**LAB1 Block1**: **Assignment 1. Spam classification with nearest neighbors**

The data file **spambase.xlsx** contains information about the frequency of various words, characters etc for a total of 2740 e-mails. Furthermore, these e-mails have been manually classified as spams (spam = 1) or regular e-mails (spam = 0).
1. Import the data into R and divide it into training and test sets (50%/50%).
2. Use logistic regression (functions *glm(), predict()*) to classify the training and test data by the classification principle

$Y$hat=1 if $pp(Y=1|X)$>0.5,otherwise $Y$hat=0
and report the confusion matrices (use table()) and the misclassification rates for training and test data. Analyse the obtained results.
3. Use logistic regression to classify the test data by the classification principle

$Y$hat=1 if $pp(Y=1|X)$>0.9,otherwise $Y$hat =0
and report the confusion matrices (use table()) and the misclassification rates for training and test data. Compare the results. What effect did the new rule have?
4. Use standard classifier *kknn()* with *K=30* from package **kknn**, report the the misclassification rates for the training and test data and compare the results with step 2.

5. Repeat step 4 for *K=1* and compare the results with step 4. What effect does the decrease of K lead to and why?

**Assignment 2. Inference about lifetime of machines**

The data file **machines.xlsx** contains information about the lifetime of certain machines, and the company is interested to know more about the underlying process in order to determine the warranty time. The variable is following:
• Length: shows lifetime of a machine

1. Import the data to R.
2. Assume the probability model $p(x|\theta) = \theta e^{-\theta x}$ for $\boldsymbol{x}$=*Length* in which observations are independent and identically distributed. What is the distribution type of $x$? Write a function that computes the log-likelihood $\log p(\boldsymbol{x}|\theta)$ for a given $\theta$ and a given data vector $\boldsymbol{x}$. Plot the curve showing the dependence of log-likelihood on $\theta$ where the entire data is used for fitting. What is the maximum likelihood value of $\theta$ according to the plot?
3. Repeat step 2 but use only 6 first observations from the data, and put the two log-likelihood curves (from step 2 and 3) in the same plot. What can you say about reliability of the maximum likelihood solution in each case?
4. Assume now a Bayesian model with $p(x|\theta) = \theta e^{-\theta x}$ and a prior $p(\theta) = \lambda e^{-\lambda\theta}, \lambda = 10$. Write a function computing $\mathrm{l}(\theta) = \log\big(p(\boldsymbol{x}|\theta)p(\theta)\big)$. What kind of measure is actually computed by this function? Plot the curve showing the dependence of $l(\theta)$ on $\theta$ computed using the entire data and overlay it with a plot from step 2. Find an optimal $\theta$ and compare your result with the previous findings.
5. Use $\theta$ value found in step 2 and generate 50 new observations from $p(x|\theta) = \theta e^{-\theta x}$ (use standard random number generators). Create the histograms of the original and the new data and make conclusions.

**Assignment 3. Feature selection by cross-validation in a linear model.**
1. Implement an R function that performs feature selection (best subset selection) in linear regression by using k-fold cross-validation without using any specialized function like lm() **(use only basic R functions**). Your function should depend on: • X: matrix containing X measurements.
• Y: vector containing Y measurements
• Nfolds: number of folds in the cross-validation.

You may assume in your code that matrix X has 5 columns. The function should plot the CV scores computed for various feature subsets against the number of features, and it should also return the optimal subset of features and the corresponding cross-validation (CV) score. Before splitting into folds, the data should be permuted, and the seed 12345 should be used for that purpose.
2. Test your function on data set **swiss** available in the standard R repository: • Fertility should be Y
• All other variables should be X
• Nfolds should be 5
Report the resulting plot and interpret it. Report the optimal subset of features and comment whether it is reasonable that these specific features have largest impact on the target.

**Assignment 4. Linear regression and regularization**
The Excel file **tecator.xlsx** contains the results of study aimed to investigate whether a near infrared absorbance spectrum can be used to predict the fat content of samples of meat. For each meat sample the data consists of a 100 channel spectrum of absorbance records and the levels of moisture (water), fat and protein. The absorbance is -log10 of the transmittance measured by the spectrometer. The moisture, fat and protein are determined by analytic chemistry.
1. Import data to R and create a plot of Moisture versus Protein. Do you think that these data are described well by a linear model?

2. Consider model $MMii$ in which Moisture is normally distributed, and the expected Moisture is a polynomial function of Protein including the polynomial terms up to power $ii$ (i.e M1 is a linear model, M2 is a quadratic model and so on). Report a probabilistic model that describes $MMii$. Why is it appropriate to use MSE criterion when fitting this model to a training data?
3. Divide the data into training and validation sets( 50%/50%) and fit models $MMii$,$ii$=1…6. For each model, record the training and the validation MSE and present a plot showing how training and validation MSE depend on *i* (write some R code to make this plot). Which model is best according to the plot? How do the MSE values change and why? Interpret this picture in terms of bias-variance tradeoff.

Use the entire data set in the following computations:
4. Perform variable selection of a linear model in which *Fat* is response and *Channel1-Channel100* are predictors by using stepAIC. Comment on how many variables were selected.
5. Fit a Ridge regression model with the same predictor and response variables. Present a plot showing how model coefficients depend on the log of the penalty factor $\lambda\lambda$ and report how the coefficients change with $\lambda\lambda$.
6. Repeat step 5 but fit LASSO instead of the Ridge regression and compare the plots from steps 5 and 6. Conclusions?
7. Use cross-validation to find the optimal LASSO model (make sure that case $\lambda\lambda$=0 is also considered by the procedure) , report the optimal $\lambda\lambda$ and how many variables were chosen by the model and make conclusions. Present also a plot showing the dependence of the CV score and comment how the CV score changes with $\lambda\lambda$.
8. Compare the results from steps 4 and 7.

**LAB2 Block1 : Assignment 1. LDA and logistic regression**

The data file **australian-crabs.csv** contains measurements of various crabs, such as Frontal lobe, Rear width and others
1. Use australian-crabs.csv and make a scatterplot of carapace length (CL) versus rear width (RW) where observations are colored by Sex. Do you think that this data is easy to classify by linear discriminant analysis? Motivate your answer.
2. Make LDA analysis with target Sex and features CL and RW and proportional prior by using lda() function in package MASS. Make a scatter plot of CL versus RW colored by the predicted Sex and compare it with the plot in step 1. Compute the misclassification error and comment on the quality of fit.
3. Repeat step 2 but use priors $pp$(Male)=0.9,$p$(Female)=0.1 instead. How did the classification result change and why?
4. Make a similar kind of classification by logistic regression (use function glm()), plot the classified data and compute the misclassification error. Compare these results with the LDA results. Finally, report the equation of the decision boundary and draw it in the plot of the classified data.

**Assignment 2. Analysis of credit scoring**
The data file **creditscoring.xls** contains data retrieved from a database in a private enterprise. Each row contains information about one customer. The variable good/bad indicates how the customers have managed their loans. The other features are potential predictors. Your task is to derive a prediction model that can be used to predict whether or not a new customer is likely to pay back the loan.
1. Import the data to R and divide into training/validation/test as 50/25/25: use data partitioning code specified in Lecture 1e.
2. Fit a decision tree to the training data by using the following measures of impurity a. Deviance
b. Gini index
and report the misclassification rates for the training and test data. Choose the measure providing the better results for the following steps.
3. Use training and validation sets to choose the optimal tree depth. Present the graphs of the dependence of deviances for the training and the validation data on the number of leaves. Report the optimal tree, report it's depth and the variables used by the tree. Interpret the information provided by the tree structure. Estimate the misclassification rate for the test data.
4. Use training data to perform classification using Naïve Bayes and report the confusion matrices and misclassification rates for the training and for the test data. Compare the results with those from step 3.
5. Use the optimal tree and the Naïve Bayes model to classify the test data by using the following principle:

$Y$hat=1 if $p(Y='good'|X)>\pi$, otherwise$Y$hat =0
where $\pi\pi$=0.05,0.1,0.15,…0.9,0.95. Compute the TPR and FPR values for the two models and plot the corresponding ROC curves. Conclusion?
6. Repeat Naïve Bayes classification as it was in step 4 but use the following loss matrix:

$$L = \begin{matrix} \\ Observed \\ \\ \end{matrix} \begin{matrix} \quad Predicted \\ good \\ bad \end{matrix} \begin{pmatrix} 0 & 1 \\ 10 & 0 \end{pmatrix}$$

and report the confusion matrix for the training and test data. Compare the results with the results from step 4 and discuss how the rates has changed and why.
**Assignment 3. Uncertainty estimation**

The data file **State.csv** contains per capita state and local public expenditures and associated state demographic and economic characteristics, 1960, and there are variables

- MET: Percentage of population living in standard metropolitan areas
- EX: Per capita state and local public expenditures ($)

1. Reorder your data with respect to the increase of MET and plot EX versus MET. Discuss what kind of model can be appropriate here. Use the reordered data in steps 2-5.
2. Use package **tree** and fit a regression tree model with target EX and feature MET in which the number of the leaves is selected by cross-validation, use the entire data set and set minimum number of observations in a leaf equal to 8 (setting *minsize* in *tree.control*). Report the selected tree. Plot the original and the fitted data and histogram of residuals. Comment on the distribution of the residuals and the quality of the fit.
3. Compute and plot the 95% confidence bands for the regression tree model from step 2 (fit a regression tree with the same settings and the same number of leaves as in step 2 to the resampled data) by using a non-parametric bootstrap. Comment whether the band is smooth or bumpy and try to explain why. Consider the width of the confidence band and comment whether results of the regression model in step 2 seem to be reliable.
4. Compute and plot the 95% confidence and prediction bands the regression tree model from step 2 (fit a regression tree with the same settings and the same number of leaves as in step 2 to the resampled data) by using a parametric bootstrap, assume $YY$~$NN$($\mu\mu ii$,$\sigma\sigma$2) where $\mu\mu ii$ are labels in the tree leaves and $\sigma\sigma$2 is the residual variance. Consider the width of the confidence band and comment whether results of the regression model in step 2 seem to be reliable. Does it look like only 5% of data are outside the prediction band? Should it be?
5. Consider the histogram of residuals from step 2 and suggest what kind of bootstrap is actually more appropriate here.

**Assignment 4. Principal components**

The data file **NIRspectra.csv** contains near-infrared spectra and viscosity levels for a collection of diesel fuels. Your task is to investigate how the measured spectra can be used to predict the viscosity.
1. Conduct a standard PCA by using the feature space and provide a plot explaining how much variation is explained by each feature. Does the plot show how many PC should be extracted? Select the minimal number of components explaining at least 99% of the total variance. Provide also a plot of the scores in the coordinates (PC1, PC2). Are there unusual diesel fuels according to this plot?
2. Make trace plots of the loadings of the components selected in step 1. Is there any principle component that is explained by mainly a few original features?
3. Perform Independent Component Analysis with the number of components selected in step 1 (set seed 12345). Check the documentation for the fastICA method in R and do the following: a. Compute $WW'$=$KK$·$WW$ and present the columns of $WW'$ in form of the trace plots. Compare with the trace plots in step 2 and make conclusions. What kind of measure is represented by the matrix $WW'$?
b. Make a plot of the scores of the first two latent features and compare it with the score plot from step 1.

**LAB 3 BLOCK 1: KERNEL METHODS AND NEURAL NETWORKS**

**1. KERNEL METHODS:**

Implement a kernel method to predict the hourly temperatures for a date and place in Sweden.
To do so, you are provided with the files stations.csv and temps50k.csv. These
files contain information about weather stations and temperature measurements in the stations
at different days and times. The data have been kindly provided by the Swedish Meteorological
and Hydrological Institute (SMHI).
You are asked to provide a temperature forecast for a date and place in Sweden. The
forecast should consist of the predicted temperatures from 4 am to 24 pm in an interval of 2
hours. Use a kernel that is the sum of three Gaussian kernels:
Y The first to account for the distance from a station to the point of interest.
Y The second to account for the distance between the day a temperature measurement
was made and the day of interest.
Y The third to account for the distance between the hour of the day a temperature measurement
was made and the hour of interest.
Choose an appropriate smoothing coefficient or width for each of the three kernels above.
Answer to the following questions:
Y Show that your choice for the kernels' width is sensible, i.e. that it gives more weight
to closer points. Discuss why your of definition of closeness is reasonable.
Y Instead of combining the three kernels into one by summing them up, multiply them.
Compare the results obtained in both cases and elaborate on why they may differ.
Note that the file temps50k.csv may contain temperature measurements that are posterior
to the day and hour of your forecast. You must filter such measurements out, i.e. they cannot
be used to compute the forecast. Feel free to use the template below to solve the assignment.

**2. SUPPORT VECTOR MACHINES: To be solved by 732A99/732A68/PhD course**

Use the function ksvm from the R package kernlab to learn a SVM for classifying the spam dataset that is included with the package. Consider the radial basis function kernel (also known as Gaussian) with a width of 0.05. For the C parameter, consider values 0.5, 1 and 5. This implies that you have to consider three models.

Y Perform model selection, i.e. select the most promising of the three models (use any method of your choice except cross-validation or nested cross-validation).

Y Estimate the generalization error of the SVM selected above (use any method of your choice except cross-validation or nested cross-validation).

Y Produce the SVM that will be returned to the user, i.e. show the code.

Y What is the purpose of the parameter C ?

**3. NEURAL NETWORKS: To be solved by TDDE01**

Train a neural network to learn the trigonometric sine function. To do so, sample 50 points uniformly at random in the interval [0; 10]. Apply the sine function to each point. The resulting pairs are the data available to you. Use 25 of the 50 points for training and the rest for validation. The validation set is used for early stop of the gradient descent. That is, you should use the validation set to detect when to stop the gradient descent and so avoid overfitting. Stop the gradient descent when the partial derivatives of the error function are below a given threshold value. Check the argument threshold in the documentation. Consider threshold values i~1000 with i = 1; : : : ; 10. Initialize the weights of the neural network to random values in the interval [−1; 1]. Use a neural network with a single hidden layer of 10 units. **Use the default values for the arguments not mentioned here**. Choose the most appropriate value for the threshold. Motivate your choice. Provide the final neural network learned with the chosen threshold. Feel free to use the following template.

```
library(neuralnet)
set.seed(1234567890)
Var <- runif(50, 0, 10)
trva <- data.frame(Var, Sin=sin(Var))
tr <- trva[1:25,] # Training
va <- trva[26:50,] # Validation
# Random initialization of the weights in the interval [-1, 1]
winit <- # Your code here
for(i in 1:10) {
nn <- neuralnet(# Your code here)
# Your code here
}
plot(nn <- neuralnet(# Your code here))
# Plot of the predictions (black dots) and the data (red dots)
plot(prediction(nn)$rep1)
points(trva, col = "red")
```

**LAB 1 BLOCK 2: ENSEMBLE METHODS AND MIXTURE MODELS**

**1. ENSEMBLE METHODS**

The file spambase.csv contains information about the frequency of various words, characters, etc. for a total of 4601 e-mails. Furthermore, these e-mails have been classified as spams (spam = 1) or regular e-mails (spam = 0). You can find more information about these data at https://archive.ics.uci.edu/ml/datasets/Spambase

Your task is to evaluate the performance of Adaboost classification trees and random forests on the spam data. Specifically, provide a plot showing the error rates when the number of trees considered are 10; 20; : : : ; 100. To estimate the error rates, use 2/3 of the data for training and 1/3 as hold-out test data.

To learn Adaboost classification trees, use the function blackboost() of the R package mboost. Specify the loss function corresponding to Adaboost with the parameter family. To learn random forests, use the function randomForest of the R package randomForest. To load the data, you may want to use the following code:

```
sp <- read.csv2("spambase.csv")
sp$Spam <- as.factor(sp$Spam)
```

**2. MIXTURE MODELS**

Your task is to implement the EM algorithm for mixtures of multivariate Benoulli distributions. Please use the template in the next page to solve the assignment. Then, use your implementation to show what happens when your mixture models has too few and too many components, i.e. set K = 2; 3; 4 and compare results. Please provide a short explanation as well.

**LAB 2 BLOCK 2: Assignment 1. Using GAM and GLM to examine the mortality rates**

The Excel document **influenza.xlsx** contains weekly data on the mortality and the number of laboratory-confirmed cases of influenza in Sweden. In addition, there is information about population-weighted temperature anomalies (temperature deficits).

1. Use time series plots to visually inspect how the mortality and influenza number vary with time (use Time as X axis). By using this plot, comment how the amounts of influenza cases are related to mortality rates.
2. Use gam() function from mgcv package to fit a GAM model in which Mortality is normally distributed and modelled as a linear function of Year and spline function of Week, and make sure that the model parameters are selected by the generalized cross-validation. Report the underlying probabilistic model.
3. Plot predicted and observed mortality against time for the fitted model and comment on the quality of the fit. Investigate the output of the GAM model and report which terms appear to be significant in the model. Is there a trend in mortality change from one year to another? Plot the spline component and interpret the plot.
4. Examine how the penalty factor of the spline function in the GAM model from step 2 influences the estimated deviance of the model. Make plots of the predicted and observed mortality against time for cases of very high and very low penalty factors. What is the relation of the penalty factor to the degrees of freedom? Do your results confirm this relationship?
5. Use the model obtained in step 2 and plot the residuals and the influenza values against time (in one plot). Is the temporal pattern in the residuals correlated to the outbreaks of influenza?

6. Fit a GAM model in R in which mortality is be modelled as an additive function of the spline functions of year, week, and the number of confirmed cases of influenza. Use the output of this GAM function to conclude whether or not the mortality is influenced by the outbreaks of influenza. Provide the plot of the original and fitted Mortality against Time and comment whether the model seems to be better than the previous GAM models.

**Assignment 2. High-dimensional methods**

The data file **data.csv** contains information about 64 e-mails which were manually collected from DBWorld mailing list. They were classified as: 'announces of conferences' (1) and 'everything else' (0) (variable Conference)

1. Divide data into training and test sets (70/30) without scaling. Perform nearest shrunken centroid classification of training data in which the threshold is chosen by cross-validation. Provide a centroid plot and interpret it. How many features were selected by the method? List the names of the 10 most contributing features and comment whether it is reasonable that they have strong effect on the discrimination between the conference mails and other mails? Report the test error.
2. Compute the test error and the number of the contributing features for the following methods fitted to the training data:
a. Elastic net with the binomial response and $\alpha\alpha$=0.5 in which penalty is selected by the cross-validation
b. Support vector machine with “vanilladot” kernel.

Compare the results of these models with the results of the nearest shrunken centroids (make a comparative table). Which model would you prefer and why?

3. Implement Benjamini-Hochberg method for the original data, and use t.test() for computing p-values. Which features correspond to the rejected hypotheses? Interpret the result.

# 3 Assignment 3: Neural Networks

In this assignment, we are required to train a neural network to learn the trigonometric sine function.

**Data**

For this, we first sample 50 points uniformly at random in the interval $[0, 10]$. We apply the sine function to each point and compute their corresponding target values. These pairs of data are what we use for this task.

Out of these 50 data points, we use 25 of them for training and remaining 25 for validation.

**Neural Network**

We are considering a neural network with a single hidden layer of 10 units. In this case we have only one input feature. Hence, each unit in the hidden layer requires one input weight, one output weight and one input bias. We will be predicting only one output variable which requires one more bias. So, totally, we will be estimating 31 values (3 x 10 + 1). We randomly generate 31 start values as initial weights in the range $[-1, 1]$. We use the `neuralnet()` function to train a neural network.

**Role of `threshold`**

In every iteration of training the neural network, the weights are updated using the gradient descent method. We repeat the iterations only if the partial derivatives of the error function are greater than a specified threshold value. The partial derivatives of the error function represent the reduction in the error function value achieved by updating the weights. Updating the weights beyond a certain extent for very minimal reduction in the error function value can lead to overfitting. Hence, we want to stop the gradient descent when the partial derivatives of the error function are below a given threshold value in order to avoid overfitting. We can set this stopping criteria in the `neuralnet()` function using the `threshold` parameter. We can further verify if overfitting has occurred by computing the validation error for the model.

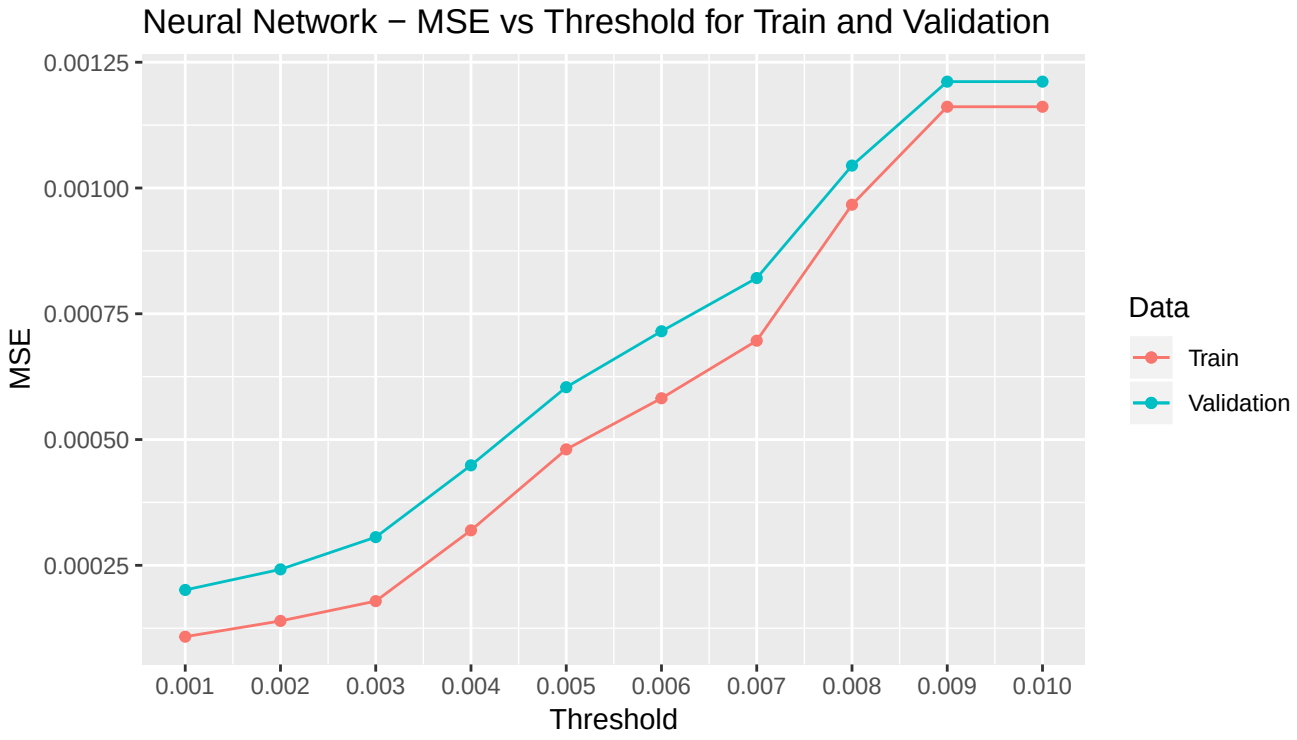
**Model Selection**

We consider the threshold values given by $i/1000$, where $i = 1, 2, 3, .., 10$. We evaluate the models obtained by different `threshold` values by computing their corresponding validation MSE values. The default values are used for all other parameters of the `neuralnet()` function.

A neural network function call looks as follows:

```
nn = neuralnet(Sin ~ Var, data = tr, hidden = 10, startweights = w_init, threshold
= i/1000)
```

By training neural networks using each of the `threshold` values, we obtain different training and validation MSE values. The following plot shows the training and validation MSE values obtained for each `threshold`.




Based on the above plot, we do not observe any overfitting for the values of `threshold` that we have considered. The train MSE consistently appears to be lower than the validation MSE as expected. As threshold is decreased, both validation and training MSE values decrease. The model which produces the minimum validation MSE can be considered as a good generalization of the data. Among the values of threshold considered, we obtain the minimum MSE value of 0.000201 for a threshold value of 0.001. Hence, the threshold value of 0.001 appears to be the most appropriate in the range of threshold values considered.

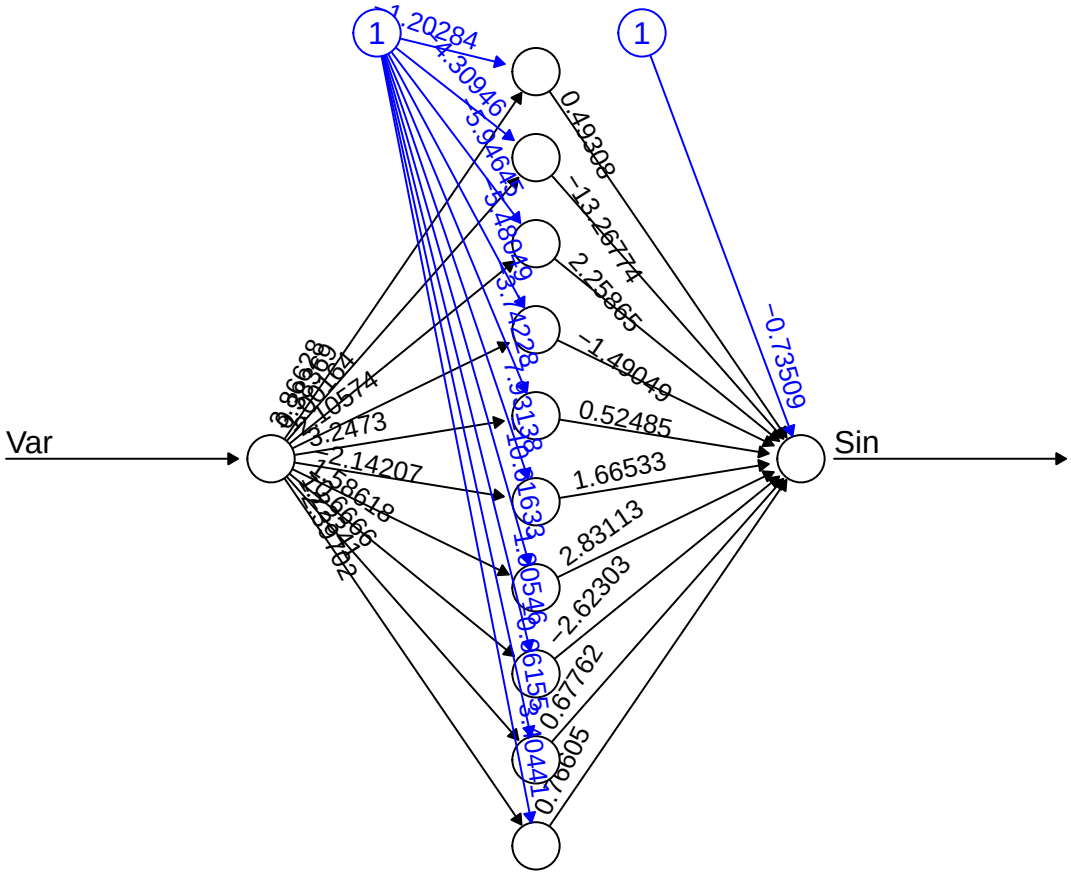
In this task, we have generated data using the sine function but the data is a perfect signal and does not contain any noise. Overfitting is usually evident when a model starts fitting the noise as well and not just the signal. So, very low threshold values like 0.001 performing optimally is not surprising in this case. We might be able to decrease the threshold value further and get an even better validation MSE as well.

### 3.0.1 Final Neural Network

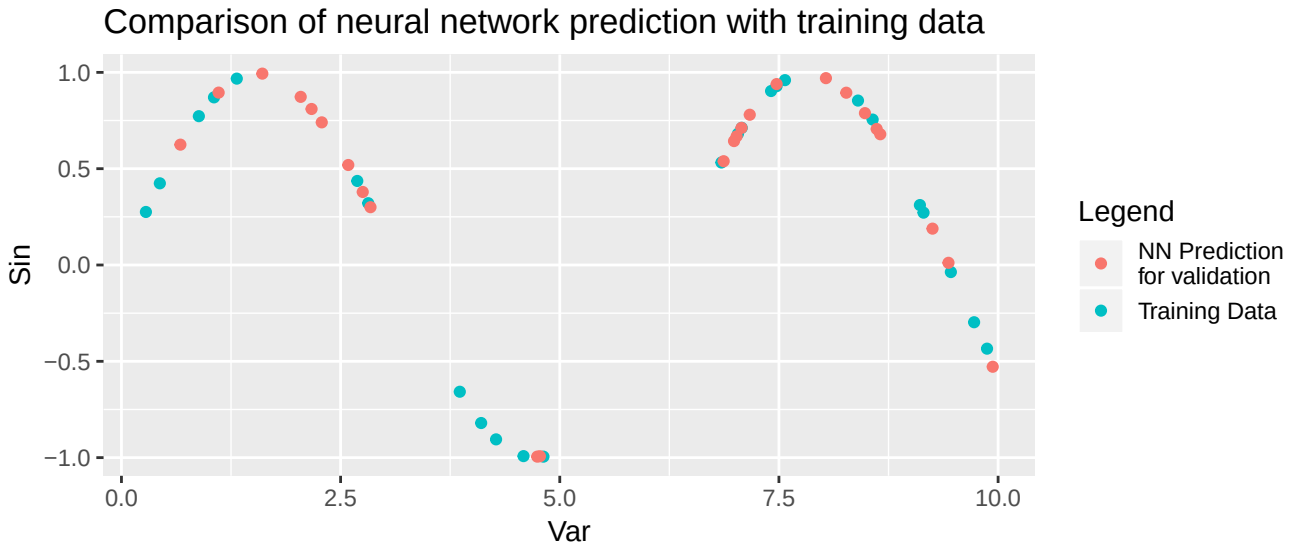
The final neural network has the following form:

```
opt_nn = neuralnet(Sin ~ Var, data = tr, hidden = 10, startweights = w_init,
threshold = 0.001)
```

The final neural network with a single hidden layer of 10 units and a threshold of 0.001 looks as follows:




In order to visually observe how closely the predictions made by the neural network match the actual data, we show the actual training data and the values predicted by the neural network for the validation data in the following plot.




From this plot, we observe that the values predicted by the neural network fall very closely on the shape that we would expect for a sine curve.

```r
kable(compare_df) %>%
  kable_styling(bootstrap_options = "striped", full_width = F, position = "center")
res <- svm_model(C = 5, train, test)
cat("Generalization error for C = 5 SVM model is:", res[[3]])

# ------------------------------------------------------------------------------
# Assignment 3, Task 1
# ------------------------------------------------------------------------------

# Generating data
set.seed(1234567890)
Var = runif(50, 0, 10)
trva = data.frame(Var, Sin = sin(Var))

# Training and validation split
tr = trva[1:25, ] # Training
va = trva[26:50, ] # Validation

nn_val_res_df = data.frame()

# Random initialization of the weights in the interval [-1, 1]
set.seed(1234567890)
w_init = runif(31, -1, 1)

for(i in 1:10) {
  print(paste("Running NN: ", i))
  set.seed(1234567890)

  # Training neural network
  nn = neuralnet(Sin ~ Var, data = tr, hidden = 10,
                 startweights = w_init, threshold = i / 1000)

  # Predicting values for train and validation
  va_res = neuralnet::compute(nn, va$Var)$net.result
  tr_res = neuralnet::compute(nn, tr$Var)$net.result

  # Computing train and validation MSE
  tr_mse = mean((tr_res - tr$Sin)^2)
  va_mse = mean((va_res - va$Sin)^2)

  # Storing data in data frame
  nn_val_res_df = rbind(nn_val_res_df,
                        data.frame(thres_num = i, thres_val = i / 1000,
                                   val_mse = va_mse, trn_mse = tr_mse))
}
```

```r
# Plot of MSE vs threshold for train and validation
ggplot(nn_val_res_df) +
  geom_point(aes(x = thres_val, y = val_mse, color = "Validation")) +
  geom_line(aes(x = thres_val, y = val_mse, color = "Validation")) +
  geom_point(aes(x = thres_val, y = trn_mse, color = "Train")) +
  geom_line(aes(x = thres_val, y = trn_mse, color = "Train")) +
  xlab("Threshold") + ylab("MSE") + labs(color = "Data") +
  scale_x_continuous(breaks = (1:10)/1000) +
  ggtitle("Neural Network - MSE vs Threshold for Train and Validation")

# Final neural network
# Best threshold = 0.001
opt_nn = neuralnet(Sin ~ Var, data = tr, hidden = 10,
                   startweights = w_init, threshold = 0.001)
plot(x = opt_nn, rep = "best", information = F)

# Plot of the predictions and the data
nn_pred_df = tr
nn_pred_df$Type = "Training Data"
nn_pred_df = rbind(nn_pred_df,
                   data.frame(Var = va$Var,
                              Sin = neuralnet::compute(opt_nn, va$Var)$net.result,
                              Type = "NN Prediction \nfor validation"))

ggplot(nn_pred_df, aes(x = Var, y = Sin, color = Type)) + geom_point() +
  ggtitle("Comparison of neural network prediction with training data") +
  labs(color = "Legend")
```

# Lab1

*Tejashree_R_Mastamardi - tejma768*

*November 16, 2018*

## Assignment1

### 1.1

```
library(readxl)
my_data <- read_xlsx("spambase.xlsx")

n=dim(my_data)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))

train=my_data[id,]
test=my_data[-id,]
```

### 1.2

```
#build model on training dataset
model <- glm(Spam ~ ., data = train, family = "binomial")
#summary(model)

#predict Y
predY_train <- predict(model, train, type = "response")

pred <- ifelse(predY_train > 0.50, "Spam", "Not Spam")

#confusionMatrix
tab <- table(train$Spam, pred)
#tab

#misClassError
1 - (sum(diag(tab))/sum(tab)) # for training data
```

```
## [1] 0.1627737
```

Here the confusion matrix describes the performance of glm function on Train data set. There are two possible classes and they are Not spam=0, and Spam=1. The confusion matrix has made 1370 predictions and that is known by adding the 4 numbers. Out of 1370 cases, the classifier has predicted Spam mails 486 times and predicted Not Spam mails 884 times. But the actual Spam mails are 425 and actual Not Spam mails are 945 mails. There are 344 True positive values and 803 True negative values. And there are 142 False positive values and 81 false negative values. Misclassification rate also known as error rate is to measure how often the classifier is wrong and in this case it is 16.27%. Intercept, word3,13, 27, 30, 31, 33, 35, 36, 37, 42, 43, 44, 45, 46 and word48 are the significant ones.

```
#predict Y
predY_test <- predict(model, test, type = "response")
```

```r
pred1 <- ifelse(predY_test > 0.50, "Spam", "Not Spam")

#confusion matrix for test dataset
tab1 <- table(test$Spam, pred1)
#tab1

#misClassError
1 - (sum(diag(tab1))/sum(tab1)) # for test data
```

```
## [1] 0.1773723
```

Here the confusion matrix describes the performance of glm function on Test data set. There are two possible classes and they are Not spam=0, and Spam=1. The confusion matrix has made 1370 predictions and that is known by adding the 4 numbers. Out of 1370 cases, the classifier has predicted Spam mails 482 times and predicted Not Spam mails 888 times. But the actual Spam mails are 433 and actual Not Spam mails are 937 mails. There are 336 True positive values and 791 True negative values. And there are 146 False positive values and 97 false negative values. Misclassification rate also known as error rate is to measure how often the classifier is wrong and in this case it is 17.73%.

## 1.3

```r
p_class2 <- ifelse(predY_train > 0.90, "Spam", "Not Spam")
table(p_class2)
```

```
## p_class2
## Not Spam     Spam
##     1363        7
```

```r
#confusion matrix for train dataset
tab2 <- table(train$Spam, p_class2)
#tab2

#misClassError
1 - (sum(diag(tab2))/sum(tab2)) # for test data
```

```
## [1] 0.3065693
```

```r
p_class3 <- ifelse(predY_test > 0.90, "Spam", "Not Spam")
table(p_class3)
```

```
## p_class3
## Not Spam     Spam
##     1363        7
```

```r
#confusion matrix for test dataset
tab3 <- table(test$Spam, p_class3)
#tab3

#misClassError
1 - (sum(diag(tab3))/sum(tab3)) # for test data
```

```
## [1] 0.3124088
```

With the change in value of probability, the misclassification rate has increased from 16.27% to 30.65% for train data set and 17.73% to 31.24% for test data set, and the number of spam mails predicted has reduced. The impact of new rule is that very less number of mails are classified as Spam.

## 1.4

```
library(kknn)

model4 <- kknn(Spam ~ .,train, train, k=30)

predictedY4 <- predict(model4)

p_class4 <- ifelse(predictedY4 > 0.50, "Spam", "Not Spam")

#confusion matrix for train dataset
tab4 <- table(train$Spam, p_class4)
#tab4

#misClassError
1 - sum(diag(tab4))/sum(tab4)
```

```
## [1] 0.1722628
```

```
model4.1 <- kknn(Spam ~ .,train, test, k=30)

predictedY4.1 <- predict(model4.1)

p_class4.1 <- ifelse(predictedY4.1 > 0.50, "Spam", "Not Spam")


#confusion matrix for test dataset
tab_4.1 <- table(test$Spam, p_class4.1)
#tab_4.1

#misClassError
1 - (sum(diag(tab_4.1))/sum(tab_4.1))
```

```
## [1] 0.329927
```

The misclassification rate for train and test data set is 17.22% and 32.99% respectively, whereas in step 2 it was 16.27% and 17.73% for train and test data set.

## 1.5

```
model5 <- kknn(Spam ~ .,train, train, k=1)

predictedY5 <- predict(model5)

p_class5 <- ifelse(predictedY5 > 0.50, "Spam", "Not Spam")

#confusion matrix for train dataset
tab5 <- table(train$Spam, p_class5)
#tab5

#misClassError
1 - sum(diag(tab5))/sum(tab5)
```

```
## [1] 0
```

```
model5.1 <- kknn(Spam ~ .,train, test, k=1)

predictedY5.1 <- predict(model5.1)

p_class5.1 <- ifelse(predictedY5.1 > 0.50, "Spam", "Not Spam")


#confusion matrix for test dataset
tab_5.1 <- table(test$Spam, p_class5.1)
#tab_5.1

#misClassError
1 - (sum(diag(tab_5.1))/sum(tab_5.1))
```

```
## [1] 0.3459854
```

The misclassification rate for train data set is 0% and for test data set it is 34.59% whereas in step 4 it was 16.27% and 17.73% for train and test data set respectively. With k=30, the areas will be much smoother, and of simpler shapes, thus will be less complex. With value of k reduced to 1, the plot becomes more complex. Thus decreasing k in k-nearest neighbours increase complexity. When the value of k is decreased, the misclassification error rate increases.

# Assignment 3

```
library(MASS)
library(ggplot2)

my_data <- swiss
b <- as.vector(my_data[,1])
a <- as.matrix(my_data[,c(2:6)])
Nf <- 5

#function for calculating weights
function_weight <- function(X,Y)
{
  Z <- ginv(t(X)%*%X)%*%t(X)%*%Y
}
n <- dim(my_data)[1]
set.seed(12345)
sampleofn <- sample(1:n)
id <- list()
cvscore <- c()

#function for calculating cv score
cvfunction <- function(X,Y,Nf)
{
  start <- 1
  for(i in 1:Nf)
    {
      if(i<Nf)
      {
        end <- start+(as.integer(n/Nf)-1)
        id[[i]] <- sampleofn[start:end]
```

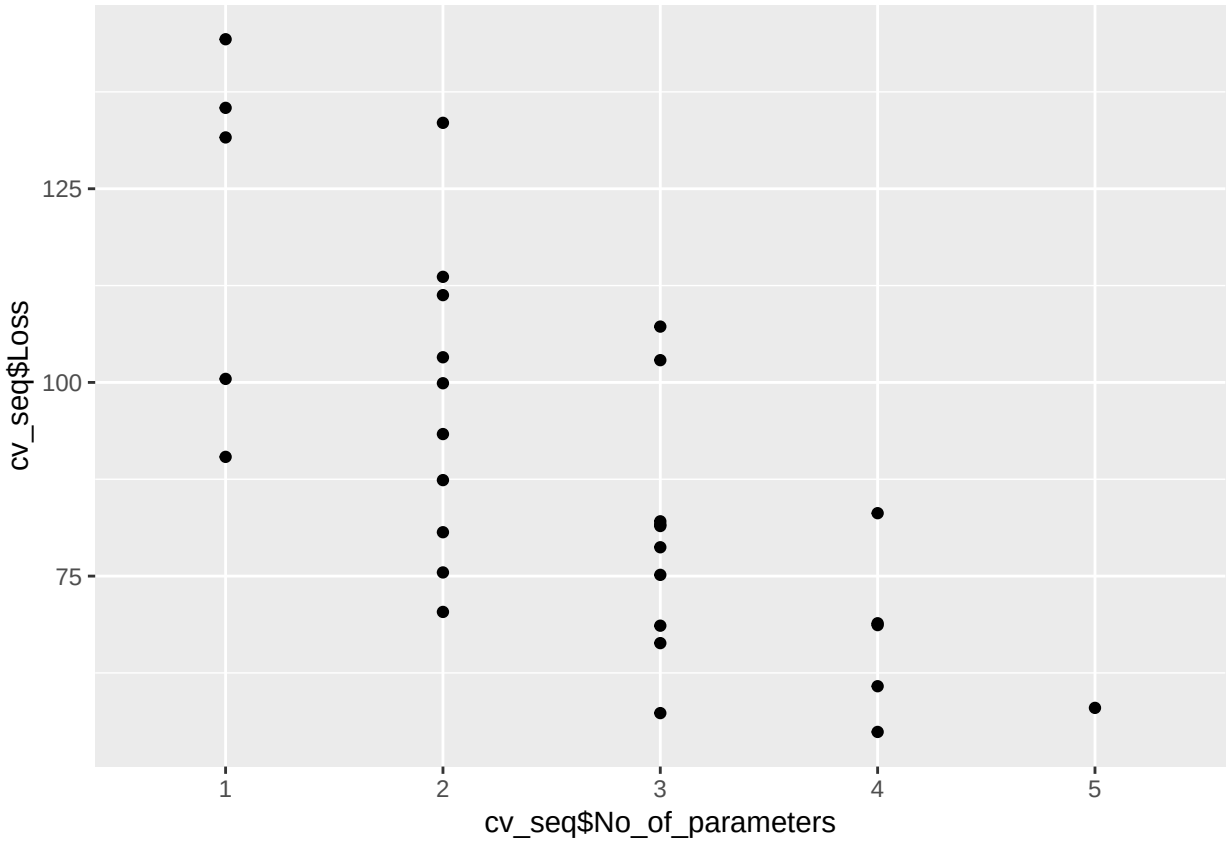
```
        start <- end+1
      }
      else if(i==Nf)
      {
        end <- n
        id[[i]] <- sampleofn[start:end]
      }
      testX <- X[as.vector(id[[i]]),]
      trainX <- X[-as.vector(id[[i]]),]
      testY <- Y[as.vector(id[[i]])]
      trainY <- Y[-as.vector(id[[i]])]

      Weight <- as.matrix(function_weight(X=trainX,Y=trainY))
      b1 <- testX%*%Weight
      loss <- b1-testY
      cv <- sum(loss*loss)/length(testY)
      cvscore[i] <- cv
  }
  average_of_cv <- sum(cvscore)/Nf
}

cv_seq <- matrix(0, nrow = 0, ncol = 3)
for(k in 1:ncol(a))
{
  combinations <- combn(1:ncol(a),k)
  for(j in 1:ncol(combinations))
  {
    a1 <- as.matrix(a[,combinations[,j]])
    a1 <- cbind(a1, 1)
    seq <- paste(combinations[,j], collapse = ",")
    avg_score <- cvfunction(X = a1,Y = b,Nf)
    cv_seq <- rbind(cv_seq, c(seq,avg_score, k))
  }
}

cv_seq = as.data.frame(cv_seq)
colnames(cv_seq) = c("Sequence", "Loss","No_of_parameters")
cv_seq$Loss = as.numeric(as.character(cv_seq$Loss))
cv_seq$Sequence = as.character(cv_seq$Seq)

cvfunction(X= a,Y= b,Nf)

cat("Optimal Subset of Features: ",cv_seq$Seq[which.min(cv_seq$Loss)])
```

```
## Optimal Subset of Features:  1,3,4,5
```

```
cat("Cross validation Score: ",min(cv_seq$Loss))
```

```
## Cross validation Score:  54.88725
```

```
p <- ggplot(cv_seq,aes(x= cv_seq$No_of_parameters,y= cv_seq$Loss))+geom_point()
p
```

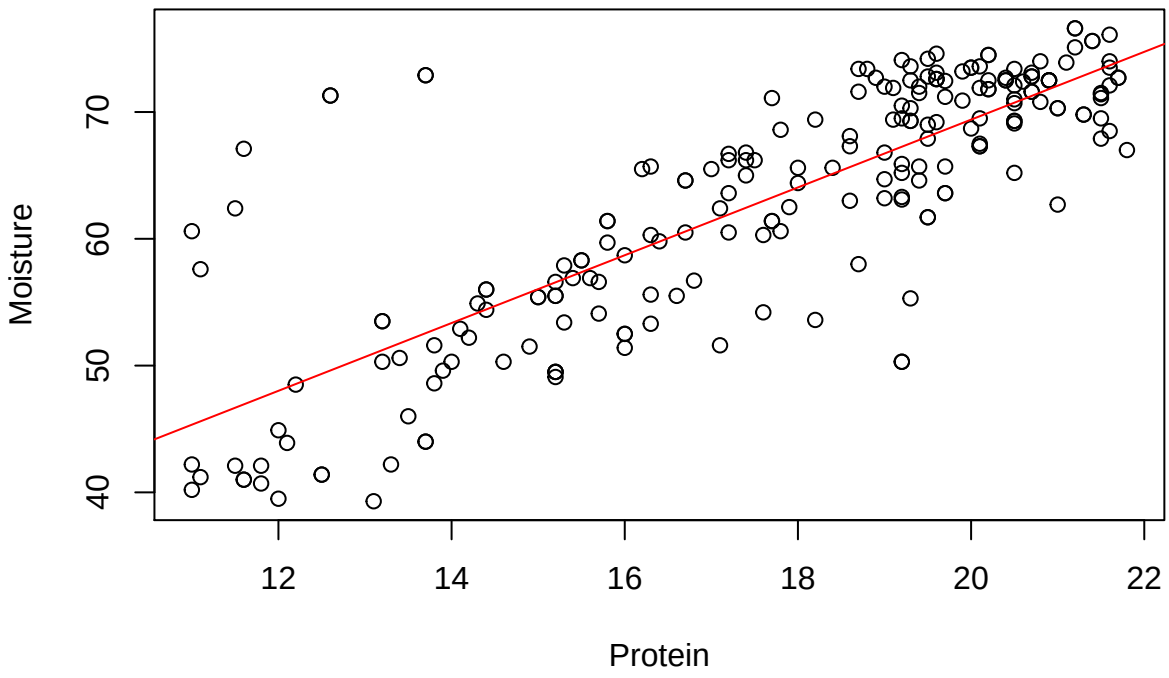



The number of features being considered increases with decrease in cross validation score. By looking at the result we get to know that the optimal subset of features had the following features: Agriculture, Education, Catholic, Infant.mortality.

# Assignment4

## 4.1

```
library(readxl)

my_data2 <- read_xlsx("tecator.xlsx")

plot(Moisture~Protein, data = my_data2)
model <- lm(Moisture~Protein, data = my_data2)
abline(model, col="red")
```




Yes the data is well described by a linear model.

## 4.2

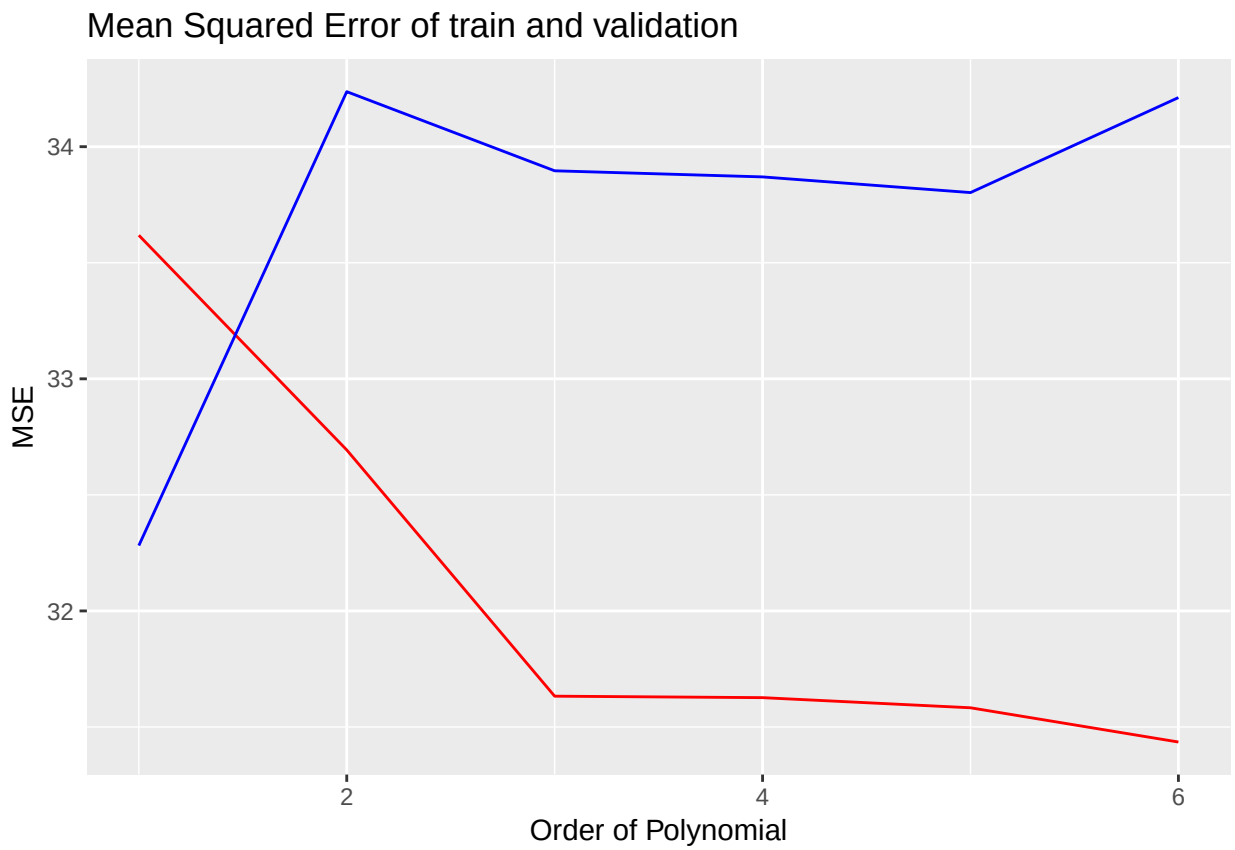
$M_i$,i = (1,2,3,..,i) yhat is the expected moisture x is protein $M_i$ : p(y|x,w) is $y_{N(aplha0+aplha1.x1+aplha2.x2+..+aplhai.xi}$~,std.devˆ2)

It is appropriate to use the MSE criterion when fitting this model to a training data because the model is fit using the Least Squares Method, where the line that is fit is chosen such that the vertical distances from the data are the least. MSE consists of two components, the squared bias and variance. MSE is also used due to the curse of dimensionality, when there is no noise in the target function , the MSE approximates to squared bias. When the target function is constant in all but one dimension, The variance dominates and is approximated by MSE.

## 4.3

```
library(readxl)
library(dplyr)
library(plotly)
library(ggplot2)

my_data2$Protein2<-(my_data2$Protein)^2
my_data2$Protein3<-(my_data2$Protein)^3
my_data2$Protein4<-(my_data2$Protein)^4
my_data2$Protein5<-(my_data2$Protein)^5
my_data2$Protein6<-(my_data2$Protein)^6
```

```r
n=dim(my_data2)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train=my_data2[id,]
valid=my_data2[-id,]

#Linear
model1 = lm(Moisture~Protein, data=train, x=TRUE, y= TRUE)
mse1 <- (sum((model1$fitted.values - model1$y)^2))/nrow(train)
model1_pred <- predict(model1, valid)
mse1.1 <- (sum((model1_pred - valid$Moisture)^2))/nrow(valid)

#Quadratic
model2 = lm(Moisture~Protein+Protein2, data=train, x=TRUE, y= TRUE)
mse2 <- (sum((model2$fitted.values - model2$y)^2))/nrow(train)
model2_pred <- predict(model2, valid)
mse2.1 <- (sum((model2_pred - valid$Moisture)^2))/nrow(valid)

#Cubic
model3 = lm(Moisture~Protein+Protein2+Protein3, data=train, x=TRUE, y= TRUE)
mse3 <- (sum((model3$fitted.values - model3$y)^2))/nrow(train)
model3_pred <- predict(model3, valid)
mse3.1 <- (sum((model3_pred - valid$Moisture)^2))/nrow(valid)

#4th degree Polynomial
model4 = lm(Moisture~Protein+Protein2+Protein3+Protein4, data=train, x=TRUE, y= TRUE)
#summary(model4)
mse4 <- (sum((model4$fitted.values - model4$y)^2))/nrow(train)
model4_pred <- predict(model4, valid)
mse4.1 <- (sum((model4_pred - valid$Moisture)^2))/nrow(valid)

#5th degree Polynomial
model5 = lm(Moisture~Protein+Protein2+Protein3+Protein4+Protein5, data=train, x=TRUE, y= TRUE)
mse5 <- (sum((model5$fitted.values - model5$y)^2))/nrow(train)
model5_pred <- predict(model5, valid)
mse5.1 <- (sum((model5_pred - valid$Moisture)^2))/nrow(valid)

#6th degree Polynomial
model6 = lm(Moisture~Protein+Protein2+Protein3+Protein4+Protein5+Protein6, data=train, x=TRUE, y= TRUE)
mse6 <- (sum((model6$fitted.values - model6$y)^2))/nrow(train)
model6_pred <- predict(model6, valid)
mse6.1 <- (sum((model6_pred - valid$Moisture)^2))/nrow(valid)

train_mse <- c(mse1, mse2, mse3, mse4, mse5, mse6)
valid_mse <- c(mse1.1, mse2.1, mse3.1, mse4.1, mse5.1, mse6.1)

MSE <- data.frame(Train_MSE=train_mse, Valid_MSE=valid_mse)

MSE %>%
  ggplot()+
  geom_line(aes(x = as.numeric(row.names(MSE)), y = train_mse),col = "red")+
  geom_line(aes(x = as.numeric(row.names(MSE)), y = valid_mse), col = "blue") +
labs(title = "Mean Squared Error of train and validation", x = "Order of Polynomial", y = "MSE")
```




According to the plot, model with equation to the power of 1(Linear Model) is the best plot as there is correct amount of Bias-Variance Tradeoff. To build a good model with less complexity, we need to minimize the total error. Estimate of test error is train error, whereas training error is not a good estimate of test error. From the plot we can make out that the Train MSE decreases in the beginning but after 3rd degree polynomial, the error rate becomes firm, whereas Valid error rate increases in the beginning n when it reaches 2nd degree polynomial it becomes firm. Since the model has large number of parameters that is from channel1 to channel100, the model is having high variance and low bias.

## 4.4

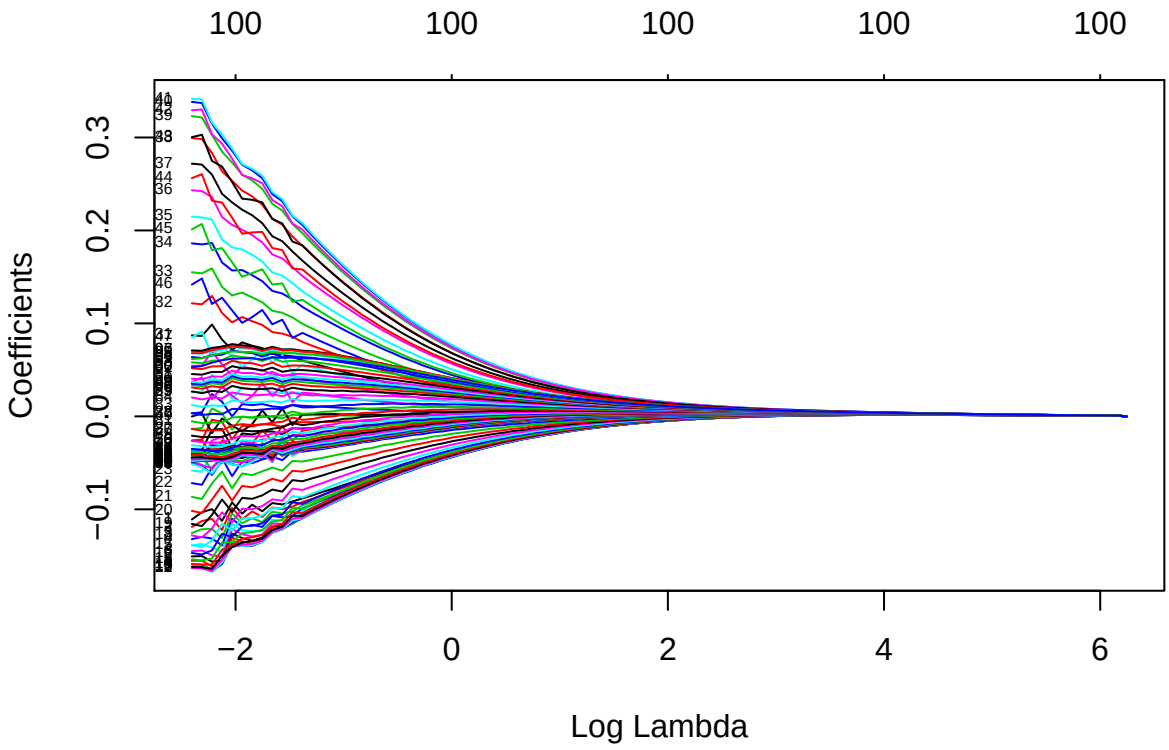
```
library(MASS)

my_data3<-my_data2[2:102]
model_fit <- lm(Fat~.,data=my_data3)

s <- stepAIC(model_fit, direction="both", trace = FALSE)
#s$anova

#summary(s)

length(s$coefficients)
```

```
## [1] 64
```

63 variables were selected.

## 4.5

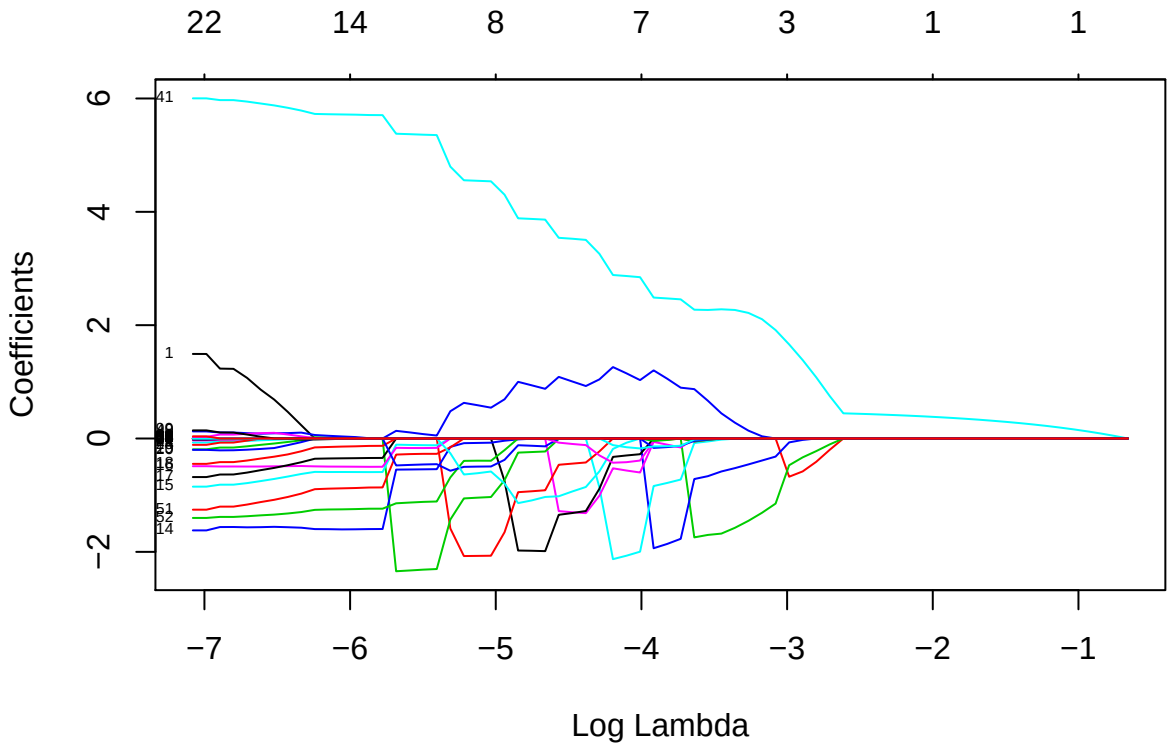
```
library(readxl)
library(glmnet)

covariates=scale(my_data3[,1:100])
response=scale(my_data3[,101])

model7=glmnet(as.matrix(covariates), response, alpha=0,family="gaussian")

plot(model7, xvar="lambda", label=TRUE)
```




As the value of Log Lambda is increasing, the coefficients are quickly converging to zero. The coefficients almost converge to 0 when the value of log lambda becomes 4.

## 4.6

```
library(readxl)
library(glmnet)

model8=glmnet(as.matrix(covariates), response, alpha=1,family="gaussian")

plot(model8, xvar="lambda", label=TRUE)
```




```
# model=cv.glmnet(as.matrix(covariates),response, alpha=1,family="gaussian")
# model$lambda.min
#
# coef(model, s="lambda.min")
# model$lambda.min
```

In Ridge regression model the coefficients quickly converge to 0, while in Lasso Regression model, each coefficient is translated by a factor of log(lambda).

## 4.7

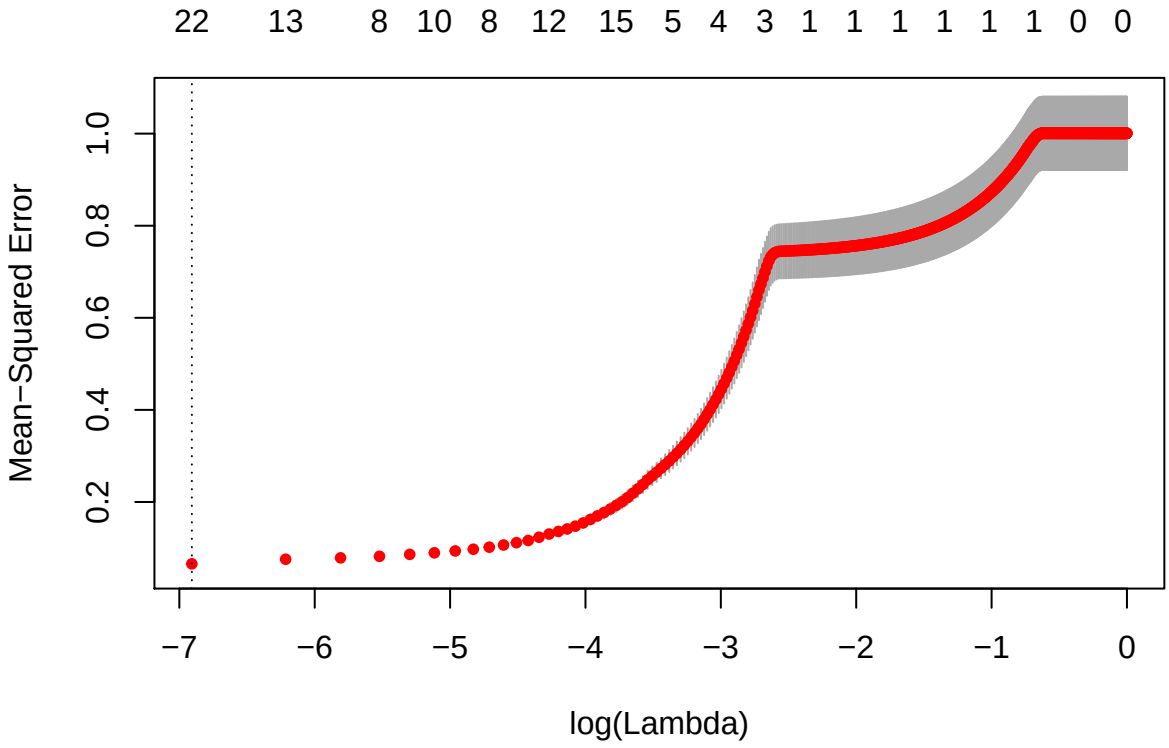
```
my_data3<-scale(my_data2[2:102])

covariates=my_data3[,1:100]
response=my_data3[,101]

model9=cv.glmnet(as.matrix(covariates), response, alpha=1, family="gaussian",lambda=seq(0,1,0.001))

y=my_data3[,101]

ynew=predict(model9, newx=as.matrix(my_data3[, 1:100]), type="response")

#Coefficient of determination
sum((ynew-mean(y))^2)/sum((y-mean(y))^2)
```

```
## [1] 0.9098411
```

```r
sum((ynew-y)^2)
```

```
## [1] 13.07625
```

```r
plot(model9)
```




```r
cat(paste("number of variables chosen =",length(coef(model9,s="lambda.min"))-1))
```

```
## number of variables chosen = 100
```

```r
optimal<-model9$lambda.min
cat(paste("optimal lambda=",optimal))
```

```
## optimal lambda= 0
```

```r
#summary(model9)
```

The value of lambda.min is 0. This makes upto 100% MSE. The printed value of lambda is the value that minimizes MSE and in this case optimal lambda is 0, which means none of the features are removed. The number of variables chosen by the model are 100. The least MSE is at log(lambda) = -7.

## 4.8

63 variables are selected by the model built in step4 using step AIC, while in the model built in step 7, all variables are selected, which means all the features are required to minimize MSE.

# Lab1_Block2

*Group A15*

*December 4, 2018*

## Ensemble methods

```
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
library(mboost)
```

```
## Loading required package: parallel
```

```
## Loading required package: stabs
```

```
## This is mboost 2.9-1. See 'package?mboost' and 'news(package  = "mboost")'
## for a complete list of changes.
```

```
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:randomForest':
##
##     combine
```

```
library(ggplot2)
```

```
##
## Attaching package: 'ggplot2'
```

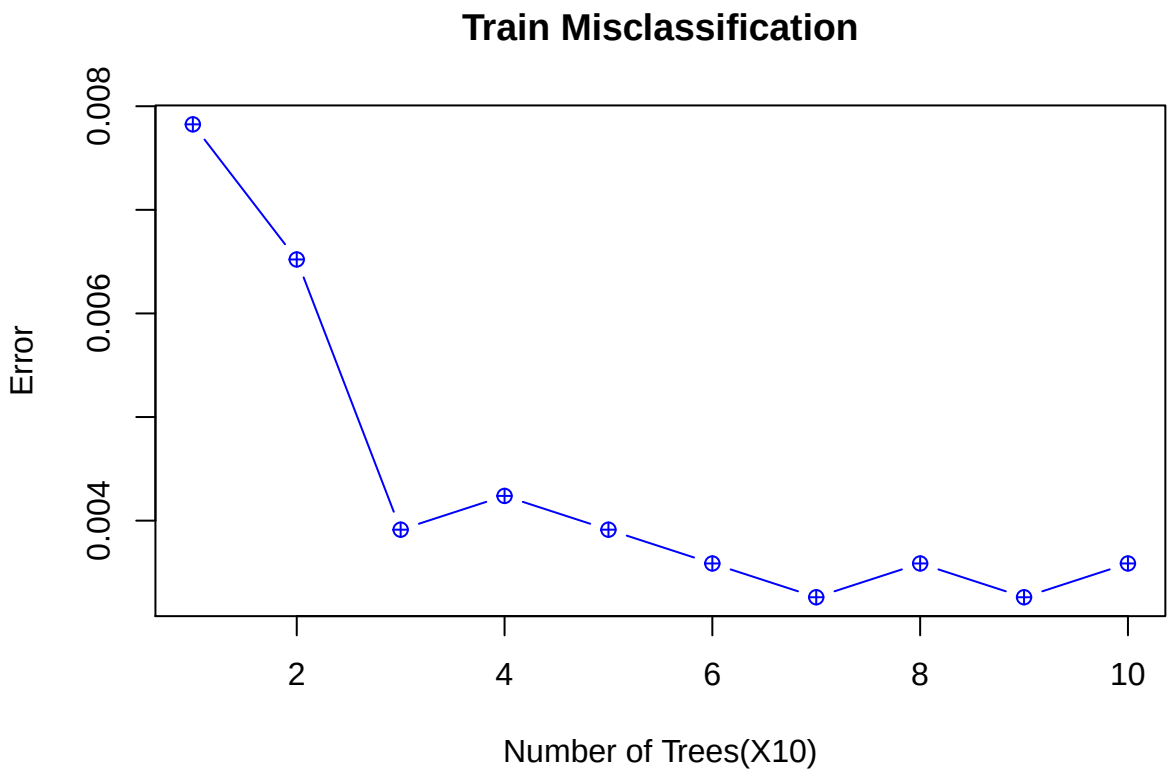
```
## The following object is masked from 'package:mboost':
##
##     %+%
```

```
## The following object is masked from 'package:randomForest':
##
##     margin
```

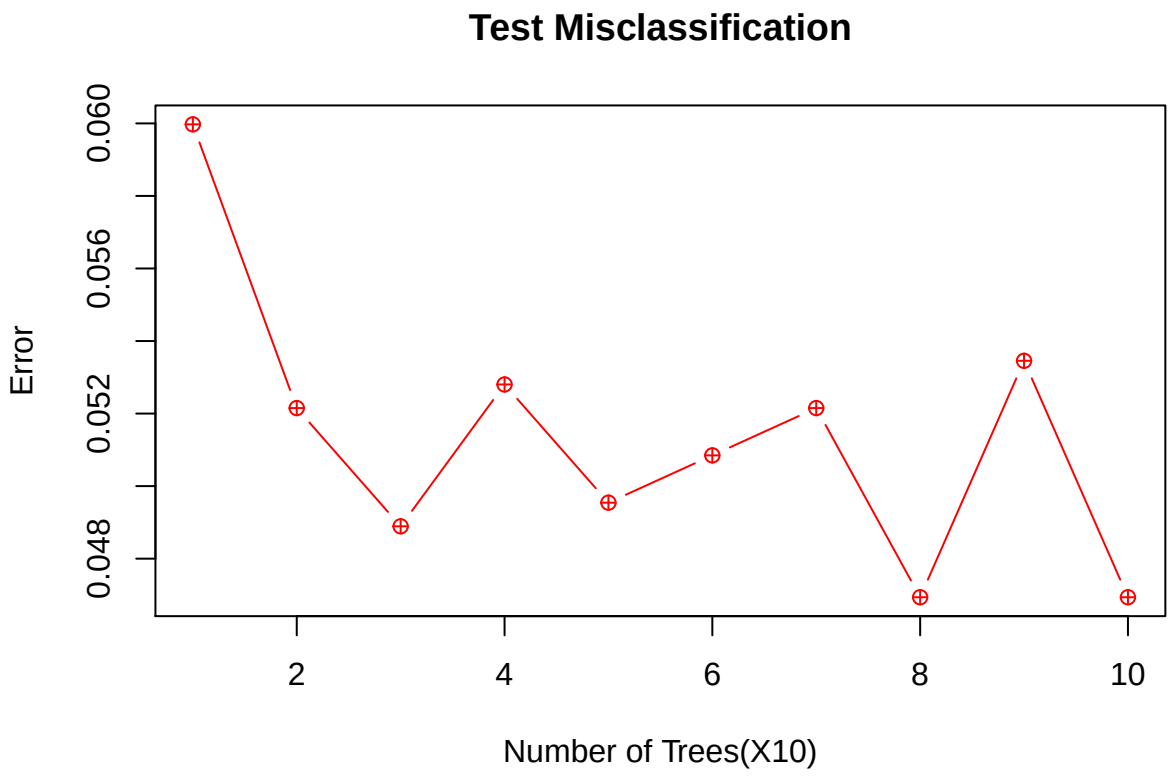
```
sp <- read.csv2("spambase.csv")
sp$Spam <- as.factor(sp$Spam)

n=dim(sp)[1]
set.seed(12345)
id=sample(1:n, floor(n*(2/3)))
train=sp[id,]
test=sp[-id,]

number_of_trees <- seq(from = 10,to = 100, by = 10)
```

### Random forest

```
random_forest <- function(ntrees){

  model_1 <- randomForest(Spam ~ .,data = sp,subset = id,ntree = ntrees,importance = T)

  predict_1 <- predict(model_1,newdata = train, type = "class")
  con_mat_1 <- table(predict_1, train$Spam)
  train_error_1 <- 1-sum(diag(con_mat_1))/sum(con_mat_1)

  predict_2 <- predict(model_1,newdata = test, type = "class")
  con_mat_2 <- table(predict_2, test$Spam)
  test_error_1 <- 1-sum(diag(con_mat_2))/sum(con_mat_2)

  return(list(train_error = train_error_1,test_error = test_error_1))
}

error_rate_rf <- as.data.frame(t(sapply(number_of_trees, random_forest)))
train_error_rf <- as.vector(unlist(error_rate_rf$train_error))
test_error_rf <- as.vector(unlist(error_rate_rf$test_error))

plot(train_error_rf,type = "b",main="Train Misclassification", xlab= "Number of Trees(X10)",
     ylab= "Error", col="blue", pch=10, cex=1)
```




```
plot(test_error_rf,type = "b",main="Test Misclassification", xlab= "Number of Trees(X10)",
     ylab= "Error", col="red", pch=10, cex=1)
```

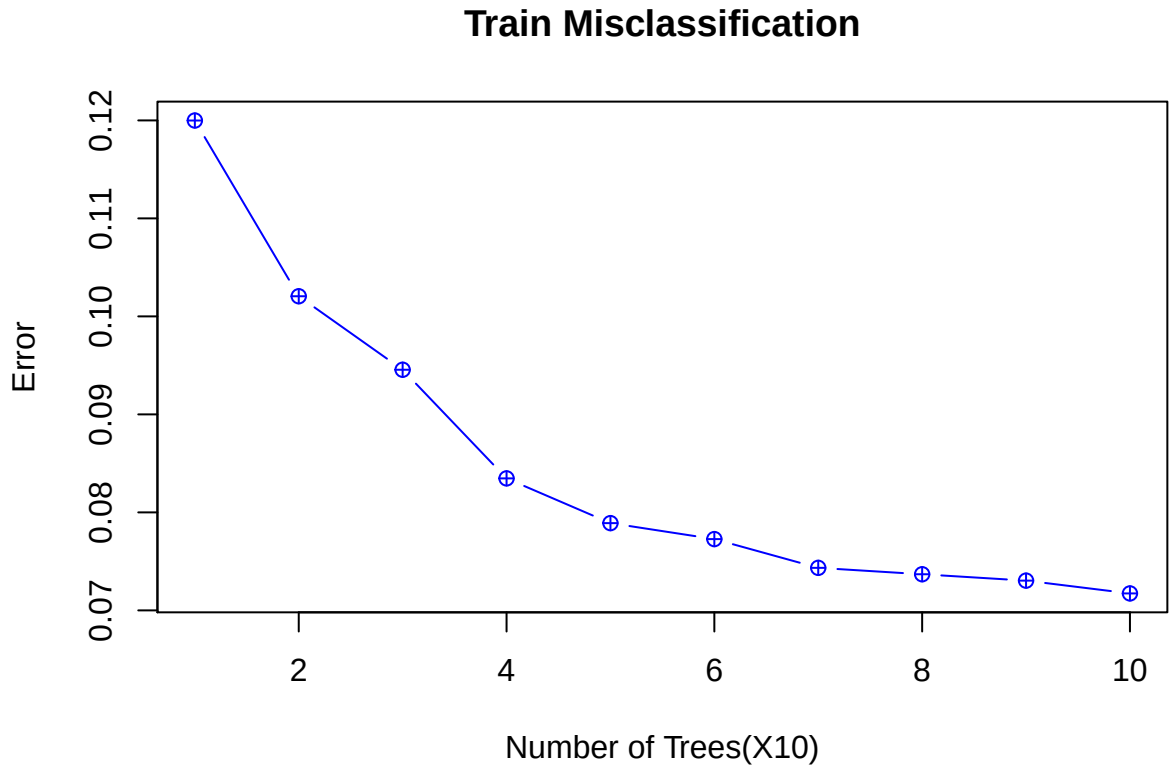



The above plots show the change in error for train and test data with respect to the number of trees considered. It can be seen that the error decreases till the number of trees considered increses upto 30 for Train data and 20 for test data. However after the number of tress increases after a particular number, in the case of train data 30 and for test data 20, the error rate increases and then again decreses. There is an almost alternate increase and decrease in the error with the increase in trees.

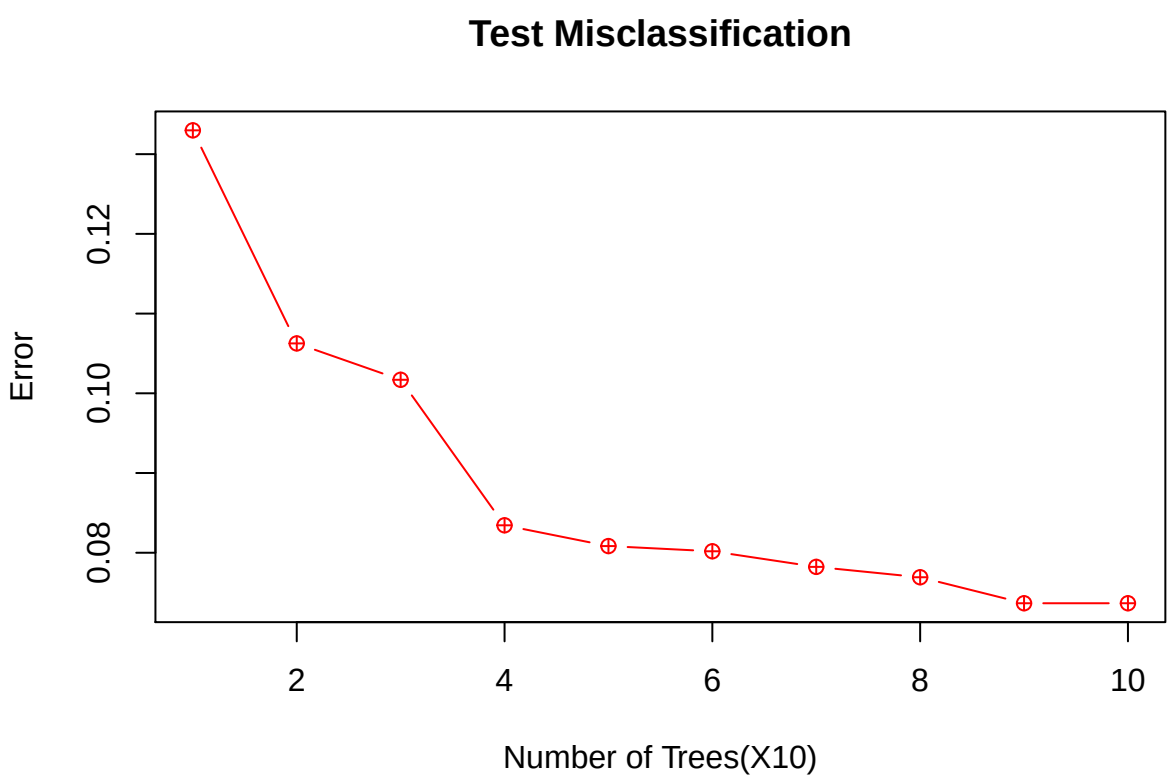
**Adaboost classi???cation trees**

```
adaboost <- function(ntrees){

  model_2 <- blackboost(Spam ~., data = train,
                        control = boost_control(mstop = ntrees, nu=0.1), family = AdaExp())

  predict_3 <- predict(model_2,newdata = train, type = "class")
  con_mat_3 <- table(predict_3, train$Spam)
  train_error_2 <- 1-sum(diag(con_mat_3))/sum(con_mat_3)

  predict_4 <- predict(model_2,newdata = test, type = "class")
  con_mat_4 <- table(predict_4, test$Spam)
  test_error_2 <- 1-sum(diag(con_mat_4))/sum(con_mat_4)

  return(list(train_error = train_error_2,test_error = test_error_2))
}
```

```r
error_rate_ada <- as.data.frame(t(sapply(number_of_trees, adaboost)))
train_error_ada <- as.vector(unlist(error_rate_ada$train_error))
test_error_ada <- as.vector(unlist(error_rate_ada$test_error))

plot(train_error_ada,type = "b",main="Train Misclassification", xlab= "Number of Trees(X10)",
     ylab= "Error", col="blue", pch=10, cex=1)
```



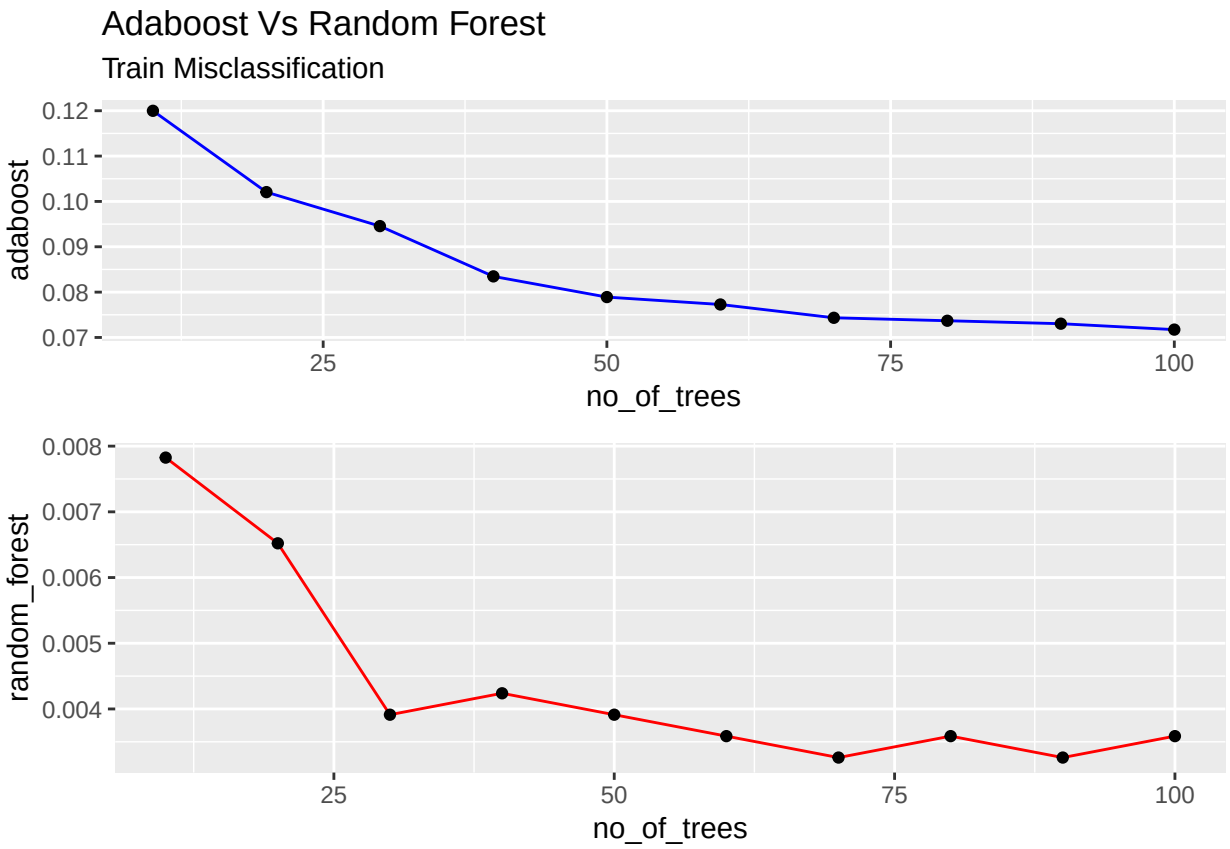

```r
plot(test_error_ada,type = "b",main="Test Misclassification", xlab= "Number of Trees(X10)",
     ylab= "Error", col="red", pch=10, cex=1)
```




The above plots show the change in error for train and test data with respect to the number of trees considered. Unlike Random forest there is no alternating between increase and decrease in error with the increase in trees, rather the error decreases continuously with the increase in number of trees.

**Comparision of error for train data**

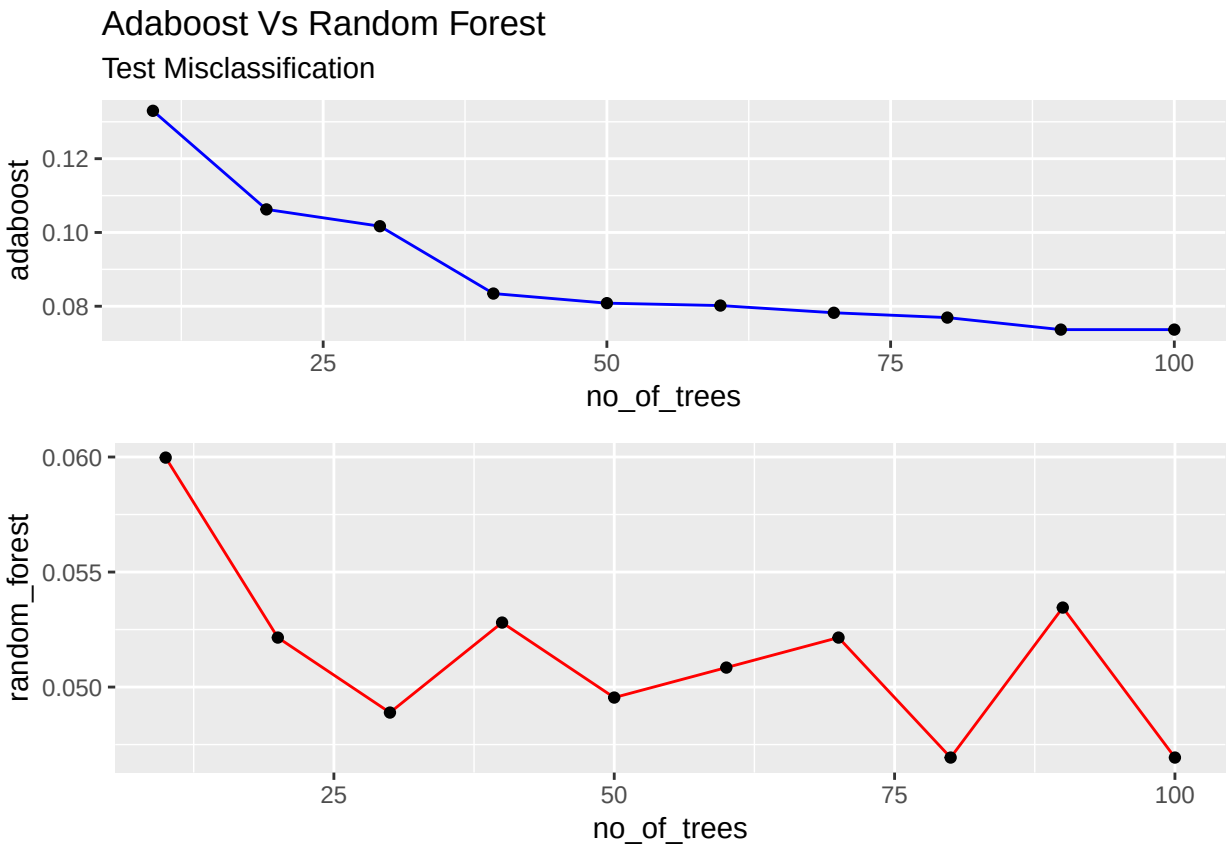
```
ada_rf_train <- data.frame(adaboost = train_error_ada, random_forest = train_error_rf,
                no_of_trees = number_of_trees)

plot_ada_train <- ggplot(ada_rf_train, aes(no_of_trees, y = adaboost)) +
  geom_line(colour="blue")+geom_point()+
  ggtitle("Adaboost Vs Random Forest","Train Misclassification")

plot_rf_train <- ggplot(ada_rf_train, aes(no_of_trees, y = random_forest)) +
  geom_line(colour="red")+geom_point()

grid.arrange(plot_ada_train,plot_rf_train,ncol=1,nrow=2)
```




The error for Adaboost seems to be much more than that for Random forest considering train data. The Random forest achieves its optimum i.e. gives least error with fairly less number of trees,approximately 30 tress in this case, compared to Adaboost.

**Comparision of error for test data**

```
ada_rf_test <- data.frame(adaboost = test_error_ada, random_forest = test_error_rf,
                  no_of_trees = number_of_trees)

plot_ada_test <- ggplot(ada_rf_test, aes(no_of_trees, y = adaboost)) +
  geom_line(colour="blue")+geom_point()+
  ggtitle("Adaboost Vs Random Forest","Test Misclassification")

plot_rf_test <- ggplot(ada_rf_test, aes(no_of_trees, y = random_forest)) +
  geom_line(colour="red")+geom_point()

grid.arrange(plot_ada_test,plot_rf_test,ncol=1,nrow=2)
```




The error for Adaboost seems to be much more than that for Random forest considering test data when less number of trees are considered. However it can be seen that as the number of trees increases the error for Adaboost decreses drastically and almost equals the error rate for Random forest, in this case when the number of trees is 100.

## Mixture Models

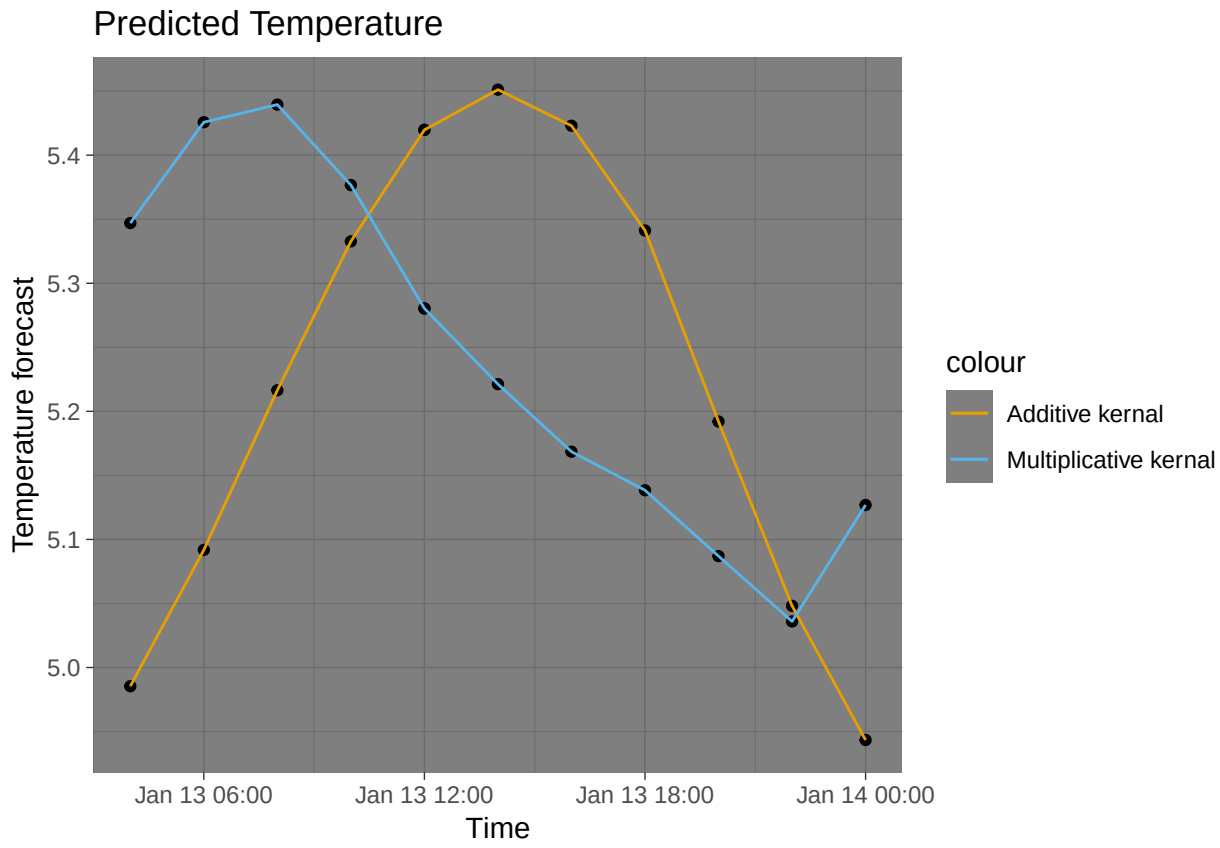
```
em_function <- function(given_k)
{
set.seed(1234567890)
max_it <- 100 # max number of EM iterations
min_change <- 0.1 # min change in log likelihood between two consecutive EM iterations
N=1000 # number of training points
D=10 # number of dimensions
x <- matrix(nrow=N, ncol=D) # training data
true_pi <- vector(length = 3) # true mixing coefficients
true_mu <- matrix(nrow=3, ncol=D) # true conditional distributions
true_pi=c(1/3, 1/3, 1/3)
true_mu[1,]=c(0.5,0.6,0.4,0.7,0.3,0.8,0.2,0.9,0.1,1)
true_mu[2,]=c(0.5,0.4,0.6,0.3,0.7,0.2,0.8,0.1,0.9,0)
true_mu[3,]=c(0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5)
plot(true_mu[1,], type="o", col="blue", ylim=c(0,1))
points(true_mu[2,], type="o", col="red")
points(true_mu[3,], type="o", col="green")
```

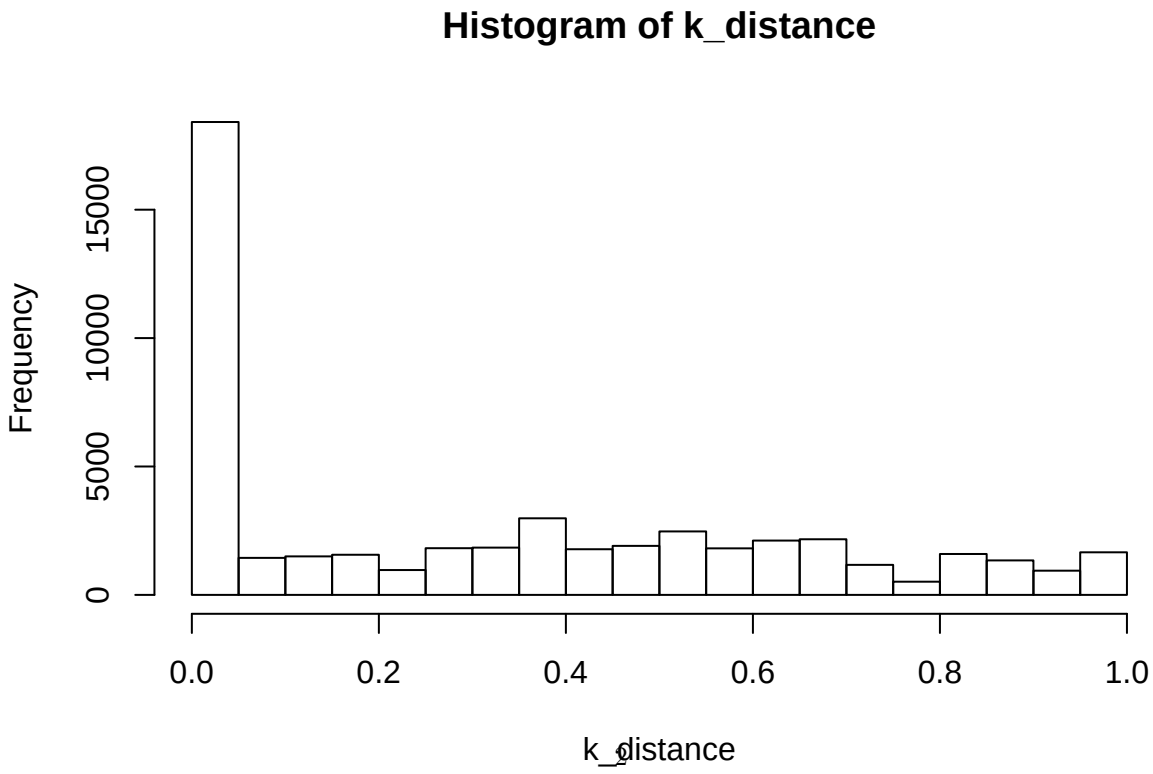
```
# Producing the training data
for(n in 1:N) {
  k <- sample(1:3,1,prob=true_pi)
  for(d in 1:D) {
    x[n,d] <- rbinom(1,1,true_mu[k,d])
  }
}
K=given_k # number of guessed components
z <- matrix(nrow=N, ncol=K) # fractional component assignments
pi <- vector(length = K) # mixing coefficients
mu <- matrix(nrow=K, ncol=D) # conditional distributions
llik <- vector(length = max_it) # log likelihood of the EM iterations
# Random initialization of the paramters
pi <- runif(K,0.49,0.51)
pi <- pi / sum(pi)
for(k in 1:K) {
  mu[k,] <- runif(D,0.49,0.51)
}
pi
mu
for(it in 1:max_it) {
  # if (k==2)
  # {
  #   plot(mu[1,], type="o", col="blue", ylim=c(0,1))
  #   points(mu[2,], type="o", col="red")
  # }
  # else if (k==3)
  # {
  #   plot(mu[1,], type="o", col="blue", ylim=c(0,1))
  #   points(mu[2,], type="o", col="red")
  #   points(mu[3,], type="o", col="green")
  # }
  # else
  # {
  #   plot(mu[1,], type="o", col="blue", ylim=c(0,1))
  #   points(mu[2,], type="o", col="red")
  #   points(mu[3,], type="o", col="green")
  #   points(mu[4,], type="o", col="yellow")
  # }

  Sys.sleep(0.5)
  # E-step: Computation of the fractional component assignments
  # Your code here
  for (n in 1:N)
  {
    prob=0
    for (k in 1:K)
    {
      prob=prob+prod(((mu[k,]^x[n,])*((1-mu[k,])^(1-x[n,]))))*pi[k]
    }
    for (k in 1:K)
    {
      z[n,k]=pi[k]*prod(((mu[k,]^x[n,])*((1-mu[k,])^(1-x[n,])))) / prob
```

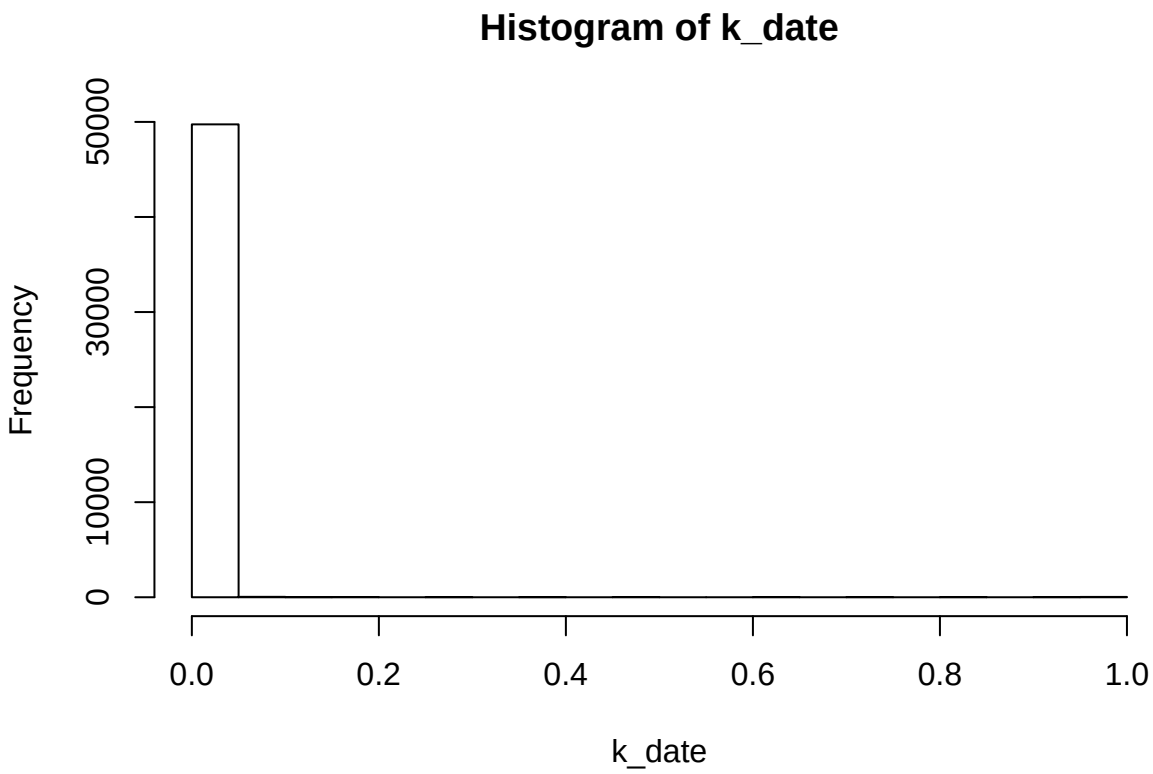
```
    }
  }
  #Log likelihood computation.
  # Your code here
  likelihood <-matrix(0,nrow =1000,ncol = K)
  llik[it] <-0
  for(n in 1:N)
  {
    for (k in 1:K)
    {
      likelihood[n,k] <- pi[k]*prod(((mu[k,]^x[n,])*((1-mu[k,])^(1-x[n,]))))
    }
    llik[it]<- sum(log(rowSums(likelihood)))
  }
  cat("iteration: ", it, "log likelihood: ", llik[it], "\n")
  flush.console()
  # Stop if the log likelihood has not changed significantly
  # Your code here
  if (it > 1)
  {
    if (llik[it]-llik[it-1] < min_change)
    {
      if(K == 2)
      {
        plot(mu[1,], type="o", col="blue", ylim=c(0,1))
        points(mu[2,], type="o", col="red")
      }
      else if(K==3)
      {
        plot(mu[1,], type="o", col="blue", ylim=c(0,1))
        points(mu[2,], type="o", col="red")
        points(mu[3,], type="o", col="green")
      }
      else
      {
        plot(mu[1,], type="o", col="blue", ylim=c(0,1))
        points(mu[2,], type="o", col="red")
        points(mu[3,], type="o", col="green")
        points(mu[4,], type="o", col="yellow")
      }
      break
    }
  }
  #M-step: ML parameter estimation from the data and fractional component assignments
  # Your code here
    mu<- (t(z) %*% x) /colSums(z)
    pi <- colSums(z)/N
  }
pi
mu
plot(llik[1:it], type="o")
}
```

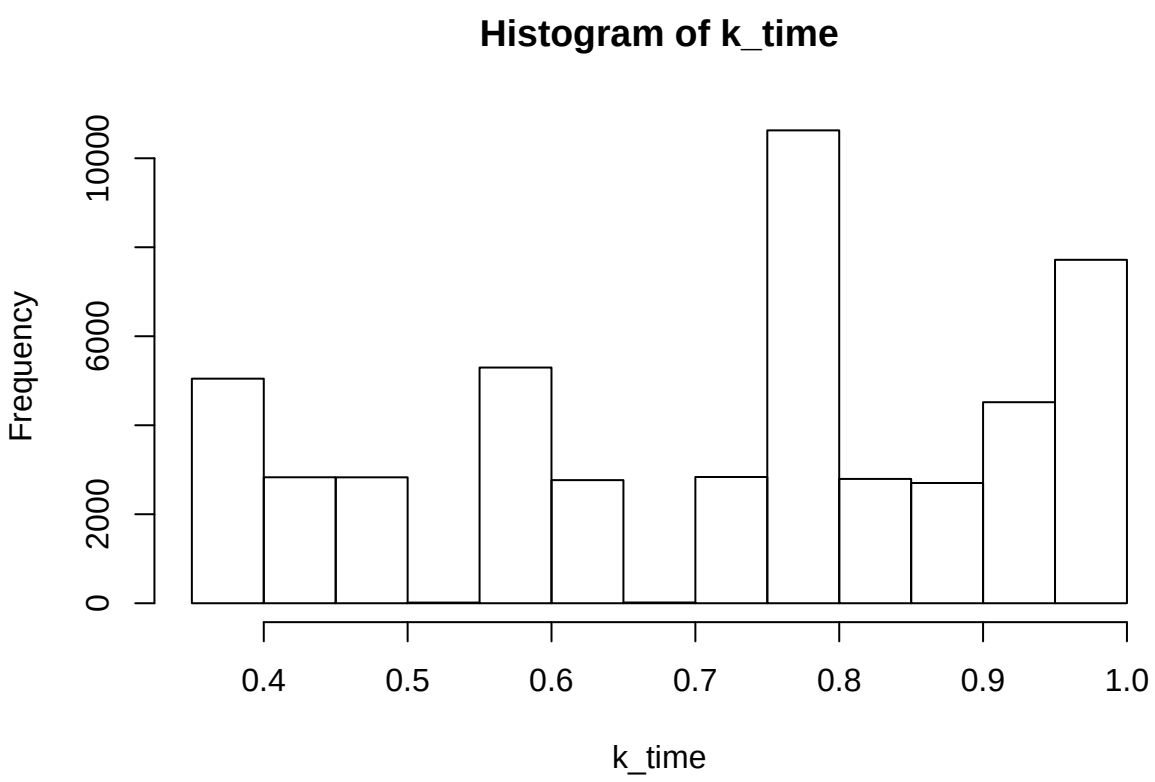
## Assignment 1.

### Kernal methods:




Histogram of k_date
Frequency
0
10000
30000
50000
0.0
0.2
0.4
0.6
0.8
1.0
k_date




The width of the distance kernel of 25Kms is chosen, Since Sweden is close to the arctic circle the temperature fluctuations remain uniform over large distances.

The width of the distance for days is taken as 7 as people generally talk about the weeks weather. It is noticebly uniform in any given week.

The width of the distance for the hours is taken as 12 as the temperature during any given day is defined by night and day which is 2 groups out of 24.

If we look at the histograms above, we see the histograms over the three kernels. As we can see we have in all three kernels values that are ranging from 0 to 1, but most of them are zero, and very few are close to one. This is reasonable since we are looking at distances only relatively close to our own position, since those are the position which are mostly related to our own weather. We are also looking at days, relatively close to the chosen day, since more than a months difference can give a major difference to air temperature. We are also looking at hours closely related to our chosen one, since just a small amount of hours can matter a lot for temperature.

The choice of kernel widths are sensitive means that more weight is given to closer points. A point can be close w.r.t. any of the3 variables (distance, date, time). In short, all plots below show that the Kernel values decrease when the 2nd point is more distant (the 1st point is fixed). I.e. the Kernels are sensitive. They gives higher weights to closer points and smaller weights to distant points. Below, some examples with Unsuitable widths are shown. We can see that the Kernel values will diminish too fast or too slow. h_distance = 0.25 , h_date = 1, h_time= 0.25 these kernel widths will make the kernel value diminish too fast and hence unsuitable. h_distance = 1000 , h_date = 200, h_time= 12 these kernel widths will make the kernel value diminish too slowly and hence also unsuitable.

## Assignment 2

### Support vector machines

```
##            Predicted svm
## Actual Test nonspam spam
##     nonspam    1346   56
##     spam        155  744
```

```
## [1] "The misclassificaiton rate is 0.0916992611907866"
```

```
##            Predicted svm
## Actual Test nonspam spam
##     nonspam    1340   62
##     spam        131  768
```

```
## [1] "The misclassificaiton rate is 0.0838765754019991"
```

```
##            Predicted svm
## Actual Test nonspam spam
##     nonspam    1336   66
##     spam        125  774
```

```
## [1] "The misclassificaiton rate is 0.0830073880921338"
```

The Misclassication error rates of the models are 0.0916,0.0838 and 0.0830 for the models with width of 0.05 as the hyperparameter for the kernal of type Radial Basis. C is the cost of contraint violation. This is the 'C' Constant of the regularisation term in the Lagrange formulation. The purpose of this is to behave as a penalty term for violation of the rules of the classification so as to not overfit the model.

### Apendix

```
knitr::opts_chunk$set(echo = TRUE)
library(geosphere)
library(kernlab)
library(ggplot2)
library(lubridate)
set.seed(1234567890)
stations <- read.csv("stations.csv")
temps <- read.csv("temps50k.csv")
st <- merge(stations,temps,by="station_number")
rm(stations,temps)
st$time <- as.POSIXct(st$time,format="%H:%M:%S")
a <- 58.4166
b <- 15.6333
hdist<-250000
hdate<-7
htime<-12
date <- "2001-11-04"
timeseq <- c("04:00:00", "06:00:00", "08:00:00","10:00:00","12:00:00","12:00:00","14:00:00","16:00:00",
```

```
timeseq <- as.POSIXct(timeseq,format="%H:%M:%S")

coords<-cbind(st$longitude, st$latitude)
ykernalsum<-c()
ykernalprod<-c()
final<-c()
for (i in 1:length(timeseq)){
  h_distance<-(distHaversine(coords,c(b,a))/hdist)
  k_distance<-exp(-(h_distance)^2)
  h_date <- abs(as.numeric(as.Date(st$date) - as.Date(date)))
  h_date[h_date > 182] <- 365 - h_date[h_date > 182]
  h_date <- h_date /hdate
  k_date <- exp(-(h_date)^2)
  h_time <- as.numeric(difftime(time1 = st$time ,time2= timeseq[i], units = "hours"))
  h_time <- abs(h_time)
  h_time[h_time > 12] = 24 - h_time[h_time > 12]
  h_time <- h_time / htime
  k_time <- exp(-(h_time)^2)
  ksum <- k_distance + k_date + k_time
  ykernalsum[i] <- sum(ksum*st$air_temperature) / sum(ksum)
  kprod <- k_distance * k_date * k_time
  ykernalprod[i] <- sum(kprod*st$air_temperature) / sum(kprod)
  df <- data.frame(Time = timeseq[i], ykernalsum = ykernalsum[i], ykernalprod = ykernalprod[i])
  final <- rbind(final, df)
}

p1 <- ggplot(final, aes(Time)) +
  geom_point(aes(y = ykernalsum)) +
  geom_point(aes(y = ykernalprod)) +
  geom_line(aes(y = ykernalsum, color = "Additive kernal")) +
  geom_line(aes(y = ykernalprod, color = "Multiplicative kernal")) +
  scale_color_manual(values=c("#E69F00", "#56B4E9")) +
  ylab("Temperature forecast") +
  theme_dark()+ggtitle("Predicted Temperature")
p1

hist(k_distance)
hist(k_date)
hist(k_time)

set.seed(1234567890)
data(spam)
n<-dim(spam)[1]
id<-sample(1:n,floor(n*0.5))
train<-spam[id,]
test<-spam[-id,]
xtrain<-as.matrix(train[,-58])
ytrain<-as.matrix(train[,58])
xtest<-as.matrix(test[,-58])
ytest<-as.matrix(test[,58])
xtrain2<-train[,-58]
svmmodel0.5<- ksvm(xtrain, ytrain, kernel="rbfdot",kpar=list(sigma=0.05),C=0.5)
svmmodel1<- ksvm(xtrain, ytrain, kernel="rbfdot",kpar=list(sigma=0.05),C=1)
```

```r
svmmodel5<- ksvm(xtrain, ytrain, kernel="rbfdot",kpar=list(sigma=0.05),C=5)

svmpredict0.5<-predict(svmmodel0.5, xtest, type="response")
svmpredict1<- predict(svmmodel1, xtest, type="response")
svmpredict5<- predict(svmmodel5, xtest, type="response")

consvm0.5<- table(ytest, svmpredict0.5)
names(dimnames(consvm0.5)) <- c("Actual Test", "Predicted svm")
consvmres0.5<-caret::confusionMatrix(consvm0.5)
consvm0.5
mse3.1<-(1-(sum(diag(consvm0.5))/sum(consvm0.5)))
paste("The misclassificaiton rate is",mse3.1)

consvm1<- table(ytest, svmpredict1)
names(dimnames(consvm1)) <- c("Actual Test", "Predicted svm")
consvmres1<-caret::confusionMatrix(consvm1)
consvm1
mse3.2<-(1-(sum(diag(consvm1))/sum(consvm1)))
paste("The misclassificaiton rate is",mse3.2)

consvm5<- table(ytest, svmpredict5)
names(dimnames(consvm5)) <- c("Actual Test", "Predicted svm")
consvmres5<-caret::confusionMatrix(consvm5)
consvm5
mse3.3<-(1-(sum(diag(consvm5))/sum(consvm5)))
paste("The misclassificaiton rate is",mse3.3)
```

# Lab2ML

*Omkar Bhutra*

*7 December 2018*

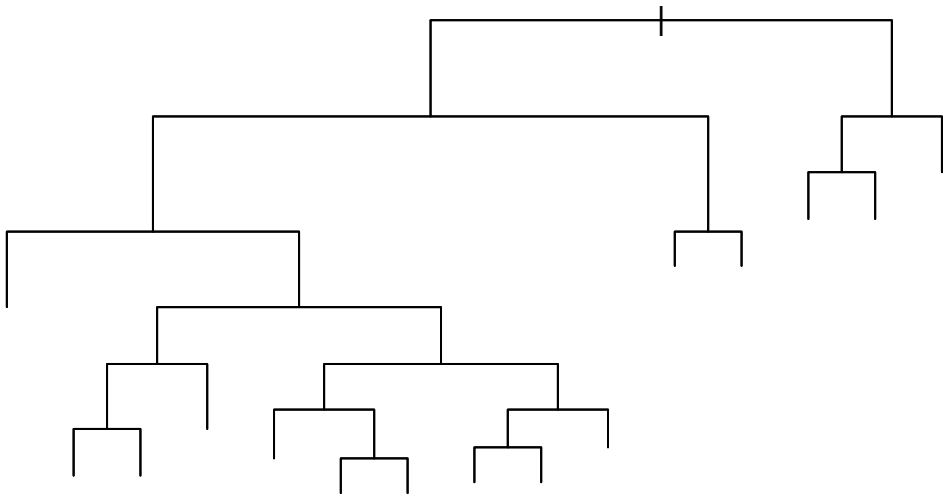
```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

## Loading required package: ggplot2

##
## Attaching package: 'plotly'

## The following object is masked from 'package:ggplot2':
##
##     last_plot

## The following object is masked from 'package:stats':
##
##     filter

## The following object is masked from 'package:graphics':
##
##     layout
```

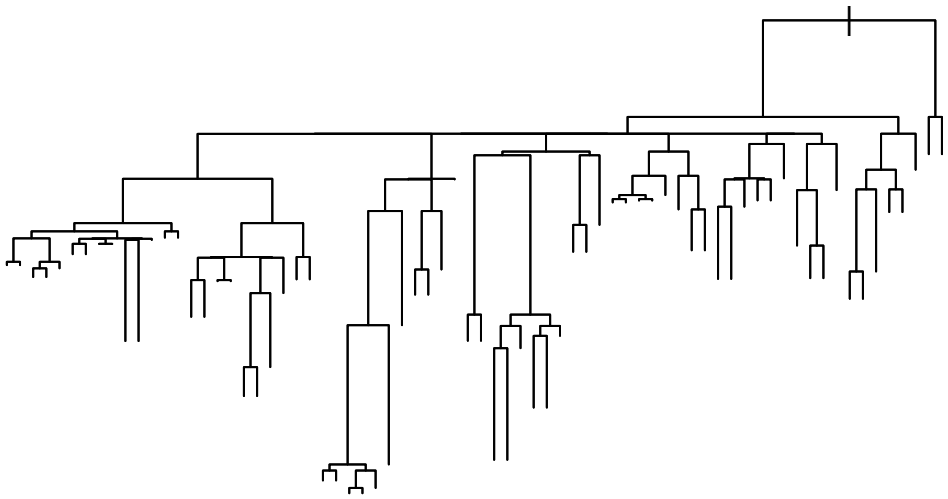
## Assignment 2

### Analysis of credit scoring

```
##
## Classification tree:
## tree(formula = as.factor(good_bad) ~ ., data = train, split = "deviance")
## Variables actually used in tree construction:
## [1] "savings"  "duration" "history"  "age"      "purpose"  "amount"
## [7] "resident" "other"
## Number of terminal nodes:  15
## Residual mean deviance:  0.9569 = 458.3 / 479
## Misclassification error rate: 0.2105 = 104 / 494
```

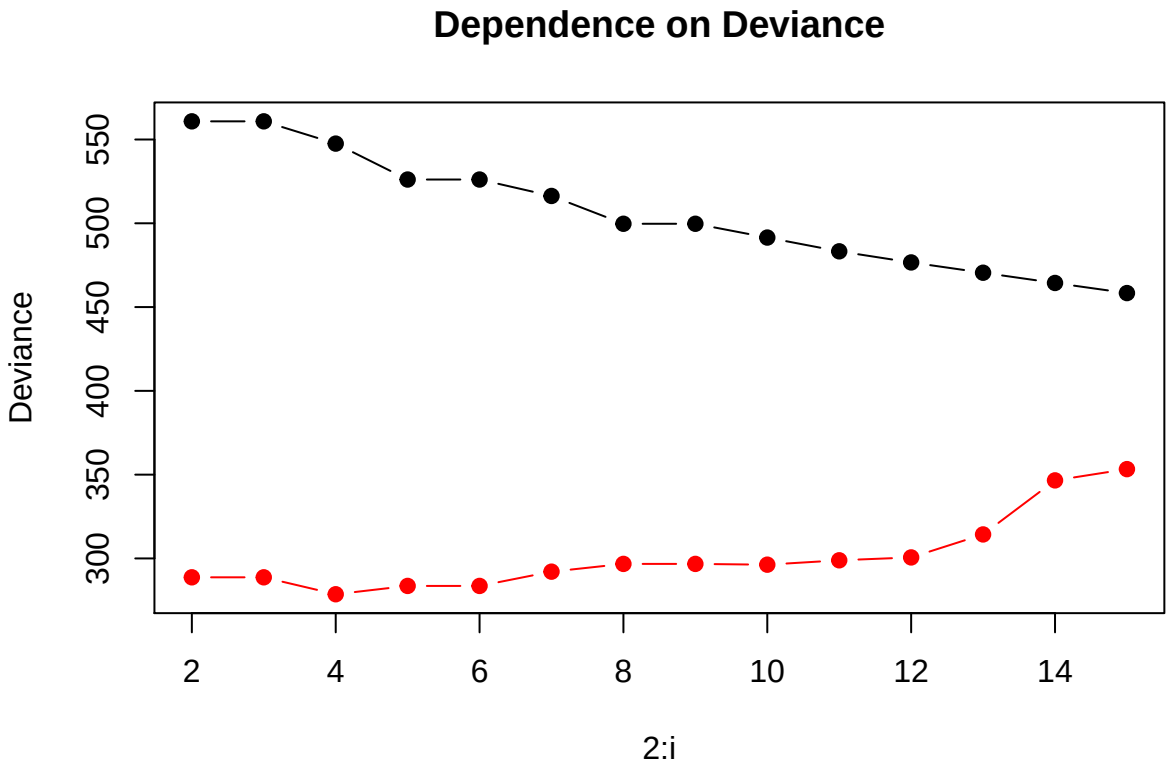
```
##
## Classification tree:
## tree(formula = as.factor(good_bad) ~ ., data = train, split = "gini")
## Variables actually used in tree construction:
##  [1] "foreign"  "coapp"    "depends"  "telephon" "existcr"  "savings"
##  [7] "history"  "property" "marital"  "duration" "employed" "age"
## [13] "housing"  "amount"   "purpose"  "resident" "job"      "installp"
## Number of terminal nodes:  72
## Residual mean deviance:  1.015 = 428.5 / 422
## Misclassification error rate: 0.2368 = 117 / 494
```





```
## [1] 0.212
```

```
## [1] 0.23
```



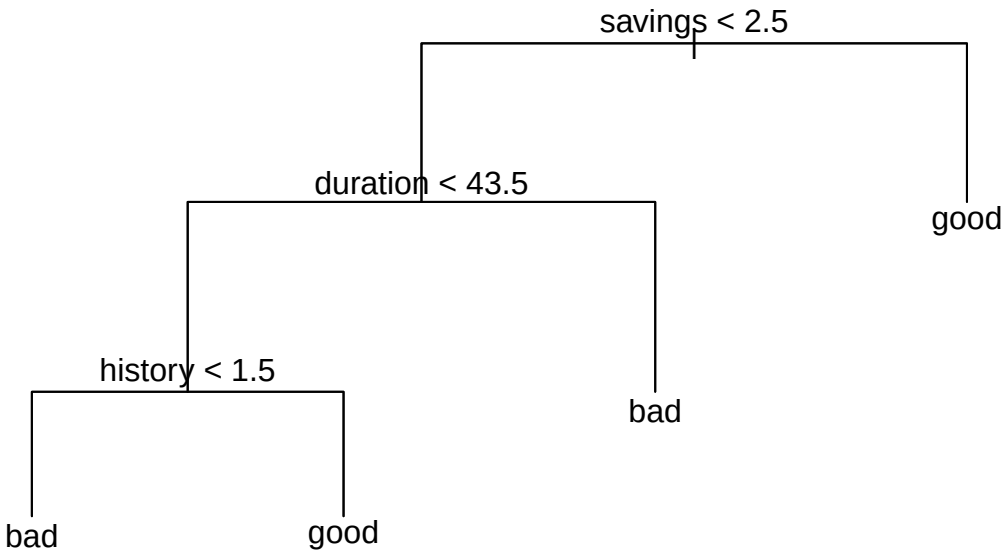

```
##
## Classification tree:
## snip.tree(tree = treestep1, nodes = c(5L, 3L, 9L))
## Variables actually used in tree construction:
## [1] "savings"  "duration" "history"
## Number of terminal nodes:  4
## Residual mean deviance:  1.117 = 547.5 / 490
## Misclassification error rate: 0.251 = 124 / 494
```

```
## [1] "Confusion Matrix"
```

```
##       fit
##        bad good
##   bad   22   53
##   good  12  163
```

```
## [1] "Misclassification rate"
```

```
## [1] 0.26
```




The misclassification rate is reported to be 26% with 22 true negatives and 163 true positives. The classification is done to find good customers that may pay back loans on time. The deviance vs the tree depth is plotted in the given figure. The line for training is shown in red vs the line for validation is shown in black. The optimal tree depth i.e the lowest deviance is present at the tree depth of 4. The optimal tree is shown in the figure. The savings lesser than 2.5 is considered bad, duration lesser than 43.5 is considered good and history lesser than 1.5 is considered bad.

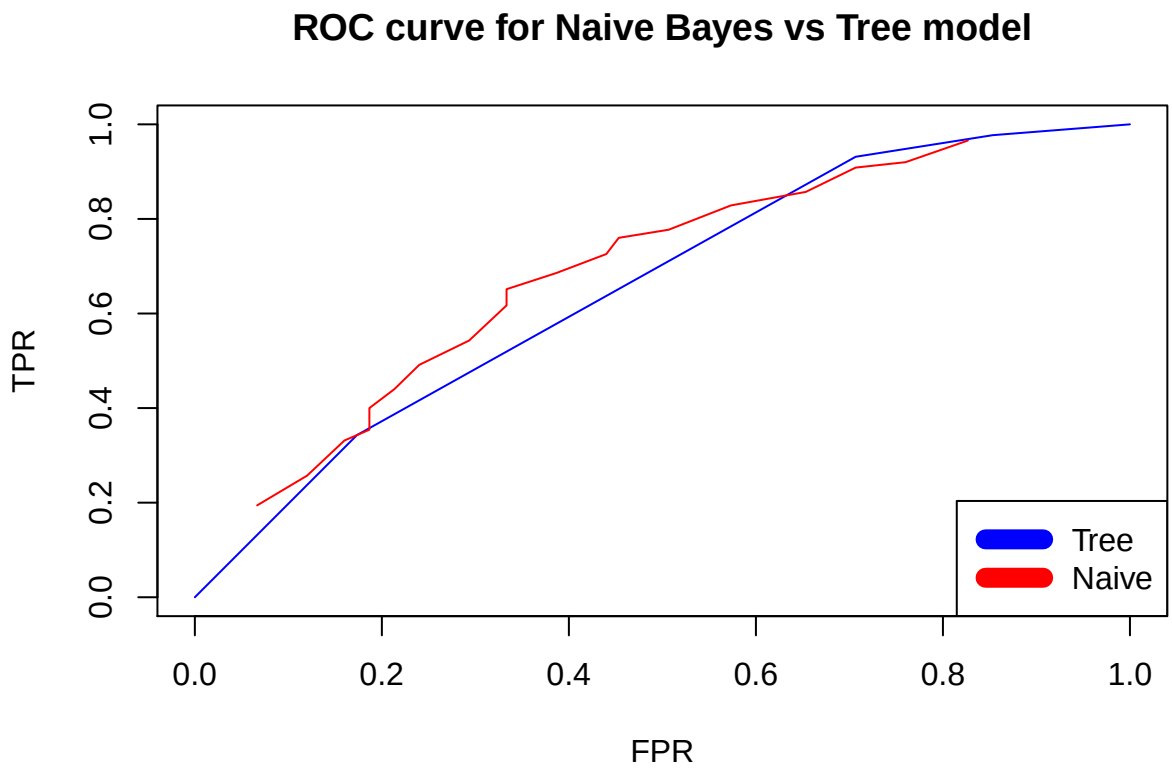
```
##       nbtest
##        bad good
##   bad   50   25
##   good  61  114
```

```
##       nbtrain
##        bad good
##   bad   95   52
##   good  98  255
```

```
## Misclassification rate on train data with Naive Bayes classification is:  0.3
```

```
## Misclassification rate on test data using Naive Bayes classification is:  0.344
```

Misclassification rate on train data with Naive Bayes classification is: 30%. Misclassification rate on test data using Naive Bayes classification is: 34.4%. The error rate has increased from step 3 from 26% to 34.4% for the test data which implies that naive baye's is not good predictor.




It is seen that Naive Baye's performs better according to the ROC plot,with a higher true positive rate and lesser false positive rate.

This type of graph is called a Receiver Operating Characteristic curve (or ROC curve.) It is a plot of the true positive rate against the false positive rate for the different possible cutpoints of a diagnostic test.

An ROC curve demonstrates several things:

It shows the tradeoff between sensitivity and specificity (any increase in sensitivity will be accompanied by a decrease in specificity). The closer the curve follows the left-hand border and then the top border of the ROC space, the more accurate the test. The closer the curve comes to the 45-degree diagonal of the ROC space, the less accurate the test.

```
##       nbtest1
##        FALSE TRUE
##   bad     66    9
##   good   130   45
```

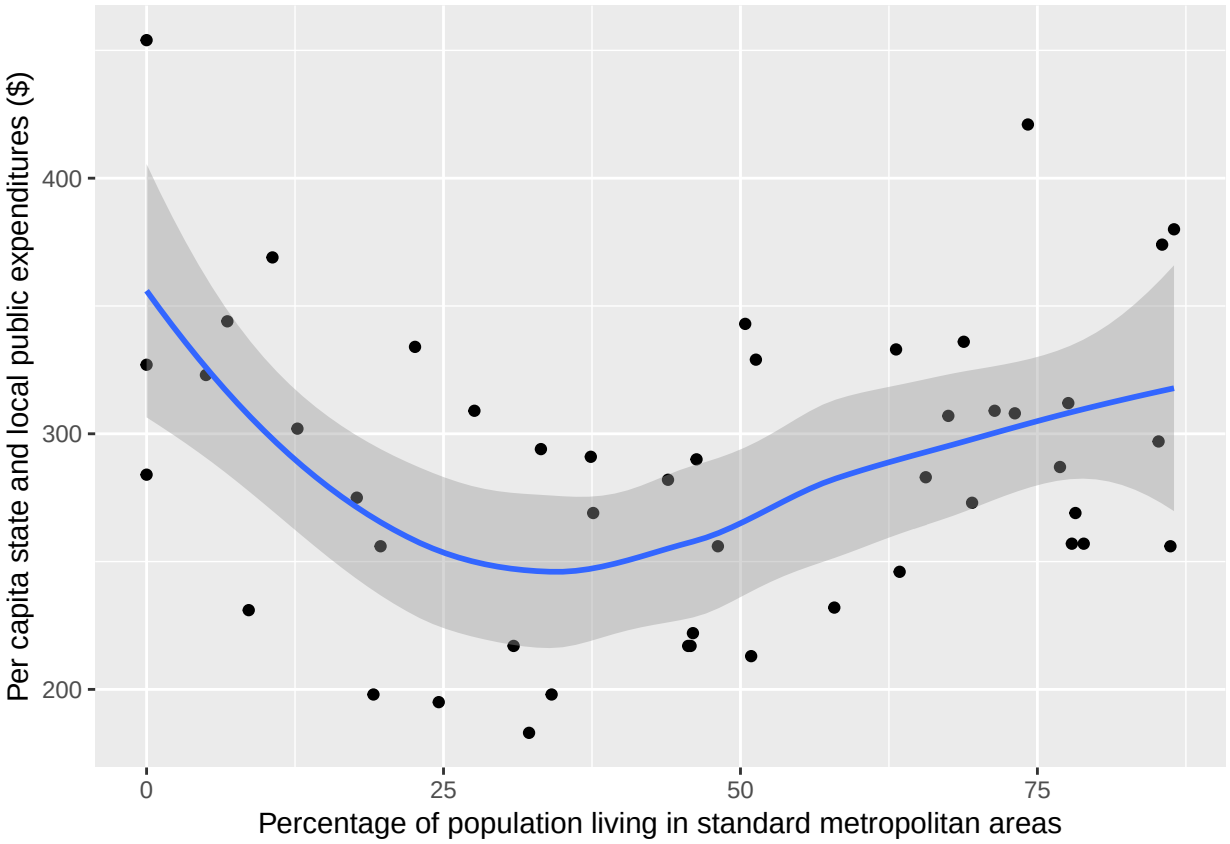
```
##       nbtrain1
##        FALSE TRUE
##   bad    137   10
##   good   263   90
```

```
## [1] 0.546
```

```
## [1] 0.556
```

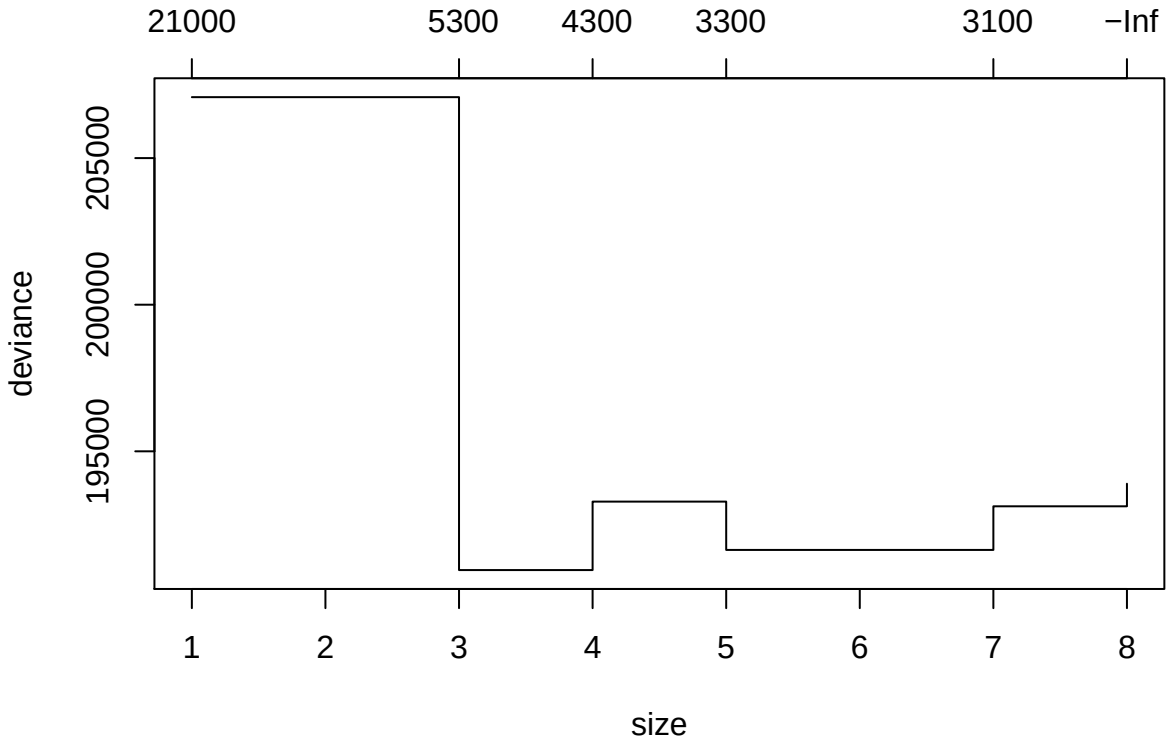
The misclassification rate for the training dataset is reported as 54.6% and slightly higher for the test at 55.6%. This error rate is much higher from Naive Baye's also which stood at 34.4%.

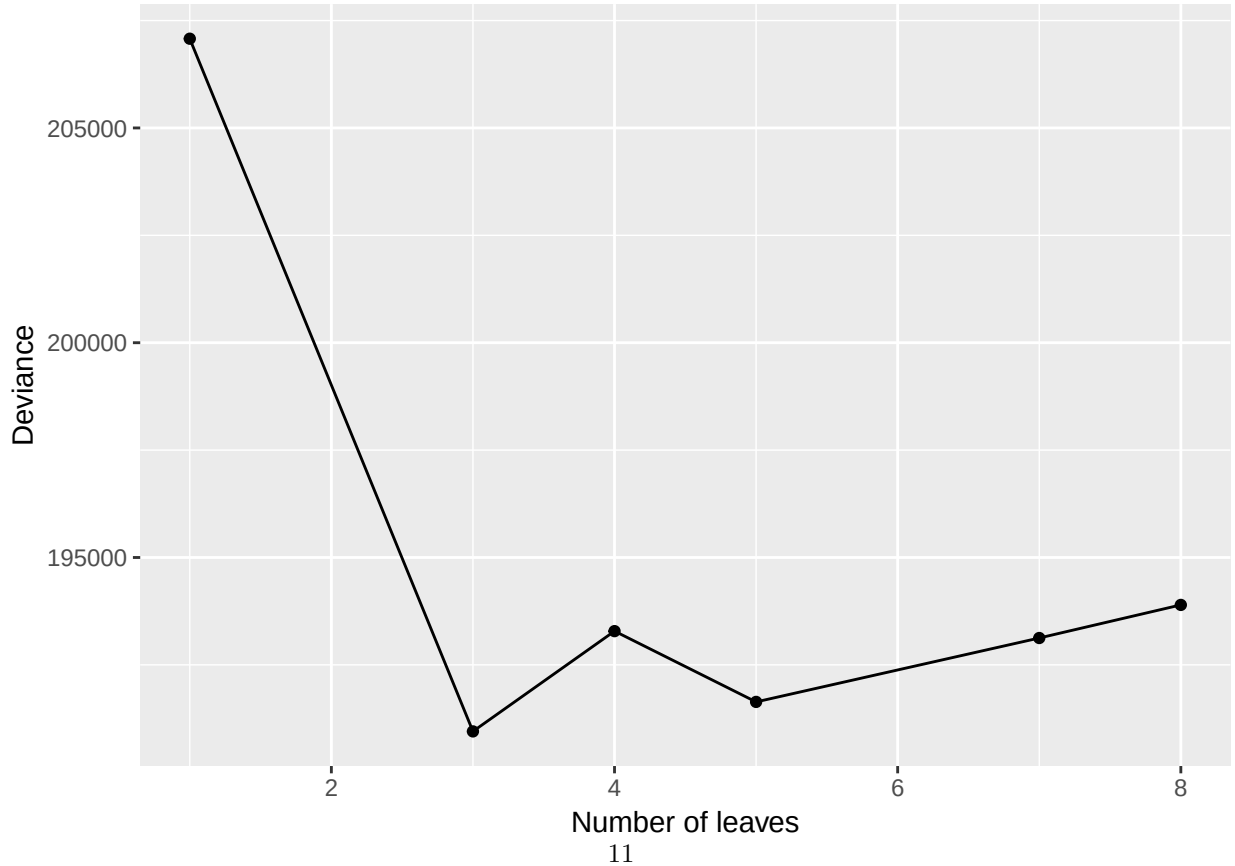
## Assignment3 Uncertainty Estimation

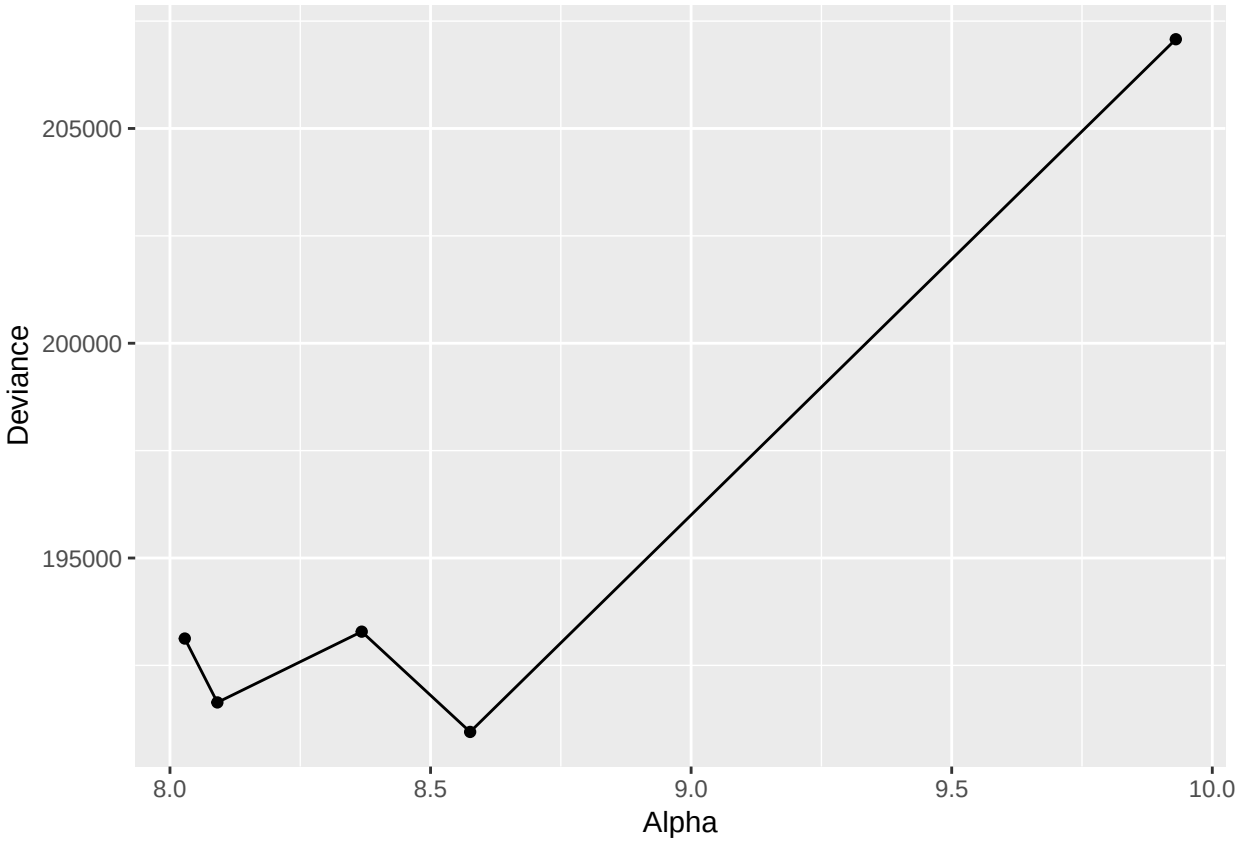


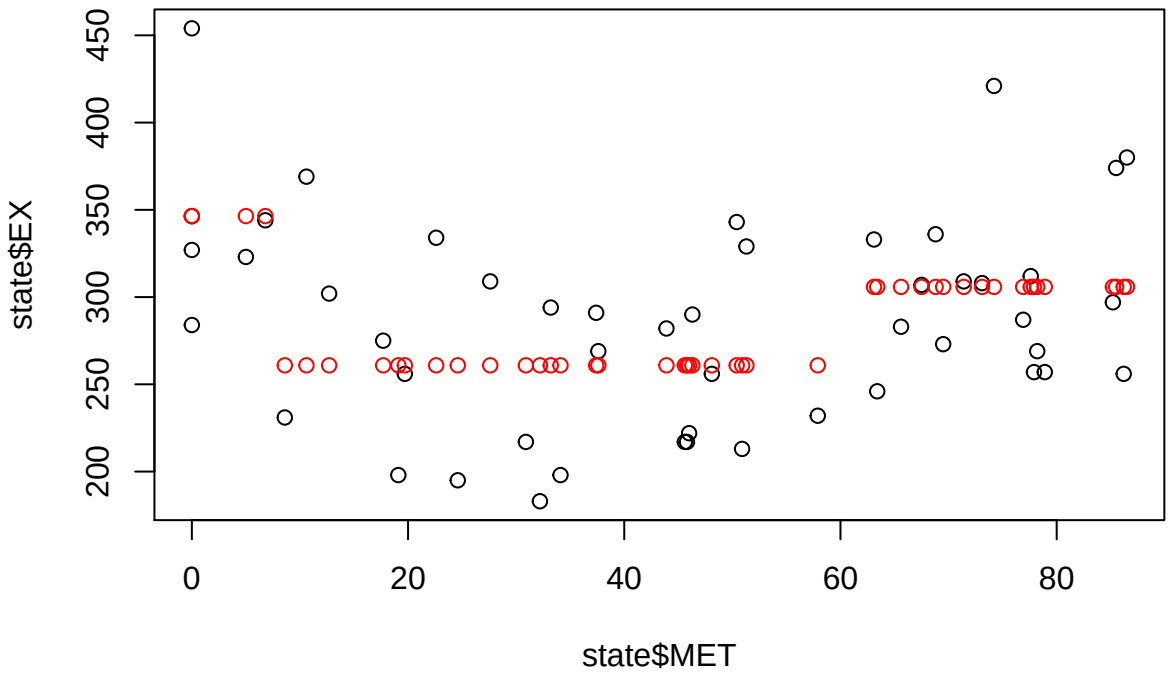

As seen from the plot data seems to be scatter all around, the variance is high. A decision tree would be better to be used here.

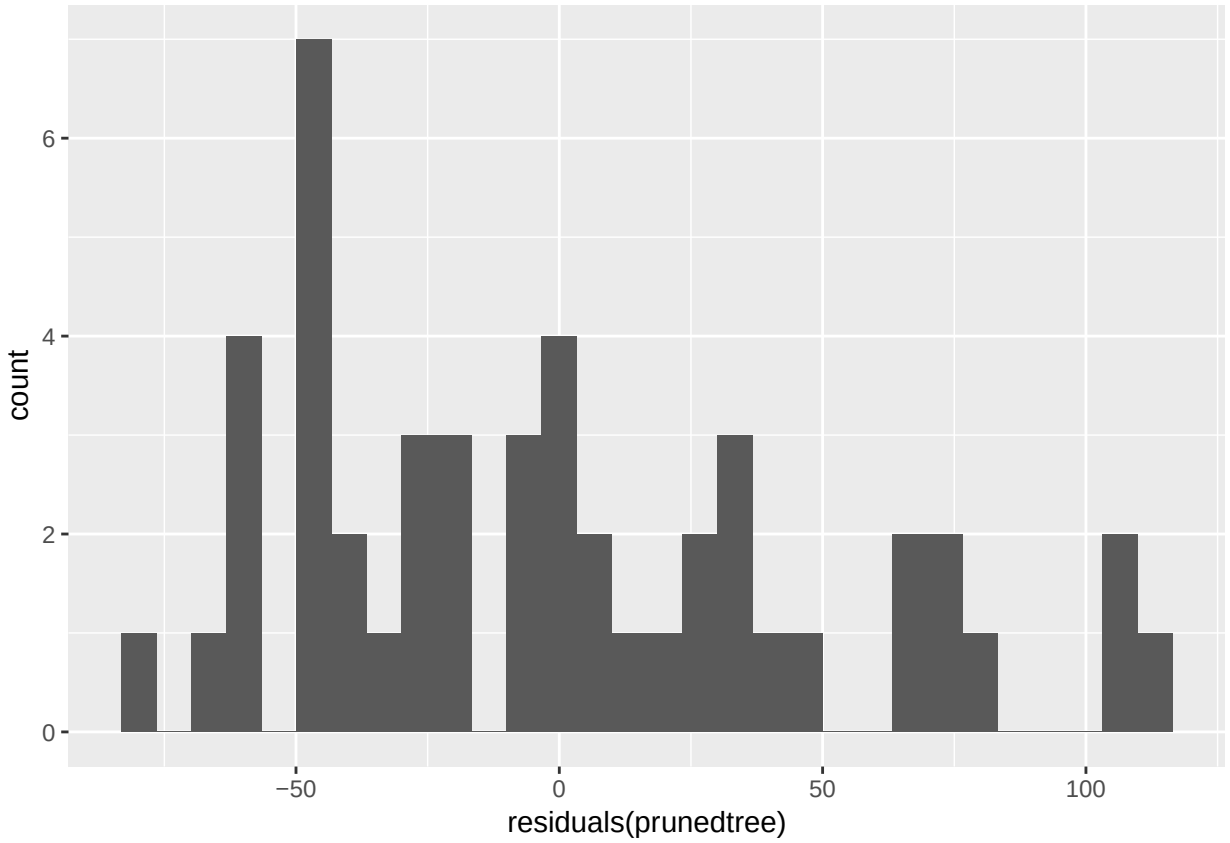
**3.2**

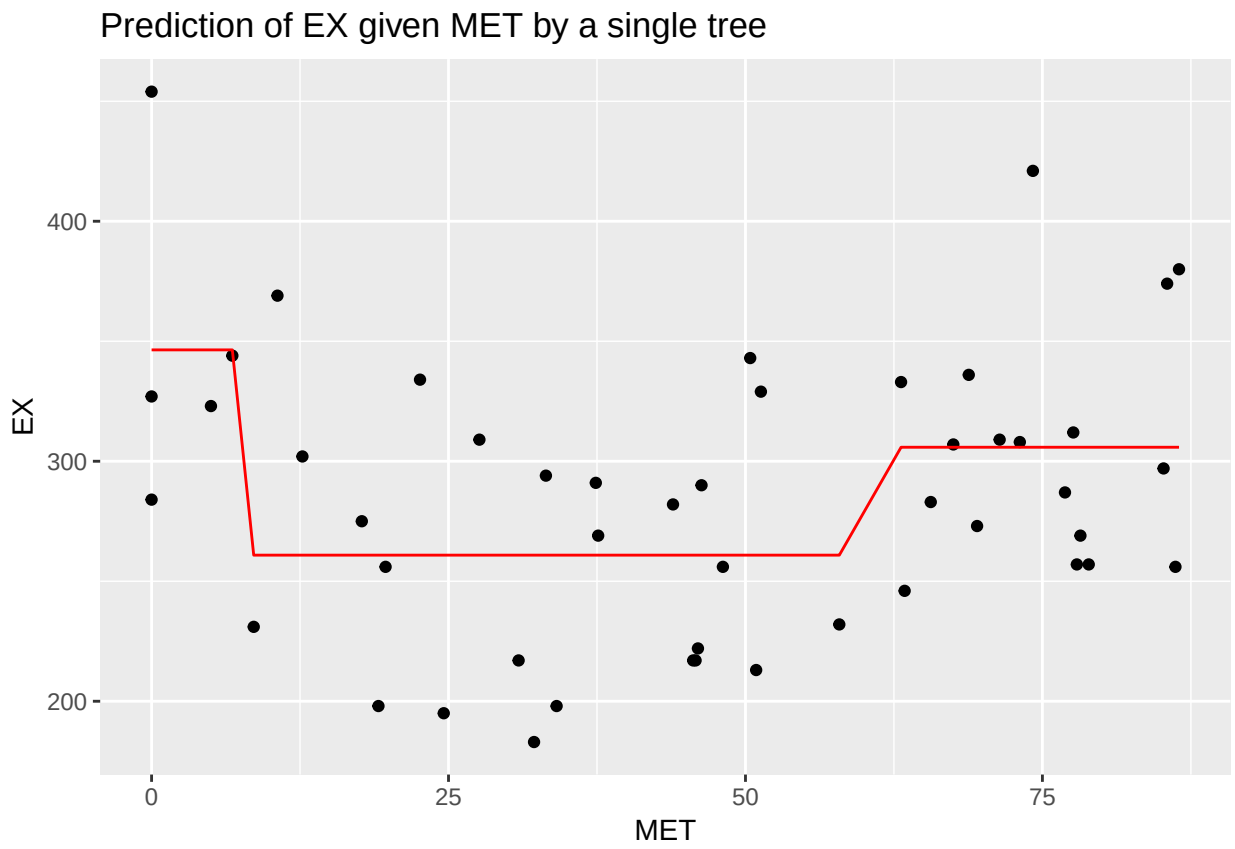

21000
5300
4300
3300
3100
−Inf
deviance
205000
200000
195000
1
2
3
4
5
6
7
8
size


Deviance
205000
200000
195000
2
4
6
8
Number of leaves


205000
200000
195000
Deviance
8.0
8.5
9.0
9.5
10.0
Alpha


state$EX
200
250
300
350
400
450
0
20
40
60
80
state$MET

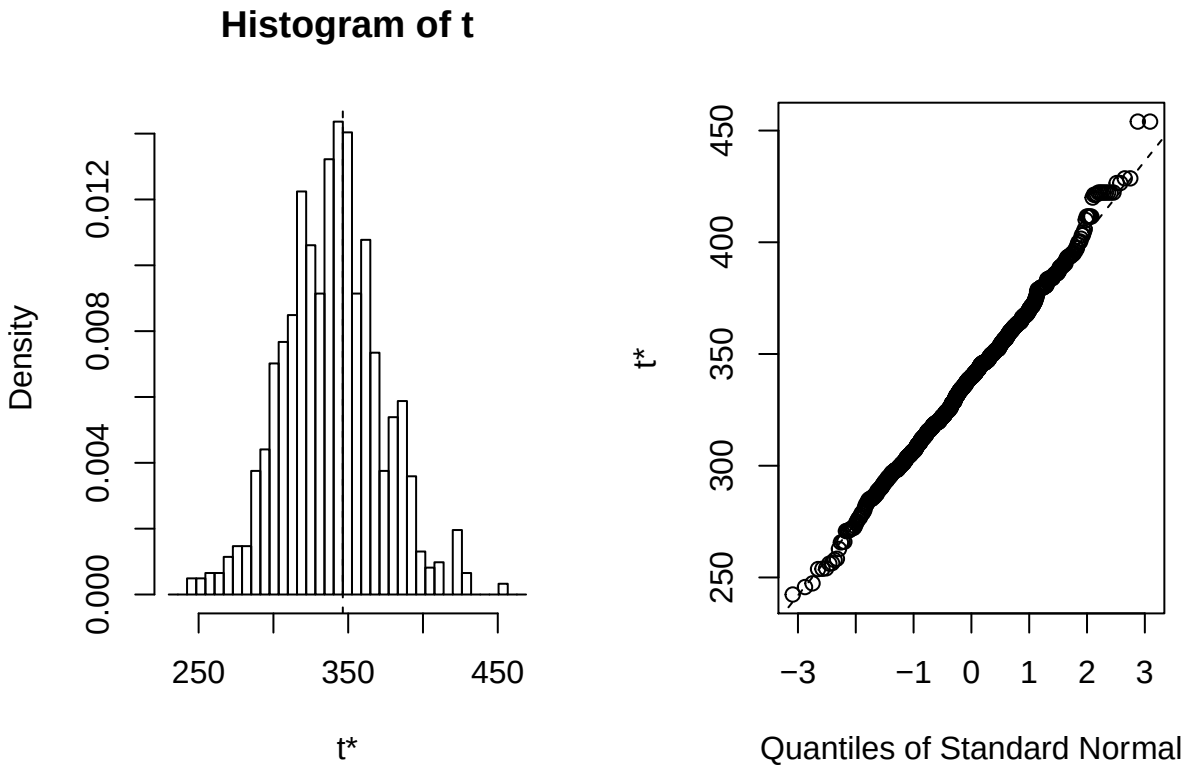

count
6
4
2
0
−50
0
50
100
residuals(prunedtree)

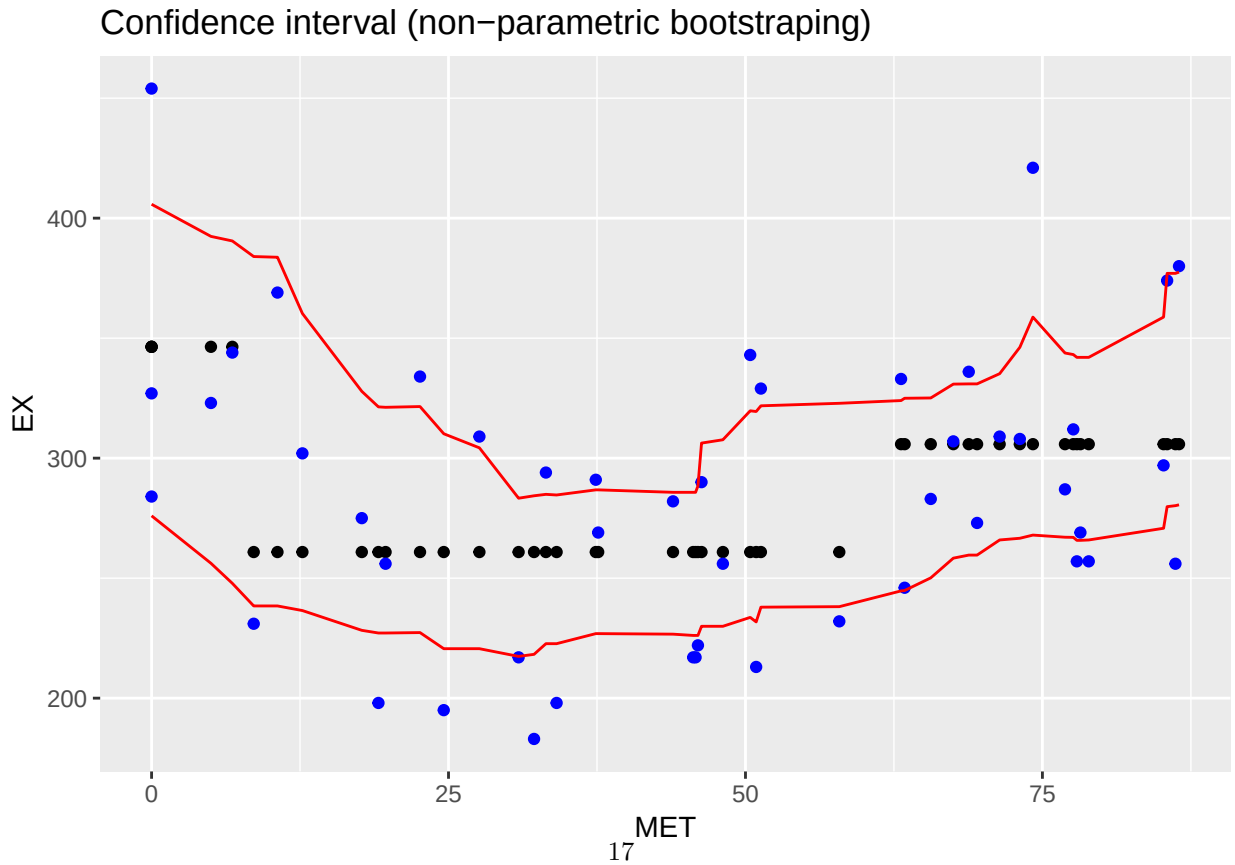



The optimal tree depth is of size 3, as it has the lowest variance when compared to others. From the plot we can say that residuals can be improved or reduced by applying better fitting.

From the plots, we can see the 3 values for each leaves on the scatter plot. The residuals seem to be normally distributed but skewed to the left, like a Chi-squared distribution that could manage negative values. This tells us that the fit is not as good as it should be, since we should expect the distribution of the residuals to be symmetric between the positive axis to the negative or vice versa.

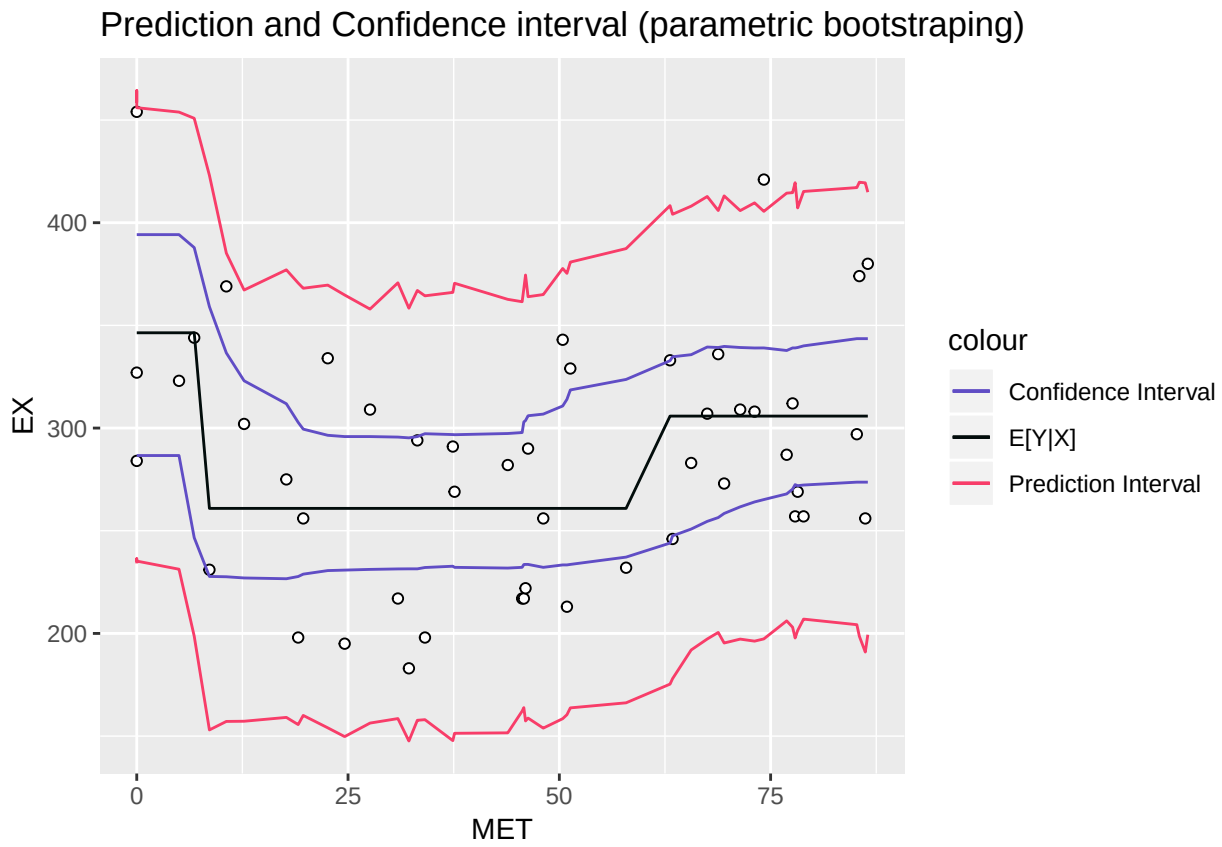
### 3.3 Non-Paramatric Bootstrap

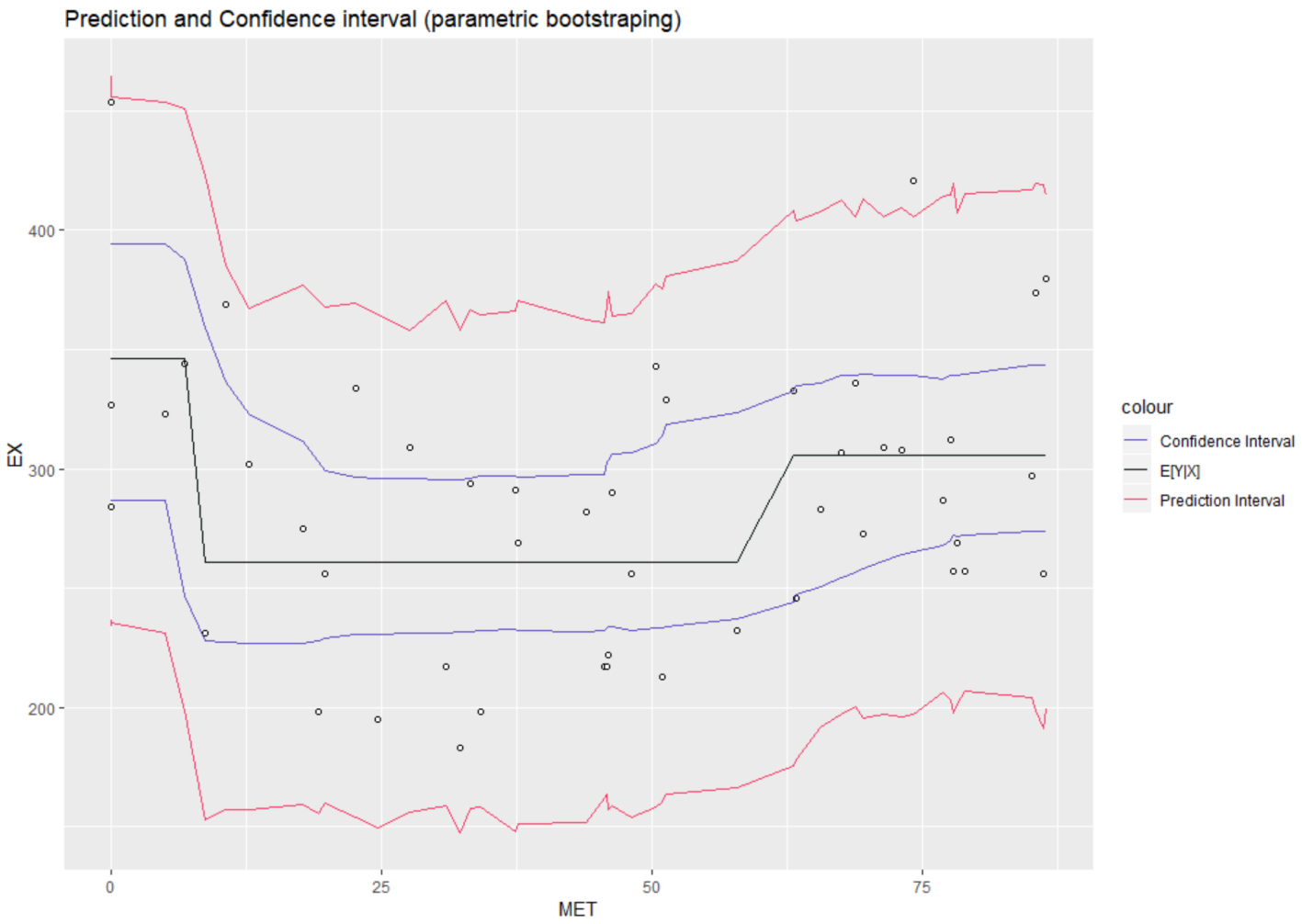


Confidence band is a combination of different confidence intervals computed for different replicates of bootstrap. The confidence band for the parametric bootstrap is volatile due to the impact of bias on bootstrap. Considering the width of confidence interval, the result of regression model computed in part 2 is reliable as it lies within the confidence band

Calculating the confidence interval using non-parametric bootstrap for the following statistic which is given by the tree: $\mu = E[Y|X]$ In this case, since no assumptions are made about the distribution, we want to generate a sample with replacement Xs and then calculate our statistic $\mu s = E[Y|Xs]$ several times (in this case 1000). After that, for each x belongs to X we get the 2.5 and 97.5 percentile of our $\mu s$,percentile and build the confidence interval. The confidence bands for this statistic seems bumpy. This is because we are calculating our statistic $\mu$ given 3 intervals (3 leaves) for each bootstrap sample which is not a smooth function and makes it bumpy when averaging over all of the samples. The predictions from our model from step 2 seems reliable since it captures the general trend of the data and it's not affected by the outliers.






In this case we assume the following distribution for our data:

$$Y \sim N(\mu, \sigma^2)$$

Where

$$\mu$$

is given by our tree:

$$\mu = \hat{f}(X|\hat{\Theta}(X,Y)) = E_{Y\sim N}[Y|X]$$

and

$$\sigma^2$$

is given by the variance of the residuals:

$$\sigma^2 = Var(Y - \mu) = Var(Y - \hat{f}(X|\hat{\Theta}(X,Y))$$

The first step is to create a sample

$$Y_s$$

from Y|X. Given this sample we want to create intervals for μ and for Y . The interval for μ is going to be constructed by calculating multiple

$$\mu_s$$

from each bootstrap sample generated from

$$Y \sim N(\mu, \sigma^2)$$

.It ss worth noting that each µs is being generated by a different

$$\hat{f}_s$$

, which means, that we are going to train a different tree for each tuple

$$(X, Y_s), s \in (1, 2, ..., S)$$

where S is the number of samplings we are going to perform (in this case 1000).

$$\text{t}_s = \hat{f}_s(X|\hat{\Theta}(X, Y_s))$$

As for the interval of Y|X, we get

$$\mu_s$$

from a bootstrap sample tuple

$$(X, Y_s)$$

and then we get our bootstrap sample

$$Y_{boot}$$

from

$$N(\mu_s, \sigma^2)$$

. We repeat this procedure 1000 times and again, we select the 2.5 and 97.5 percentile to create the 95% interval. The width of the confidence band seems to resemble the one from the previous task. So, as stated above, the predictions from the model in step 2 seems to be reliable. As for the prediction band, to the naked eye it doesn't seem that 5% of the data is outside of it and it's totally fine to happen since we assume the following distribution for the data

$$Y \sim N(\mu, \sigma^2)$$

. This means, that if the plot contained all of the samples Yboot we would be able to see that 5% of the sampled data is outside of the prediction interval. Another way to confirm that this interval seems right is to remember that

$$2\sigma$$

from the mean

$$\mu$$

amounts for 95% of the observed data. In this case, the standard deviation of the residuals if roughly 50%, which means that the prediction interval of 95% should be ±100 around the mean.

# Assignment4

## 4.1

```
##   [1] 1.489914e-02 9.998545e-04 2.954195e-05 1.608532e-05 1.091077e-05
##   [6] 3.939315e-06 1.414911e-06 4.981545e-07 4.262849e-07 2.577774e-07
##  [11] 2.080001e-07 1.587511e-07 1.425823e-07 1.126727e-07 7.232246e-08
##  [16] 6.878939e-08 5.307373e-08 4.373598e-08 3.975200e-08 3.627181e-08
##  [21] 3.473207e-08 2.838554e-08 2.750156e-08 2.356802e-08 2.057859e-08
##  [26] 1.921151e-08 1.772579e-08 1.719151e-08 1.546958e-08 1.450458e-08
##  [31] 1.349010e-08 1.229577e-08 1.210005e-08 1.144210e-08 1.068630e-08
```

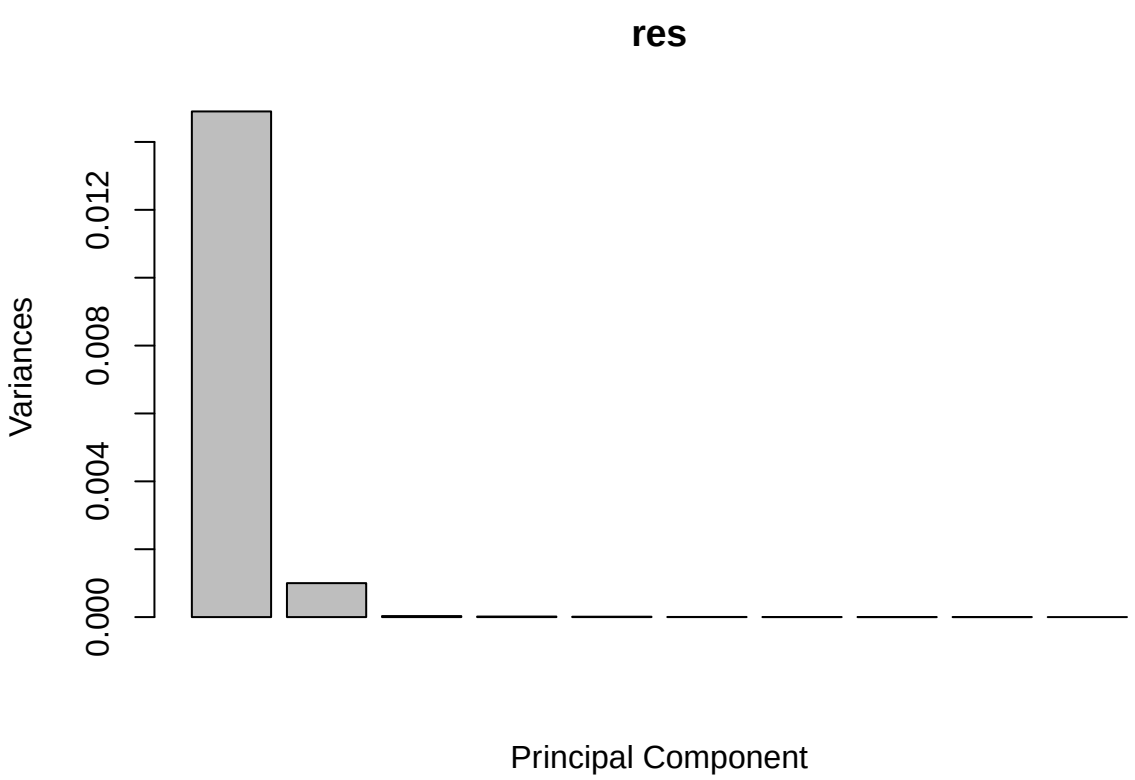
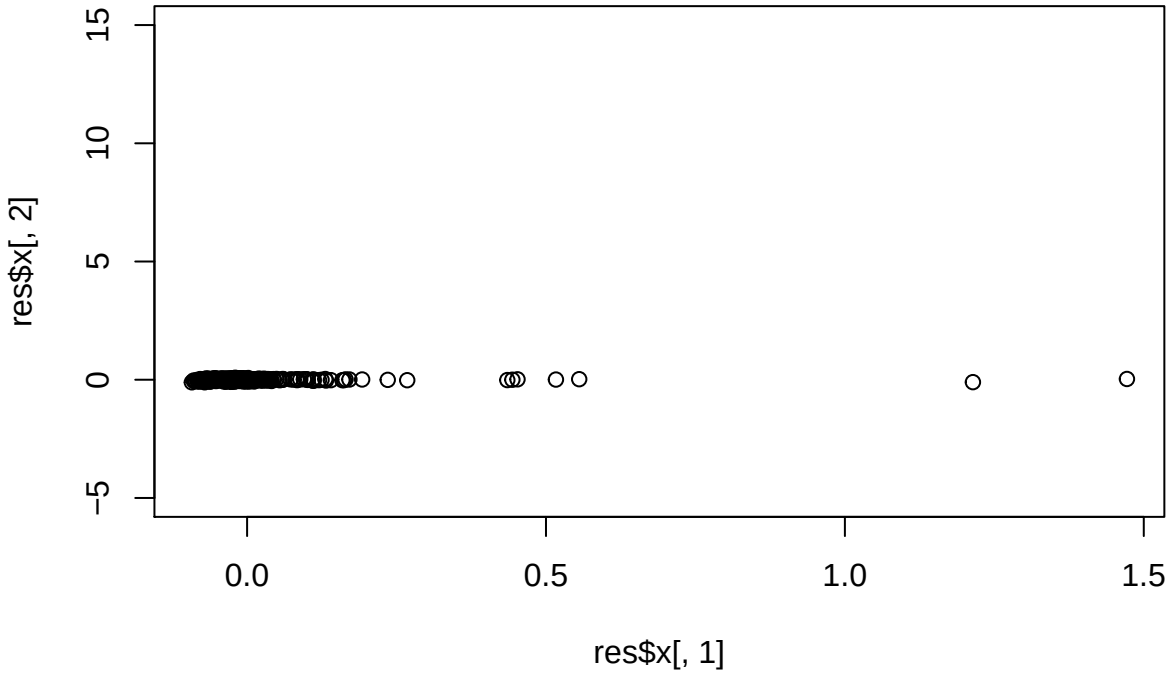
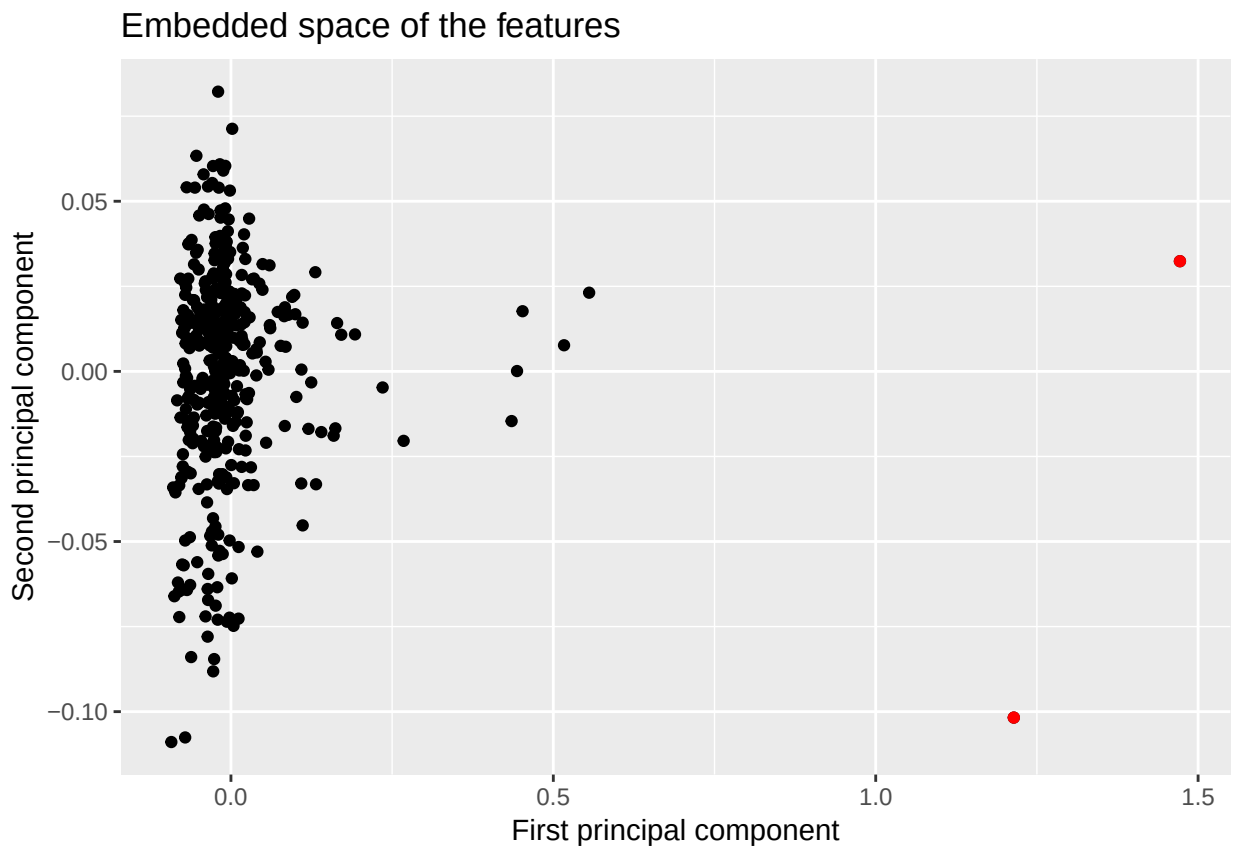

res
Variances
0.000
0.004
0.008
0.012
Principal Component

Table 1: Eigenvalues for the top axis of the new basis

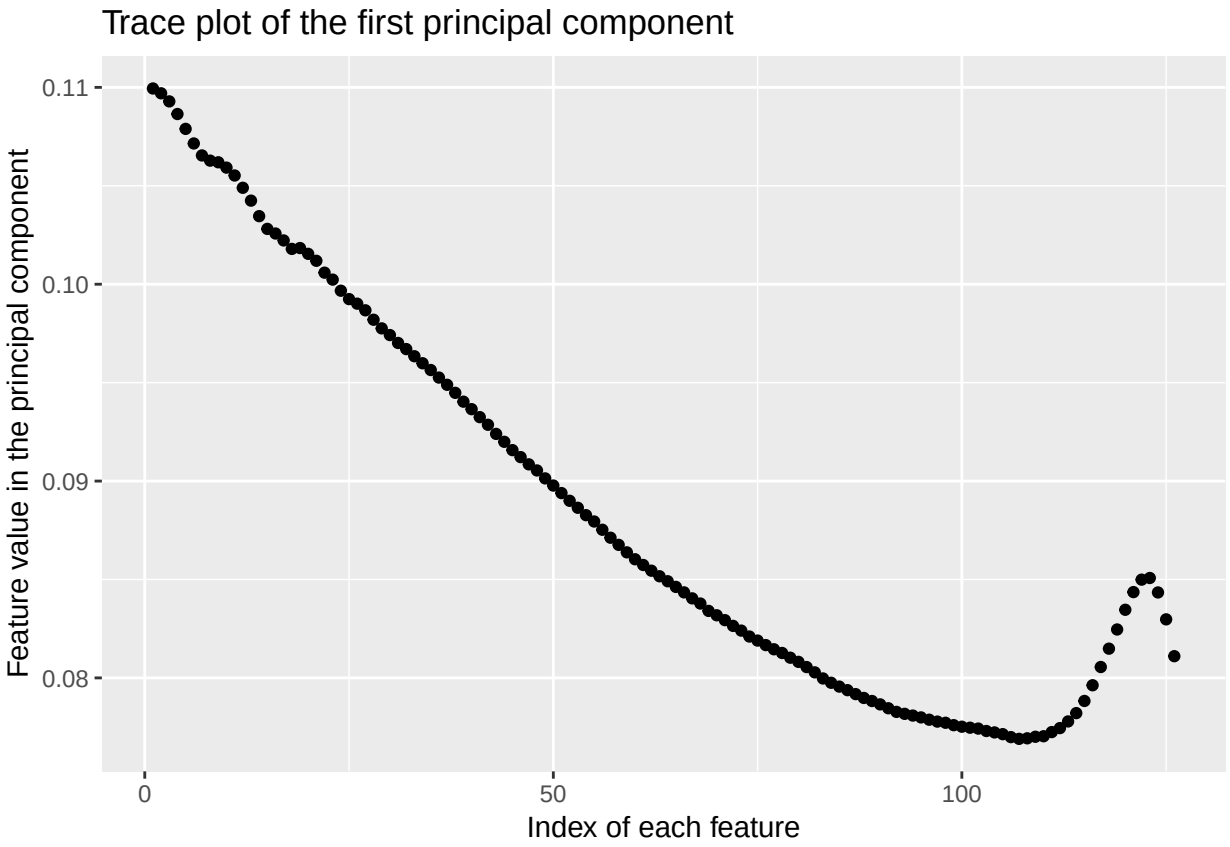
| x |
|---|
| 0.0148991 |
| 0.0009999 |

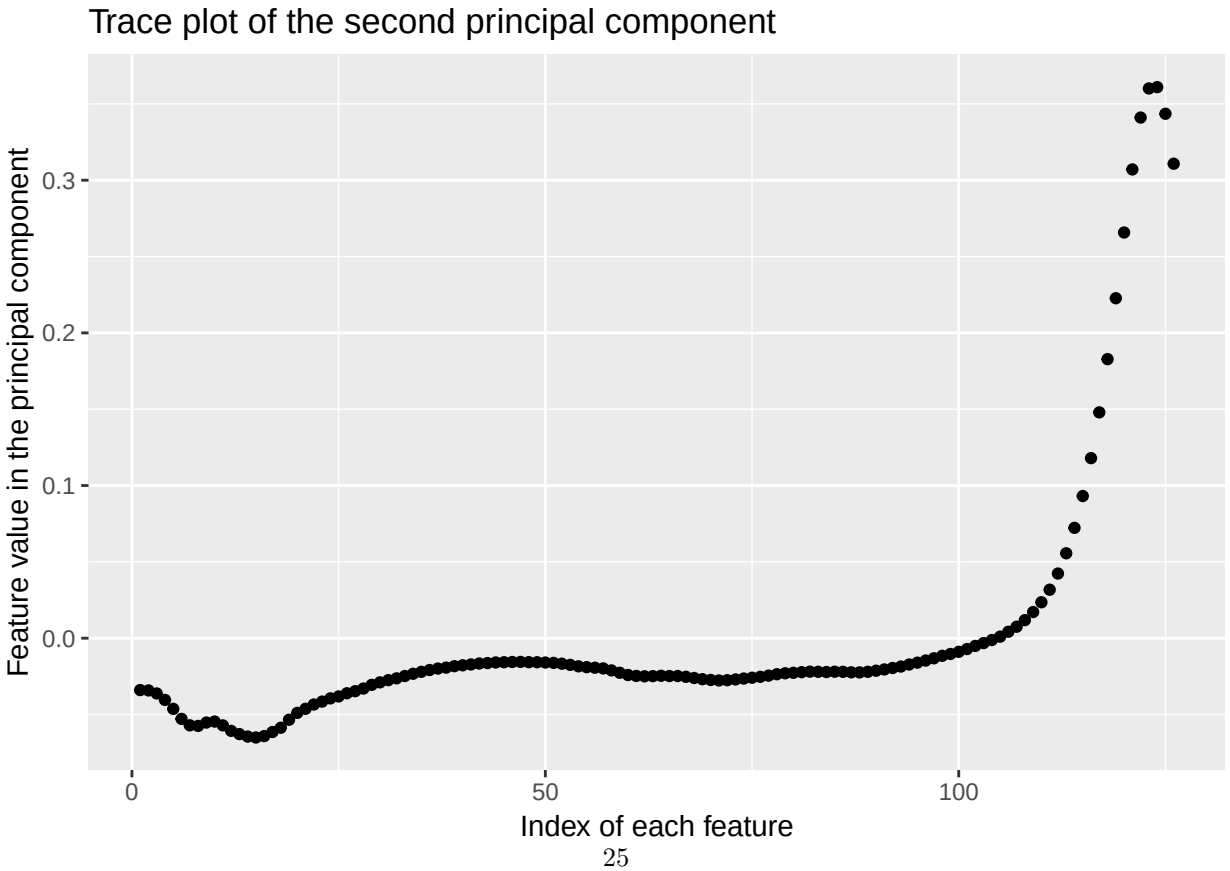




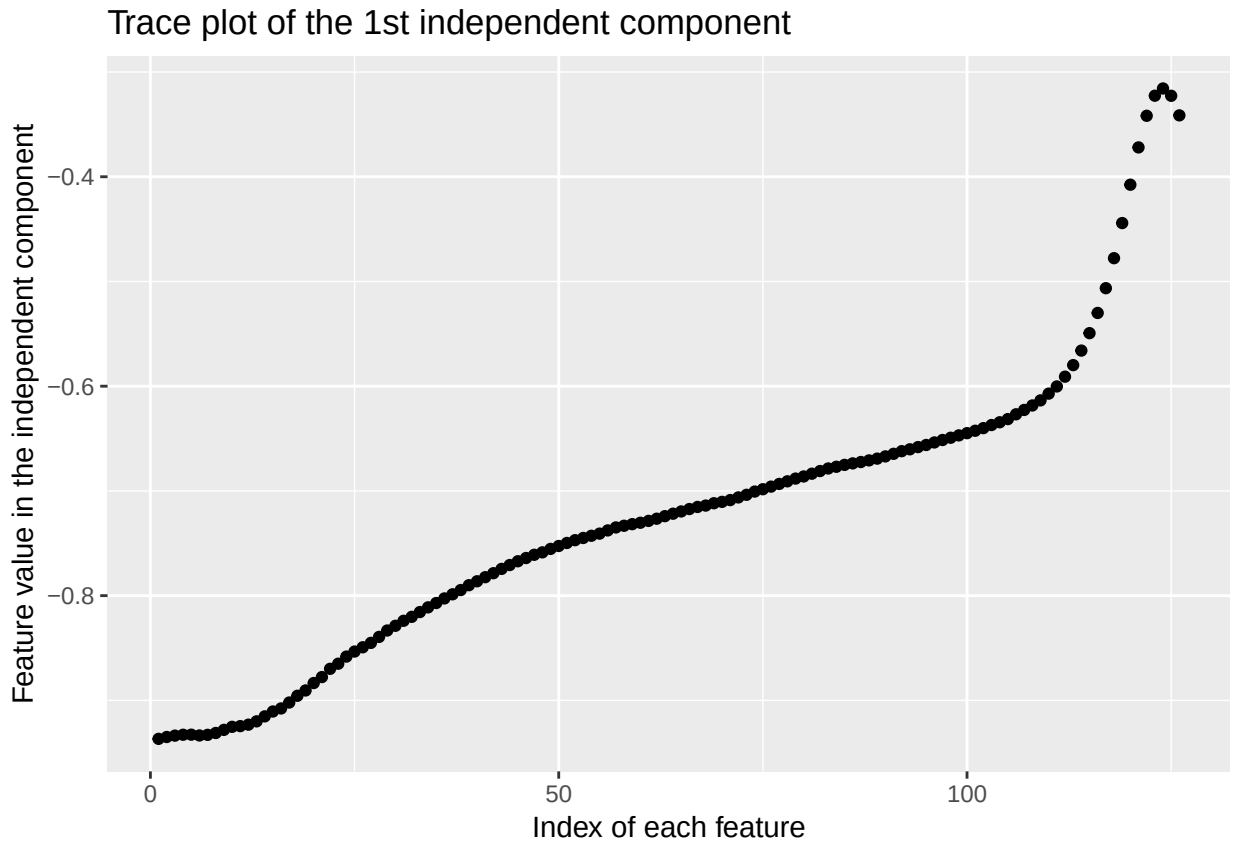

The first plot shows how many Principal componenets should be extracted. According to the plot the first 2 principal components should be extracted. Yes ,Unusual diesel fuels are seen as outliers in the 2nd plot.

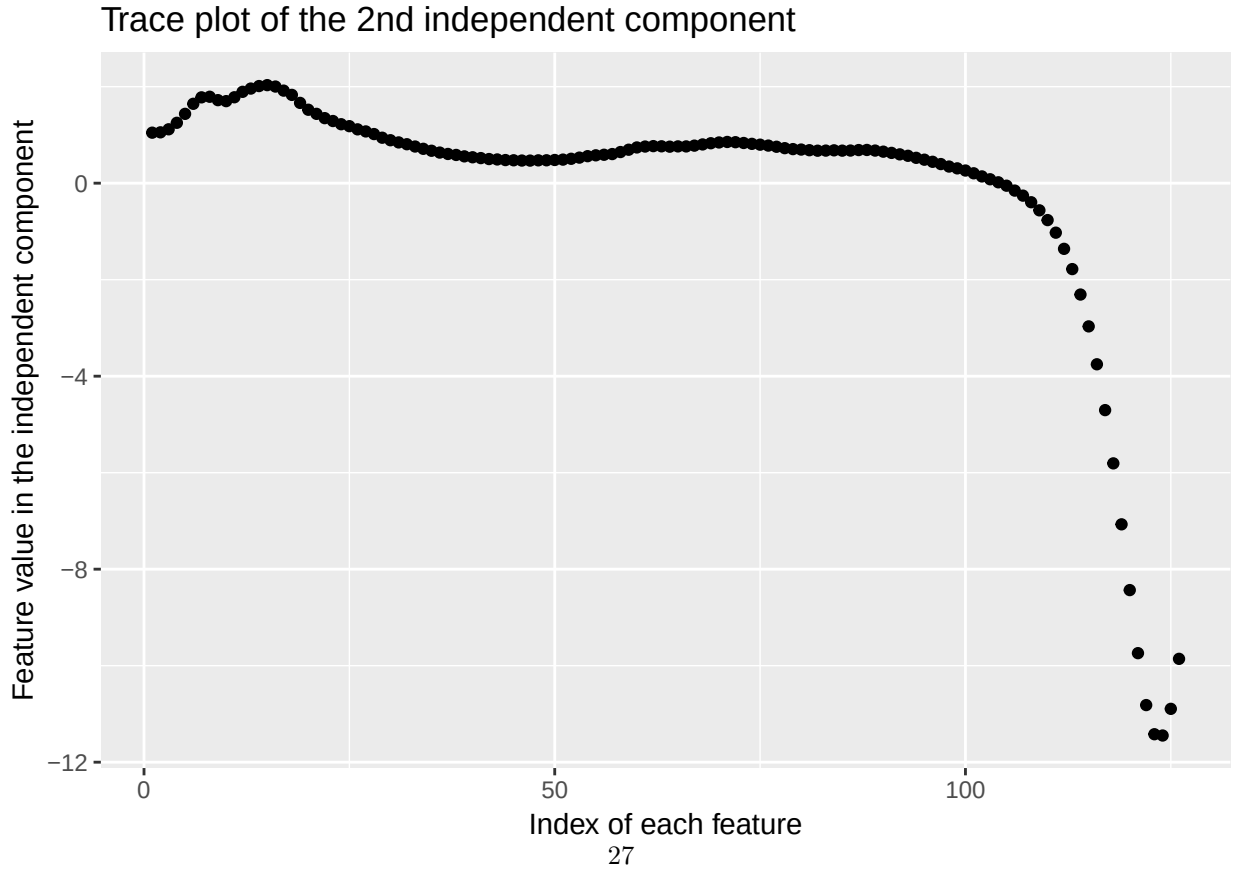
## 4.2

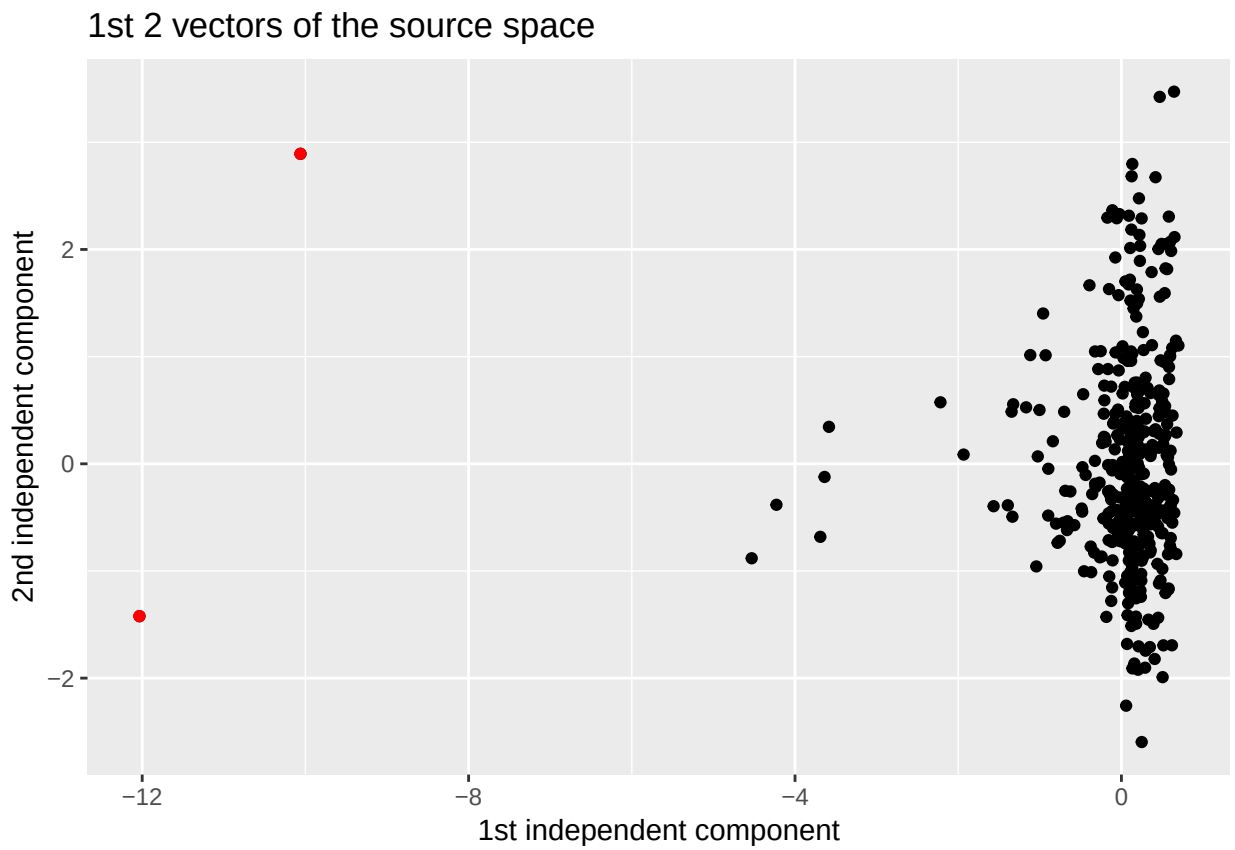

Trace plot of the first principal component
Feature value in the principal component
0.11
0.10
0.09
0.08
0
50
100
Index of each feature


Trace plot of the second principal component
Feature value in the principal component
0.3
0.2
0.1
0.0
0
50
100
Index of each feature

**4.3**


Trace plot of the 1st independent component
Feature value in the independent component
−0.4
−0.6
−0.8
0
50
100
Index of each feature


Trace plot of the 2nd independent component
Feature value in the independent component
0
−4
−8
−12
0
50
100
Index of each feature




The trace plots of W' show that ICA found a similar basis as the one that PCA found. The only difference is that the axis is inverted. This can be seen on the plot with the scores. This means that the latent variables found by PCA and ICA have the properties of being statistically independent, non-gaussian and identifiable. In this case W' represents a direct projection from the feature space to the source space found by ICA. It first projects the data to the PCA space S = X W' = (XK)W and then it projects it again to the source space

$$S = XW' = X_{white}W$$

. As stated above, we get the same latent space but with the axis inverted and elongated. In addition we also get the same two unusual diesel fuels and once again, they are coloured in red.

### Apendix

```
knitr::opts_chunk$set(echo = TRUE)
library(dplyr)
library(plotly)
library(ggplot2)
library(xlsx)
library(readxl)
library(tree)
library(e1071)
library(boot)
library(kableExtra)
library(fastICA)
library(knitr)
```

```
creditscoring = read_excel("creditscoring.xls")
n=dim(creditscoring)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train=creditscoring[id,]
tester=creditscoring[-id,]
m=dim(tester)[1]
id1<-sample(1:m,floor(m*0.5))
test<-tester[id1,]
validation<-tester[-id1,]
treestep1<-tree(as.factor(good_bad) ~ .,data=train,split = "deviance")
summary(treestep1)

treestep2<-tree(as.factor(good_bad) ~ .,data=train,split = "gini")
summary(treestep2)

devfit1<-predict(treestep1, newdata = train, type = "class")
ginifit1<-predict(treestep2, newdata = train, type = "class")

plot(treestep1)

plot(treestep2)
tabdev1<-table(devfit1,train$good_bad)
tabgini1<-table(ginifit1,train$good_bad)
msrdev1 <- 1-sum(diag(tabdev1))/sum(tabdev1)
msrdev1
msrgini1 <- 1-sum(diag(tabgini1))/sum(tabgini1)
msrgini1

devfit2<-predict(treestep1,newdata = test,type = "class")
ginifit2<-predict(treestep2, newdata = test, type = "class")
plot(devfit2)
plot(ginifit2)
tabdev2<-table(devfit2,test$good_bad)
tabgini2<-table(ginifit2,test$good_bad)
msrdev2 <- 1-sum(diag(tabdev2))/sum(tabdev2)
msrdev2
msrgini2 <- 1-sum(diag(tabgini2))/sum(tabgini2)
msrgini2
i<-summary(treestep1)[4]$size
trainscore<-rep(0,i)
testscore<-rep(0,i)
for (o in 2:i) {
  sniptree<-tree::prune.tree(treestep1,best=o)
  pred=predict(sniptree, newdata=validation,type="tree")
  trainscore[o]=deviance(sniptree)
  testscore[o]=deviance(pred)
}
plot(2:i,trainscore[2:i],col="Black",type = "b", main = "Dependence on Deviance",
     ylim=c(min(testscore[2:i]),max(trainscore)), pch=19, cex=1, ylab="Deviance")
points(2:i,testscore[2:i],col="Red",type="b", pch=19, cex=1)

msrtest = prune.tree(treestep1, best = 4)
```

```
summary(msrtest)
fit = predict(msrtest, newdata = test, type="class")
testconf= table(test$good_bad,fit)
print("Confusion Matrix")
testconf
mcr <- 1-sum(diag(testconf))/sum(testconf)
print("Misclassification rate")
mcr
plot(msrtest)
text(msrtest)
nb<-naiveBayes(as.factor(good_bad) ~ . , data=train)
nbtest = predict(nb, newdata = test[,-20], type = "class") #removing the last column with 'good_bad'
nbtrain = predict(nb,newdata = train[,-20])
# Confusion Matrix Using Naive Bayes
nbtesttab<- table(test$good_bad,nbtest)
print(nbtesttab)
nbtraintab<-table(train$good_bad,nbtrain)
print(nbtraintab)
mcrnbtrain <- 1-sum(diag(nbtraintab))/sum(nbtraintab)
cat("Misclassification rate on train data with Naive Bayes classification is: ", mcrnbtrain)
# Missclassification test data value Using Naive Bayes
mcrnbtest <- 1-sum(diag(nbtesttab))/sum(nbtesttab)
cat("Misclassification rate on test data using Naive Bayes classification is: ", mcrnbtest)
#Q5
df = data.frame(pi=double(), tree_tpr=double(), tree_fpr=double(),
naive_tpr=double(), naive_fpr=double())
for (pi in seq(0.05, 0.95, by=0.05)) {
pred_tree = as.data.frame(predict(msrtest, test))
pred_naive = as.data.frame(predict(nb, test, type="raw"))
pi_tree = ifelse(pred_tree$good > pi, "good", "bad")
pi_naive = ifelse(pred_naive$good > pi, "good", "bad")
tree_table = table(test$good_bad, factor(pi_tree, levels=c("bad", "good")))
naive_table = table(test$good_bad, factor(pi_naive, levels=c("bad", "good")))

df = rbind(df, c(pi,
tree_table[4]/(tree_table[4]+tree_table[2]),
tree_table[3]/(tree_table[3]+tree_table[1]),
naive_table[4]/(naive_table[4]+naive_table[2]),
naive_table[3]/(naive_table[3]+naive_table[1])
))
}
colnames(df) = c("pi", "tpr_tree", "fpr_tree", "tpr_naive", "fpr_naive")
plot(-1, -1, xlim=c(0, 1), ylim=c(0, 1), xlab="FPR", ylab="TPR",
main="ROC curve for Naive Bayes vs Tree model")
lines(df$fpr_tree, df$tpr_tree, lwd=1, col="blue")
lines(df$fpr_naive, df$tpr_naive, lwd=1, col="red")
legend("bottomright", c("Tree", "Naive"), col=c("blue", "red"), lwd=10)

#q6
nbtest1 = predict(nb, test[,-20] , type="raw")
nbtrain1 = predict(nb, train[,-20] , type="raw")
# loss matrix
nbtest1 = (nbtest1[, 2] / nbtest1[, 1]) > 10 # compare with loss matrix
```

```
nbtrain1 = (nbtrain1[, 2] / nbtrain1[, 1]) > 10
# confusion matrix for train & test
nbtabtest = table(test$good_bad,nbtest1)
nbtabtest
nbtabtrain = table(train$good_bad,nbtrain1)
nbtabtrain
# missclasification rates for train and test respectively
1-sum(diag(nbtabtrain))/sum(nbtabtrain)
1-sum(diag(nbtabtest))/sum(nbtabtest)
# 3.1 Data import, reorder and Plot
set.seed(12345)
state = read.csv2("State.csv", header = TRUE)
state = state[order(state$MET),]
ggplot(state)+geom_point(aes(x=MET,y=EX))+geom_smooth(aes(x=MET,y=EX),method = "loess")+labs(x = "Perce
set.seed(12345)
control_parameter = tree.control(nobs = nrow(state),minsize = 8)
fit_tree = tree(formula = EX ~ MET,data = state,control = control_parameter)
leaffit = cv.tree(fit_tree)
plot(leaffit)
#plotting deviance against number of leaves
p = ggplot() +
geom_line(aes(x=leaffit$size, y=leaffit$dev)) +
geom_point(aes(x=leaffit$size, y=leaffit$dev)) +
labs(x="Number of leaves", y="Deviance")
print(p)
# Plotting Deviance vs alpha.
p = ggplot() +
geom_line(aes(x=log(leaffit$k), y=leaffit$dev)) +
geom_point(aes(x=log(leaffit$k), y=leaffit$dev)) +
labs(x="Alpha", y="Deviance")
print(p)

prunedtree = prune.tree(fit_tree,best = leaffit$size[which.min(leaffit$dev)])

plot(prunedtree)
text(prunedtree, pretty=1, cex = 0.8, xpd = TRUE)

fitted_val = predict(prunedtree, newdata=state)

plot(state$MET, state$EX)
points(state$MET, fitted_val, col="red")

# Plotting the histogram of the residuals.
p = ggplot() +
geom_histogram(aes(residuals(prunedtree)), bins = 30)
print(p)

# Plotting the original data and the predictions.
p = ggplot() +
geom_point(aes(x=state$MET, y=state$EX)) +
geom_line(aes(x=state$MET, y=fitted_val), colour='red') +
labs(x="MET", y="EX", title="Prediction of EX given MET by a single tree")
print(p)
```

```
f_np = function(state,index){
  sample = state[index,]
  Ctrl = tree.control(nrow(sample), minsize = 8)
  fit = tree( EX ~ MET, data=sample, control = Ctrl)
  optimal_tree = prune.tree(fit, best= leaffit$size[which.min(leaffit$dev)])
  return(predict(optimal_tree, newdata=state))
}
np_bs = boot(state, statistic = f_np, R=1000)
conf_bound = envelope(np_bs,level=0.95) # For 95% Confidence interval

predictions = predict(prunedtree,state)
plot(np_bs)
fig_data = data.frame(orig = state$EX, x=state$MET, pred=predictions,
                      upper=conf_bound$point[1,], lower=conf_bound$point[2,])

p = ggplot(fig_data, aes(x,predictions,upper,lower)) + geom_point(aes(x, pred)) +
  geom_point(aes(x, orig),colour="blue") +
  geom_line(aes(x,upper),colour="red") +
  geom_line(aes(x,lower),colour="red")+
labs(x='MET',
y='EX',
title='Confidence interval (non-parametric bootstraping)',
color="Type")
p
# 3.4 Paramatric Bootstrap
set.seed(12345)
best_nleaves = leaffit$size[which.min(leaffit$dev)]
original_data = state
reg = tree(EX ~ MET,
data=state,
control=tree.control(nobs=nrow(state), minsize=8))
reg = prune.tree(reg, best=best_nleaves)
original_reg = reg
# Creating a function that is going to sample from our prior.
# Our prior is that the data is distributed as a normal distribution.
rng = function(state, model)
{
# Getting the parameters for the normal distribution.
nobs = nrow(state) # Number of observations.
y_hat = predict(model, newdata=state) # Predictions.
y = state$EX # Real values.
resid = y - y_hat # Residuals.
state$EX = rnorm(nobs, # Normal distribution.
y_hat,
sd(resid))
return(state)
}
# Function that is going to return the predictions, given new
# random generated data. This will be used for getting the
# confidence interval for the predictions.
citree_prediction = function(state)
{
# Training the model.
```

```
reg = tree(EX ~ MET,data=state,control=tree.control(nobs=nrow(state), minsize=8))
reg = prune.tree(reg, best=best_nleaves)
# Predicting over my original data.
y_hat = predict(reg, newdata=original_data)
return(y_hat)
}
# Function that is going to return the predictions, given our new
# random generated data. This will be used for getting the
# prediction interval for our predictions.
ptree_prediction = function(state)
{
# Fitting our tree given our bootstrap sample.
reg = tree(EX ~ MET,data=state,control=tree.control(nobs=nrow(state), minsize=8))
reg = prune.tree(reg, best=best_nleaves)
# Getting E[Y|X] given our bootstrap model
# on the original data.
y_hat = predict(reg, newdata=original_data)
# Getting the residuals from the original model.
y = original_data$EX
resid = y - predict(original_reg, newdata=original_data)
# Sampling from N(E[Y|X], Var(residuals)).
nobs = nrow(original_data)
y_hat = rnorm(nobs,y_hat,sd(resid))
return(y_hat)
}
#Running the bootstrap for the confidence interval.
results = boot(state,statistic=citree_prediction,R=1000,mle=reg,ran.gen=rng,sim="parametric")
ci_results = envelope(results)
# Running the bootstrap for
# the prediction interval.
results = boot(state,statistic=ptree_prediction,R=1000,mle=reg,ran.gen=rng,sim="parametric")
p_results = envelope(results)
# Getting the predictions.
reg = tree(EX ~ MET,data=state,control=tree.control(nobs=nrow(state), minsize=8))
reg = prune.tree(reg, best=best_nleaves)
y_hat = predict(reg, newdata=state)
# Plotting the results.
z = ggplot() +
geom_point(aes(x=state$MET, y=state$EX), fill='white', shape=21) +
geom_line(aes(x=state$MET, y=p_results$point[1, ], color="Prediction Interval")) +
geom_line(aes(x=state$MET, y=p_results$point[2, ], color="Prediction Interval")) +
geom_line(aes(x=state$MET, y=ci_results$point[1, ], color="Confidence Interval")) +
geom_line(aes(x=state$MET, y=ci_results$point[2, ], color="Confidence Interval")) +
geom_line(aes(x=state$MET, y=y_hat, color="E[Y|X]")) +
scale_colour_manual(values=c("#604dc5", "#020c0b", "#f83d69")) +
labs(x='MET', y='EX',
title='Prediction and Confidence interval (parametric bootstraping)')
print(z)
knitr::include_graphics("bootstraping.png")
NIRSpectra <- read.csv2("NIRSpectra.csv", header = TRUE)
data1 <- NIRSpectra
data1$Viscosity = c()
res=prcomp(data1)
```

```r
lambda=res$sdev^2
lambda
sprintf("%2.3f", lambda/sum(lambda)*100)
screeplot(res,xlab = "Principal Component")
U=res$rotation
#plot of scores
plot(res$x[,1],res$x[,2],ylim=c(-5,15))

# Getting the variance explained by each eigenvector.
eigen_values = res$sdev^2
pcvariance = eigen_values * 100 / sum(eigen_values)
# Components that explain at least 99% of the variation.
toppc = res$rotation[, c(1, 2)]
topev = eigen_values[c(1, 2)]
kable(topev, caption="Eigenvalues for the top axis of the new basis")
# Getting the outliers.
mask = res$x[, 1] > 1
outliers = res$x[mask, ]
# Embeded space of the coordinates
p = ggplot() +
geom_point(aes(x=res$x[, 1], y=res$x[, 2])) +
geom_point(aes(x=outliers[, 1], y=outliers[, 2]), color='red') +
labs(title="Embedded space of the features",
x="First principal component",
y="Second principal component")
print(p)
#U=res$rotation
#plot(U[,1], main="Traceplot, PC1")
#plot(U[,2],main="Traceplot, PC2")

# TASK 4.2
# Traceplots.
# First component.
p = ggplot() +
geom_point(aes(x=1:nrow(toppc), y=toppc[, 1])) +
labs(title="Trace plot of the first principal component",
x="Index of each feature",
y="Feature value in the principal component")
print(p)
# Second component.
p = ggplot() +
geom_point(aes(x=1:nrow(toppc), y=toppc[, 2])) +
labs(title="Trace plot of the second principal component",
x="Index of each feature",
y="Feature value in the principal component")
print(p)
set.seed(12345)
# Calculating W'
ica = fastICA(data1, 2)
W_prime = ica$K %*% ica$W # Latent variable, same as the eigenvalues for PCA.
# This operation maps the basis constructed from ICA
# and maps it into the eigenspace. This is mapping the PCA basis to the
# ica basis.
```

```
# Trace plots.
# First component.
p = ggplot() +
geom_point(aes(x=1:nrow(W_prime), y=W_prime[, 1])) +
labs(title="Trace plot of the 1st independent component",
x="Index of each feature",
y="Feature value in the independent component")
print(p)
# Second component.
p = ggplot() +
geom_point(aes(x=1:nrow(W_prime), y=W_prime[, 2])) +
labs(title="Trace plot of the 2nd independent component",
x="Index of each feature",
y="Feature value in the independent component")
print(p)
# Getting the outliers.
mask = ica$S[, 1] < -8
outliers = ica$S[mask, ]
# Embeded space of the coordinates
p = ggplot() +
geom_point(aes(x=ica$S[, 1], y=ica$S[, 2])) +
geom_point(aes(x=outliers[, 1], y=outliers[, 2]), color='red') +
labs(title="1st 2 vectors of the source space",
x="1st independent component",
y="2nd independent component")
print(p)
```

```
loglikelihood1 = function(data, theta){ #Data == x

  #Declare the vector to return the values in.
  return_vec = 1:length(theta)

  #For every value of theta, calculate the probability for each data
  #For every iteration: save the log product of all probabilities
  for(i in 1 : length(theta)){
    probability_vec = theta[i]*exp(-theta[i]*data)
    return_vec[i] = log(prod(probability_vec))
  }

  return(return_vec)
}


loglikelihood2 = function(data, theta, lambda){ #Data == x

  #Declare the vector to return the values in.
  return_vec = 1:length(theta)

  #For every value of theta, calculate the probability for each data
  #For every iteration: save the log product of all probabilities times prior p
  for(i in 1 : length(theta)){
    probabillity_vec = theta[i]*exp(-theta[i]*data)
    prior_vec = lambda*exp(-theta[i]*lambda)
    return_vec[i] = log(prod(probabillity_vec)*prior_vec)
  }

  return(return_vec)
}

theta = seq(0, 5, by = 0.001)

#-------------------------------------------------------------------Assignment 2.2
result1.llhood = loglikelihood1(machines$Length, theta)
result1 = data.frame(theta = theta, likelihod = result1.llhood)

plot(result1, type="l", col="red")
print(max(result1.llhood))
print(theta[as.numeric(which.max(result1.llhood))])

#-------------------------------------------------------------------Assignment 2.3
result2.llhood = loglikelihood1(machines$Length[1:6], theta)
result2 = data.frame(theta= theta, likelihood = result2.llhood)
```

```
lines(result2, type="l", col="blue")
print(max(result2.llhood))
print(theta[as.numeric(which.max(result2.llhood))])


#-------------------------------------------------------------------Assignment 2.4
result3.llhood = loglikelihood2(machines$Length, theta, 10)
result3 = data.frame(theta = theta, likelihood = result3.llhood)

lines(result3, type="l", col="green")
print(max(result3.llhood))
print(theta[as.numeric(which.max(result3.llhood))])

#-------------------------------------------------------------------Assignment 2.5

set.seed(12345)
random_nr = rexp(50, rate=1.126)
hist(random_nr, breaks=50, col="red")
hist(machines$Length, breaks=50, col="blue")
```

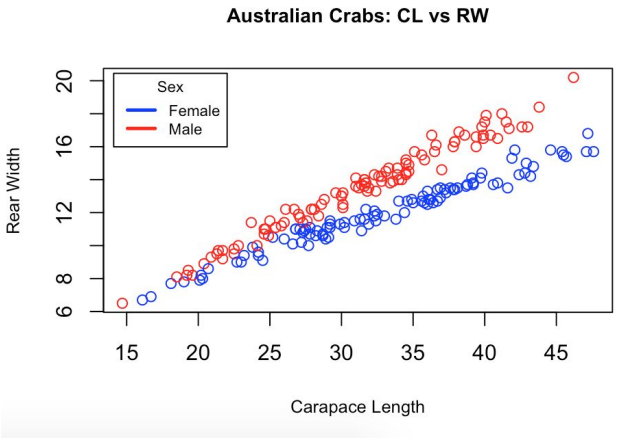
# C Code for assignment 4

```
setwd("C:/Users/elete/Desktop/Uni/TDDE01/Labs/Lab 1/Lab 1")
require(readxl)
data = read_excel("tecator.xlsx")

# 4.1

plot(x=data$Moisture, y=data$Protein, type = "p")

# 4.2

#Function to calculate MSE
mse = function(y, yhat)
{

  return(mean((y - yhat)^2))
}

mean_squared_error = function(y, y_hat) {
  squared_error = (y - y_hat)^2
  sum_squared_error = sum(squared_error)
  n = length(squared_error)
  return(sum_squared_error/n)
}
```

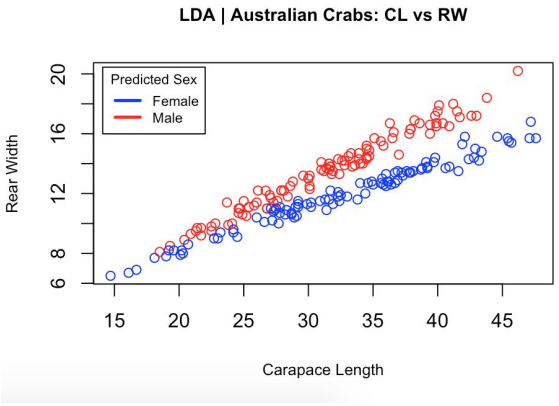
# Laboration 2 - danjo390 danro880 toblo956

# Assignment 1

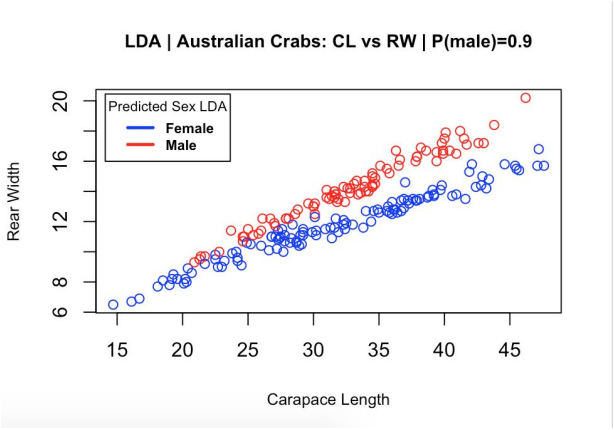
1. Yes due to the appearance of the graph and the distinct difference between the male and female it is fairly easy to classify with linear discriminant analysis.



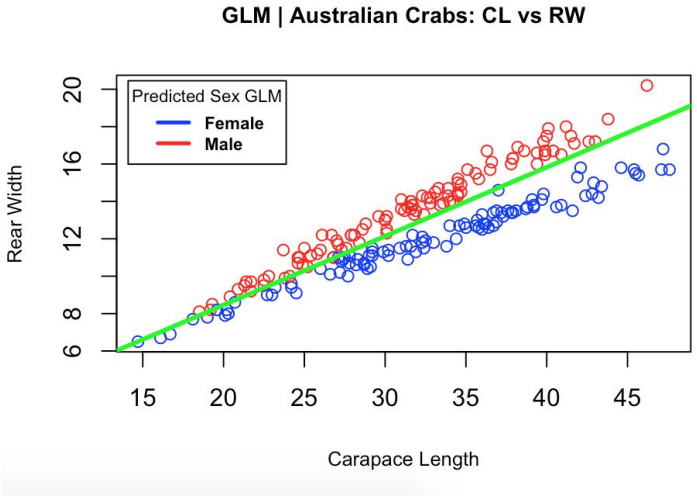

2. You can see that there is a utmost difference between the plot for the predicted Sex and the actual plot in 1). This corresponds to the misclassification error of 0.035, meaning the quality of fit is extremely accurate.




3. When changing the priors in 2) to $p(Male) = 0.9, p(Female) = 0.1$ instead we get the misclassification error to 0.08, which is larger than the previous but still considerably small. More aggressive classification as male since the prior now indicates that 90 % are males.




4. When using logistic regression we got this graph, where also the decision boundary is drawn. The misclassification error is 0.035 - the same as in 2) - but some points are classified differently when looking closer. The equation for the decision boundary was y=0.3685758 x + 1.08379



# Appendix

## Assignment 1

```
setwd("~/Lab2")
library(readxl)
library(MASS)
crab_data <- read.csv("~/Desktop/australian-crabs.csv")
set.seed(12345)
#2.1.1
cl <- crab_data$CL
rw <- crab_data$RW
sex <- crab_data$sex

plot(cl, rw, main='Australian Crabs: CL vs RW', cex.main=0.9, cex.lab=0.8, xlab='Carapace
Length', ylab='Rear Width', col=c('red','blue')[sex])
legend("topleft", title='Sex',  cex=0.7, text.font=1, inset=.02, c("Female", "Male"),
lwd=c(2.5,2.5),col=c('blue','red'))

#2.1.2
lda_analysis <- lda(sex ~ cl + rw, data=crab_data)
lda_analysis.p <- predict(lda_analysis,sex)
ct <- table(sex,lda_analysis.p$class)
sum(diag(prop.table(ct)))
# Missclass LDA = 1 - 0.965 = 0.035
plot(cl, rw, main='LDA | Australian Crabs: CL vs RW', cex.main=0.9, cex.lab=0.8,
xlab='Carapace Length', ylab='Rear Width', col=c('red','blue')[lda_analysis.p$class])
legend("topleft", title='Predicted Sex', cex=0.7,text.font=1,inset=.02,,c("Female", "Male"),
lwd=c(2.5,2.5),col=c('blue','red'))

#2.1.3
lda_analysis_2 <- lda(sex ~ cl + rw, prior= c(0.1,0.9),data=crab_data)
lda_analysis_2.p <- predict(lda_analysis_2,sex)
ct_2 <- table(sex,lda_analysis_2.p$class)
sum(diag(prop.table(ct_2)))
# Missclass LDA 0.9= 1 - 0.92 = 0.08
plot(cl, rw, main='LDA | Australian Crabs: CL vs RW | P(male)=0.9', cex.main=0.9,
cex.lab=0.8, xlab='Carapace Length', ylab='Rear Width',
col=c('red','blue')[lda_analysis_2.p$class])
legend("topleft", title='Predicted Sex LDA', cex=0.7, text.font=2,inset=.02,c("Female",
"Male"), lwd=c(2.5,2.5),col=c('blue','red'))

#2.1.4
glm_analysis <- glm(sex~ cl + rw, data=crab_data, family=binomial())
```

```
glm_analysis.p <- (predict(glm_analysis)>0)
ct_3 <- table(prediction=glm_analysis.p, data=sex)
sum(diag(prop.table(ct_3)))
# Missclass GLM = 1 - 0.965 = 0.035
plot(cl, rw, main='GLM | Australian Crabs: CL vs RW', cex.main=0.9, cex.lab=0.8,
xlab='Carapace Length', ylab='Rear Width', col=c('red','blue')[glm_analysis.p+1])
legend("topleft", title='Predicted Sex GLM', cex=0.7, text.font=2,inset=.02,c("Female",
"Male"), lwd=c(2.5,2.5),col=c('blue','red'))
coeffs <- glm_analysis$coefficients
par(new=TRUE)
abline(-coeffs[1]/coeffs[3], -coeffs[2]/coeffs[3], col="green", lwd = 3)
```

# MLLab2Block2

*Omkar Bhutra*

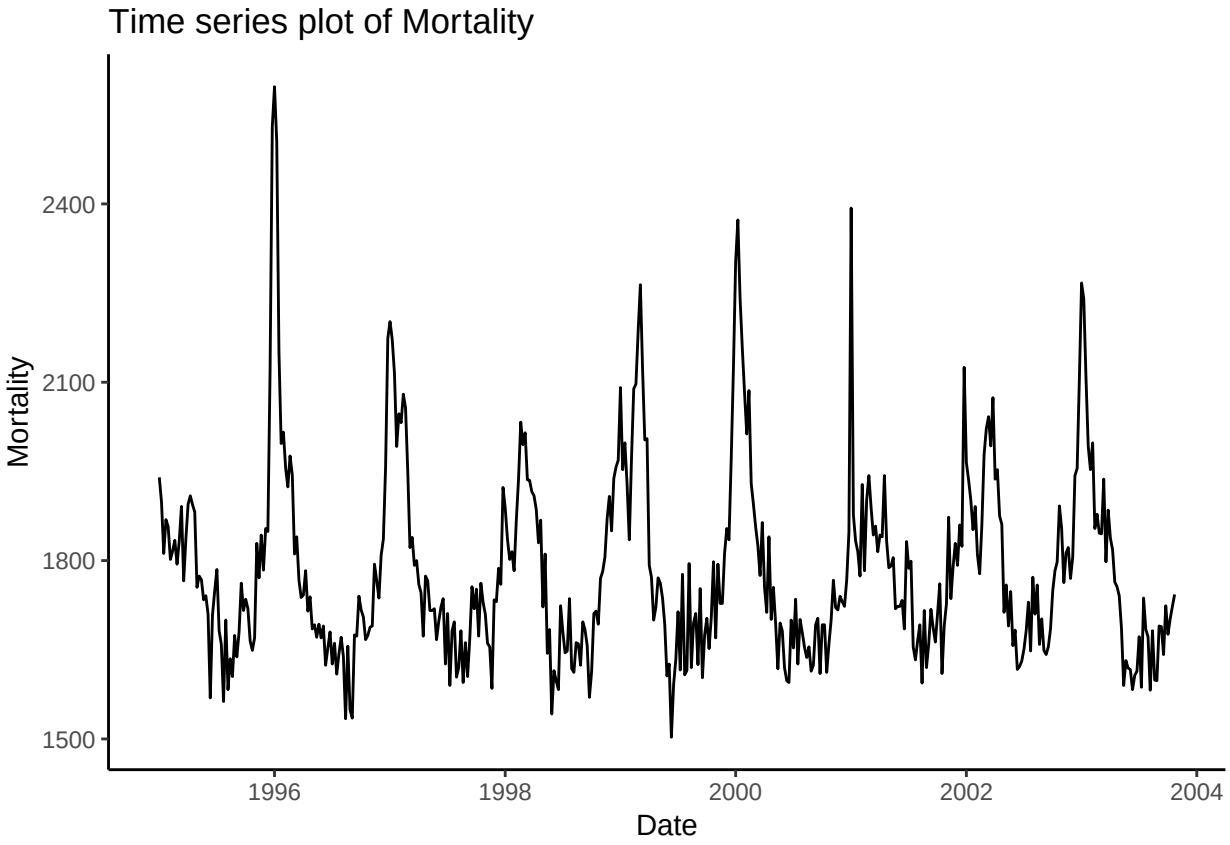
*11 December 2018*

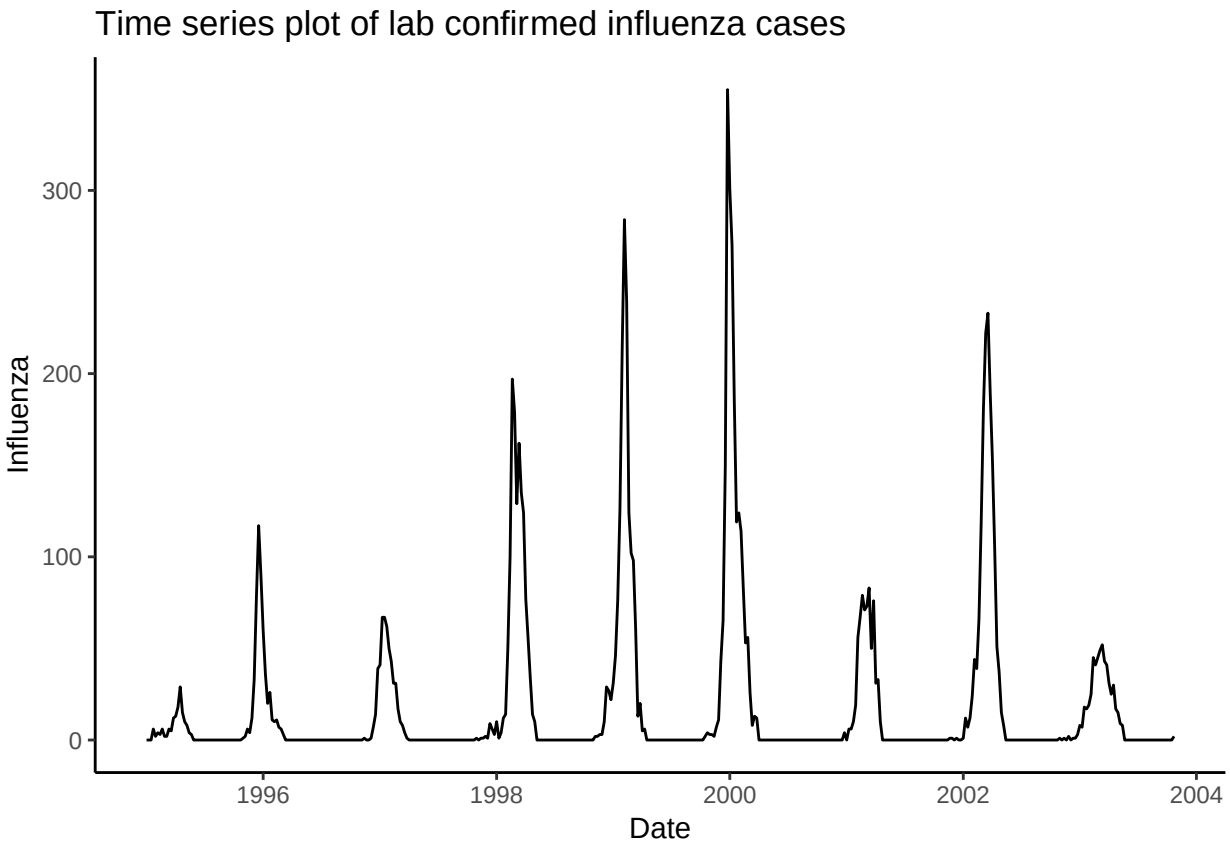
## Assignment 1.

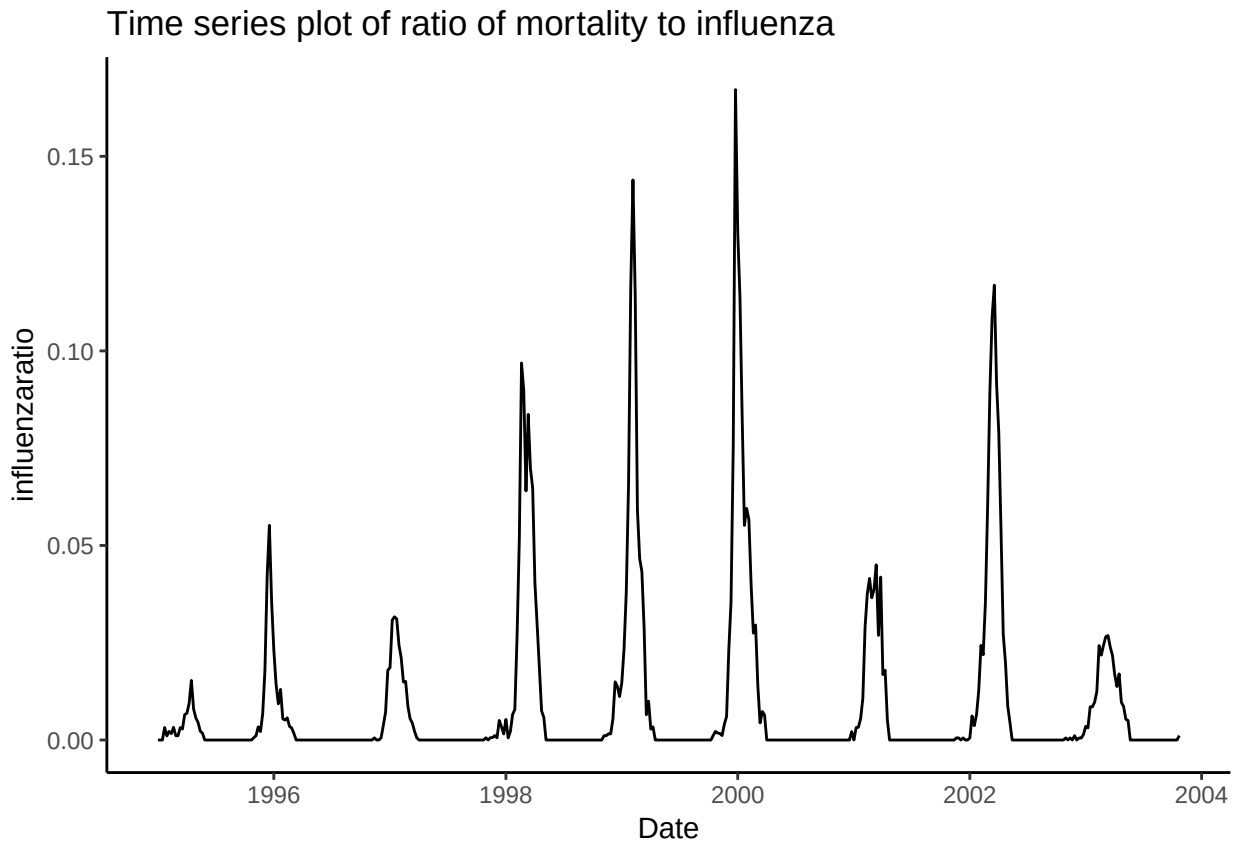
### Using GAM and GLM to examine the mortality rates

#### Q1

From the plots we can see that, Mortality and Influenza peaking during the same time of each year which is the 1st quarter (Jan to March) with Influenza peaking sometimes in December of the previous as well. Although, The highest mortality is in January of 1996 with 2597 deaths and the highest laboratory-confirmed cases of influenza is found in December of 1999 with 355 cases. The third plot shows the percentage of influenza cases that directly attributed to death and it confirms that the two variables are highly correlated.




Time series plot of lab confirmed influenza cases
Influenza
300
200
100
0
1996
1998
2000
2002
2004
Date




**Q2**

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Mortality ~ Year + s(Week, k = length(unique(Influenza$Week)))
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -680.598   3367.760  -0.202    0.840
## Year           1.233      1.685   0.732    0.465
##
## Approximate significance of smooth terms:
##           edf Ref.df     F p-value
## s(Week) 14.32  17.87 53.86  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Rank: 52/53
## R-sq.(adj) =  0.677   Deviance explained = 68.8%
## GCV = 8708.6  Scale est. = 8398.9    n = 459
```

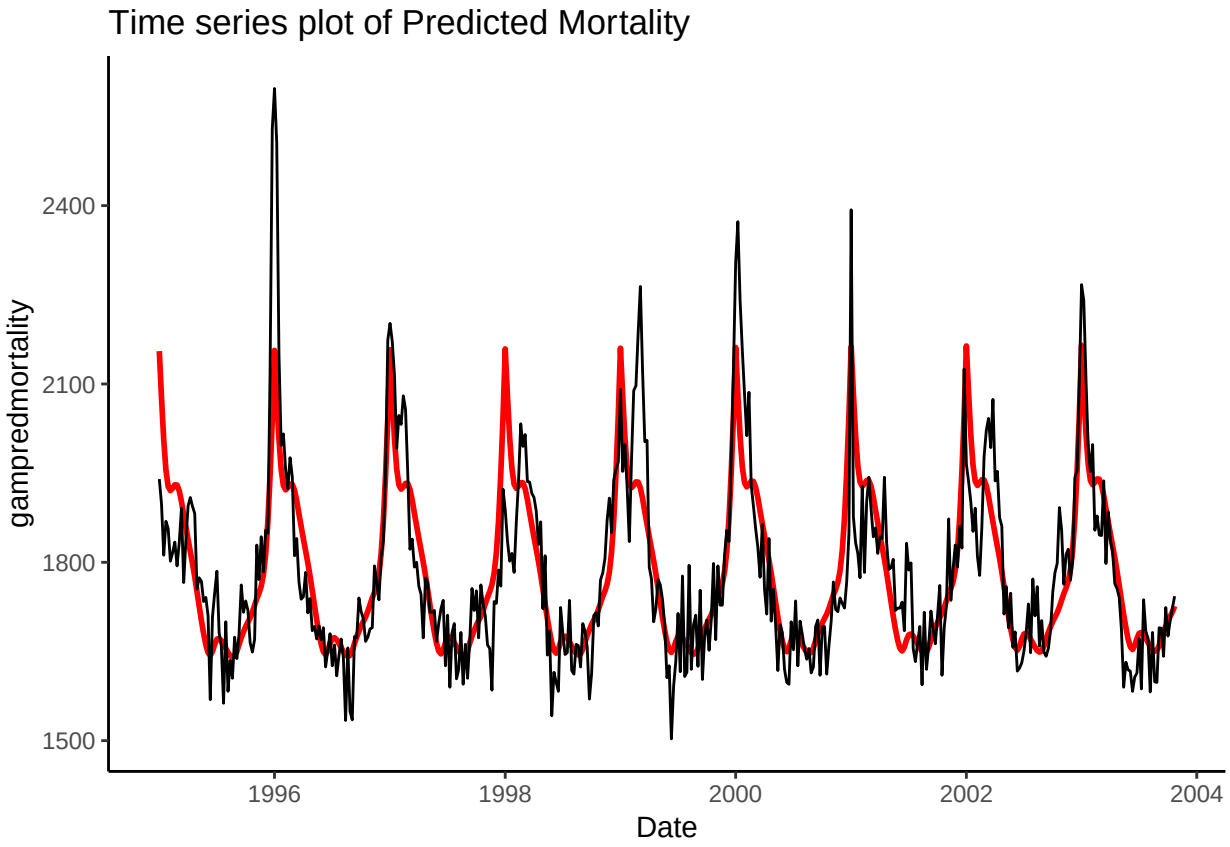
Underlying probablistic equation of the model :

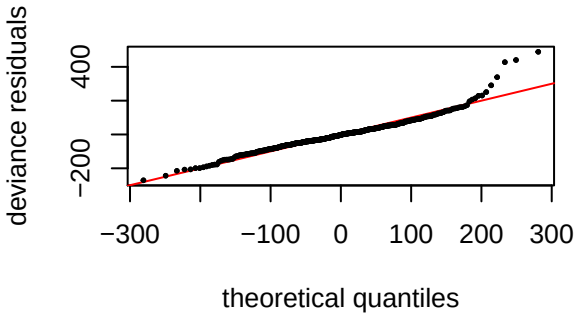
$$Mortality = N(\mu, \sigma^2)$$

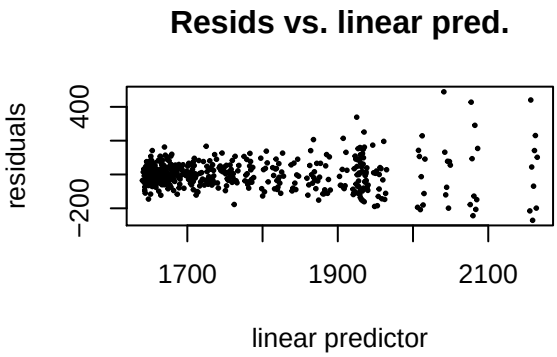
$$g(\mu) = Intercept + Beta_{year} * Year + s(Week)$$

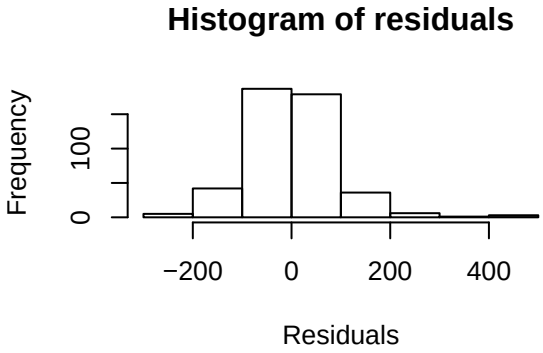
Where g is the link function, in this case it is a normal distribution

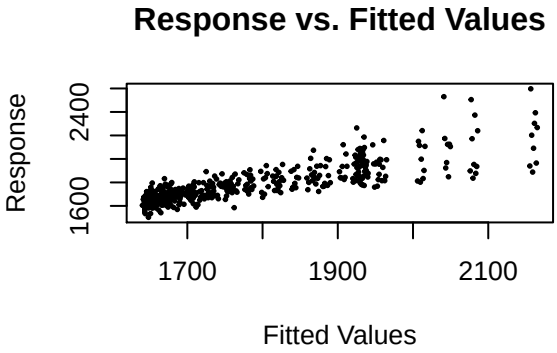
**Q3**



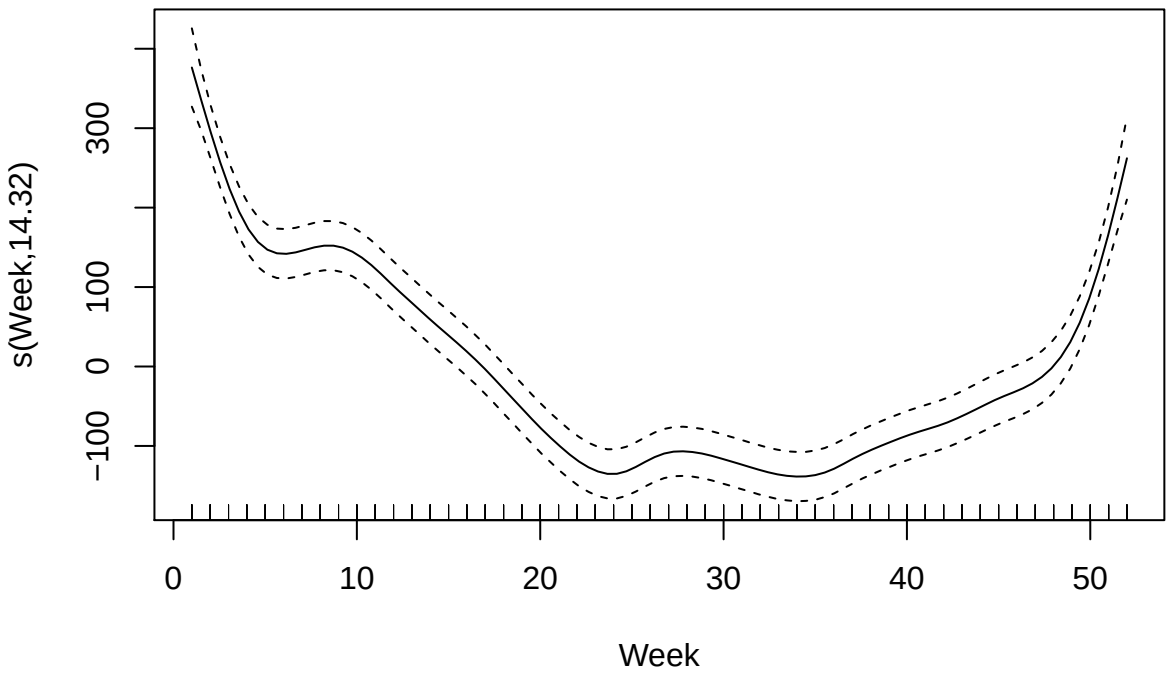



```
##
## Method: GCV   Optimizer: magic
## Smoothing parameter selection converged after 9 iterations by steepest
## descent step failure.
## The RMS GCV score gradient at convergence was 0.00106719 .
## The Hessian was positive definite.
## Model rank =  52 / 53
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##           k'  edf k-index p-value
## s(Week) 51.0 14.3    1.09    0.98
```



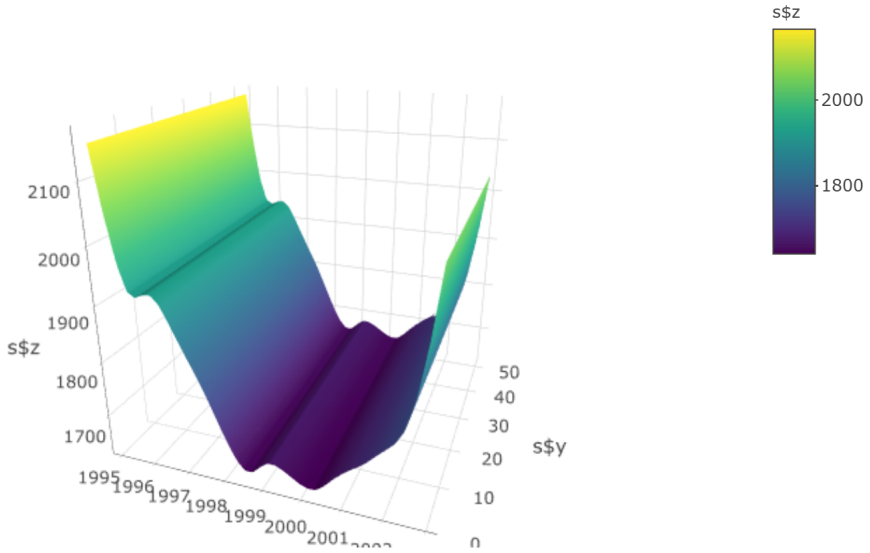

The predicted mortality fits quite well with the time (x axis) i.e the peaks and troughs match with the actual mortality value but it is a repeating function that does not capture the the mortality values in the model and hence not a very good model to predict. It is observed that the linear component of year is not significant but the spline component of Week is a significant term with a very low p value. From the plot of the spline component it is seen that mortality peaks in the winter of each year and are the least in the summer of each year.

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Mortality ~ Year + s(Week, k = length(unique(Influenza$Week)))
##
## Estimated degrees of freedom:
## 14.3  total = 16.32
##
## GCV score: 8708.581     rank: 52/53
```

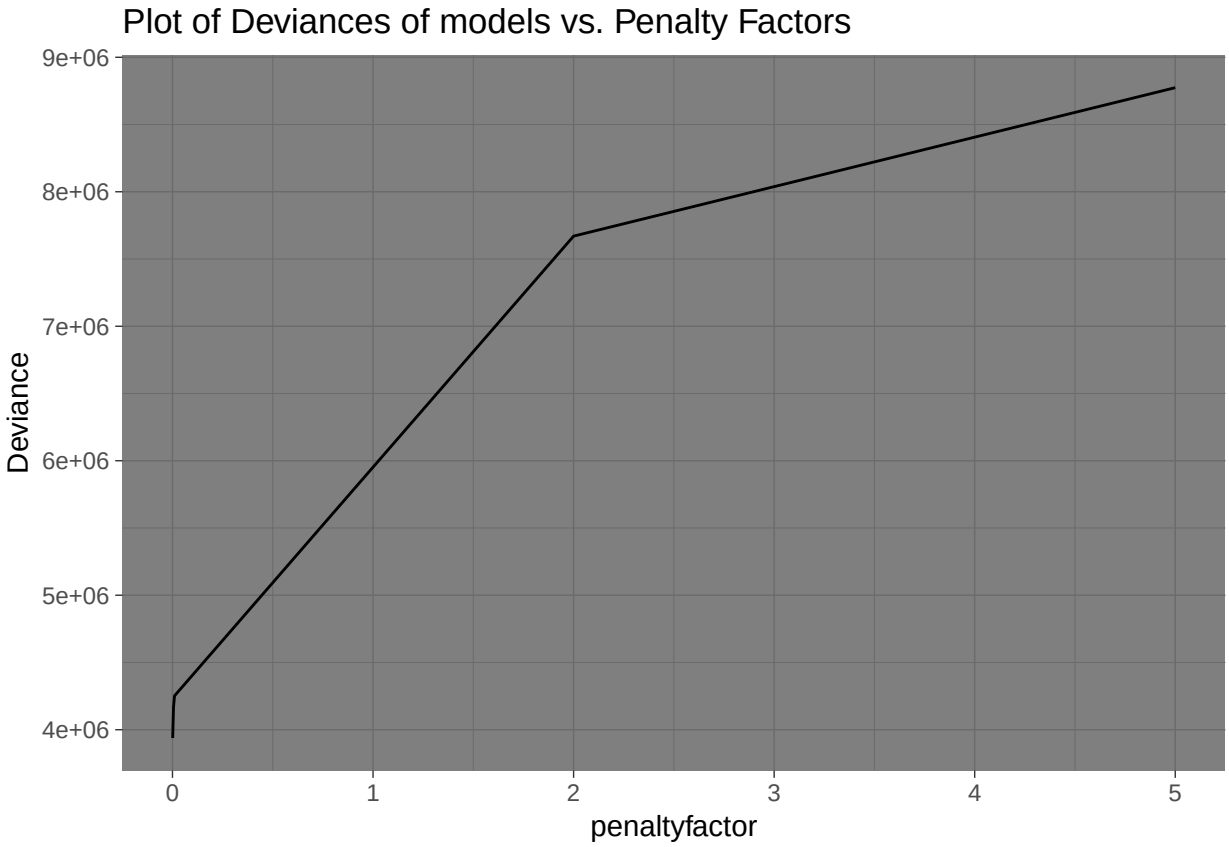
```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Mortality ~ Year + s(Week, k = length(unique(Influenza$Week)))
##
```

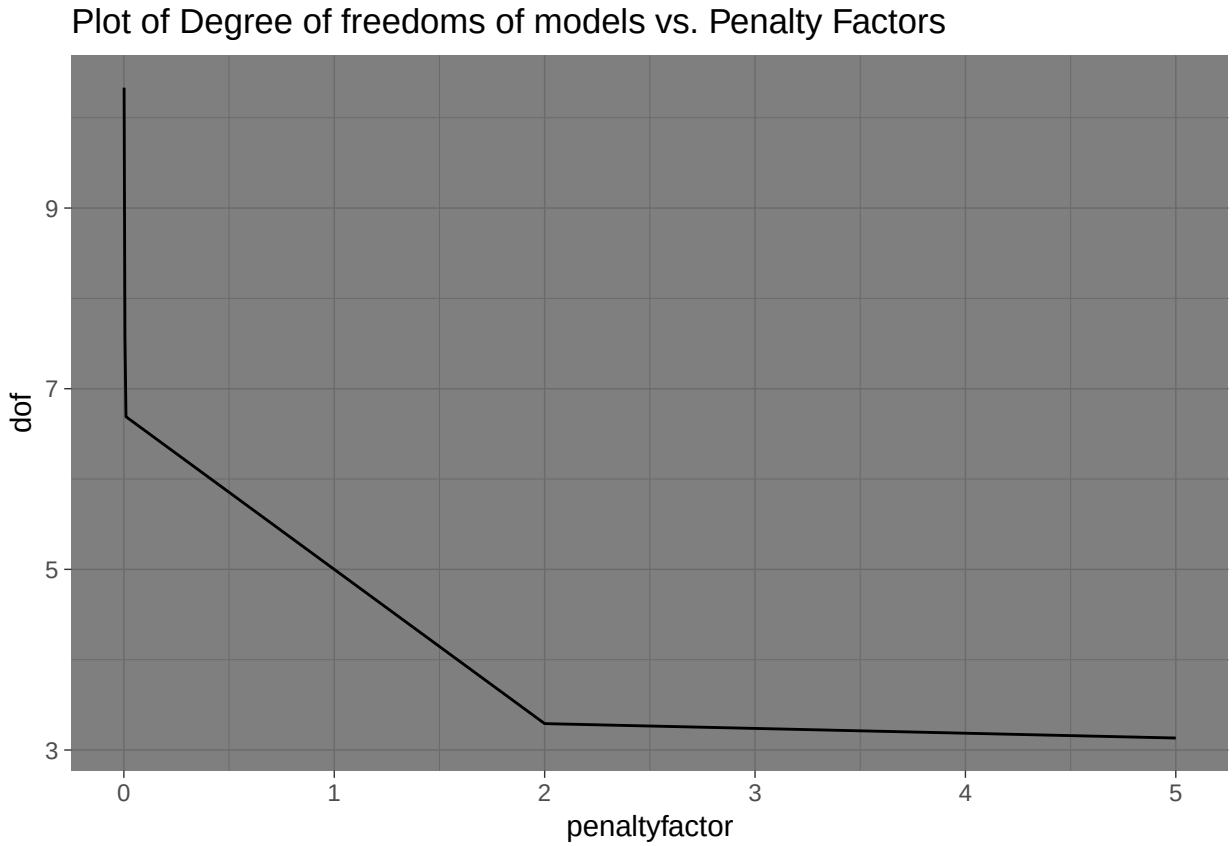
```
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -680.598   3367.760  -0.202    0.840
## Year           1.233      1.685   0.732    0.465
##
## Approximate significance of smooth terms:
##           edf Ref.df     F p-value
## s(Week) 14.32  17.87 53.86  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Rank: 52/53
## R-sq.(adj) =  0.677   Deviance explained = 68.8%
## GCV = 8708.6  Scale est. = 8398.9    n = 459
```

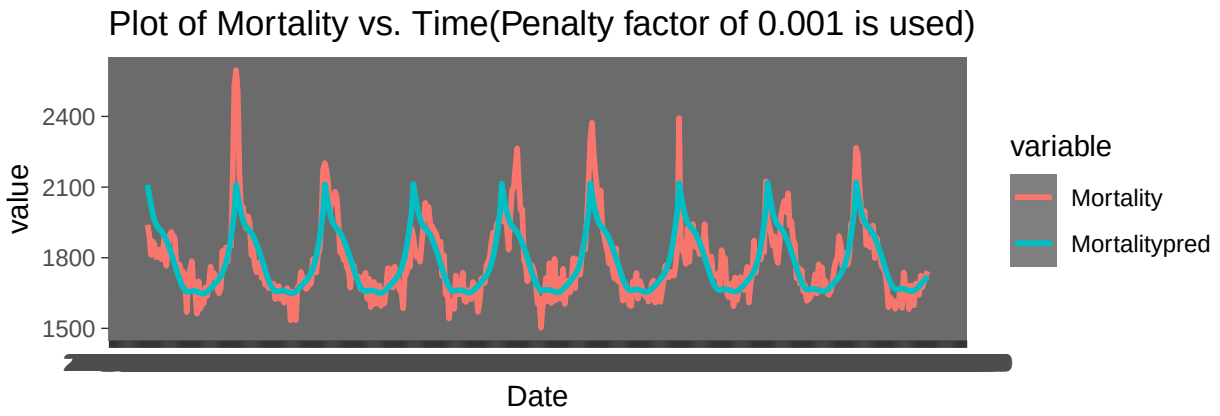
```
##      s(Week)
## 0.0001131932
```

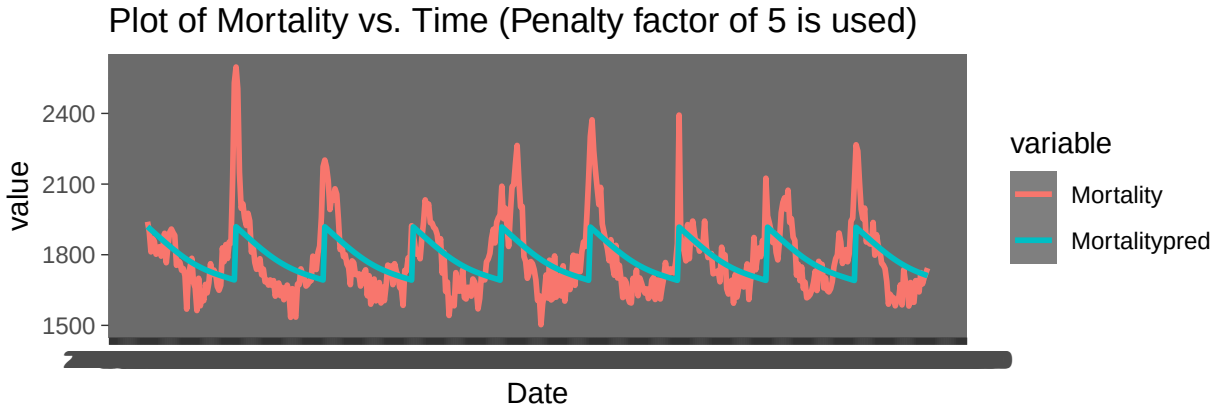



**Q4**


Plot of Deviances of models vs. Penalty Factors
Deviance
9e+06
8e+06
7e+06
6e+06
5e+06
4e+06
0
1
2
3
4
5
penaltyfactor

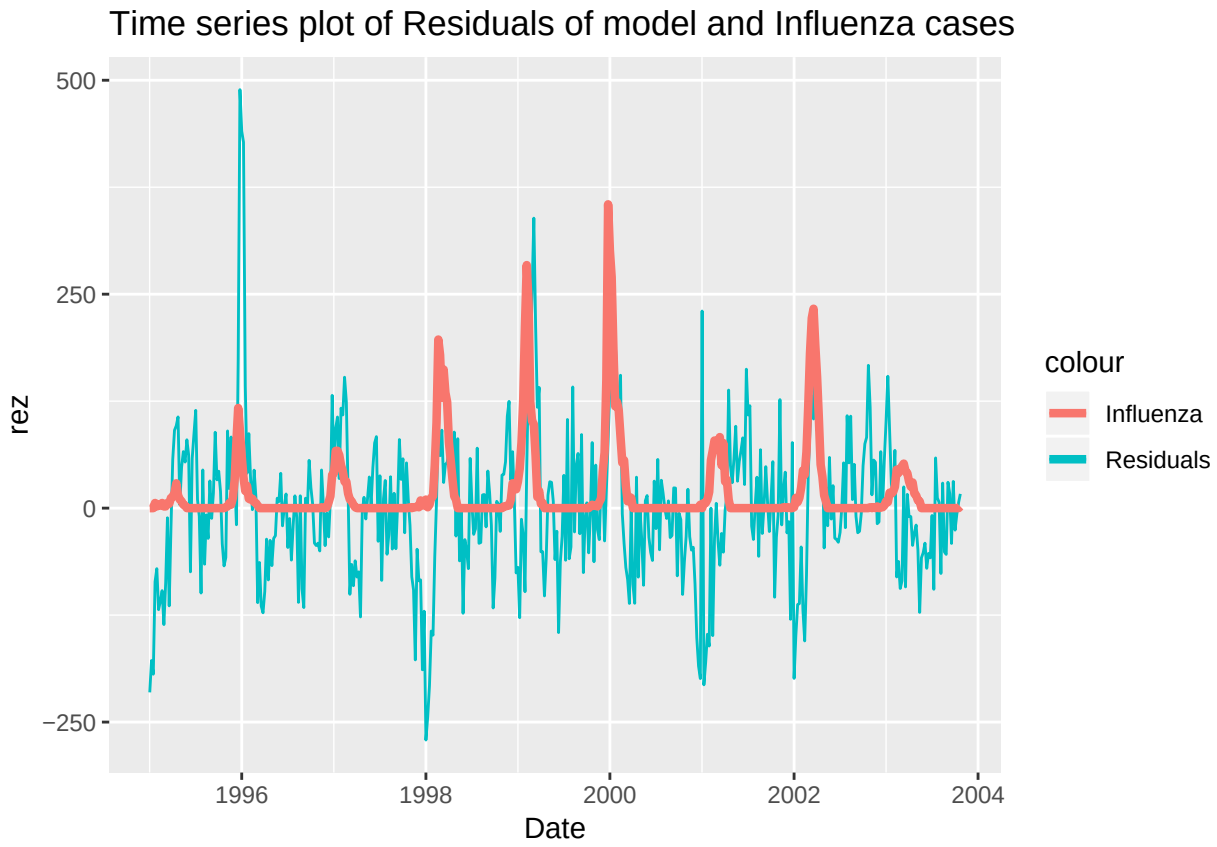

Plot of Degree of freedoms of models vs. Penalty Factors
dof
9
7
5
3
0
1
2
3
4
5
penaltyfactor



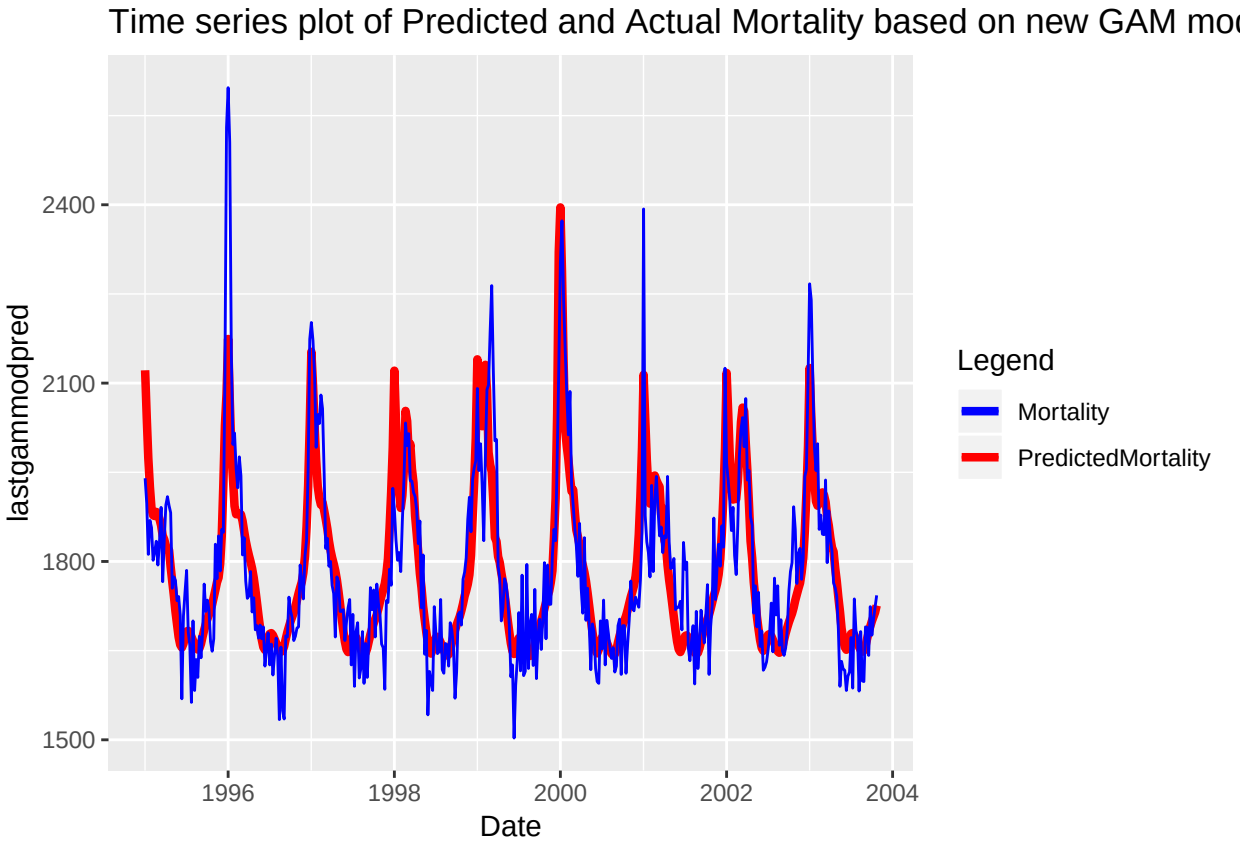

A directly propotional relationship is seen between penalty factor and deviance. Higher the penalty factor , higher is the deviance. With a higher penalty factor comes less complexity and more bias in the model. An inverse relationship is seen between penalty factor and degree's of freedom. Lower the penalty factor, Higher is the degree of freedom. yes, this is confirmed from our results.

**Q5**




the temporal pattern in the residuals can be linked to the periodic outbreak of influenza to an extent. The Three largest outbreaks of influenza also have residuals peaking in the positive direction while it is seen that the residuals have negative troughs right before the influenza peaks that is for the last quarter of the year.

**Q6**



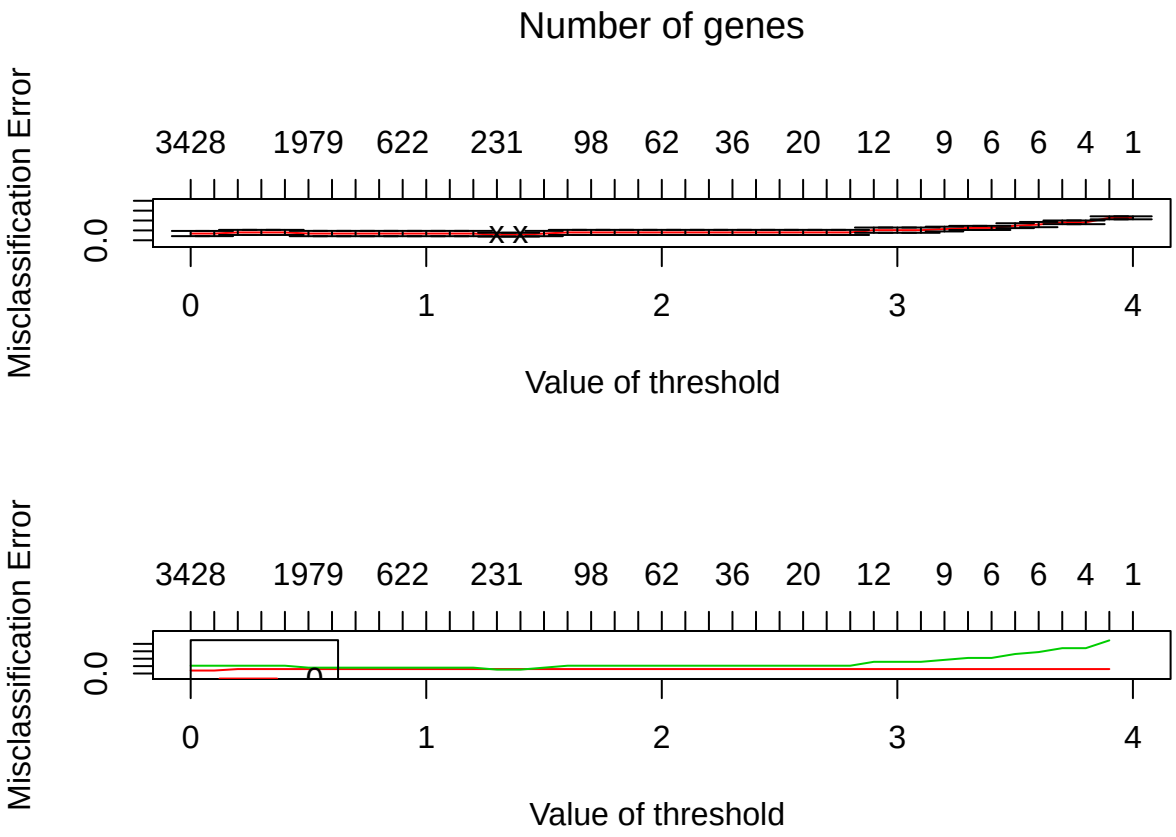

Yes, this Generalised Additive Model is better than the previous models as the predicted fit is good not only in the x axis but also matches the actual value peaks and troughs. It can be concluded that Mortality can be described well with non linear spline functions of Year and Week along with the linear function of Influenza. Hence, Outbreaks of Influenza in the winters have a direct effect on Mortality.

## Assignment 2.

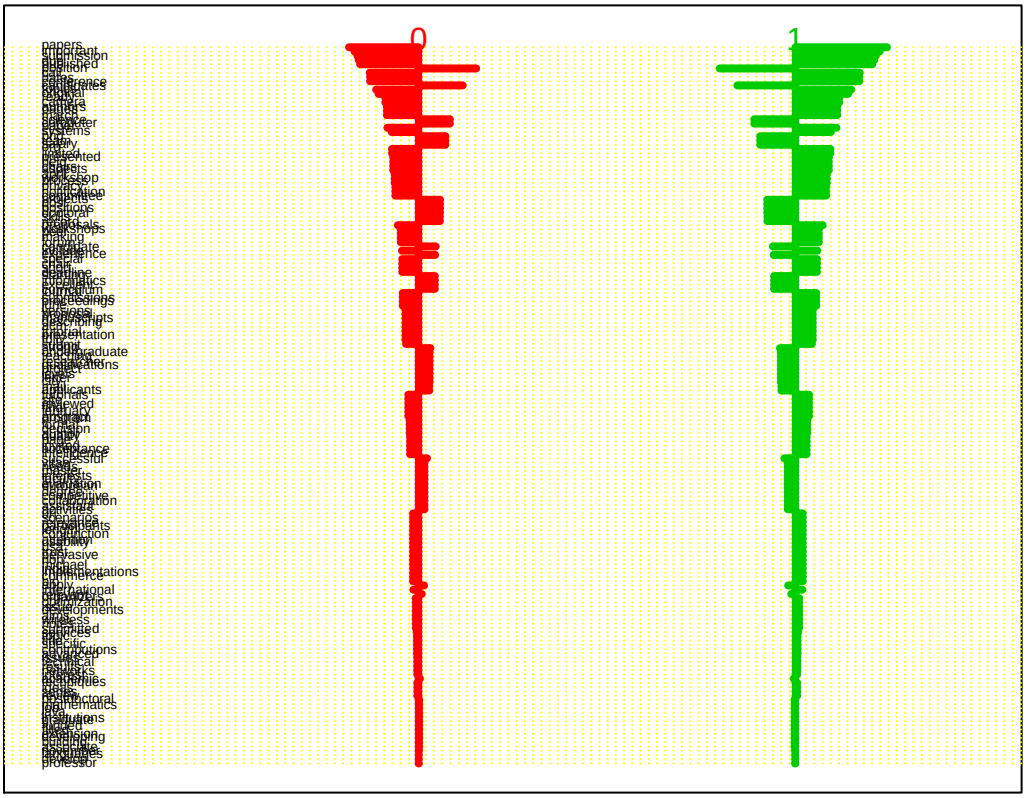
### High-dimensional methods

### Q1

```
## 12345678910111213141516171819202122232425262728293031323334353637383940 41

## 12Fold 1 :12345678910111213141516171819202122232425262728293031323334353637383940 41
## Fold 2 :12345678910111213141516171819202122232425262728293031323334353637383940 41
## Fold 3 :12345678910111213141516171819202122232425262728293031323334353637383940 41
## Fold 4 :12345678910111213141516171819202122232425262728293031323334353637383940 41
## Fold 5 :12345678910111213141516171819202122232425262728293031323334353637383940 41
## Fold 6 :12345678910111213141516171819202122232425262728293031323334353637383940 41
## Fold 7 :12345678910111213141516171819202122232425262728293031323334353637383940 41
## Fold 8 :12345678910111213141516171819202122232425262728293031323334353637383940 41
## Fold 9 :12345678910111213141516171819202122232425262728293031323334353637383940 41
## Fold 10 :12345678910111213141516171819202122232425262728293031323334353637383940 41

## Call:
```


Number of genes
Misclassification Error
3428 1979 622 231 98 62 36 20 12 9 6 6 4 1
0.0
0 1 2 3 4
Value of threshold


Misclassification Error
3428 1979 622 231 98 62 36 20 12 9 6 6 4 1
0.0
0 1 2 3 4
Value of threshold




```
##         id   0-score 1-score
##    [1,] 3036 -0.369   0.4856
##    [2,] 2049 -0.3396 0.4468
##    [3,] 4060 -0.3244 0.4269
##    [4,] 1262 -0.3178 0.4181
##    [5,] 3364 -0.31   0.4079
##    [6,] 3187 0.3056  -0.4022
##    [7,] 596  -0.2593 0.3412
##    [8,] 869  -0.2574 0.3387
##    [9,] 1045 -0.2574 0.3387
##   [10,] 607  0.2344  -0.3085
##   [11,] 4282 -0.2252 0.2963
##   [12,] 2990 -0.2123 0.2793
##   [13,] 599  -0.1765 0.2322
##   [14,] 3433 -0.1765 0.2322
##   [15,] 389  -0.1684 0.2216
##   [16,] 2588 -0.1684 0.2216
##   [17,] 3022 -0.1684 0.2216
##   [18,] 850  0.1661  -0.2186
##   [19,] 3725 0.1661  -0.2186
##   [20,] 3035 -0.1654 0.2176
##   [21,] 4129 -0.1427 0.1878
##   [22,] 3125 0.1427  -0.1878
##   [23,] 4177 0.1424  -0.1874
##   [24,] 3671 0.1424  -0.1874
##   [25,] 2974 -0.141  0.1856
```

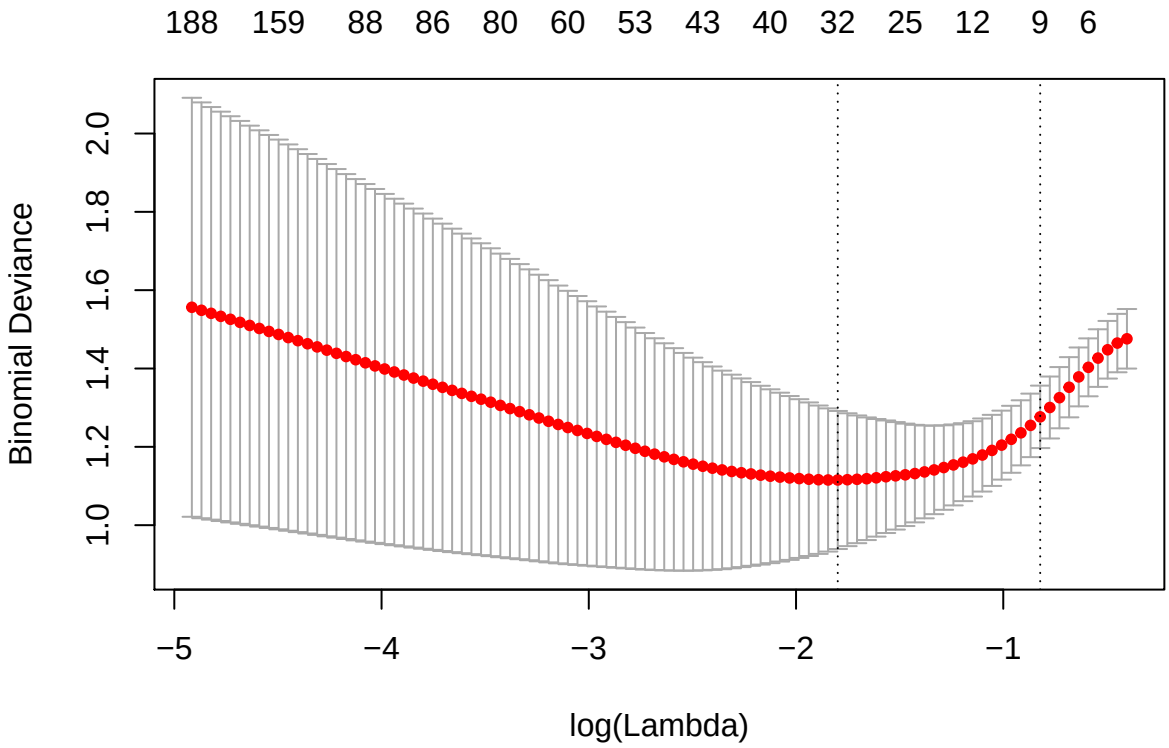
```
## [170,] 3295 0.0017  -0.0022
```

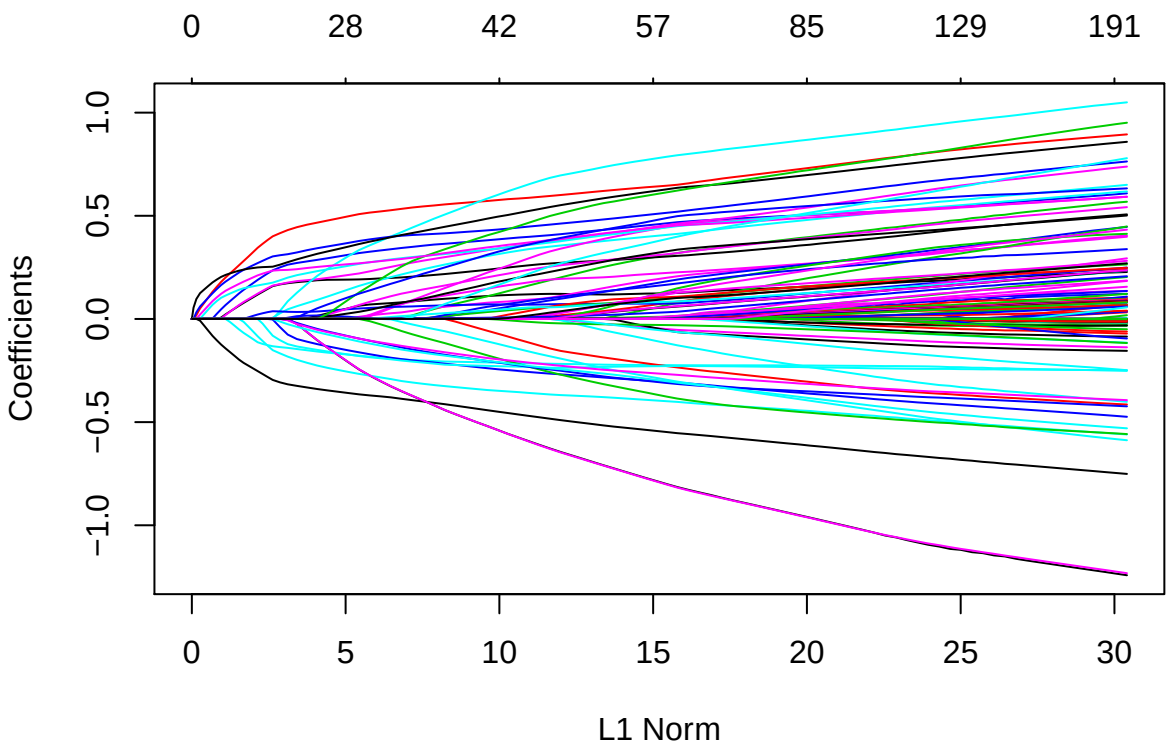
Table 1: Top 10 Important features by NSC

| x |
|---|
| acceptance |
| X59â |
| adhere |
| X1st |
| acquiring |
| accessibility |
| agenda |
| aicit |
| X5102011 |
| agents |

From the plot generated of threshold vs misclassification error. It is observed that when the threshold value is 1.4, the misclassification error is at its lowest. 170 features were selected by this model and top 10 features are listed below. The misclassification error rate is 10%. The confusion matrix reveals that 'everything else' is classified with 10/10 times while 'announces of conferences' is classified 8/10 times.

**Q2a**






```
##       elasticpredict
## ytest2  0  1
##      0 10  0
##      1  2  8
```

```
## [1] "The misclassificaiton rate is 0.1"
```

The Elastic net model has a misclassification error rate of 10%. This model selects the least number of features i.e 39 features.

**Q2b**

```
##  Setting default kernel parameters
```

```
##            Predicted svm
## Actual Test  0  1
##           0 10  0
##           1  1  9
```

```
## [1] "The misclassificaiton rate is 0.05"
```

```
## 25 9.090038e-05     3458  FALSE        record
## 26 9.090038e-05     3891  FALSE        skills
## 27 1.529174e-04     1891  FALSE          held
## 28 1.757570e-04     4177  FALSE          team
## 29 2.007353e-04     3022  FALSE         pages
## 30 2.007353e-04     4628  FALSE      workshop
## 31 2.117020e-04      810  FALSE     committee
## 32 2.117020e-04     3285  FALSE   proceedings
## 33 2.166414e-04      272  FALSE         apply
## 34 2.246309e-04     4039  FALSE        strong
## 35 2.295684e-04     2175  FALSE international
## 36 3.762328e-04     1088  FALSE        degree
## 37 3.762328e-04     1477  FALSE     excellent
## 38 3.762328e-04     3191  FALSE          post
## 39 3.765147e-04     3243  FALSE     presented
```

39 features correspond to the rejecting the null hypothesis, according to the BH rejection threshold. These contain variable names such as 'notification','workshop','conference','candidates','published','topics' to name a few of the 39 features. These reject that the null hypothesis that states that these features have no effect in the classification of into conference and non-conference.

From the first table , it is observed that 281 features have significant p values. Features such as 'committee', 'conference','process','optimization','arrangements' make sense in the usage.

### Apendix

```
knitr::opts_chunk$set(echo = TRUE)
library(dplyr)
library(plotly)
library(ggplot2)
library(xlsx)
library(readxl)
library(tidyr)
library(lubridate)
library(stringr)
library(mgcv)
library(gridExtra)
library(akima)
library(reshape)
library(pamr)
library(glmnet)
library(pROC)
library(kernlab)
library(e1071)
Influenza = read.xlsx("Influenza.xlsx",sheetName = "Raw data",header = TRUE)
Influenza$Date=date_decimal(Influenza$Time)
Influenza$influenzaratio<-((Influenza$Influenza)/(Influenza$Mortality))
p1<-ggplot(Influenza,aes(Date,Mortality))+geom_line(color="black")+scale_fill_brewer()+theme_classic()+g
p1

p2<-ggplot(Influenza,aes(Date,Influenza))+geom_line(color="black")+scale_fill_brewer()+theme_classic()+g
p2
```

```
p3<-ggplot(Influenza,aes(Date,influenzaratio))+geom_line(color="black")+scale_fill_brewer()+theme_classi
p3

gammer<-mgcv::gam(data=Influenza, Mortality ~ Year + s(Week,k=length(unique(Influenza$Week))), method="
summary(gammer)

Influenza$gampredmortality<-mgcv::predict.gam(gammer,newdata = Influenza,type = "link")


p4<-ggplot(Influenza)+geom_line(aes(x=Date,y=gampredmortality),color="red",size=1)+geom_line(aes(x=Date

p4

gam.check(gammer,pch=19,cex=.3)
plot(gammer)
gammer1<-mgcv::gam(data=Influenza, Mortality ~ Year + s(Week,k=length(unique(Influenza$Week))))

s=interp(Influenza$Year, Influenza$Week, fitted(gammer1))
print(gammer1)
summary(gammer1)
gammer1$sp
#plot_ly(x=~s$x, y=~s$y, z=~s$z, type="surface")
knitr::include_graphics("surface.png")

modeldev <- NULL
for(sp in c(0.001, 0.01, 0.005, 2, 5))
{
  k=length(unique(Influenza$Week))
gammod <- mgcv::gam(data = Influenza, Mortality~Year+s(Week, k=k, sp=sp), method = "GCV.Cp")
temp <- cbind(gammod$deviance, gammod$fitted.values, gammod$y, Influenza$Date,
              sp, sum(influence(gammod)))
modeldev <- rbind(temp, modeldev)
}

modeldev <- as.data.frame(modeldev)
colnames(modeldev) <- c("Deviance", "Mortalitypred", "Mortality", "Date",
                            "penaltyfactor", "dof")

modeldev$Date <- as.Date(modeldev$Date, origin = '1995-01-01')
#deviance plot
p5 <- ggplot(data=modeldev, aes(x = penaltyfactor, y = Deviance)) +geom_line() +theme_dark() +
ggtitle("Plot of Deviances of models vs. Penalty Factors")
p5
#degree of freedom plot
p6 <- ggplot(data=modeldev, aes(x = penaltyfactor, y = dof)) +geom_line() +theme_dark() +
ggtitle("Plot of Degree of freedoms of models vs. Penalty Factors")
p6

modeldevwide <- melt(modeldev[,c("Date", "penaltyfactor",
                                          "Mortality", "Mortalitypred")],
                           id.vars = c("Date", "penaltyfactor"))

#predicted vs observed mortality
```

```
p7 <- ggplot(data=modeldevwide[modeldevwide$penaltyfactor == 0.001,], aes(x= Date, y = value)) +
  geom_line(aes(color = variable), size=1) +scale_fill_brewer() +theme_dark() +ggtitle("Plot of Mortali

p8 <- ggplot(data=modeldevwide[modeldevwide$penaltyfactor == 5,], aes(x= Date, y = value)) + geom_line(a

grid.arrange(p7,p8,ncol=1)

Influenza$rez<-gammer$residuals
p9<-ggplot(Influenza,aes(x=Date))+geom_line(aes(y=rez,color="Residuals"))+geom_line(aes(y=Influenza,colo
p9

lastgammod <- mgcv::gam(data = Influenza, Mortality~s(Year,k=length(unique(Influenza$Year)))+s(Week, k=

Influenza$lastgammodpred<-mgcv::predict.gam(lastgammod,newdata = Influenza,type = "link")

p10<-ggplot(Influenza,aes(x=Date))+geom_line(aes(y=lastgammodpred,color="PredictedMortality"),size=1.5)+
p10

data<-read.csv2("data.csv",header = TRUE,sep=";")
email<-as.data.frame(data)
email$Conference<-as.factor(email$Conference)
rownames(email)=1:nrow(email)

n=dim(email)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.7))
train=email[id,]
test=email[-id,]
xtrain=t(train[,-4703])
ytrain=train[[4703]]
xtest=t(test[,-4703])
ytest=test[[4703]]
myemailtrain=list(x=xtrain,y=ytrain,geneid=as.character(1:nrow(xtrain)),genenames=rownames(xtrain))
myemailtest=list(x=xtest,y=ytest,geneid=as.character(1:nrow(xtest)),genenames=rownames(xtest))
model=pamr.train(myemailtrain,threshold = seq(0,4,0.1))

cvmodel=pamr.cv(model,myemailtrain)
print(cvmodel)

pamr.plotcv(cvmodel)

pamr.plotcen(model,myemailtrain,threshold=1.4)

a=pamr.listgenes(model,myemailtrain,threshold=1.4)
cat(paste(colnames(myemailtrain)[as.numeric(a[,1])],collapse = '\n'))

predicted <- pamr.predict(model, newx = xtest, threshold = 1.4)

contab <- table(ytest, predicted)
contab
names(dimnames(contab)) <- c("Test Actual", "Predicted by Nearest Shrunken Centroid on test")
contabres<-caret::confusionMatrix(contab)
mse1<-(1-(sum(diag(contab))/sum(contab)))
```

```r
paste("The misclassificaiton rate is",mse1)

var<- as.data.frame(pamr.listgenes(model, myemailtrain, threshold = 1.4))
knitr::kable(colnames(data[,head(var$id,10)]), caption = "Top 10 Important features by NSC ")

xtrain2<-as.matrix(train[,-4703])
ytrain2<-as.matrix(train[,4703])
xtest2<-as.matrix(test[,-4703])
ytest2<-as.matrix(test[,4703])

cvmodel2<-cv.glmnet(x=xtrain2,y=ytrain2,alpha = 0.5,family="binomial")
model2<-glmnet(x=xtrain2,y=ytrain2,alpha = 0.5,family="binomial")
elasticpredict<-predict.cv.glmnet(cvmodel2, newx = xtest2, s = "lambda.min", type = "class")
elasticpredict2<-predict(model2, xtest2, type = "response")
contab22 <- table(ytest2, elasticpredict)
plot(cvmodel2)
plot(model2)
contab2 <- table(ytest2, elasticpredict)
contab2
contab2res<-caret::confusionMatrix(contab2)
mse2<-(1-(sum(diag(contab2))/sum(contab2)))
paste("The misclassificaiton rate is",mse2)
names(dimnames(contab2)) <- c("Actual Test", "Predicted by ElasticNet model")

elasticcoefs<- coef(cvmodel2, s = "lambda.min")
elasticvars <- list(name = elasticcoefs@Dimnames[[1]][elasticcoefs@i + 1])
knitr::kable(elasticvars, caption = "Contributing features of elastic net model")
set.seed(12345)
svmmodel<- ksvm(xtrain2, ytrain2, kernel="vanilladot",scaled=FALSE)
svmpredict<- predict(svmmodel, xtest2, type="response")

consvm<- table(ytest2, svmpredict)
names(dimnames(consvm)) <- c("Actual Test", "Predicted svm")
consvmres<-caret::confusionMatrix(consvm)
consvm
mse3<-(1-(sum(diag(consvm))/sum(consvm)))
paste("The misclassificaiton rate is",mse3)
comptab<- as.data.frame(cbind(contabres$overall[[1]]*100,
                      contab2res$overall[[1]]*100,
                      consvmres$overall[[1]] *100))
countf <- cbind(nrow(var), length(elasticcoefs@i), length(svmmodel@coef[[1]]))
mse <- c(mse1,mse2,mse3)
comptab <- rbind(comptab, countf)
comptab <- rbind(comptab, mse)
colnames(comptab) <- c("Nearest Shrunken Centroid Model",
                           "ElasticNet Model", "SVM Model")
rownames(comptab) <- c("Accuracy", "Number of Features","Misclassifcation error rate")
knitr::kable(comptab, caption = "Comparsion of the models")

set.seed(12345)
p<-c()
x<-email[,-4703]
for (i in 1:(length(email)-1)){
```

```
  x<-email[,i]
res<-t.test(x~Conference,data=email,alternative="two.sided")
p[i]<-res$p.value
}

pvalues<- data.frame(pvalue=p,variable=1:(length(email)-1))
pvalues<- pvalues[order(pvalues$pvalue),]


alpha<-0.05
l<-c()
o<-1
for(j in 1:length(p)){
 if( pvalues$pvalue[j]< alpha*(j/nrow(pvalues)) ){
   l[o]<-j
   o<-o+1
 }
}
pl = pvalues$pvalue[max(l)]
pl
for(j in 1:nrow(pvalues)){
  if(pvalues$pvalue[j]<= pl){
    pvalues$status[j]<-FALSE
  }
  else{
    pvalues$status[j]<-TRUE
  }
}

significantp<-filter(pvalues,pvalue<=0.05)
significantp<-cbind(significantp,Variable_name=colnames(email[significantp$variable]))
significantp

finalbh<-filter(pvalues,status==FALSE)
finalbh<-cbind(finalbh,Variable_name=colnames(email[finalbh$variable]))
finalbh
```

## Assignment 1 (7p)

The data file "glass.csv" contains information about the contents of chemicals in various samples of the glass. In the tasks below, you are assumed to investigate how the contents of Aluminium (Al) can be explained by the other chemicals. Therefore, consider Al as target variable and the remaining variables as thefeatures in the models below.

1. Partition data into training, validation and test sets (50/25/25) by using seed 12345. Use training and validation trees to fit the regression trees of different sizes and estimate the predictive error. Provide the plot showing the dependence of the training and validation errors on the tree size and comment which tree is optimal and why. Interpret this graph also in terms of bias-variance tradeoff. **(2p)**
2. Investigate the optimal tree from step 1 and report which variables were chosen. Report also the test error (assume that the target is normally distributed). **(1p)**
3. Fit a PLS regression model in which the amount of variables are chosen by cross validation. Answer the following questions **(3p)**:
   a. How many variables are enough to explain at least 90% of variation in the feature space?
   b. How many variables are enough to explain at least 90% of variation of the target?
   c. What is the optimal amount of variables according to the cross-validation?
   d. Which variables contribute mostly to the first principle component?
   e. What is the equation of the Target in the coordinates of the principle components?
   f. What is the prediction error of the optimal PLS model for the test data?
4. Compare the test errors of the optimal tree and PLS models and answer which model has a better predictive power. Comment on why choosing the holdout principle for these data is much less reliable than using cross-validation. **(1p)**

## Assignment 2 (8p)

In this assignment, you are going to analyze dataset **mtcars** available in basic R. Scale predictors *qsec* and *hp* for further analysis.

1. Plot the data in the coordinates *hp* versus *qsec* where the data are colored by *am*. Does it seem that the Linear Discriminant analysis will be able to separate these data perfectly if class priors are chosen appropriately? Do these data seem to fulfill assumptions of LDA? **(1p)**
2. Perform LDA with response *am* and predictors *qsec* and hp and
   a) Equal priors
   b) Proportional priors

   Plot the classified data and compare the results obtained by a) and b). Which method seems to classify the data better? How has the parameters of the decision boundary (intercept and slope) changed from case a) to case b) and why? **(3p)**
3. Implement kernel density estimation with Epanechnikov kernel that uses matrices X, Xtest and a scalar λ to estimate density from X and predict it at Xtest (observations in the matrices are given in the rows). Estimate the kernel density for λ =0.2 and for
   a. X is matrix with two columns: column 1 are *qsec* values such that *am=0*, column 2 are *hp* values such that *am=0*, Xtest is matrix with two columns: column 1 are all *qsec* values, column 2 are all *hp* values
   b. X is matrix with two columns: column 1 are *qsec* values such that *am=1*, column 2 are *hp* values such that *am=1*, Xtest is matrix with two columns: column 1 are all *qsec* values, column 2 are all *hp* values

   Use the estimated densities and the Bayesian rule to classify your data (assume equal class priors) and plot the classified data. Comment on the quality of fit. How does the performance of this classifier change when λ is set as a very small or very large value and why? **(4p)**

## Assignment 3 (5p)

The file **wine.csv** contain 130 bottles of wine of two different types and some measurements of levels within these bottles.

1. First change the class "2" to "-1" in order to model a classification problem using the *neuralnet* package.
2. Separate the data set into 70 percent training and 30 percent test using seed 12345.
3. Fit a neural network with seed 12345, 0 hidden nodes and *tanh* as the activation function of the output layer. Present the weights and state which variable is deemed most and least important. **(1p)**
4. Compute the misclassification rate of the training and test set and interpret the results. (Note that the model predictions need to be converted to the -1, 1 classes, this is easily done by the sign-function in R.) **(1p)**
5. Fit a new neural network with the same seed as in step 3, 1 hidden node and *tanh* as the activation function of the hidden layer. The output layer should have no activation function. Repeat step 4 and compare with the model from step 3. **(2p)**
6. Reflect on what type of model the neuralnet-function in step 3 and 5 is trying to fit, specifically the error and activation functions. Given the data set and the response variable, is this type of architecture the proper one to use? Motivate your answer. **(1p)**

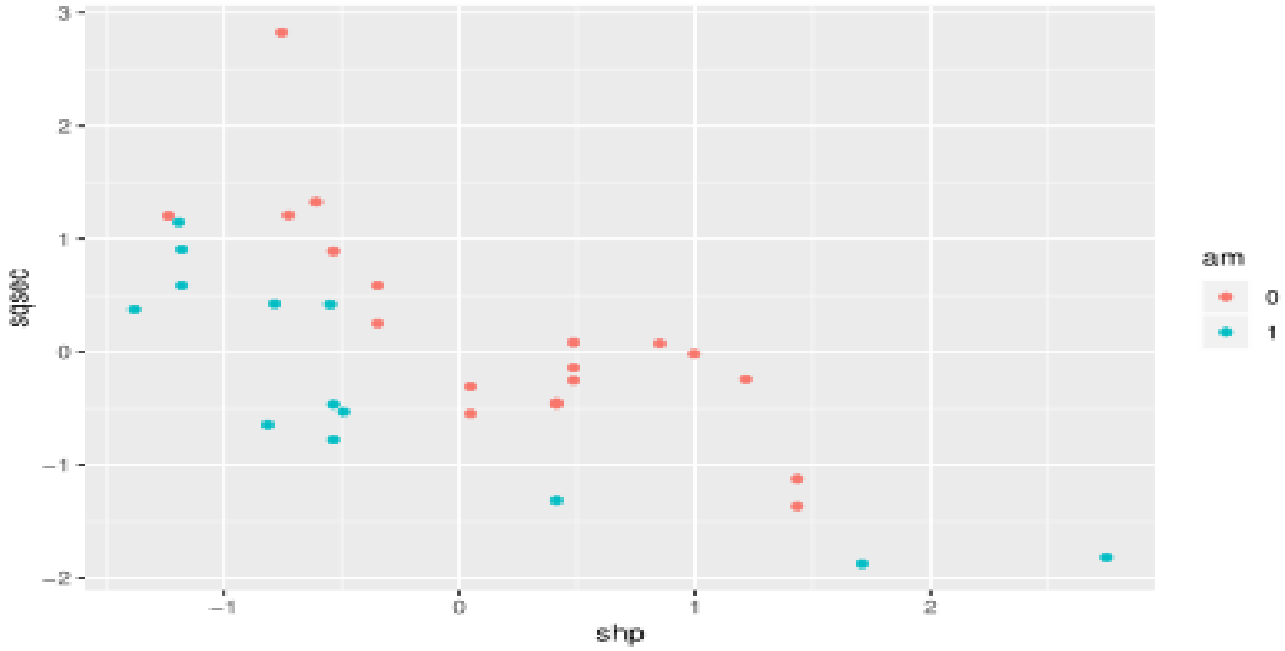
## Assignment 1

```
library(tree)
library(ggplot2)
library(pls)

data <- read.csv2("../data/glass.csv")

set.seed(12345)
n <- nrow(data)

train_size <- floor(n * 0.5)
validation_size <- floor(n * 0.25)
test_size <- n - train_size - validation_size

idx <- 1:n
train_idx <- sample(x=idx, size=train_size)
validation_idx <- sample(x=idx[-train_idx], size=validation_size)
test_idx <- idx[-c(train_idx, validation_idx)]

train <- data[train_idx,]
validation <- data[validation_idx,]
test <- data[test_idx,]

## 1
sizes <- 2:8
validation_errors <- rep(0, length(sizes))
train_errors <- rep(0, length(sizes))
fit <- tree(Al ~ ., data=train)

for (size in sizes) {
    fit_pruned <- prune.tree(fit, best=size)
    validation_errors[size-1] <- mean((predict(fit_pruned, newdata=validation) - validation$Al)^2)
    train_errors[size-1] <- mean((predict(fit_pruned, newdata=train) - train$Al)^2)
}

plot_data <- data.frame(x=sizes, y1=validation_errors, y2=train_errors)

ggplot() +
    xlab("# of terminal nodes") + ylab("Mean Squarred Error") +
geom_line(data=plot_data, aes(x=x, y=y2), color="blue") +
    geom_line(data=plot_data, aes(x=x, y=y1), color="red")

## 2
optimal_size <- which.min(validation_errors) + 1
optimal_tree <- prune.tree(fit, best=optimal_size)
test_error <- mean((predict(optimal_tree, newdata=test) - test$Al)^2)
test_error

plot(optimal_tree)
text(optimal_tree, pretty=TRUE)

## 3
```

```
set.seed(12345)
fit <- plsr(Al ~ ., data=train, validation="CV")

summary(fit)
fit$validation
fit$scores
fit$loadings

optimal_fit <- plsr(Al ~ ., data=train, ncomp=6)

## a) 3 variables
## b) 6 variables
## c) According to CV the model with 6 components is best
## d) Na Mg Si Ca Ba
## e) Y_score = z1 + z2 + z3 + z4 + z5 + z6
rowSums(optimal_fit$scores)

## f)
test_error <- mean((predict(optimal_fit, newdata=test) - test$Al)^2)
test_error

## 4
```

## Assignment 2

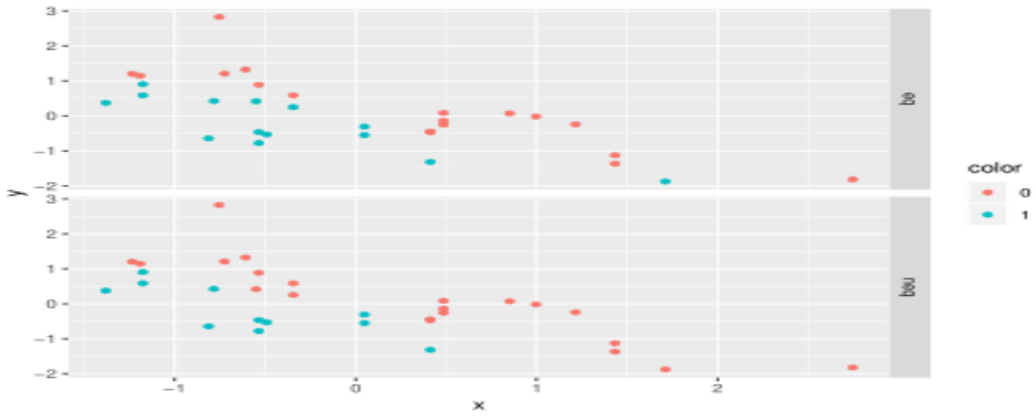
```
library(ggplot2)
library(MASS)

data <- mtcars
data$shp <- scale(data$hp)
data$sqsec <- scale(data$qsec)
data$am <- as.factor(data$am)

## 1
ggplot() +
    geom_point(data=data, aes(x=shp, y=sqsec, color=am))
```




```
## No, the data is not linearly separable

## 2
prior <- c(1, 1) / 2
fit_eq <- lda(am ~ shp + sqsec, data=data, prior=prior)
fit_eq
```

```
prior <- as.numeric(table(data$am) / sum(table(data$am)))
fit_neq <- lda(am ~ shp + sqsec, data=data, prior=prior)
fit_neq
```

```
prediction_eq <- predict(fit_eq, data)$class
prediction_neq <- predict(fit_neq, data)$class

plot_data1 <- data.frame(x=data$shp, y=data$sqsec, color=prediction_eq, type="eq")
plot_data2 <- data.frame(x=data$shp, y=data$sqsec, color=prediction_neq, type='neq')

plot_data <- rbind(plot_data1, plot_data2)

ggplot() +
    geom_point(data=plot_data, aes(x=x, y=y, color=color)) +
    facet_grid(type ~ .)
```

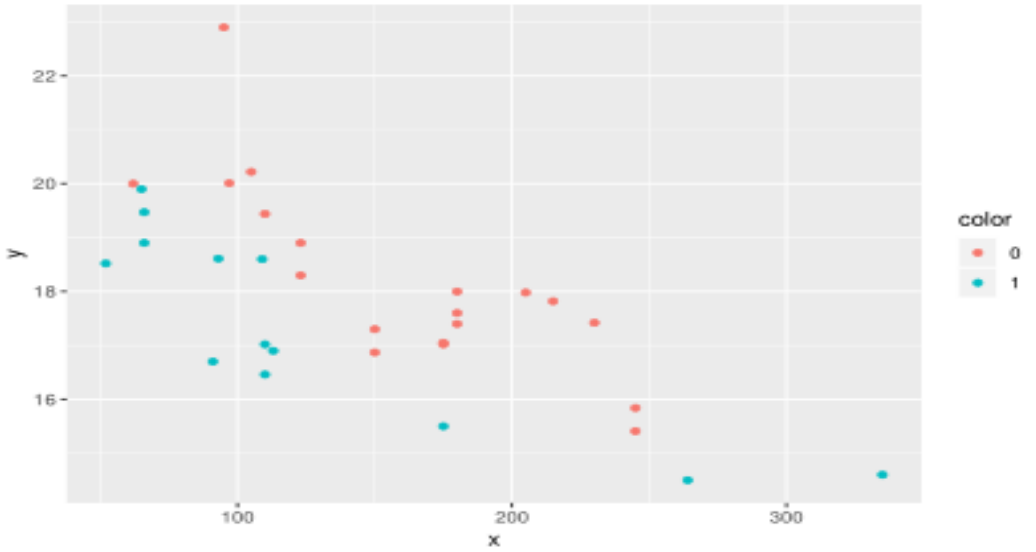



```
## 3
euclidean <- function(u) {
    sqrt(sum(u^2))
}

kernel.epan <- function(u) {
    (1 - euclidean(u)^2) * as.numeric((euclidean(u) <= 1))
}

kernel.density <- function(X, Xtest, lambda) {
    apply(Xtest, 1, function(x){
        s <- 0

        for (i in 1:nrow(X)) {
            s <- s + kernel.epan((X[i, ] - x) / lambda)
        }

        s / nrow(X)
    })
}

lambda <- 0.2

idx1 <- which(data$am == 0)
X1 <- as.matrix(data.frame(data$qsec[idx1], data$hp[idx1]))
Xtest1 <- as.matrix(data.frame(data$qsec, data$hp))
```

```
density1 <- kernel.density(X1, Xtest1, lambda)

idx2 <- which(data$am == 1)
X2 <- data.frame(data$qsec[idx2], data$hp[idx2])
Xtest2 <- data.frame(data$qsec, data$hp)
density2 <- kernel.density(X2, Xtest2, lambda)

densities <- data.frame(density1, density2)
prediction <- apply(densities, 1, function(x) which.max(x))

prediction_error <- mean(as.numeric(data$am) != prediction)
prediction_error
```

```
## [1] 0
```

```
plot_data <- data.frame(x=data$hp, y=data$qsec, color=as.factor(prediction - 1))

ggplot() +
    geom_point(data=plot_data, aes(x=x, y=y, color=color))
```



## Assignment 3

```
library(neuralnet)

data <- read.csv("../data/wine.csv")
data$class[which(data$class == 2)] <- -1

set.seed(12345)
train_idx <- sample(1:nrow(data), size=floor(nrow(data) * 0.7))
train <- data[train_idx,]
test <- data[-train_idx,]

## 3
set.seed(12345)
formula <- paste("class ~ ", paste(names(data)[-1], collapse=" + "))
fit <- neuralnet(formula=formula, data=train, hidden=0, act.fct="tanh", linear.output=FALSE)
plot(fit)

weights <- fit$weights[[1]][[1]][-1,]
weights
variables <- fit$model.list$variables[order(abs(weights), decreasing=TRUE)]
variables

## 4
train_error <- mean(sign(compute(fit, train[, -1])$net.result) != train$class)
train_error

test_error <- mean(sign(compute(fit, test[, -1])$net.result) != test$class)
test_error

## 5
set.seed(12345)
formula <- paste("class ~ ", paste(names(data)[-1], collapse=" + "))
fit <- neuralnet(formula=formula, data=train, hidden=1, act.fct="tanh", linear.output=TRUE)
plot(fit)

train_error <- mean(sign(compute(fit, train[, -1])$net.result) != train$class)
train_error

test_error <- mean(sign(compute(fit, test[, -1])$net.result) != test$class)
test_error

## 6
## 1. A tanh function
## 2. A translated tanh function
## 3. Parabola
```

```
# JMP

library(neuralnet)

# two layers

set.seed(1234567890)

Var <- runif(50, 0, 10)

trva <- data.frame(Var, Sin=sin(Var))

tr <- trva[1:25,] # Training

va <- trva[26:50,] # Validation

# plot(trva)

# plot(tr)

# plot(va)

restr <- vector(length = 10)

resva <- vector(length = 10)

winit <- runif(22, -1, 1) # Random initializaiton of the weights in the interval [-1, 1]

for(i in 1:10) {

  nn <- neuralnet(formula = Sin ~ Var, data = tr, hidden = c(3,3), startweights = winit,

                  threshold = i/1000, lifesign = "full")

   # nn$result.matrix

   aux <- compute(nn, tr[,1])$net.result # Compute predictions for the trainig set and their squared error

  restr[i] <- sum((tr[,2] - aux)**2)/2

   aux <- compute(nn, va[,1])$net.result # The same for the validation set

  resva[i] <- sum((va[,2] - aux)**2)/2

}

plot(restr, type = "o")

plot(resva, type = "o")

restr

resva

# one layer

set.seed(1234567890)

Var <- runif(50, 0, 10)

trva <- data.frame(Var, Sin=sin(Var))

tr <- trva[1:25,] # Training

va <- trva[26:50,] # Validation

# plot(trva)

# plot(tr)

# plot(va)

restr <- vector(length = 10)

resva <- vector(length = 10)

winit <- runif(41, -1, 1) # Random initializaiton of the weights in the interval [-1, 1]

for(i in 1:10) {

  nn <- neuralnet(formula = Sin ~ Var, data = tr, hidden = c(10), startweights = winit,

                  threshold = i/1000, lifesign = "full")


  # nn$result.matrix


  aux <- compute(nn, tr[,1])$net.result # Compute predictions for the trainig set and their squared error

  restr[i] <- sum((tr[,2] - aux)**2)/2


  aux <- compute(nn, va[,1])$net.result # The same for the validation set

  resva[i] <- sum((va[,2] - aux)**2)/2

}

plot(restr, type = "o")

plot(resva, type = "o")

restr

resva


# estimate generalization error for the best run above (one layer with threshold 4/1000)


Var <- runif(50, 0, 10)

te <- data.frame(Var, Sin=sin(Var))


winit <- runif(31, -1, 1)

nn <- neuralnet(formula = Sin ~ Var, data = trva, hidden = 10, startweights = winit,

                  threshold = 4/1000, lifesign = "full")

sum((te[,2] - compute(nn, te[,1])$net.result)**2)/2
```