    pctWInvInc    pctWSocSec   pctWPubAsst
     0.000000     13.422522      0.000000      0.000000      0.000000
> lasso.coef[lasso.coef!=0]
 (Intercept) racepctblack pctWFarmSelf
  948.058163    -1.066765    13.422522
```

```
> mean(( lasso.pred -y.test)^2)
[1] 180634.9
> sqrt(mean((lasso.pred-y.test)^2))/mean(y.test)
[1] 0.4146465
```

PCR

- *Using PCR we get 124 components for the dataset. The cross-validation was done using 10 random segments.*

```
> summary(pcr.fit)
Data:   X dimension: 319 124
        Y dimension: 319 1
Fit method: svdpc
Number of components considered: 124

VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
CV           786.4    627.5    632.6    789.3    762.3    618.6    621.6
adjCV        786.4    624.7    630.1    773.3    745.4    611.5    614.5
       7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
CV       593.4    525.7    516.0     515.2     515.5     518.7     520.8
adjCV    587.2    518.2    513.7     513.0     513.2     516.0     517.8
       14 comps  15 comps  16 comps  17 comps  18 comps  19 comps  20 comps
CV        526.4     526.8     520.1     514.8     516.1     518.1     516.3
adjCV     522.9     523.0     517.1     512.5     513.6     516.3     513.4
       21 comps  22 comps  23 comps  24 comps  25 comps  26 comps  27 comps
CV        520.7     541.6     549.1     551.0     555.3     556.1     557.4
adjCV     517.4     536.8     544.1     545.9     549.0     550.3     551.7
       28 comps  29 comps  30 comps  31 comps  32 comps  33 comps  34 comps
CV        560.6     594.6     581.4     588.0     590.5     594.9     592.6
adjCV     555.0     585.5     574.6     579.9     582.4     586.3     584.3
```

- *We get the minimum Mean Squared Error in case of 18 components where we get a RMS error of about 516.1.*

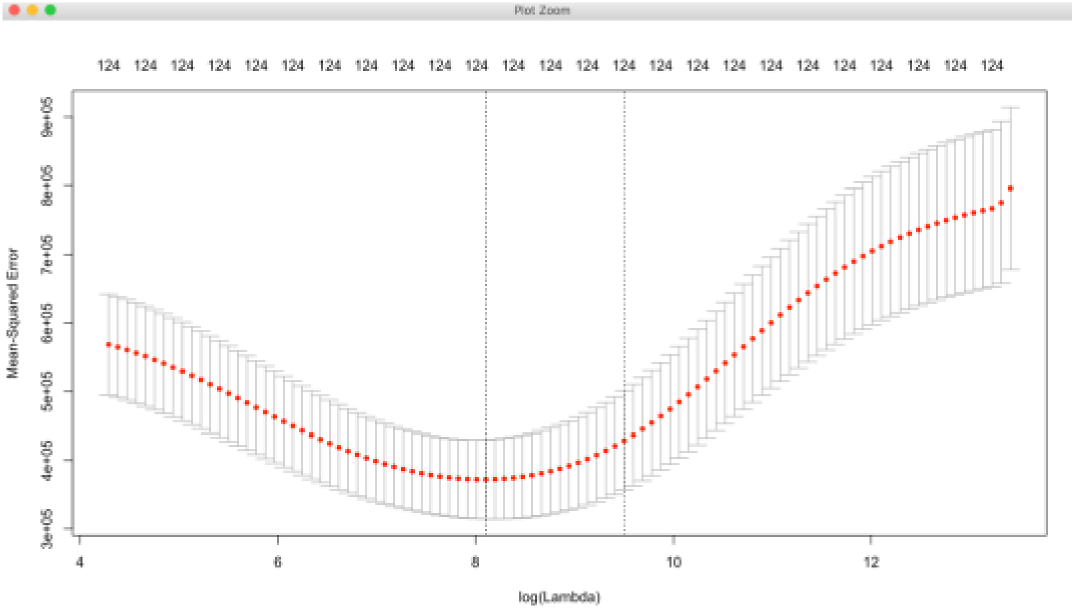
KNN

- *We next try KNN with 5 NN using which we get a MSE of about 378536.7. This turns out to be high compared to Ridge, Lasso Regression and PCR.*
- *We try KNN using a range of values for nearest neighbors. We get a least MSE of about 350893.7 for 15 NN.*

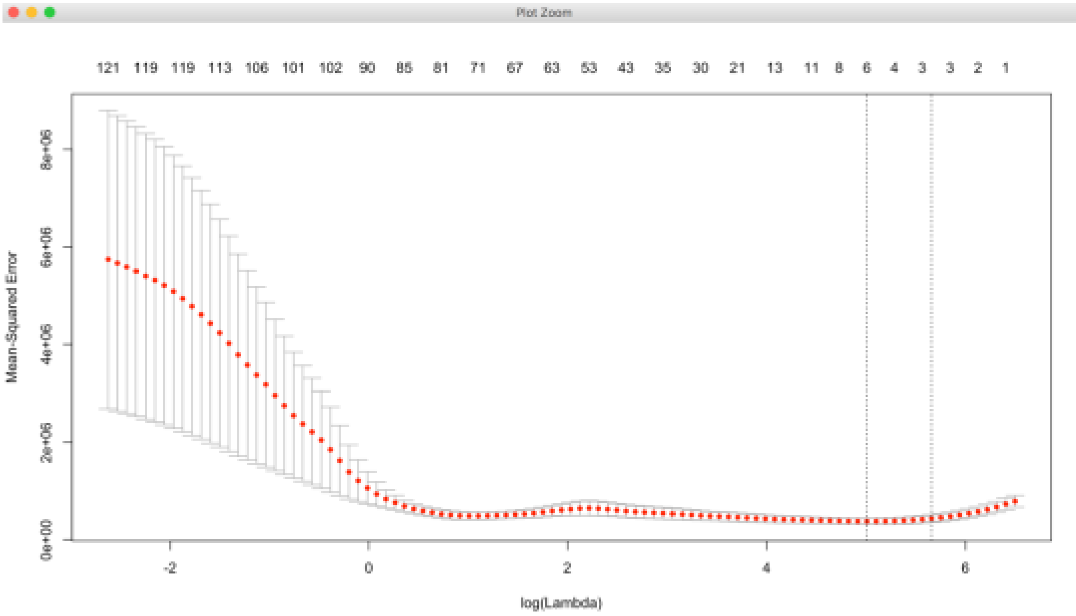
```
> errs
 [1] 584161.8 442330.7 392206.4 394070.6 378536.7 375254.1 378204.3 379007.4
 [9] 380110.4 381414.6 377316.9 361970.3 360766.3 354197.7 350893.7
> min(errs)
[1] 350893.7
```

Graphs:

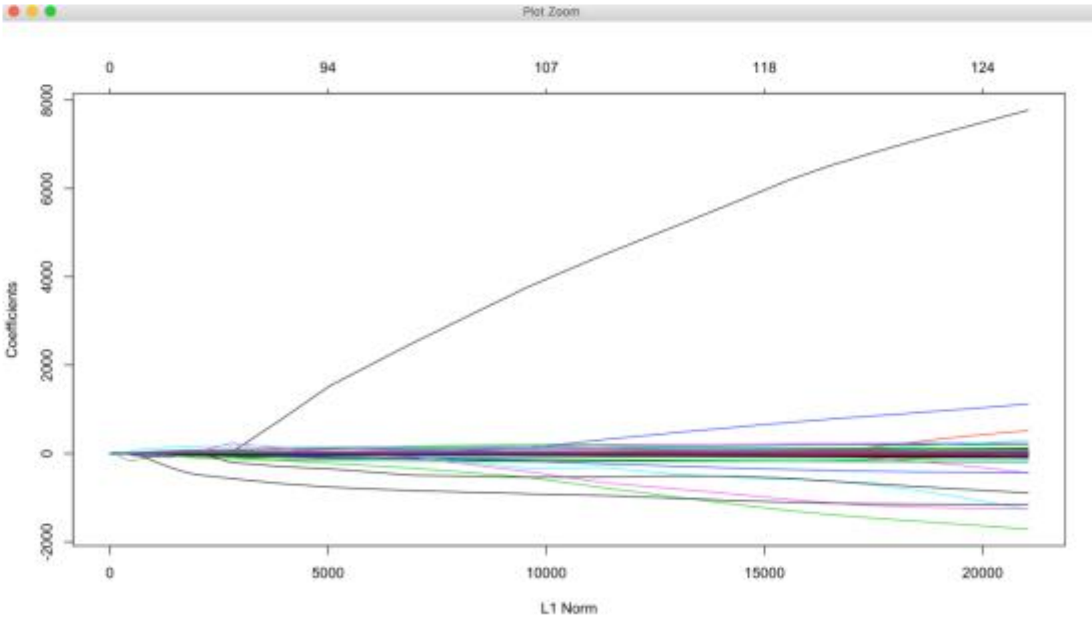
- Ridge Regression(Mean Squared Error vs Log Lambda)



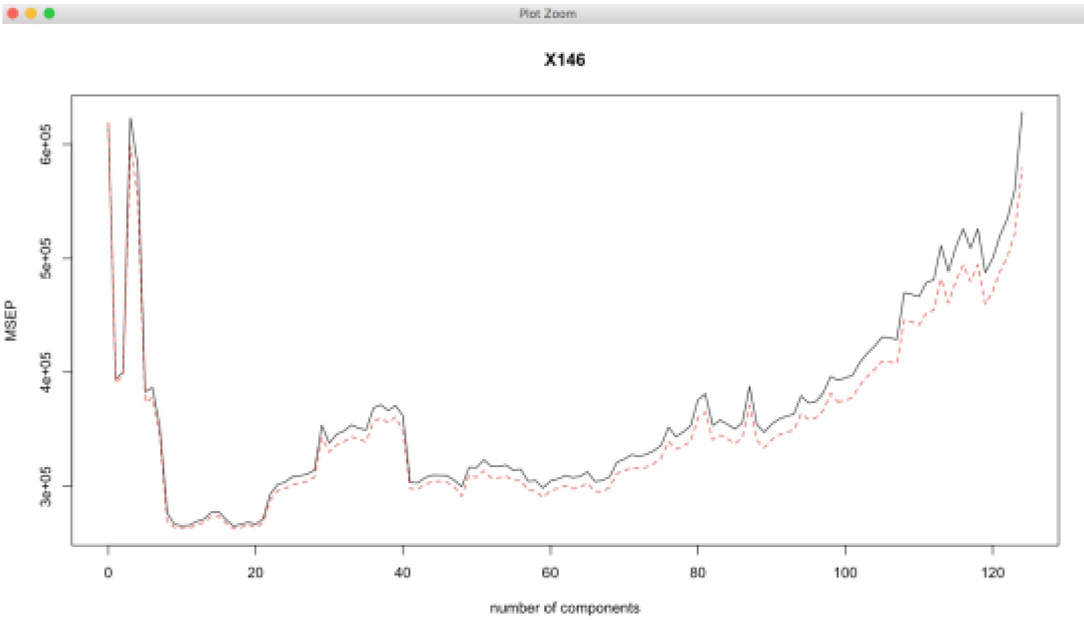
- Lasso Regression(Mean Squared Error vs Log Lambda)



- Lasso Regression(Coefficients vs L1 Norm)



- PCR Validation plot



Summary:

- We get the best results using Lasso Regression. Using KNN with 5 NN yields the worst result out of all which is improved when we try KNN with 8NN starting with a range of NN values and calculating binErrors. However the error percentage in all the cases turns out to be quite high for the dataset at around 41% which may be a result of absence of a strong correlation between the various factors present in the dataset.