732A99/732A68/ TDDE01 Machine Learning Exam Help

Vyshnavi Pisupati

25/12/2019

Block 1 Lab 1

Assignment 1. Spam classification with nearest neighbors

1.1 Import and divide the data into training and test

1)The data was imported into R and then divided into 50% train and 50% test data.

1.2 Logistic regression to predict spam using the threshold 0.5

Model was built for logistic regression model using the function glm() in R.The train and test errors were predicted using the model. The threshold probability used here to classify as spam is 0.5. The results are as below.

Table 1: Confusion Matrix Training Data threshold 0.5

	0	1
0	804	127
1	93	346

Table 2: Confusion Matrix Test Data threshold 0.5

	0	1
0	808	143
1	92	327

Misclassification Rates LR threshold 0.5

	Training	Test
Misclassification Rate	0.1605839	0.1715328

Looking at the results, it can be observed that the test error is higher than the training error. This is due to the fact that the model has been trained on the traning data which is already seen by the model.

1.3 Logistic regression to predict usinng threshold 0.8

Table 3: Confusion Matrix Training Data threshold 0.8

	0	1
0	921	10
1	333	106

Table 4: Confusion Matrix Test Data threshold 0.8

	0	1
0	931	20
1	314	105

Misclassification Rates LR threshold 0.8

	Training	Test
Misclassification Rate	0.250365	0.2437956

Comparing the results, the misclassification rate has increased due to the new rule,though the test error still remains higher than the train error. The false positive rate(i.e. a normal being classified as a spam mail) has decreased using the second rule which is desired in this particular case of spam mail filtering.

Also, with the appropriate loss matrix which specifies the penality for the false positives either of the models can be used.If loss for marking a non-spam email as spam is much greater than marking a spam as non-spam,the model with threshold 0.8 performs better.

1.4 K nearest neighbours with K=30

K-nearest neighbour model was built with K=30 and the results are as below.

Table 5: Confusion Matrix Training Data K=30

	0	1
0	779	152
1	77	362

Table 6: Confusion Matrix Test Data K=30

	0	1
0	702	249
1	180	239

Misclassification Rates for kknn K=30

	Training	Test
Misclassification Rate	0.1671533	0.3131387

Comparing the results using K-nearest neighbours and logistic regression with threshold 0.5,though the train errors in both the cases is almost same,the later has higher the test error and false positives than the former.Hence logistic regression model with threshold probability with 0.5 is better than K-nearest neigbour classification with K=30.

Comparing the results using K-nearest neighbours and logistic regression with threshold 0.8, the former has higher train error,train error and false positive rate compared to the later.Hence logistic regression with threshold probability 0.8 is better than K-nearest neighbour classification with K=30.Among the two logistic regressions, the one with threshold probability 0.8 is better as it has lower false positve rates.

Overall,the logistic regression with threshold 0.8 is the best model among the 3 models.

1.5 K-nearest neighbours with K=1

K-nearest neighbour model was built with K=1 and the results are as below.

Table 7: Confusion Matrix Training Data K=30

	0	1
0	931	0
1	0	439

Table 8: Confusion Matrix Test Data K=30

	0	1
0	644	307
1	185	234

Misclassification Rates for kknn K=30

	Training	Test
Misclassification Rate	0	0.3591241

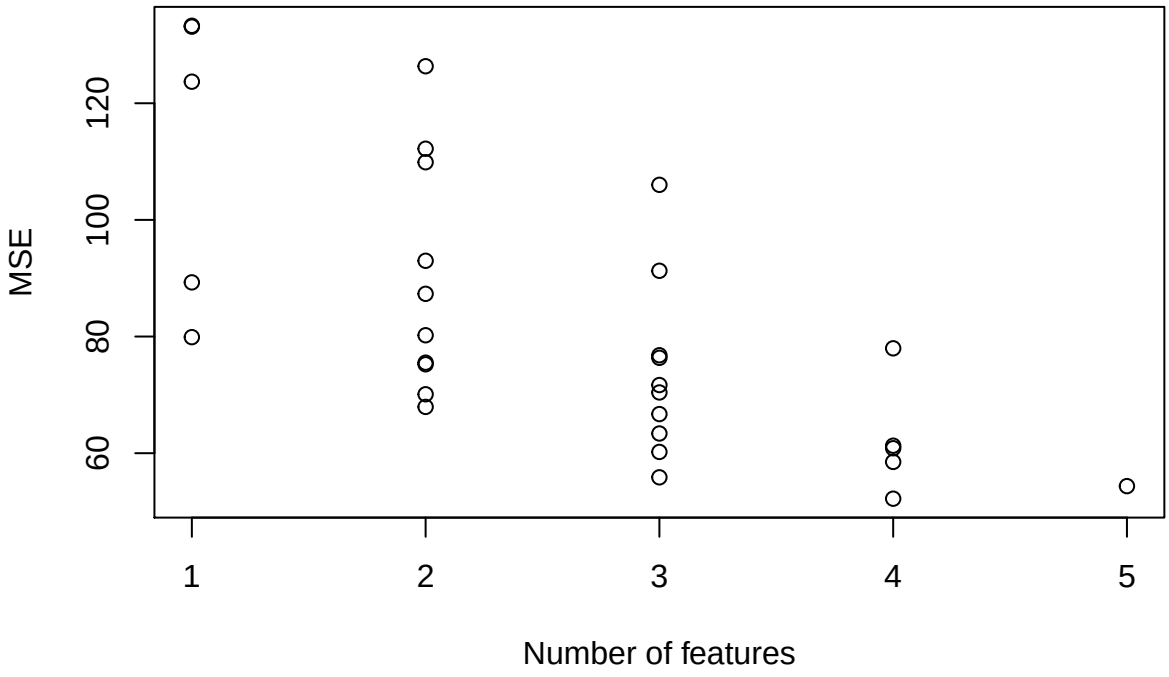
The change in K to 1 has lead to increase in test misclassification rate and has caused over-fitting of the training data compared to the logistic model with threshold 0.5 ot 0.8.This can be concluded from the zero traning misclassification error and high test misclassification error.

NOTE: The 0 misclassification rate in case of the training dataset can be attributed to the fact that the same dataset is used to train as well as test the model. Hence, every point in the dataset used for testing the model is closest to itself for k = 1 nearest neighbour classification.

Assignment 3. Feature selection by cross-validation in a linear model.

3.1 Writing functions for linear regression and Cross-validation.

3.2 Testing with "swiss" data set



	CV	Features
Best Fit Model	52.19978	1 0 1 1 1

Selected Features = Agriculture, Education, Catholic, Infant.Mortality

Correlation matrix for swiss dataset:

	Fertility	Agriculture	Examination	Education	Catholic	Infant.Mortality
Fertility	1.0000000	0.3530792	-0.6458827	-0.6637889	0.4636847	0.4165560
Agriculture	0.3530792	1.0000000	-0.6865422	-0.6395225	0.4010951	-0.0608586
Examination	-0.6458827	-0.6865422	1.0000000	0.6984153	-0.5727418	-0.1140216
Education	-0.6637889	-0.6395225	0.6984153	1.0000000	-0.1538589	-0.0993218
Catholic	0.4636847	0.4010951	-0.5727418	-0.1538589	1.0000000	0.1754959
Infant.Mortality	0.4165560	-0.0608586	-0.1140216	-0.0993218	0.1754959	1.0000000

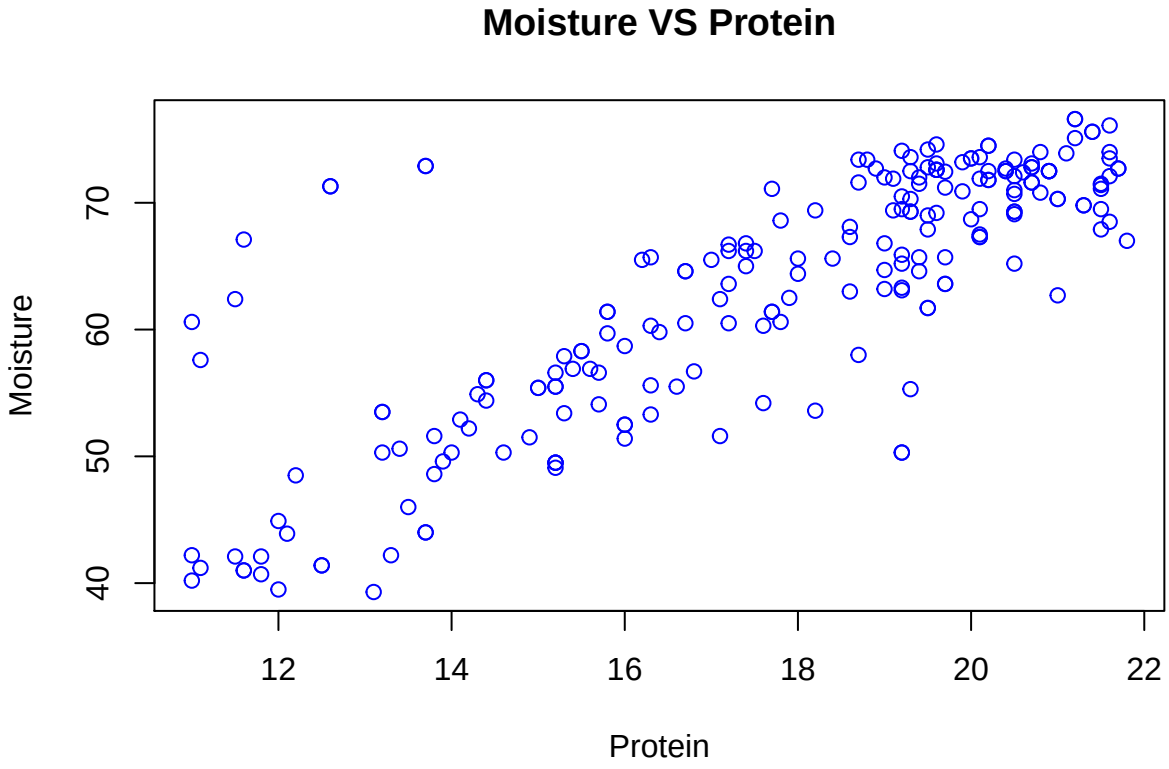
The optimal model has a subset of 4 features - Agriculture, Education, Catholic and Infant.Mortality. It is reasonable that this subset of features has the largest impact on the target feature "Fertility" because this subset has the lowest MSE. High multicollinearity reduces the precision of the estimated coefficients (increases standard error) as it becomes harder to determine the effect of individual predictors on the target. Hence, eliminating this highly correlated feature results in better estimates of the target for changes in each

of the remaining individual predictors and lowers the MSE of the model predictions. This notion is supported by the results obtained as the second feature "Examination" was dropped due to its high multicolinearilty with the other features by the cross-validation algorithm.This can be seen from the correlation matrix as shown above.

It is obsered from the plot, the variance of the model decreases and the number of the features increases(i.e as the model complexity increases).The model MSE score reaches the minimum when the combination of the 4 features "Agriculture","Education","Catholic","Infant.mortality" is selected.The MSE increases when all the features were included.We can conclude that "Examination" does not play a significant role in the target feature "Fertility".

Assignment 4. Linear regression and regularization

4.1 Import the data into R and plotting the data



It is seen from the graph clearly that moisture and protein follow a linear relation with a few outliers.So the data can be described well with a linear model.

4.2 Usage of MSE criterion

$$P(Y_i|\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y_i-\mu)^2}{2\sigma^2}}$$

where, $\mu = w^T X_i$ where X_i is a column matrix with $j = 1, 2, ..M$ number of rows representing the powers of X and σ^2 is the variance of the noise in the data which is also normal on $\mu = 0$ and σ.

The optimal parameters $\hat{w}$ must be determined such that Y is closest to the target variable t. For this we use maximize the likelyhood function with respect to w.

Calculating the maximum likelyhood of Y, we get

$$l = \prod_{i=1}^{n} \frac{1}{(\sqrt{2\pi}\sigma)^n} e^{-\frac{\sum_{i=1}^{n}(y_i - w^T X_i)^2}{2\sigma^2}}$$

To simplify, we take ln of l then to find $\hat{w}$, we maximize $ln\,(l)$ or minimize $-ln\,(l)$.

Taking $-ln(l)$

$$-ln(l) = \frac{n}{2} ln(2\pi\sigma^2) + \frac{\sum_{i=1}^{n}(Y_i - w^T X_i)^2}{2\sigma^2}$$

Since $\frac{n}{2} ln(2\pi\sigma^2)$ consists of constants, in order to minimize the $-ln(l)$, we must minimize $\sum_{i=1}^{n}(Y_i - w^T X_i)^2$ which means applying the crieteria of minimizing the MSE is appropriate.

4.3 Fitting different polynomial models to training 50% and testing 50% and determining the best model

From the graphs, it can be seen that when the degree of polynomial is 1 both the traning error and test error are high and also the variance(this can be see by the difference between the Traning and Test error)is high.When the degreee of the polynomial is 2,there is drops and reaches the minimum value, though the train MSE is not at its minumum,the variance is for this model is low.When the polynomial degree is 3,the traning MSE drops drastically, while, there is surge in the test MSE. This trend continues for the all higher polynomial degress. Though the traning MSE is decreasing the test MSE is increasing, indicating that model

is overfitting the data.Hence, model 2 with polynomial degree 2 is the best model which balances bias and variance as there is always a trade off between these two.

Taking a look at the loss function below, for a given model, as the variance decreses, the bias increases and vice-versa. Hence there is always a trade-off between the bias and variance. Choosing a model with low variance, increases the bias.In this case model 2 is choosen, though the variance is low, the bias is high.

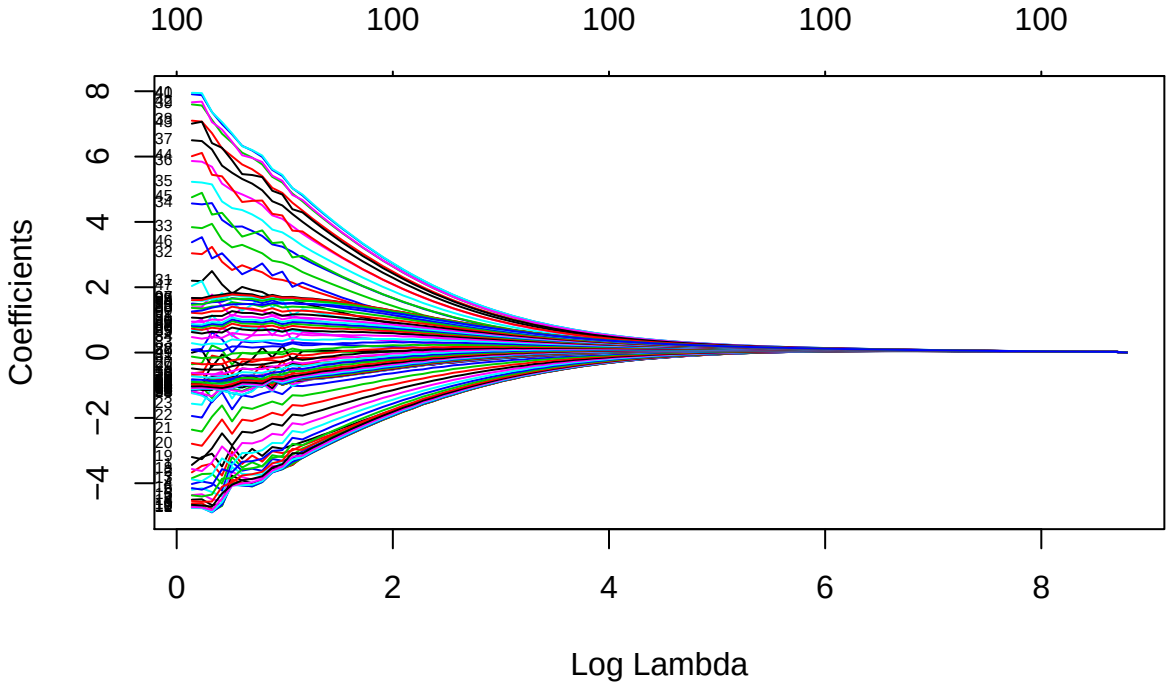
$$R\left(Y\left(x_o\right), \hat{Y}\left(x_o\right)\right) = \sigma^2 + Bias\left(\hat{Y}\left(x_o\right)\right)^2 + Var\left(\hat{Y}\left(x_o\right)\right)$$

4.4 Variable selection using stepAIC

The below features are identified as the most significant : [1] “Channel1” “Channel2” “Channel4” “Channel5” “Channel7” [6] “Channel8” “Channel11” “Channel12” “Channel13” “Channel14” [11] “Channel15” “Channel17” “Channel19” “Channel20” “Channel22” [16] “Channel24” “Channel25” “Channel26” “Channel28” “Channel29” [21] “Channel30” “Channel32” “Channel34” “Channel36” “Channel37” [26] “Channel39” “Channel40” “Channel41” “Channel42” “Channel45” [31] “Channel46” “Channel47” “Channel48” “Channel50” “Channel51” [36] “Channel52” “Channel54” “Channel55” “Channel56” “Channel59” [41] “Channel60” “Channel61” “Channel63” “Channel64” “Channel65” [46] “Channel67” “Channel68” “Channel69” “Channel71” “Channel73” [51] “Channel74” “Channel78” “Channel79” “Channel80” “Channel81” [56] “Channel84” “Channel85” “Channel87” “Channel88” “Channel92” [61] “Channel94” “Channel98” “Channel99” Number of fetures selected = 63 SSE: 0.8598985

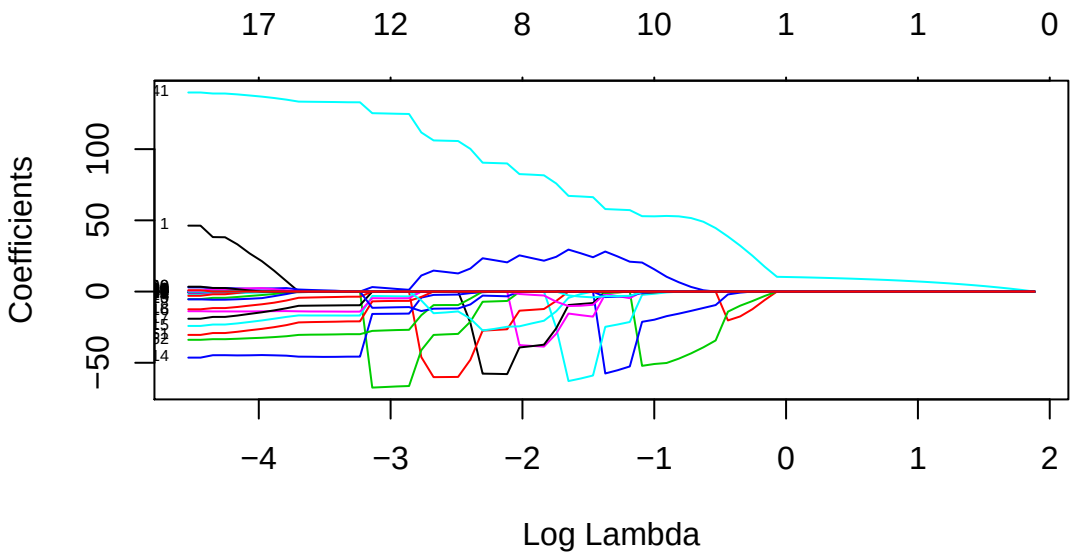
A total of 100 features were given as the input to the stepAIC function which eveluates removes and adds the features based on the AIC values calculated in each iteration.Finally, 63 features were identified to be most significant in determining Fat.

4.5 Ridge regression

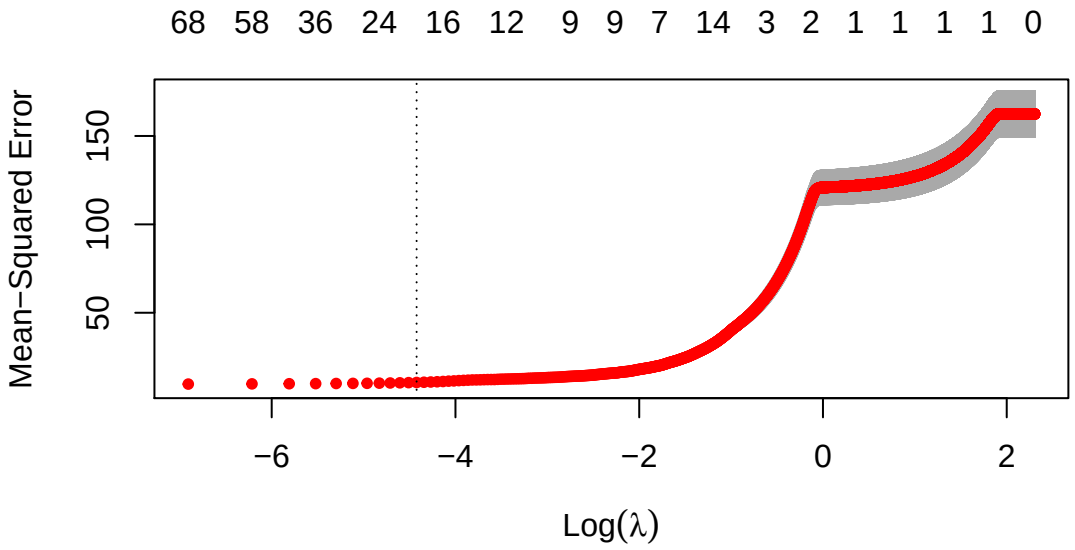


As λ increases, the number of the coefficients tending to zero increases.After the value $Log(\lambda)$ is greater than 6, all the coeffcients tend to zero. Hence an optimal value of λ needs to be choosen which lets the coefficients of less significant features tend to zero.This can be done by cross-validation.

4.6 Fitting a LASSO model



4.7 Cross-Validation for LASSO



	Lambda	Measure	SE	Nonzero
Min	0.000	9.488881	1.131688	100
1se	0.012	10.552218	1.013034	17

It can be seen from the graph that as the penalty factor λ increases, the MSE also increases. Combining the results from exercise 4.6, it can be said that as the lambda increases, more and more features are dropped from the model (coefficients shrink to zero) making the model simpler which results in a higher mean squared error.

Hence the lowest MSE recorded is 9.49 for $\lambda = 0$ (simple linear regression) where all of the 100 features are selected in the model. But while $\lambda.min$ gives the least MSE, it is clearly not the most optimum. A good model would be one which is not overly complex, but at the same time has low MSE with an optimal selection of features. It can be observed that $\lambda.1se = 0.012$ gives the smallest subset of features such that the MSE is within 1 standard error of the minimum MSE. In other words, for a small increase in MSE value from the minimum MSE, the model is optimized by selecting only 17 features out of 100.

4.8 Compare results from 4.4 and 4.7

	Measure	Features
step AIC	0.8598985	63
CV Lasso	10.5522180	17

In step 4, which uses AIC method for feature selection, 63 features were selected as the most significant features while in step 7 which uses LASSO a coefficient decomposition technique,17 features were selected. LASSO model should be selected as it is less complex when compared to the stepAIC model.

Block 1 Lab 2

Assignment 2. Analysis of credit scoring

2.1 Importing data and diving into train,validation and test

2.2 Fitting a model using Deivance and Gini imputities

a. Deviance Method

Deviance is one of the possible crieteria to buld the tree i.e create a split. Deviance is also known as cross-entrpy of a system. It is the MLE for decision trees. Its formula is given as below.

Say we have K classes and m regions, the cross-entropthy is calculated for each regionR_m,E_m.

$$E_m = -\sum_{k=1}^{K} \hat{p_{mk}} ln(\hat{p_{mk}})$$

Where,$\hat{p_{mk}}$ is the probalility of a class $k = 1, 2, ..., K$ in the region R_m.

The decision tree aims for a minimum deviance. Ideally when $E_m = 0$, all the classes are classified correctly without any misclassification. But in such cases, over-fitting of the training data occurs where out tree does not give a general decision tree to predict the data. There is always a tradeoff between E_m and the quality of fit for testing data.

Misclassification Rates with Deviance method

	Training	Test
Misclassification Rate	0.212	0.264

Table 10: Confusion Matrix Training Data(Deviance)

	bad	good
bad	61	86
good	20	333

Table 11: Confusion Matrix Test Data(Deviance)

	bad	good
bad	30	47
good	19	154

b. Gini Index

Gini index is one of the possible measure to build the decision tree, i.e create a split of the parent region. It is similar to deviance measure. The formula is given as below.

Say we have K classes and m regions, the gini index is calculated for each regionR_m,G_m.

$$G_m = \sum_{k=1}^{K} \hat{p_{mk}}(1 - \hat{p_{mk}})$$

Where,$\hat{p_{mk}}$ is the probalility of a class $k = 1, 2, ..., K$ in the region R_m.

Table 12: Confusion Matrix Training Data(Gini)

	bad	good
bad	66	81
good	38	315

Table 13: Confusion Matrix Test Data(Gini)

	bad	good
bad	26	51
good	28	145

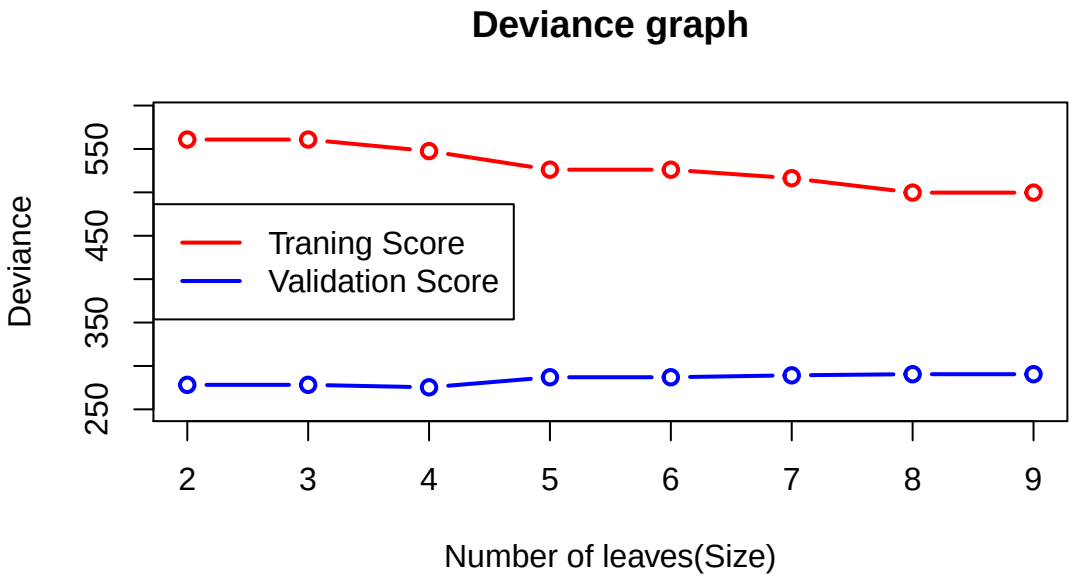
Misclassification Rates for Gini method

	Training	Test
Misclassification Rate	0.238	0.316

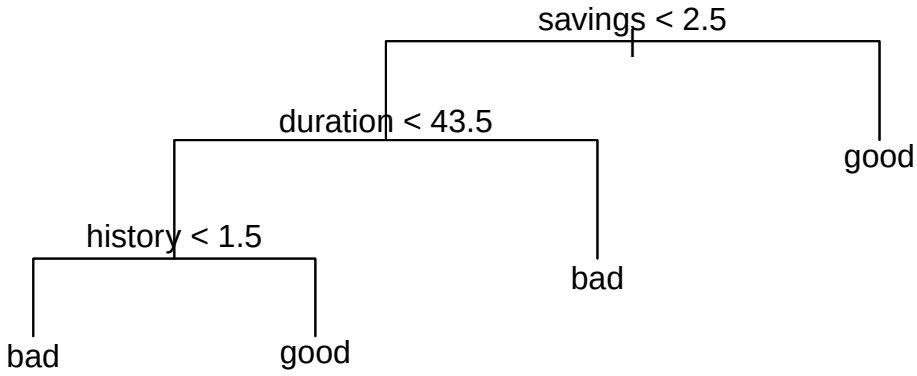
Comparing both the methods, both the test and train errors are higher using the gini method. Hence deviance method is choosen for the below steps.

2.3 Determining the optimal tree depth

As expalined earlier, minimizing the entrophy completely can cause over-fitting hence is always recommanded to prune the tree,i.e. remove some branches of the tree so that the test misclassification is reduced though the train misclassification increases.



Best tree test misclassification rate : 0.264



The best fitting model has 3 variables which are “savings”,“duration” and “history” hence the optimal tree depth is 3 with 4 leaves i.e. terminal nodes.

2.4 Naive Bayes classification

The classifier uses probability to classify the data. The probability of an outcome occuring due to each variable is calculated.

Table 14: Confusion Matrix Training Data(Naive Bayes)

	bad	good
bad	95	52
good	98	255

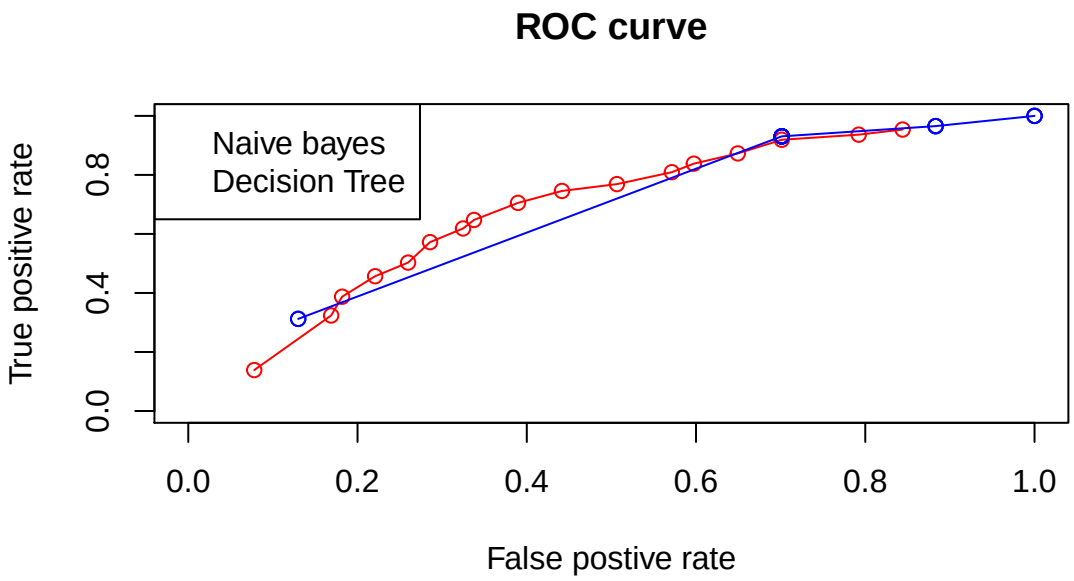
Table 15: Confusion Matrix Test Data(Naive Bayes)

	bad	good
bad	51	26
good	61	112

Misclassification Rates for Naive Bayes

	Training	Test
Misclassification Rate	0.3	0.348

2.5 ROC curves for Decision tree and Naive Bayes with a principle



As seen from the ROC curve of decision tree and naive bayes, when the FPR rate is less than approximately 0.62, the naive bayes is above the decision tree curve implying for the same FPR, the TPR for naive bayes is higher(area under the curve is also higher)which is desired for this scenario. For higher FPR, the decision tree performs better.Hence depending on the accepatable FPR rate we the model should be choosen. Genrally high FPR is not acceptable. Hence, we can conclude that naive bayes is a better classifier.

2.6 Naive Bayes with a loss matrix

Table 16: Confusion Matrix Train Data(NB with LM)

	bad	good
bad	137	10
good	263	90

Table 17: Confusion Matrix Test Data(NB with LM)

	bad	good
bad	65	12
good	139	34

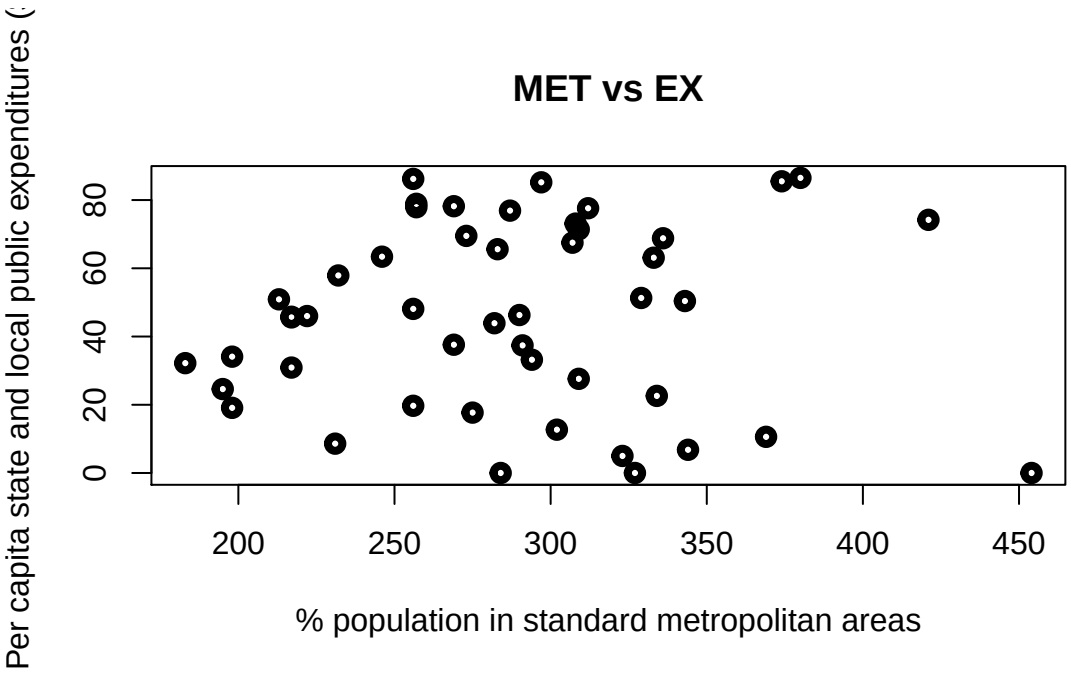
Parameters to compare step 4 and 6 :

	Naive.Bayes	Naive.Bayes.with.LM
Misclassification Rate	0.3480000	0.6040000
False Positive Rate	0.3376623	0.1558442

When comparing the results from 4 and 6, we see that though the misclassification rate has increased when the naive bayes uses a loss matrix,the FPR decreases as using the loss matrix due to the high penality factor introduced for classifying a "bad" customer as "good" customer. Hence it is always better to use a loss matrix specific to the problem with a naive bayes to be on a safer side.

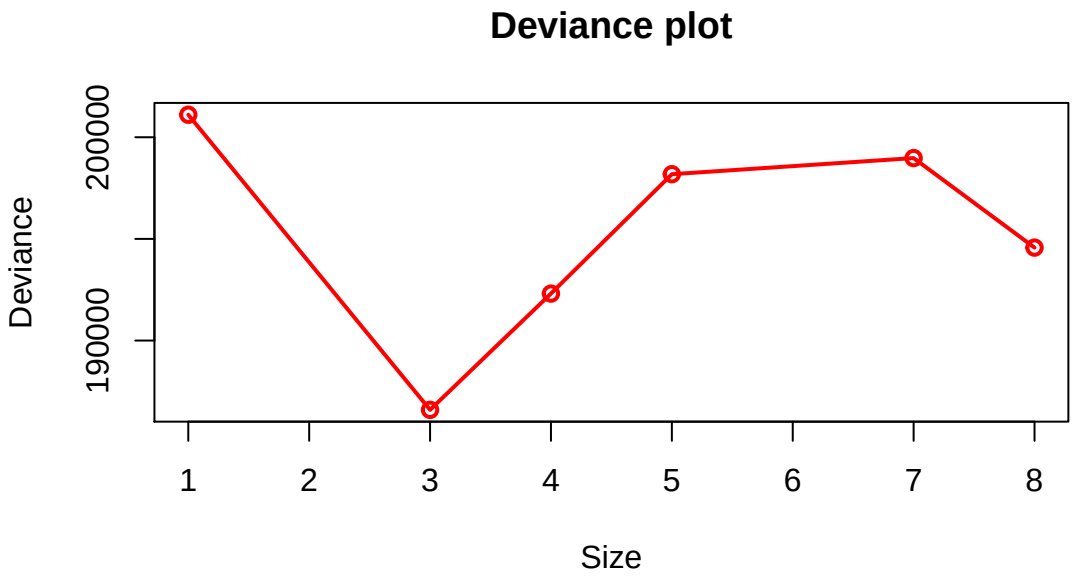
Assignment 3. Uncertainty estimation

3.1 Seeing the data trend

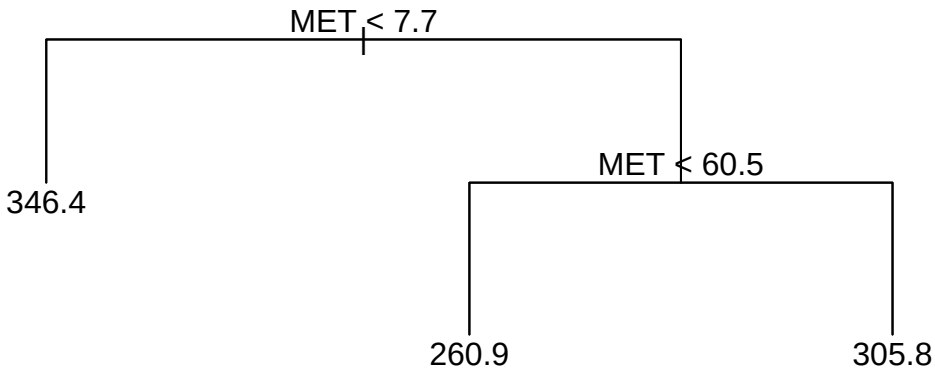


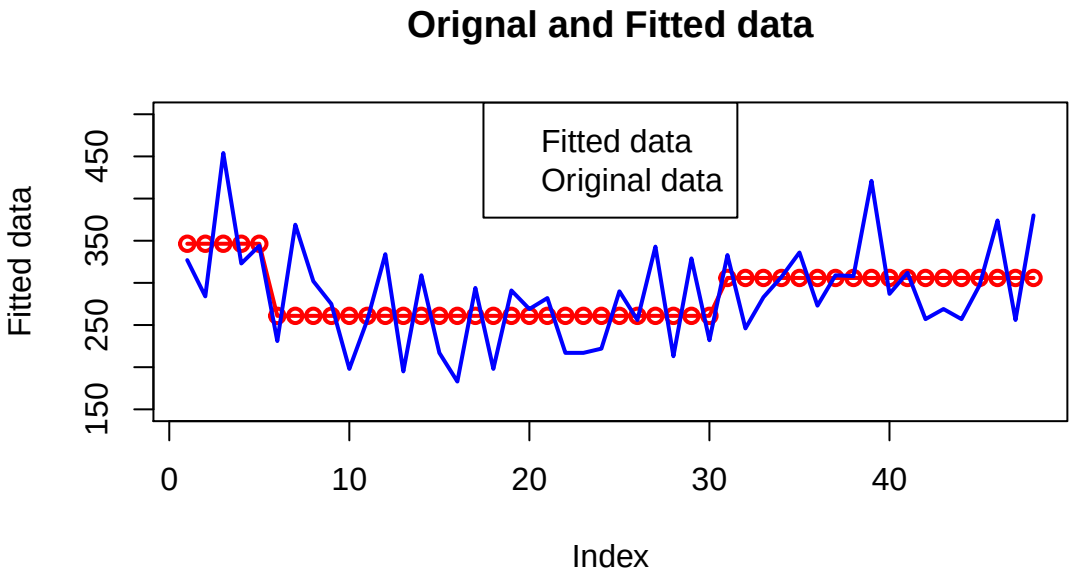
As observed from the 2-D scatter-plot of the data, it can be concluded that the dependent variable(EX) is not linearly dependent on the independent variable(MEX),hence we can not apply linear regression to the data as is. Converting the data into a feature space and then checking the linearlity is one option that can be used. A simpler option would be a regression tree.

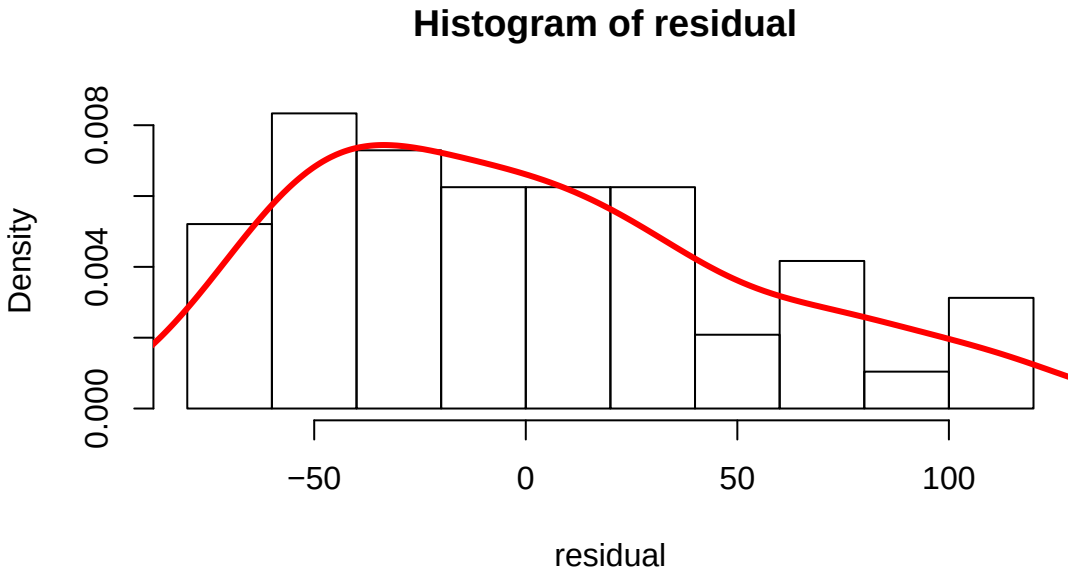
3.2 Fitting a regression tree model and checking quality of fit



The optimal tree is for the model: 5

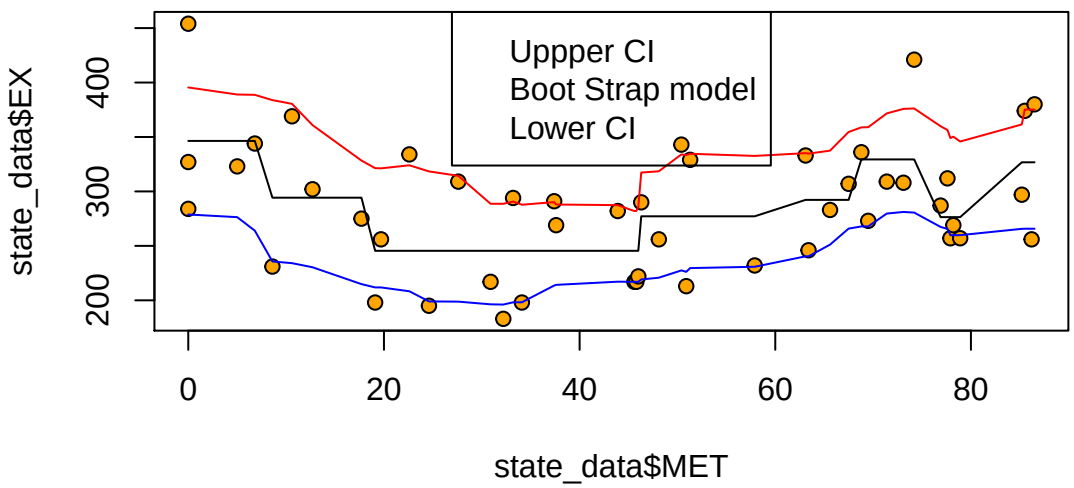






The residuals do not follow a perfect normal distribution clearly as it is slightly skewed towards the right.Hence assuming a linear model with residuals following a normal distribution is not correct to be used.Hence, a regression tree model which can be used when the distribution of the residuals is unknown is used.Though an optimal tree with minimum deviance was choosen,(3 terminal nodes and 2 levels) the quality of fit is not good as the predicted values do not capture the peaks in the original data.A general linear model which assumes a beta or gamma distribution can be used for a better fit.

3.3 Non-parametric bootstrap

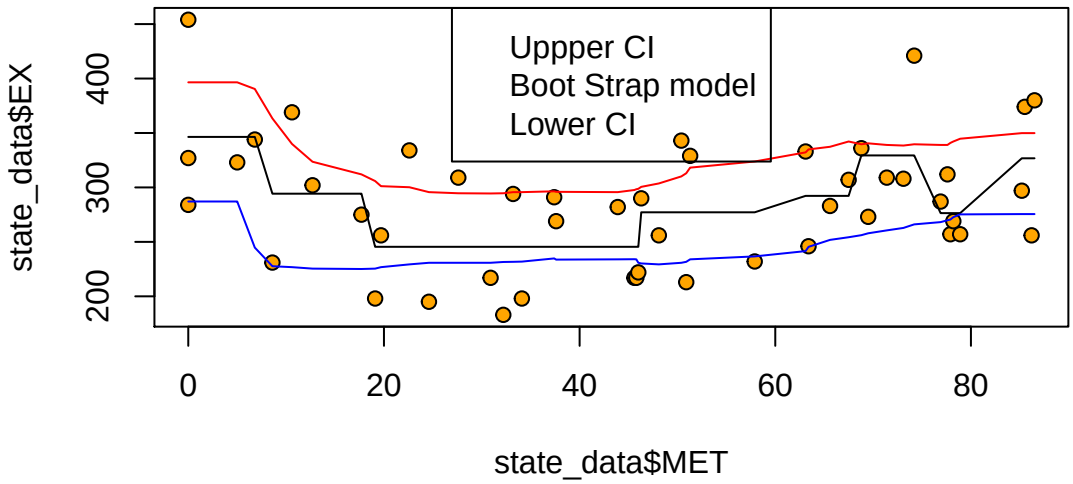


Comparing the Non-parametric and parametric bootstrap :

	Non.parametric.bootstrap
Mean width	99.99272
Variance width	333.07576

The width of the interval is 99.9 units aproximately and also, the CI is very bumpy impies that the prediction made in step 2 doesnt seem reliable.

3.4 Parametric bootstrap



Comparing the Non-parametric and parametric bootstrap :

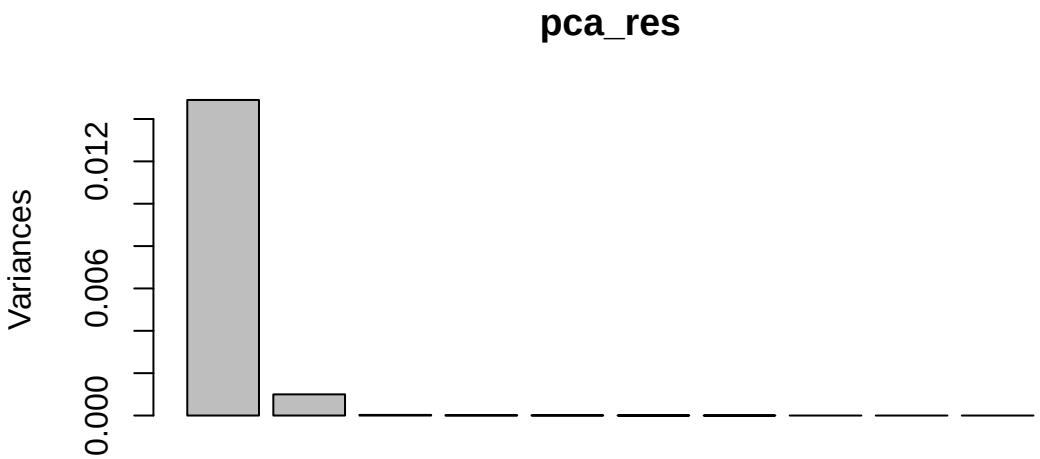
	Non.parametric.bootstrap	Paramaetric.bootstrap
Mean width	99.99272	80.80047
Variance width	333.07576	358.85859

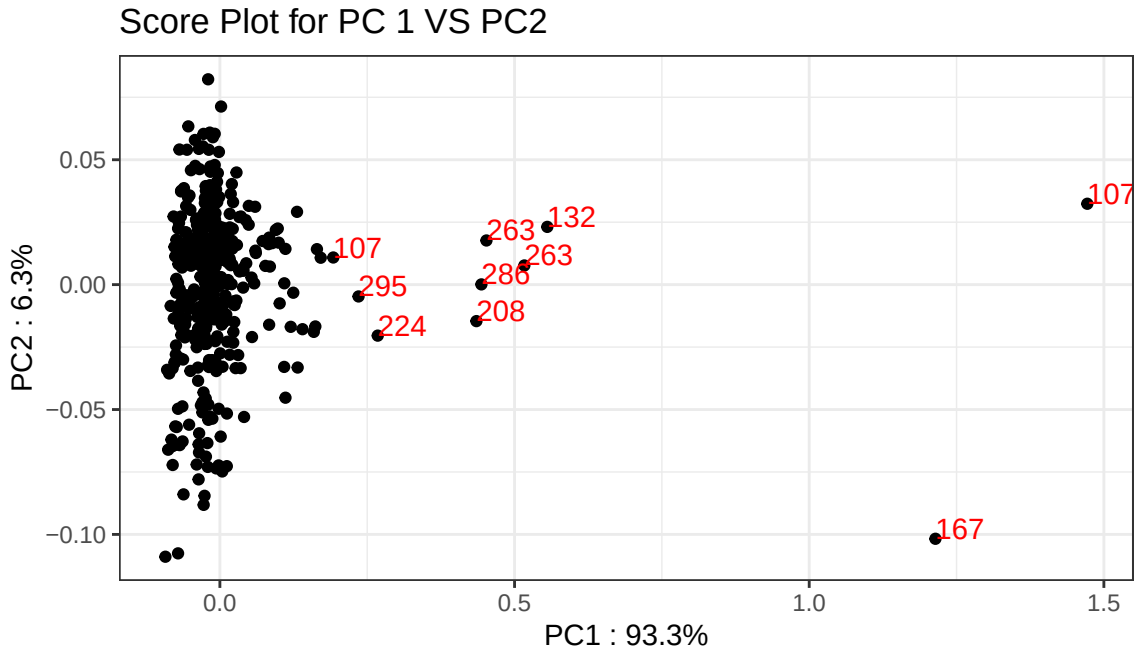
We have assumed that the output variable follows a normal distribution which is nt correct so it does not make any sense to check how much percentage is inside the confidence interval.Even if the assumption is correct,due to the randomness in data more than 5% of the data will be out of the confidence interval.

3.5 Histogram of residuals and appropriate bootstrap method.

The parametric boot strap CI is narrower than the non-parametric bootstrap model.This does not imply that the parametric bootstrap model is a better model compared to the non-parametric model a the paramtric model assumes that the true values are normally distributed and with the variance same as the variance of the residuals.As seen from the residuals graph, the residuals are not perfectly distributed and are skewed right. Hence a beta or gamma would have been a better assumption in this case.

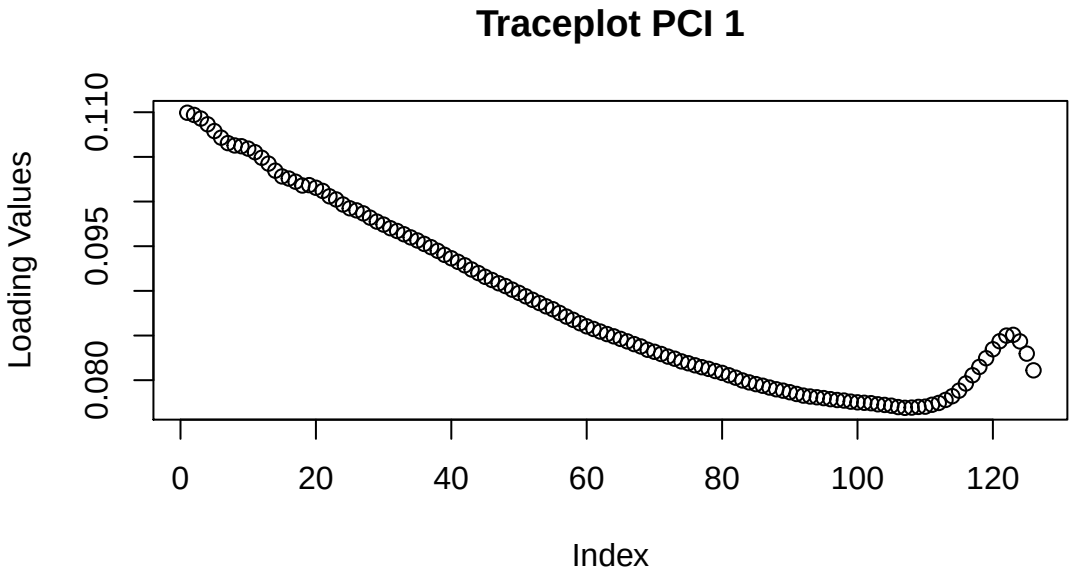
Assignment 4. Principal components

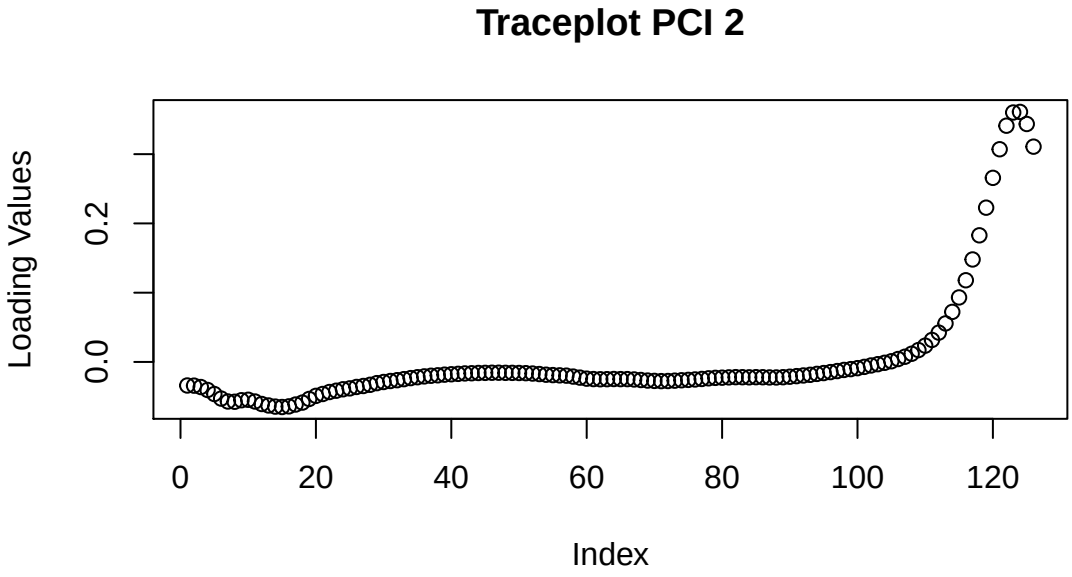




As seen from the screeplot, the 99.6 % of total variation is shown by PC 1 and PC 2. Hence rest of the PCs can be ignored.According to the score plot,some of the diesel fuels (labelled in red with the column number of the feature) that are on the right extreme of the PC1 co-ordinate axis.These unusual values (outliers) cause the high variance along PC1.

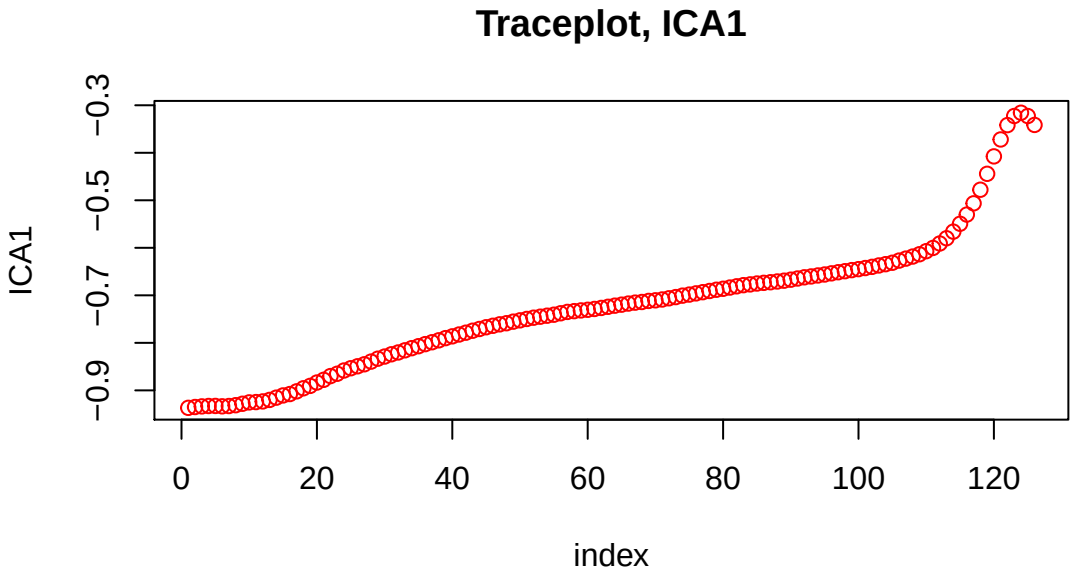
4.2 Analyzing the trace plots

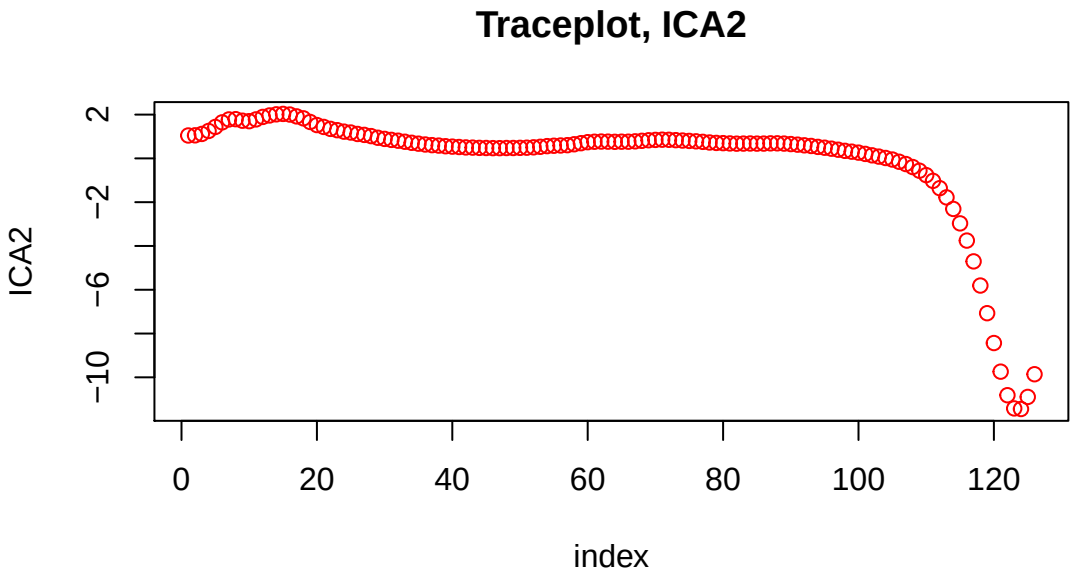




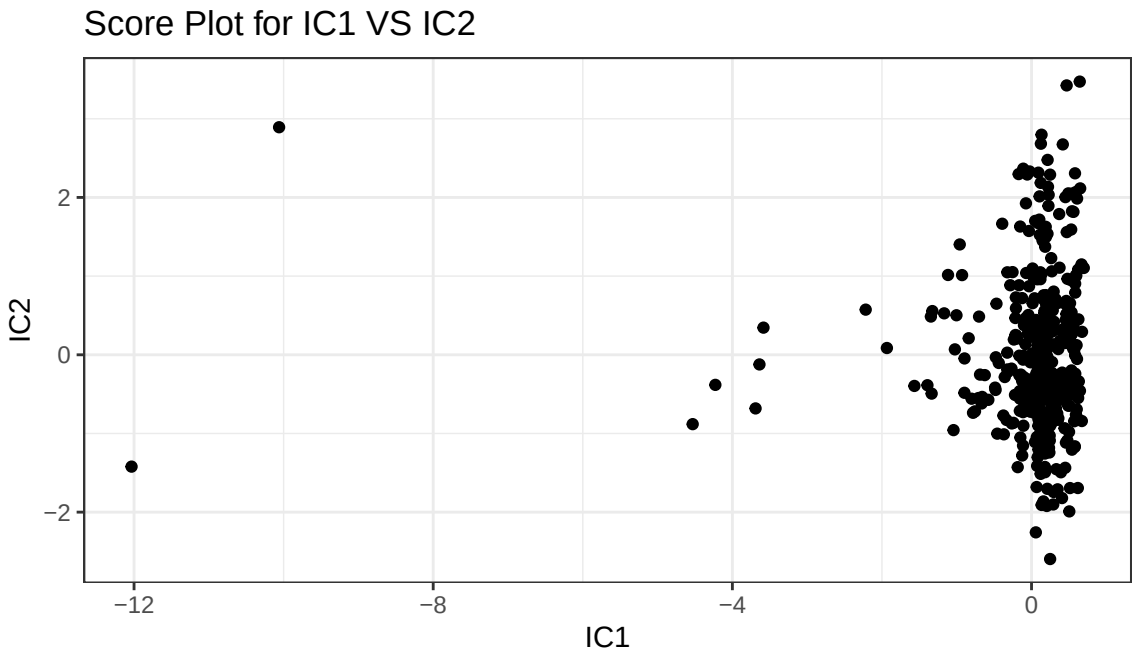
As seen from the trace plots, the loading values of the variables for PC 2 are very less approximately till 115th variable. So the loading values of approximately, 11 variables are significant.Hence this can be considered as the PC explained by few features.

4.3a Independent Component Analysis





The data matrix is taken and then PCA is done on these to get the direction of maximum variation.K matrix gives the principal components and also called the pre-whitening matrix. W matrix is the unmixing matrix which gives the ICs.W' is calculated from W and the K matrix.Compared with the trace plots in step 2 we see that the latent ICA component are mirrors of the PCA components. This is because PCA tries to find correlation between the features by maximizing the variance to allow reconstruction. ICA works maximizing independence through transforming the feature space into a new feature space where all features are mutually independent.



Just like in a) we see that also this is a mirror reflection of the scores from PCA but through performing ICA the features are scaled differently.

Block 1 Lab 3

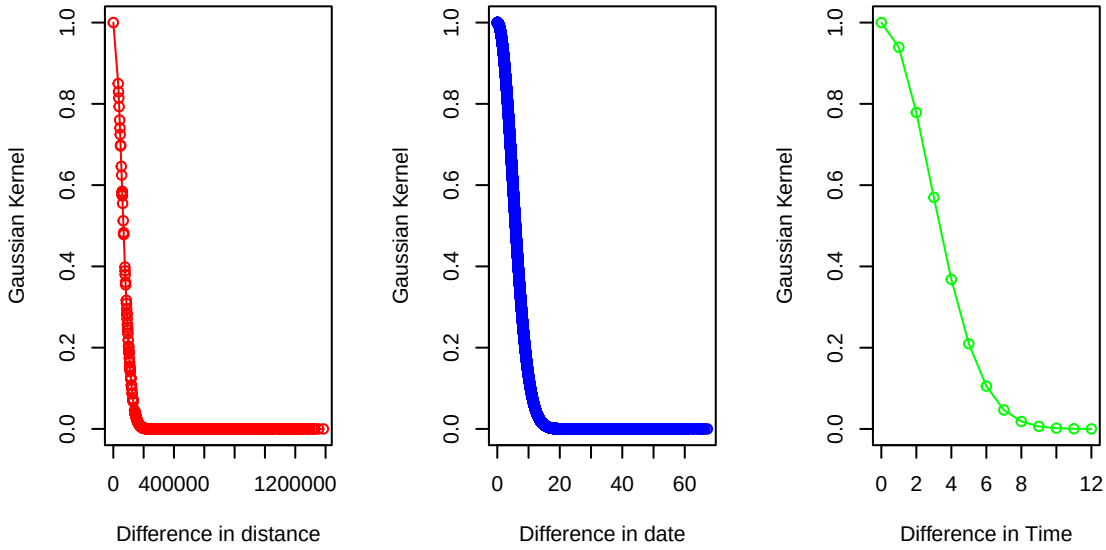
1. KERNEL METHODS

1.1 Accounting for the distances in location,date and time

The differences in the diatance, location, date and time are calculated using distHaversine() and difftime() functions respectively.

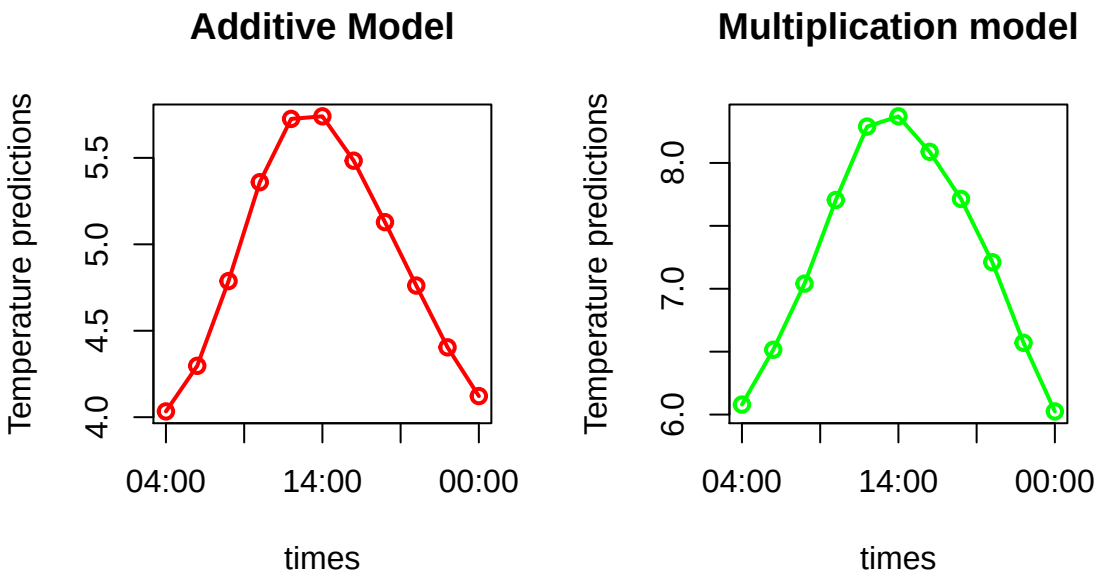
Three individual guassian Kernels are built and then combined both by addition and multiplication to build a final kernel.

1.2 Choosing the smoothing co-efficents



As seen from the graphs, of the kernels, the points are choosen in such a way that maximum number of points are near the kernel mean with a minimum standard deviation.

1.3 Additive Kernel and Product Kernel Model



The multiplication model is s better predictor here as seen from the graphs as it accounts to the closeness of distance time and date in a more appropriate way.

2. SUPPORT VECTOR MACHINES

2.1 Model selection from 3 SVMS with C=0.5,1,5

To select the model, holdout method is used as an alternative to cross-validation. It is used when the available data is very large and it is computationally expensive to perform a cross-validation algorithm.

Hold-out Method

1)In this method, all the avalible data is divided into traning(50%), validation(25%) and testing(25%) datasets.
2)The different models($C = 0.5, 1, 5$) are tranined using the traning data set and the misclassification error is calculated for the validation dataset. The value of C that gives a minimum misclassification error is choosen.
3)Again a new model is tranined using both the traning and validation datasets and the misclassification error is calculated using the test dataset. This gives the generalization error for the model trained.
4)Finally, the model is tranined using the entire data and returned to the user.

Table 18: Confusion Matrix C=0.5

	nonspam	spam
nonspam	670	29
spam	66	385

Table 19: Confusion Matrix C=1

	nonspam	spam
nonspam	666	33
spam	57	394

Table 20: Confusion Matrix C=5

	nonspam	spam
nonspam	666	33
spam	51	400

Parameters to analyze

	TPR	FPR	MC
C=0.5	0.9103261	0.0700483	0.0826087
C=1	0.9211618	0.0772834	0.0782609
C=5	0.9288703	0.0762125	0.0730435

From the results we can see that, the FPR rate (which is desired to br minimum in spam filtering) is the least for $C = 1$ and the misclassification rate is minimum in the case of $C = 1$. Inspite of TPR for $C = 5$ being the highest, the misclassification rate and FPR are given more weightage and $C = 1$ model is selected.

2.2 Determining the generalized error for the SVM

Generalized Error : 0.07037359

2.3 SVM returned to the user

Support Vector Machine object of class “ksvm”

SV type: C-svc (classification) parameter : cost C = 1

Gaussian Radial Basis kernel function. Hyperparameter : sigma = 0.0287419476636351

Number of Support Vectors : 1403

Objective Function Value : -809.1044 Training error : 0.04586

2.4 Pupose of the C parameter.

The C parameter is a measure how much we penalize a misclassified data i.e. it is a penality factor. Higher the value of C, higher the penalization.

Appendix

```
#############################################################
#Assignment 1. Spam classification with nearest neighbors
#############################################################
#1.1reading the data
data<-xlsx::read.xlsx("D:/Perriod 2/Machine Learning/Lab/Block 1 lab1/spambase.xlsx",
                      sheetName = "spambase_data")
#diving the data into test and train dataset
n<-dim(data)[1]
set.seed(12345)
id<-sample(1:n, floor(n*0.5))
train<-data[id,]
test<-data[-id,]
#------------------------------------------------------------------------


#1.2fitting the model
fit<-glm(formula = Spam~.,family = binomial(link = 'logit'),data = train)
#predicting the probability of spam for both testing and training data
spam_hut_train<-predict(fit,train,type="response")
spam_hut_test<-predict(fit,test,type="response")
#using 0.5 as thresholld
#predicting the train values
spam_hut_train_1<-spam_hut_train
spam_hut_train_1<-ifelse(spam_hut_train>0.5,1,0)
#confusion matrix and misclassification train
cm_train_1<-table(train$Spam,spam_hut_train_1,dnn=c("True Values","Predicted Vlues"))
mc_train_1<-1-sum(diag(cm_train_1))/sum(cm_train_1)

#predicting the test values
spam_hut_test_1<-spam_hut_test
spam_hut_test_1<-ifelse(spam_hut_test>0.5,1,0)
#confusion matrix and misclassification test
cm_test_1<-table(test$Spam,spam_hut_test_1,dnn=c("True Values","Predicted Vlues"))
mc_test_1<-1-sum(diag(cm_test_1))/sum(cm_test_1)


#Displaying the reults in laetx form and analyzing them
library(knitr)
library(xtable)
t1<-kable(x=cm_train_1,format="latex")
t2<-kable(x=cm_test_1,format="latex")

cat(c("\\begin{table}[!htb]
       \\begin{minipage}{.5\\linewidth}
       \\caption{Confusion Matrix Training Data threshold 0.5}
       \\centering",
       t1,
      "\\end{minipage}%
       \\begin{minipage}{.5\\linewidth}
       \\caption{Confusion Matrix Test Data threshold 0.5}
       \\centering",
```

```
       t2,
      "\\end{minipage}
       \\end{table}"
))

cat("Misclassification Rates LR threshold 0.5")
knitr::kable(x = data.frame("Training" = mc_train_1,
                            "Test"     = mc_test_1,
                            row.names  = c("Misclassification Rate")),
             format = "latex")
#-------------------------------------------------------------------------


#using the treshold 0.8
#training data
spam_hut_train_2<-spam_hut_train
spam_hut_train_2<-ifelse(spam_hut_train>0.8,1,0)
#confusion matrix and misclassification
cm_train_2<-table(train$Spam,spam_hut_train_2,dnn=c("True Values","Precited Values"))
mc_train_2<-1-sum(diag(cm_train_2))/sum(cm_train_2)

#test data
spam_hut_test_2<-spam_hut_test
spam_hut_test_2<-ifelse(spam_hut_test>0.8,1,0)
#confusion matrix and misclassification
cm_test_2<-table(test$Spam,spam_hut_test_2,dnn=c("True Values","Precited Values"))
mc_test_2<-1-sum(diag(cm_test_2))/sum(cm_test_2)
#Displaying the results in latex form and andlyzing them
library(knitr)
library(xtable)
t1<-kable(x=cm_train_2,format="latex")
t2<-kable(x=cm_test_2,format="latex")

cat(c("\\begin{table}[!htb]
       \\begin{minipage}{.5\\linewidth}
       \\caption{Confusion Matrix Training Data threshold 0.8}
       \\centering",
       t1,
      "\\end{minipage}%
       \\begin{minipage}{.5\\linewidth}
       \\caption{Confusion Matrix Test Data threshold 0.8}
       \\centering",
       t2,
      "\\end{minipage}
       \\end{table}"
))

cat("Misclassification Rates LR threshold 0.8")
knitr::kable(x = data.frame("Training" = mc_train_2,
                            "Test"     = mc_test_2,
                            row.names  = c("Misclassification Rate")),
             format = "latex")
#-------------------------------------------------------------------------
```

```
#1.4 K nearest neighbour classification
#Fitting a model using kknn model for K=30
output_test_30<-kknn::kknn(formula=Spam~.,train=train,test=test,k=30)
output_train_30<-kknn::kknn(formula=Spam~.,train=train,test=train,k=30)
fv_test_30<-output_test_30$fitted.values
fv_train_30<-output_train_30$fitted.values
#Classifying as spam or not
fv_test_30<-ifelse(fv_test_30>0.5,1,0)
fv_train_30<-ifelse(fv_train_30>0.5,1,0)
#confusion matrix and misclassification
cm_kknn_train_30<-table(train$Spam,fv_train_30,dnn=c("True Values","Precited Values"))
mc_kknn_train_30<-1-sum(diag(cm_kknn_train_30))/sum(cm_kknn_train_30)
cm_kknn_test_30<-table(test$Spam,fv_test_30,dnn=c("True Values","Precited Values"))
mc_kknn_test_30<-1-sum(diag(cm_kknn_test_30))/sum(cm_kknn_test_30)


#Displaying the results in latex form and andlyzing them
library(knitr)
library(xtable)
t1<-kable(x=cm_kknn_train_30,format="latex")
t2<-kable(x=cm_kknn_test_30,format="latex")

cat(c("\\begin{table}[!htb]
       \\begin{minipage}{.5\\linewidth}
       \\caption{Confusion Matrix Training Data K=30}
       \\centering",
       t1,
      "\\end{minipage}%
       \\begin{minipage}{.5\\linewidth}
       \\caption{Confusion Matrix Test Data K=30}
       \\centering",
       t2,
      "\\end{minipage}
       \\end{table}"
))

cat("Misclassification Rates for kknn K=30")
knitr::kable(x = data.frame("Training" = mc_kknn_train_30,
                            "Test"     = mc_kknn_test_30,
                            row.names  = c("Misclassification Rate")),
             format = "latex")
#-------------------------------------------------------------------------


#1.5 K nearest neighbour classification
#Fitting a model using kknn model for K=1
output_test_1<-kknn::kknn(formula=Spam~.,train=train,test=test,k=1)
output_train_1<-kknn::kknn(formula=Spam~.,train=train,test=train,k=1)
fv_test_1<-output_test_1$fitted.values
fv_train_1<-output_train_1$fitted.values
#Classifying as spam or not
fv_test_1<-ifelse(fv_test_1>0.5,1,0)
```

```
fv_train_1<-ifelse(fv_train_1>0.5,1,0)
#confusion matrix and misclassification
cm_knn_train_1<-table(train$Spam,fv_train_1,dnn=c("True Values","Precited Values"))
mc_knn_train_1<-1-sum(diag(cm_knn_train_1))/sum(cm_knn_train_1)
cm_knn_test_1<-table(test$Spam,fv_test_1,dnn=c("True Values","Precited Values"))
mc_knn_test_1<-1-sum(diag(cm_knn_test_1))/sum(cm_knn_test_1)

#Displaying the results in latex form and andlyzing them
library(knitr)
library(xtable)
t1<-kable(x=cm_knn_train_1,format="latex")
t2<-kable(x=cm_knn_test_1,format="latex")

cat(c("\\begin{table}[!htb]
       \\begin{minipage}{.5\\linewidth}
       \\caption{Confusion Matrix Training Data K=30}
       \\centering",
       t1,
      "\\end{minipage}%
       \\begin{minipage}{.5\\linewidth}
       \\caption{Confusion Matrix Test Data K=30}
       \\centering",
       t2,
      "\\end{minipage}
       \\end{table}"
))

cat("Misclassification Rates for kknn K=30")
knitr::kable(x = data.frame("Training" = mc_knn_train_1,
                            "Test"     = mc_knn_test_1,
                            row.names  = c("Misclassification Rate")),
             format = "latex")
#-------------------------------------------------------------------------

###############################################################################
#Assignment 3. Feature selection by cross-validation in a linear model
###############################################################################
#3.1 Writing the functions.
#linear regression using basic R operations
mylin=function(X,Y, Xpred){
  Xpred1=cbind(1,Xpred)
  X=cbind(1,X)
 #compute beta
  beta=solve(t(X)%*%X)%*%t(X)%*%Y
  Res=Xpred1%*%beta
  return(Res)
}
#Nfold cross-validation
myCV=function(X,Y,Nfolds){
  n=length(Y)
  p=ncol(X)
  suppressWarnings(RNGversion("3.5.9"))
  set.seed(12345)
```

```
  ind=sample(n,n)
  X1=X[ind,]
  Y1=Y[ind]
  sf=(n/Nfolds)
  MSE=numeric(2^p-1)
  Nfeat=numeric(2^p-1)
  Features=list()
  curr=0

  #we assume 5 features.

  for (f1 in 0:1)
    for (f2 in 0:1)
      for(f3 in 0:1)
        for(f4 in 0:1)
          for(f5 in 0:1){
            model= c(f1,f2,f3,f4,f5)
            if (sum(model)==0) next()
            SSE=0

            for (k in 1:Nfolds){
              #compute which indices should belong to current fold
              indices<-(((k-1)*sf)+1):(k*sf)
                    #implement cross-validation for model with
              #features in "model" and iteration i.
              X_test<-X1[indices,which(model==1)]
              X_train<-X1[-indices,which(model==1)]
              Yp<-Y1[indices]
              Y_train<-Y1[-indices]
              Ypred<-mylin(X_train,Y_train,X_test)
              #Get the predicted values for fold 'k',
              #Ypred, and the original values for folf 'k', Yp.
              SSE=SSE+sum((Ypred-Yp)^2)
            }
            curr=curr+1
            MSE[curr]=SSE/n
            Nfeat[curr]=sum(model)
            Features[[curr]]=model
          }

  #plot MSE against number of features
  plot(Nfeat,MSE,xlab = "Number of features")
  i=which.min(MSE)
  return(list(CV=MSE[i], Features=Features[[i]]))
}
#------------------------------------------------------------------------------


#3.2 Testing with swiss data set
library(knitr)
library(xtable)
data("swiss")
swiss<-swiss
```

```
li<-myCV(X=as.matrix(swiss[,2:6]),Y=swiss[[1]],Nfolds=5)
kable(x = data.frame(CV= li[[1]],
                             Features  = paste(li[[2]], collapse = " "),
                             row.names = c("Best Fit Model")),
            format = "latex")
cat("Selected Features = ",
    paste(names(swiss[, 2:6])[which(li$Features == 1)], collapse = ", "))
cat("\n","Correlation matrix for swiss dataset:")
kable(cor(swiss))
#-------------------------------------------------------------

#####################################################
#Assignment 4. Linear regression and regularization
#####################################################
#4.1Import data to R
soil_data<-xlsx::read.xlsx("D:/Perriod 2/Machine Learning/Lab/Block 1 lab1/tecator.xlsx",
                           sheetName = "data")
#plot of Moisture versus Protein.
plot(x=soil_data$Protein,y=soil_data$Moisture,xlab = "Protein",ylab = "Moisture",
     main="Moisture VS Protein",col="blue")
#---------------------------------------------------------------------


#4.3 Diving the data into traning and test data sets
nn<-dim(soil_data)[1]
set.seed(12345)
id<-sample(1:nn, floor(nn*0.5))
soil_train<-soil_data[id,]
soil_test<-soil_data[-id,]
M<-list()
mse_train<-c()
mse_test<-c()

#fitting different models

for(i in 1:6){
  M$i<-lm(soil_train$Moisture~poly(soil_train$Protein,i,raw = TRUE),data = soil_train)
  y_train<-predict(M$i,soil_train,type="response")
  y_test<-predict(M$i,soil_test,type="response")
  mse_train[i]<-mean((y_train-soil_train$Moisture)^2)
  mse_test[i]<-mean((y_test-soil_test$Moisture)^2)
}
i<-1:6
g<-mse_test-mse_train
analyse<-data.frame("Polynomial"=i,"Train"=mse_train,
                    "Test"=mse_test,"Test-Train"=g)
train_mse<-ggplot2::ggplot(analyse,
                           ggplot2::aes(i,mse_train))+
  ggplot2::geom_point(col="red")+
  ggplot2::ggtitle("MSE plot for train data") +
  ggplot2::ylab("Empirical error") +
  ggplot2::xlab("Polynimial degree")
```

```
test_mse<-ggplot2::ggplot(analyse,
                          ggplot2::aes(i,mse_test))+
  ggplot2::geom_point(col="red")+
  ggplot2::ggtitle("MSE plot for test data") +
  ggplot2::ylab("Predicted error") +
  ggplot2::xlab("Polynimial degree")

ggpubr::ggarrange(train_mse,test_mse,
                  ncol = 2, nrow = 1)
#-----------------------------------------------------------------


lm_fs<-lm(Fat~.,data = soil_data[,2:102])
step<-MASS::stepAIC(lm_fs,direction="both",trace = FALSE)
my_pars<-colnames(step$model)
cat("The below features are identified as the most significant : \n")
my_pars[-1]
cat("Number of fetures selected =",(length(my_pars)-1),"\n")
cat("SSE:\n",mean(step$residuals^2))
#-------------------------------------------------------------------------


#4.5Fit a model
model_ridge=glmnet::glmnet(as.matrix(soil_data[,2:101]),
                           soil_data$Fat, alpha=0,family="gaussian")
#plot coefficients and log of lambda
ridge_plot<-plot(model_ridge, xvar="lambda", label=TRUE)
#-------------------------------------------------------------------


#4.6 LASSO regression
model_lasso=glmnet::glmnet(as.matrix(soil_data[,2:101]),
                           soil_data$Fat, alpha=1,family="gaussian")
#plot coefficients and log of lambda
lasso_plot<-plot(model_lasso, xvar="lambda", label=TRUE)
#-----------------------------------------------------------------------------


cv_lasso<-glmnet::cv.glmnet(as.matrix(soil_data[,2:101]),
                            soil_data$Fat,
                            alpha=1,
                            family="gaussian",
                            lambda = seq(0,10,0.001))
plot(cv_lasso)

# Format results in latex
lambda.min <- which(cv_lasso$lambda == cv_lasso$lambda.min)
lambda.1se <- which(cv_lasso$lambda == cv_lasso$lambda.1se)
knitr::kable(
  x = data.frame(Lambda    = c(cv_lasso$lambda.min, cv_lasso$lambda.1se),
                 Measure   = cv_lasso$cvm[c(lambda.min, lambda.1se)],
                 SE        = cv_lasso$cvsd[c(lambda.min, lambda.1se)],
                 Nonzero   = cv_lasso$nzero[c(lambda.min, lambda.1se)],
```

```
                row.names = c("Min", "1se")),
  format = "latex")
#-------------------------------------------------------------------------

# Calculate MSE for model with selected features
AICmodel <- lm(Fat ~ Channel1  + Channel2  + Channel4  + Channel5  +
                     Channel7  + Channel8  + Channel11 + Channel12 + Channel13 +
                     Channel14 + Channel15 + Channel17 + Channel19 + Channel20 +
                     Channel22 + Channel24 + Channel25 + Channel26 + Channel28 +
                     Channel29 + Channel30 + Channel32 + Channel34 + Channel36 +
                     Channel37 + Channel39 + Channel40 + Channel41 + Channel42 +
                     Channel45 + Channel46 + Channel47 + Channel48 + Channel50 +
                     Channel51 + Channel52 + Channel54 + Channel55 + Channel56 +
                     Channel59 + Channel60 + Channel61 + Channel63 + Channel64 +
                     Channel65 + Channel67 + Channel68 + Channel69 + Channel71 +
                     Channel73 + Channel74 + Channel78 + Channel79 + Channel80 +
                     Channel81 + Channel84 + Channel85 + Channel87 + Channel88 +
                     Channel92 + Channel94 + Channel98 + Channel99, soil_data)
Ypred.AIC <- predict(AICmodel, soil_data)
MSE.AIC   <- mean((Ypred.AIC - soil_data$Fat) ^ 2)

# Format results in latex
knitr::kable(
  x = data.frame(Measure  = c(MSE.AIC, cv_lasso$cvm[lambda.1se]),
                 Features = c(length(step$coefficients) - 1, cv_lasso$nzero[lambda.1se]),
                 row.names = c("step AIC", "CV Lasso")),
  format = "latex")
#-------------------------------------------------------------------------B1 L1 ENDS
########################################
#Assignment 2. Analysis of credit scoring
########################################
# 2.1 Reading the data into R from the excel and dividing into train,validation and test
credit_scores<-xlsx::read.xlsx(
  "D:/Perriod 2/Machine Learning/Lab/Block 1 lab2/creditscoring.xls",
                              sheetName = "credit")
#tarining data
n<-nrow(credit_scores)
suppressWarnings(RNGversion("3.5.1"))
set.seed(12345)
id1<-sample(1:n, floor(n*0.5))
credit_train<-credit_scores[id1,]
#testing and validation
cred<-credit_scores[-id1,]
idrem<-setdiff(1:n,id1)
suppressWarnings(RNGversion("3.5.1"))
set.seed(12345)
id2<-sample(idrem, n*0.25)
id3<-setdiff(idrem,id2)
credit_test<-credit_scores[id2,]
credit_validation<-credit_scores[id3,]
#-------------------------------------------------------------------------

# a.Deviance
```

```
library(tree)
credit_dt_deviance<-tree(good_bad~.,
                         data = credit_train,
                         split = "deviance",
                         method = "recursive.partition")
y_hut_test_deviance<-predict(credit_dt_deviance,credit_test,type="class")
y_hut_train_deviance<-predict(credit_dt_deviance,credit_train,type = "class")
cm_train_deviance<-table(credit_train$good_bad,y_hut_train_deviance,
                          dnn=c("Actual","Predicted"))
cm_test_deviance<-table(credit_test$good_bad,y_hut_test_deviance,
                        dnn=c("Actual","Predicted"))
mc_train_deviance<-1-sum(diag(cm_train_deviance))/sum(cm_train_deviance)
mc_test_deviance<-1-sum(diag(cm_test_deviance)/sum(cm_test_deviance))

#Displaying the results in latex form and andlyzing them
library(knitr)
library(xtable)
t1<-kable(x=cm_train_deviance,format="latex")
t2<-kable(x=cm_test_deviance,format="latex")

cat(c("\\begin{table}[!htb]
       \\begin{minipage}{.5\\linewidth}
       \\caption{Confusion Matrix Training Data(Deviance)}
       \\centering",
       t1,
      "\\end{minipage}%
       \\begin{minipage}{.5\\linewidth}
       \\caption{Confusion Matrix Test Data(Deviance)}
       \\centering",
       t2,
      "\\end{minipage}
       \\end{table}"
))

cat("Misclassification Rates with Deviance method")
knitr::kable(x = data.frame("Training" = mc_train_deviance,
                            "Test"     = mc_test_deviance,
                            row.names  = c("Misclassification Rate")),
             format = "latex")
#-------------------------------------------------------------------------

# b.gini method
credit_dt_gini<-tree(good_bad~.,
                     data = credit_train,
                      split = "gini",
                      method = "recursive.partition")
y_hut_train_gini<-predict(credit_dt_gini,credit_train,type="class")
y_hut_test_gini<-predict(credit_dt_gini,credit_test,type = "class")
cm_train_gini<-table(credit_train$good_bad,y_hut_train_gini,
                     dnn=c("Actual","Predicted"))
cm_test_gini<-table(credit_test$good_bad,y_hut_test_gini,
                    dnn = c("Actual","Predicted"))
mc_train_gini<-1-sum(diag(cm_train_gini))/sum(cm_train_gini)
```

```r
mc_test_gini<-1-sum(diag(cm_test_gini)/sum(cm_test_gini))

#Displaying the results and analyzing them.
t1<-kable(x=cm_train_gini,format="latex")
t2<-kable(x=cm_test_gini,format="latex")

cat(c("\\begin{table}[!htb]
       \\begin{minipage}{.5\\linewidth}
       \\caption{Confusion Matrix Training Data(Gini)}
       \\centering",
       t1,
      "\\end{minipage}%
       \\begin{minipage}{.5\\linewidth}
       \\caption{Confusion Matrix Test Data(Gini)}
       \\centering",
       t2,
      "\\end{minipage}
       \\end{table}"
))

cat("Misclassification Rates for Gini method")
knitr::kable(x = data.frame("Training" = mc_train_gini,
                            "Test"     = mc_test_gini,
                            row.names  = c("Misclassification Rate")),
             format = "latex")
#-------------------------------------------------------------------------


library(ggplot2)
library(tree)
#2.3 choose the optimal tree depth
train_score<-numeric(9)
valid_score<-numeric(9)
#fitting a model using tree
fit<-tree(good_bad~.,data = credit_train)
#vizualizing the tree
#plot(x=fit,y=NULL)
#text(x=fit,pretty=NULL)
for(i in 2:9){
  #refining the tree by returning the best model for i number of nodes
  refined_fit<-prune.tree(fit,best=i)
  train_score[i]<-deviance(refined_fit)
  pred<-predict(refined_fit,newdata=credit_validation,type="tree")
  valid_score[i]<-deviance(pred)
}
#graphs of deviances for the training and the validation data
#on the number of leaves.

plot(2:9, train_score[2:9], type="b", col="red",ylim = c(250,590),lwd="2",
     xlab = "Number of leaves(Size)",ylab = "Deviance", main = "Deviance graph")
points(2:9, valid_score[2:9], type="b", col="blue",lwd="2")
legend("left", col=c("red", "blue"), legend=c("Traning Score", "Validation Score"),
       lwd=c(2,2))
```

```r
#Report the optimal tree,report it's depth and the variables used by the tree.
#Interpret the information provided by the tree structure.
best_tree<-prune.tree(fit,best=4)
y_hut_test_prune<-predict(best_tree,credit_test,type="class")
#Estimate the misclassification rate for the test data.
cm_test_prune<-table(credit_test$good_bad,y_hut_test_prune,
                     dnn=c("Actual","Predicted"))
mc_test_prune<-1-sum(diag(cm_test_prune))/sum(cm_test_prune)
cat(" Best tree test misclassification rate :",mc_test_prune,"\n")
plot(x=best_tree,y=NULL)
text(x=best_tree,pretty=NULL)
#-------------------------------------------------------------------------------

library(e1071)
#2.4 Training data to perform classification using Naïve Bayes
naive_fit<-naiveBayes(good_bad~.,data = credit_train)
#predicting for the training and test data
y_hut_train_naive<-predict(naive_fit,newdata=credit_train,type="class")
y_hut_test_naive<-predict(naive_fit,newdata=credit_test,type="class")
#Confusion matrices and misclassification rates for train and test
cm_train_naive<-table(credit_train$good_bad,y_hut_train_naive,
                      dnn = c("Actual","Predicted"))
mc_train_naive<-1-sum(diag(cm_train_naive))/sum(cm_train_naive)
cm_test_naive<-table(credit_test$good_bad,y_hut_test_naive,
                     dnn = c("Actual","Predicted"))
mc_test_naive<-1-sum(diag(cm_test_naive))/sum(cm_test_naive)
fpr_nb<-cm_test_naive[1,2]/sum(cm_test_naive[1,])
#Displaying the results and analyzing them.
t1<-kable(x=cm_train_naive,format="latex")
t2<-kable(x=cm_test_naive,format="latex")

cat(c("\\begin{table}[!htb]
       \\begin{minipage}{.5\\linewidth}
       \\caption{Confusion Matrix Training Data(Naive Bayes)}
       \\centering",
       t1,
      "\\end{minipage}%
       \\begin{minipage}{.5\\linewidth}
       \\caption{Confusion Matrix Test Data(Naive Bayes)}
       \\centering",
       t2,
      "\\end{minipage}
       \\end{table}"
))

cat("Misclassification Rates for Naive Bayes")
knitr::kable(x = data.frame("Training" = mc_train_naive,
                            "Test"     = mc_test_naive,
                            row.names  = c("Misclassification Rate")),
             format = "latex")
#-------------------------------------------------------------------------------
#2.5 ROC curve
j<-1
```

```
tpr_tree<-numeric()
fpr_tree<-numeric()
tpr_naive<-numeric()
fpr_naive<-numeric()
y_opt<-predict(best_tree,newdata=credit_test)
y_opt<-as.data.frame(y_opt)
y_nb<-predict(naive_fit,newdata=credit_test,type="raw")
y_nb<-as.data.frame(y_nb)
for(pi in seq(0.05, 0.95, 0.05)){
  #pruned tree
  y_piopt<-ifelse(y_opt$good>pi,1,0)
  cm_piopt<-table(credit_test$good_bad,y_piopt,
                        dnn=c("Actual","Predicted"))

  #naive bayes
  y_pinb<-ifelse(y_nb$good>pi,1,0)
  cm_pinb<-table(credit_test$good_bad,y_pinb,
                           dnn=c("Actual","Predicted"))
  tryCatch({
    #tpr and fpr for naive bayes
    tpr_naive[j]<-cm_pinb["good","1"]/sum(cm_pinb["good",])
    fpr_naive[j]<-cm_pinb["bad","1"]/sum(cm_pinb["bad",])
    #tpr and fpr for opt
    tpr_tree[j]<-cm_piopt["good","1"]/sum(cm_piopt["good",])
    fpr_tree[j]<-cm_piopt["bad","1"]/sum(cm_piopt["bad",])
    j<-j+1
  },
  error = function(e) {}
  )
}
#plotting the fpr vs tpr
plot(x=fpr_naive,y=tpr_naive,xlab = "False postive rate",
     ylab = "True positive rate",main = "ROC curve",type = "o",
     col="red",xlim = c(0,1),ylim = c(0,1))
points(x=fpr_tree,y=tpr_tree,col="blue",type = "o")
legend("topleft",c("Naive bayes","Decision Tree"),col =c("red","blue"))
#2.6 naive bayes with loss matrix
y_hut_train_naive_raw<-predict(naive_fit,credit_train,type="raw")
y_hut_test_naive_raw<-predict(naive_fit,credit_test,type="raw")

y_hut_train_naive_raw[,1]<-y_hut_train_naive_raw[,1]*10
y_hut_test_naive_raw[,1]<-y_hut_test_naive_raw[,1]*10
y_hut_train_lmnaive<-
  ifelse(y_hut_train_naive_raw[,2]>y_hut_train_naive_raw[,1],"good", "bad")
y_hut_test_lmnaive<-
  ifelse(y_hut_test_naive_raw[,2]>y_hut_test_naive_raw[,1],"good", "bad")
cm_train_lmnaive<-table(credit_train$good_bad,y_hut_train_lmnaive,
                        dnn=c("Actual","Predicted"))
mc_train_lmnaive<-1-sum(diag(cm_train_lmnaive))/sum(cm_train_lmnaive)
cm_test_lmnaive<-table(credit_test$good_bad,y_hut_test_lmnaive,
                       dnn=c("Actual","Predicted"))
mc_test_lmnaive<-1-sum(diag(cm_test_lmnaive))/sum(cm_test_lmnaive)
fpr_nblm<-cm_test_lmnaive[1,2]/sum(cm_test_lmnaive[1,])
```

```
#Displaying the results and analyzing them.

t1<-kable(x=cm_train_lmnaive,format="latex")
t2<-kable(x=cm_test_lmnaive,format="latex")
cat(c("\\begin{table}[!htb]
       \\begin{minipage}{.5\\linewidth}
       \\caption{Confusion Matrix Train Data(NB with LM)}
       \\centering",
       t1,
      "\\end{minipage}%
       \\begin{minipage}{.5\\linewidth}
       \\caption{Confusion Matrix Test Data(NB with LM)}
       \\centering",
       t2,
      "\\end{minipage}
       \\end{table}"
))


cat("Parameters to compare step 4 and 6 :")
knitr::kable(x = data.frame("Naive Bayes"=c(mc_test_naive,fpr_nb),
                            "Naive Bayes with LM"= c(mc_test_lmnaive,fpr_nblm),
                            row.names  = c("Misclassification Rate",
                                           "False Positive Rate")),
             format = "latex")
#-------------------------------------------------------------------------------

library(tree)
library(boot)
# 3.1 importing the data into R and orderding with increase in MET
state_data_un<-read.csv2(file = "D:/Perriod 2/Machine Learning/Lab/Block 1 lab2/state.csv",sep=";")
#Odereding the data as per the lab.
state_data<-state_data_un[order(state_data_un$MET),]
#plotting the dependednt variable with respect to independent.
plot(state_data$EX,state_data$MET,
     xlab="% population in standard metropolitan areas",
     ylab="Per capita state and local public expenditures ($)",
     main="MET vs EX",
     lwd="4")
#-------------------------------------------------------------------------------
# 3.2 regression model with tree
#fitting a tree with minimum number of leaves =8
tree_model<-tree(EX~MET,data = state_data,
              control=tree.control(nobs=nrow(state_data),minsize = 8))
#Performing cross validation for the fitted model
#to find the best number of nodes
cv_tree_model<-cv.tree(tree_model)
#plotting the deviances
plot(cv_tree_model$size,cv_tree_model$dev,
     xlab="Size",
     ylab = "Deviance",
     main="Deviance plot",
     lwd="2",
```

```
     col="red",
     type="o")
#reporting the optimal tree
cat("The optimal tree is for the model:",which.min(cv_tree_model$dev))
#refining the tree to best number of nodes
refined_tree_model<-prune.tree(tree_model,best=3)
plot(refined_tree_model)
text(x=refined_tree_model,pretty=NULL)
#predicting with the best model
y_hut_rftree<-predict(refined_tree_model,data=state_data)
residual<-residuals(refined_tree_model)
#ploting the original data vs the predicted data
plot(y_hut_rftree,ylab="Fitted data",
     ylim = c(150,500),col="red",type = "o",
     main = "Orignal and Fitted data",lwd="2")
points(state_data$EX,col="blue",type = "l",lwd="2")
legend("top",c("Fitted data","Original data"),col = c("red","blue"))
#histogram of residuals
hist(residual, prob=TRUE,xlab = "residual",
main = "Histogram of residual ")
lines(density(residual),lwd=3,
col="red")
#-------------------------------------------------------------------------
# 3.3 95% confidence bands for non parametric boot strap
# computing bootstrap samples
f=function(data, ind){
  data1=data[ind,]# extract bootstrap sample
  res=tree(EX~MET, data=data1) #fit linear model
  #predict values for all Area values from the original data
  metP=predict(res,newdata=data)
  return(metP)
}
res<-boot(data=state_data,
         statistic=f,
         R=1000) #make bootstrap
#comupting 95% confidence bands
e<-envelope(res)
boot_tree<-tree(EX~MET,
                data=state_data,
                control=tree.control(nobs=nrow(state_data),minsize = 8))
metp<-predict(boot_tree)
plot(state_data$MET,state_data$EX,pch=21,bg="orange")#ploting the data points
points(state_data$MET,metp,type="l")#plotting the predicted values
points(state_data$MET,e$point[2,],type="l",col="blue")
points(state_data$MET,e$point[1,],type="l",col="red")
legend("top",c("Uppper CI","Boot Strap model","Lower CI"),
       col=c("red","black","blue"))
width_npb<-mean(e$point[1,]-e$point[2,])
var_npb<-var(e$point[1,]-e$point[2,])
#-------------------------------------------------------------------------
cat("Comparing the Non-parametric and parametric bootstrap :")
knitr::kable(x = data.frame("Non-parametric bootstrap"=c(width_npb,var_npb),
                            row.names  = c("Mean width",
```

```
                                          "Variance width")),
             format = "latex")
#-------------------------------------------------------------------------------
# 3.4
mle <- prune.tree(tree(EX ~ MET, state_data, minsize = 8), best = 3)

rng<- function(data, mle) {
  datanew <- data.frame(EX = data$EX, MET = data$MET)
  n       <- length(data$EX)
  # Generate new Expenditure
  datanew$EX <- rnorm(n, predict(mle, newdata = datanew), sd(residuals(mle)))
  return(datanew)
}

fp <- function(data){
  # Fit regression tree
  regtree <- prune.tree(tree(EX ~ MET, data, minsize = 8), best = 3)
  # Predict values for all MET values from the original data
  exPredict <- predict(regtree, newdata = state_data)
  return(exPredict)
}
# Make bootstrap
para_bootstrap = boot(state_data,
                      statistic = fp,
                      R = 1000,
                      mle = mle,
                      ran.gen = rng,
                      sim = "parametric")
ev<-envelope(para_bootstrap)
para_boot_tree<-tree(EX~MET,data=state_data,
                control=tree.control(nobs=nrow(state_data),minsize = 8))
metp<-predict(para_boot_tree)
plot(state_data$MET,state_data$EX,pch=21,bg="orange")#ploting the data points
points(state_data$MET,metp,type="l")#plotting the predicted values
points(state_data$MET,ev$point[2,],type="l",col="blue")
points(state_data$MET,ev$point[1,],type="l",col="red")
legend("top",c("Uppper CI","Boot Strap model","Lower CI"),
       col=c("red","black","blue"))
width_pb<-mean(ev$point[1,]-ev$point[2,])
var_pb<-var(ev$point[1,]-ev$point[2,])
cat("Comparing the Non-parametric and parametric bootstrap :")
knitr::kable(x = data.frame("Non-parametric bootstrap"=c(width_npb,var_npb),
                            "Paramaetric bootstrap"= c(width_pb,var_pb),
                            row.names  = c("Mean width",
                                           "Variance width")),
             format = "latex")
#-------------------------------------------------------------------------------
library(ggplot2)
# 4.1 Doing PCI Analysis
spectra<-read.csv2(file = "D:/Perriod 2/Machine Learning/Lab/Block 1 lab2/NIRspectra.csv",sep=";")
spectrum<-spectra
spectrum$Viscosity<-c()
#doing a PCA analysis
```

```r
pca_res<-prcomp(spectrum)
#getting the eigen vectors
lambda<-pca_res$sdev^2#represents the variation along that axis
#proportion of variation with respect to each feature
percentage_variation <- round((lambda/sum(lambda))*100, 1)
screeplot(pca_res)
#PLotting the PCAs
pca.data <- data.frame(label=1:nrow(pca_res$x),
                       X=pca_res$x[,1],
                       Y=pca_res$x[,2])
unusual_points<-ifelse((pca_res$x[,1]>4*IQR(pca_res$x[,1])|
                          pca_res$x[,2]>4*IQR(pca_res$x[,2])),
                       which((pca_res$x[,1]>4*IQR(pca_res$x[,1])|
                                pca_res$x[,2]>4*IQR(pca_res$x[,2]))),NA)
ggplot(data=pca.data, aes(x=X, y=Y))+geom_point()+
  geom_text(aes(label=unusual_points),col="red",hjust=0,vjust=0)+
  xlab(paste("PC1 : ", percentage_variation[1], "%", sep=""))+
  ylab(paste("PC2 : ", percentage_variation[2], "%", sep="")) +
  theme_bw() +
  ggtitle("Score Plot for PC 1 VS PC2")
#-------------------------------------------------------------------------
# 4.2 Trace plots for the selected
trace<-pca_res$rotation
plot(trace[,1],main = "Traceplot PCI 1",ylab ="Loading Values")
plot(trace[,2],main = "Traceplot PCI 2",ylab ="Loading Values")
#4.3a Trace Plot
library("fastICA")
set.seed(12345)
ica_obj = fastICA(spectrum, n.comp=2, verbose=TRUE)
#X=SA;XK=SA;S=XKW
#K Gives the first n.comp PCA components
W<-ica_obj$W
K<-ica_obj$K
WK<-K%*%W
# Making traceplot
plot(WK[,1],
     main="Traceplot, ICA1",
     xlab="index",
     ylab="ICA1",
     type="b",
     col="red")
plot(WK[,2],
     main="Traceplot, ICA2",
     xlab="index",
     ylab="ICA2",
     type="b",
     col="red")
ica.data <- data.frame(label=1:nrow(ica_obj$S),
                       X=ica_obj$S[,1],
                       Y=ica_obj$S[,2])
#-------------------------------------------------------------------------
#4.3b plot of the scores of the first two latent features
ggplot(data=ica.data, aes(x=X, y=Y))+geom_point()+
```

```
  xlab("IC1")+
  ylab("IC2") +
  theme_bw() +
  ggtitle("Score Plot for IC1 VS IC2")
#-------------------------------------------------------------------------
library("geosphere")
library("hms")
set.seed(1234567890)
# reading the places data
stations <- read.csv(
  file = "D:/Perriod 2/Machine Learning/Lab/Block 1 lab3/stations.csv",
  header = TRUE)
# reading temperature data
temps <- read.csv(
  file = "D:/Perriod 2/Machine Learning/Lab/Block 1 lab3/temps50k.csv",
  header = TRUE)
set.seed(1234567890)
st <- merge(stations, temps, by = "station_number")
# Format Date and Time data
st$date <- as.Date(st$date)
st$time <- strptime(st$time, "%H")
#kernel widths
h_distance <-80000 # These three values are up to the students
h_date <-7
h_time <-4
#lat long of the point to predict
lat <- 58.4274 # The point to predict (up to the students)
long <- 14.826
d <- as.Date("2008-11-08") # Date to predict
times <- c("04:00:00", "06:00:00", "08:00:00", "10:00:00", # Times to predict
           "12:00:00", "14:00:00", "16:00:00", "18:00:00",
           "20:00:00", "22:00:00", "24:00:00")
# formatting the time
times <- strptime(times, "%H")
#taking all the values prior to the selected date
st_prior <- st[st$date<d, ]
#Temperatureas predictions for sum and product kernel
temp_sum<-numeric(length(times))
temp_prod<-numeric(length(times))
#-------------------------------------------------------------------------------------
#distance from a station to the point of interest
diff_dist<-distHaversine(p1=st_prior[,c("latitude", "longitude")], p2 = c(lat,long))
#distance between the day a temperature measurement
#was made and the day of interest
diff_date<-abs(as.numeric(difftime(d,st_prior$date)))
for(i in 1:length(diff_date)){
  diff_date[i]<-min(as.numeric(diff_date[i])/365,
                    365-(as.numeric(diff_date[i])/360))
}
#distance between the hour of the day a temperature measurement
#was made and the hour of interest
diff_time <- sapply(times, function(x) {
  timediff <- difftime(x, st_prior$time, units = "hours")
```

```
  timediff <- sapply(abs(as.numeric(timediff)), function(y) {
    min(y, 24 - y)
  })
  return(timediff)
})
#-------------------------------------------------------------------------
#Gaussian kernels
#Distance Kernel
distance_kernel <- exp(-1*(diff_dist/h_distance)^2)
#Date Kernel
date_kernel <- exp(-1*(diff_date/h_date)^2)
#Time Kernel
time_kernel<-exp(-1*(diff_time/h_time)^2)
#Plotting the gaussian Kernels
par(mfrow = c(1,3))
order_dist <- diff_dist[order(diff_dist)]
order_disk <- exp(-1 *(order_dist/ h_distance) ^ 2)
plot(x = unique(order_dist),
     y = unique(order_disk),
     xlab = "Difference in distance",
     ylab = "Gaussian Kernel",
     type = "o",
     col = "red")
order_date <- diff_date[order(diff_date)]
order_datk <- exp(-1 * (order_date / h_date) ^ 2)
plot(x = unique(order_date),
     y = unique(order_datk),
     xlab = "Difference in date",
     ylab = "Gaussian Kernel",
     type = "o",
     col = "blue")
order_time <- diff_time[order(diff_time[, 1]), 1]
order_timk <- exp(-(order_time/ h_time)^2)
plot(x = unique(order_time),
     y = unique(order_timk),
     xlab = "Difference in Time",
     ylab = "Gaussian Kernel",
     type = "o",
     col = "green")
#-------------------------------------------------------------------------
#Adding the three kernels
kernel_sum<-distance_kernel+date_kernel+time_kernel
#Predicting the temperature
for(i in 1:ncol(kernel_sum)){
  temp_sum[i]<-sum(kernel_sum[, i]*st_prior$air_temperature) / sum(kernel_sum[, i])
}

#Product of three kernel
kernel_prod<-distance_kernel*date_kernel*time_kernel
#predicting temperature
for(i in 1:ncol(kernel_sum)){
  temp_prod[i]<-sum(kernel_prod[, i]*st_prior$air_temperature) / sum(kernel_prod[, i])
}
```

```r
par(mfrow = c(1,2))
plot(x = times,
     y = temp_sum,
     ylab = "Temperature predictions",
     type = "o",
     col = "red",
     main="Additive Model",lwd="2")
plot(x = times,
     y = temp_prod,
     ylab = "Temperature predictions",
     type = "o",
     col = "green",
     main="Multiplication model",lwd="2")
#2.1 model selection
library(kernlab)
#SVM function
svm_function<-function(tr,c,te){
  svm_obj<-ksvm(type~.,
                    data=tr,
                    kernel="rbfdot",
                    type="C-svc",C=c,width=0.05)
  #getting the predicted values of the test data
  y_hut_pred<-predict(object=svm_obj,
                          newdata=te,
                          type="response")
  #confusion matrix
  cm<-table(te$type,y_hut_pred,
                dnn=c("True Values","Predicted values"))
  #True positive rate and false positive rate
  tpr<-cm[1,1]/sum(cm[,1])
  fpr<-cm[1,2]/sum(cm[,2])
  #Misclassification rate
  mc<-1-sum(diag(cm))/sum(cm)
  return(list(CM=cm,TPR=tpr,FPR=fpr,MC=mc,model=svm_obj))
}
data(spam)
n<-nrow(spam)
#grep("type",colnames(spam))
#Randomizing the data
ind=sample(n,n)
spam<-spam[ind,]

#SVM with C=0.5
output0.5<-svm_function(tr=spam[1:floor(n*0.5),],#half of the data for traning
              c=0.5,
              te=spam[(floor(n*0.5)+1):floor(n*0.75),])#next 1/4 of the data for test

#SVM with C=1
output1<-svm_function(tr=spam[1:floor(n*0.5),],#half of the data for traning
              c=1,
              te=spam[(floor(n*0.5)+1):floor(n*0.75),])#next 1/4 of the data for test
```

```
#SVM with C=5
output5<-svm_function(tr=spam[1:floor(n*0.5),],#half of the data for traning
              c=5,
              te=spam[(floor(n*0.5)+1):floor(n*0.75),])#next 1/4 of the data for test
#C=1 was selcted, now with we train with 75% of the data train+valid data
#Estimating the generalization error
svm_gen<-svm_function(tr=spam[1:floor(n*0.75),],
                      c=1,
                      te=spam[(floor(n*0.75)+1):n,])
cat("Generalized Error : ",svm_gen$MC)
#Final SVM is when tranined with the whole data
fin<-svm_function(tr=spam,
                  c=1,
                  te=spam)$model
print(fin)
```