Detection of Epileptic Seizure

ISEN 613 : Engineering Data Analysis

07-Dec-2018

TEAM

Ganesh Praneeth Arcot	127004004	(20%)
Keerthi Panganamala	427009338	(20%)
Nishitha Battu	527004879	(20%)
Sowmya Gaja Kumar.	526009328	(20%)
Suyashkumar Chavan.	228001537	(20%)

Table of Contents

Introduction

Importance of the problem:
Epilepsy is a neurological disorder that is marked by sudden recurrent episodes of sensory disturbances, loss of consciousness associated with abnormal activity in the brain. Every year about 150,000 Americans are diagnosed with Epilepsy and over a lifetime, 1 in 26 Americans will be diagnosed with Epilepsy. It is the fourth most common neurological disease. It is most commonly diagnosed before age 20 or after the age of 65.

Symptoms of Epilepsy include a staring spell, loss of consciousness or recognition, confusion, uncontrollable movement and repetitive movements. Diagnosing Epilepsy requires several tests and studies to ensure the symptoms and sensations, are indeed, the result of Epilepsy. Most common tests are Blood based, EEG (Electroencephalogram), Neurological Examination, MRI, CT Scan, Pet scan etc. Of all this, EEG is the tool that most successfully diagnoses Epilepsy.

Objective :
This project (or the data set) was chosen with the main intention as to classify people if or not they have Epilepsy using both Supervised and Unsupervised techniques. These kind of analyses can be extrapolated and can be used to predict Epilepsy while it's still early, followed with either medication or surgery.

One of the most important task is to increase the prediction accuracy of the response variable. Results of the classification techniques used in this report are presented in the form of a table (or a Confusion Matrix). The best technique is the one with the highest value of TP (True Positive). While FP (False Positive) prediction is not harmful, FN (False Negative) rate has to be as low as possible as we don't want to classify people with Epilepsy as negative.

Specifics :
The dataset from UCI Machine learning repository was only partially processed and is a widely used dataset. The problem initially had 5 classification labels, Class 1 for positive Epilepsy and Negative for the rest. This was then transformed to a binary classification for ease of prediction.

Scope of Work :
The Machine Learning techniques used in this report (a few beyond the scope of the class) aim at classifying in order to aid the detection of Epilepsy. The techniques used are:

- Linear Discriminant Analysis
- Logistic Regression
- K - Nearest Neighbours
- Decision Trees
- Support Vector Machines
- Neural Networks
- Naive Bayes

These models are then classified based on the Accuracy and False Negative rate.

Acknowledgements

This project has been a learning experience in understanding the uncertainties associated with data systems in the real-world scenario. It has given us an opportunity to work with a dataset which has real time applications in the field of medicine.

Firstly, we would like to express our sincere gratitude to our course instructor Dr. Na Zou for the continuous support and related help in our project, for her patience, motivation, and immense knowledge. Her guidance helped us in all the time of subject and this project. We could not have imagined having a better course instructor for this subject.

We would also like to thank our Teaching Assistant Ming Li , for his insightful comments and encouragement.

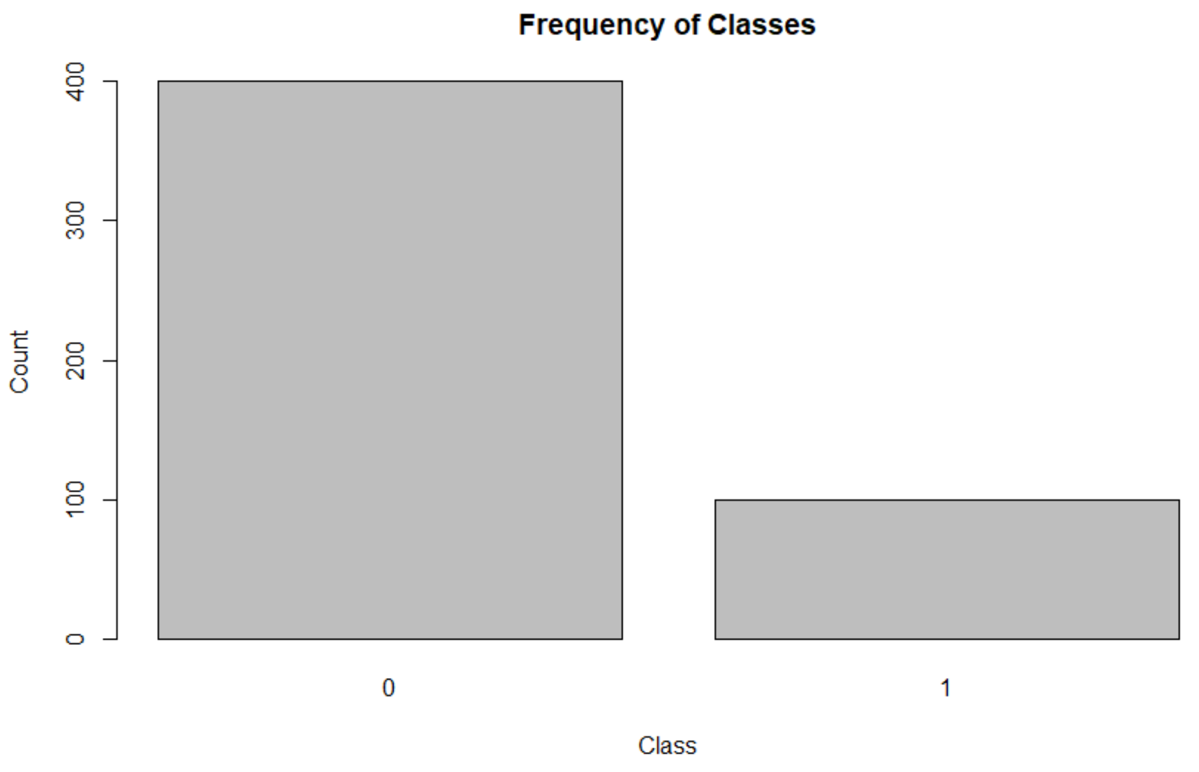
We would also like to acknowledge the support and resources Texas A&M University has provided us throughout this project.

Project Approach

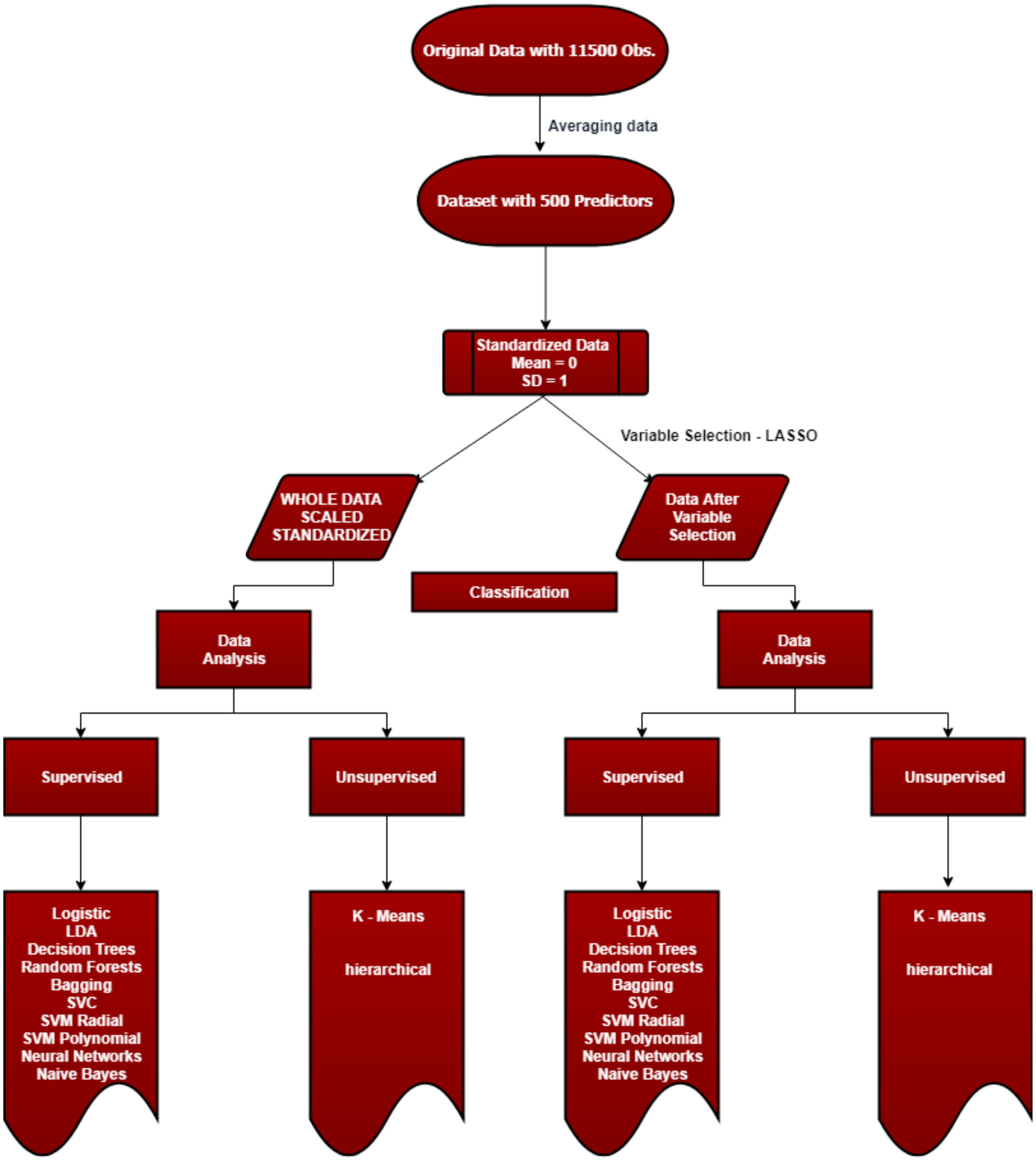
Data Description :

The original data file for this project was obtained from UC Irvine's Machine Learning repository. The original data had 5 different folders, each with 100 files, each one representing a single subject. Each of these files is a recording of the brain activity for 23.6 seconds. Every data point is the value of the EEG recording, so we have a total of 4097 data points for 23.6 seconds. Then, every 4097 data points were shuffled into 23 chunks (each containing 178 data points for 1 sec). So, now we have a total of 500 x 23 = 11500 rows of observation. Each row has 178 data points (attributes) and last column is the response, y (1,2,3,4,5).

This data was modified, averaged to bring back the readings of the 500 people. So the final data now has 500 observations with 178 predictors and 1 response. The response variable has 5 classes (Classes 1, 2, 3, 4 and 5). All the subjects in classes 2, 3, 4 and 5 did not have epileptic seizures. Only the subjects in class 1 had seizures and so, this data was again modified to become a binary classification with Classes 1 and 0. Subjects in Class 1 had seizures and subjects in class 0 did not. Figure 1 shows the frequency of both the classes in the data.



Methodology



The data was first checked to find any missing observations and remove them. This data was then standardised with Mean 0 and SD 1.The response variable, was encoded as factor variables. **dimension reduction and**. This data set was split into training and test set on random (70% and 30% respectively). This training data was used to train the data with the techniques mentioned above and for each of these, both the training and test error were calculated and test error was used to select the best out of these techniques. Cross validation was also used to calculate the test error for these.

Reason for Approach

Numerous methods were introduced talked about in the course and it is clear that "There is no free lunch". The initial data had a lot of predictors and hence subset selection felt more reasonable. After calculating the test error, the new data was not better in all the cases. Few of the methods used in prediction are standard and are very reliable. Additional methods were used to improvise upon and to increase the prediction accuracy. This is because, a model that's a better fit in a few situation might not be the best overall. Hence, the main goal of this project is to compare all the techniques and find a suitable model that fits the data best.

Implementation

Preprocessing:
The data did not have any missing values and initially had 5 classes. This was reduced to 2 classes (Class 1 - Positive epilepsy and the rest for negative). The first two columns were deleted as they were the patient ID's and will not be useful in any way. This was scaled to then have a mean of 0 and standard deviation of 1.

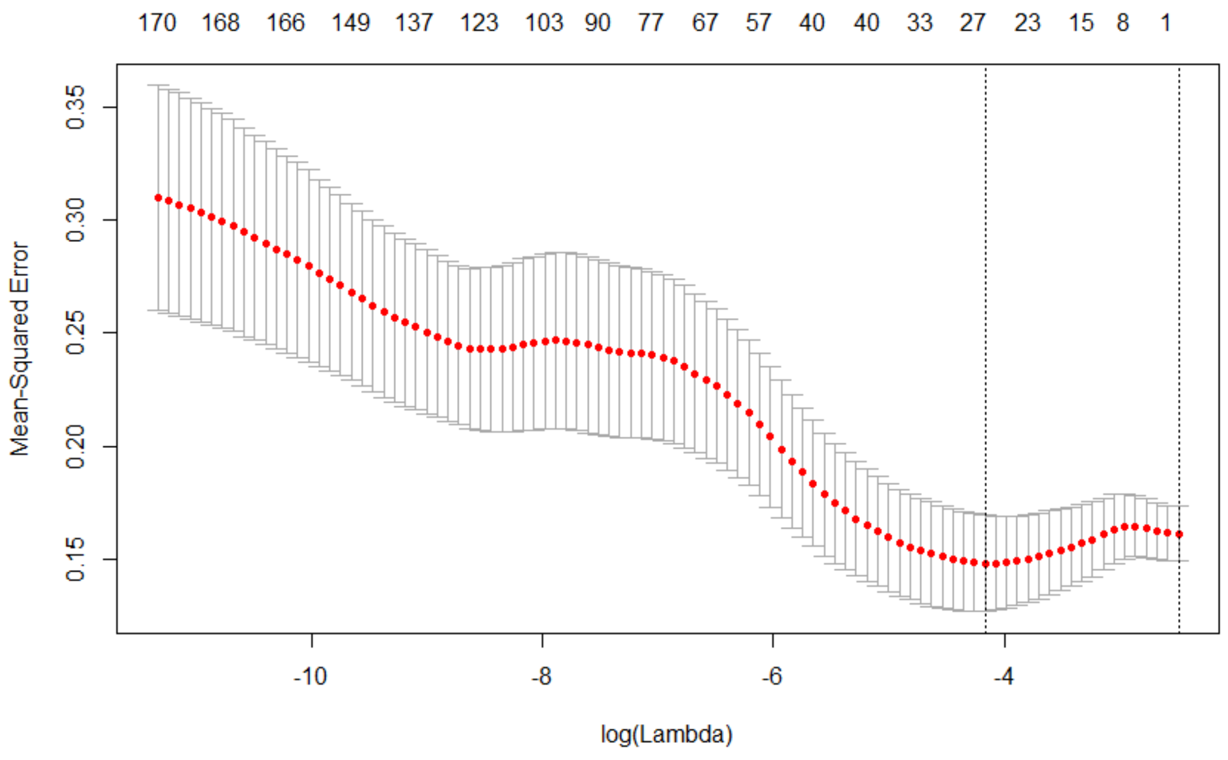
```
data = data[c(-1,-2)]
data$y[data$y != 1] = 0
#data$y = as.factor(data$y)
data[,1:178] = scale(data[,-179])
```

The next step is best subset selection using Lasso.

Variable Selection:

The methods used for variable selection is Lasso Regression. The best lambda was found out to be ***0.01555411341.*** And the result had 16 variables in the matrix, "X7" "X11" "X12" "X27" "X33" "X34" "X44" "X57" "X94" "X95" "X96" "X97" "X102" "X152" "X158" "X160". These variables were used to create a new data set including only these. Through out this report, we will be using both the data sets - One with all the variables and also the one with reduced variables to analyse and compare.

```
#Lasso Regression
library(glmnet)
x = model.matrix(y~., data)[,-c(179)]
y = data$y
fit.cv = cv.glmnet(x,y,alpha = 1)
fit.cv$lambda.min
plot(fit.cv)
lasso.fit = glmnet(x,y,alpha=1, lambda = fit.cv$lambda.min)
abc = as.matrix(coef(lasso.fit)[,1])
```



Classification Methods:

1. Linear Discriminant Analysis:

Using the Whole data:

LDA, a classification algorithm is generally used when the number of classes is greater than 1. LDA works on the assumption that all the predictors follow a normal distribution with equal variances. If this assumptions fails, LDA might not produce a good model.

LDA was fit as shown below:

The training error for LDA was 0.0133. The test error with the test data is 0.128 or 12.8%.

Accuracy : 0.872

Sensitivity : 0.9708738

Specificity : 0.4090909

This data tells us that 87.2% of the subjects have been classified correctly and 12.8% have been classified incorrectly. The number of subjects that had the seizures but were of incorrectly predicted is around 10%.

For the Data with selected variables, the training error is 0.16533 and the test error is 16.8%. The confusion matrix for this data is

LDA	Actual	
Predicted	0	1
0	100	13
1	3	9

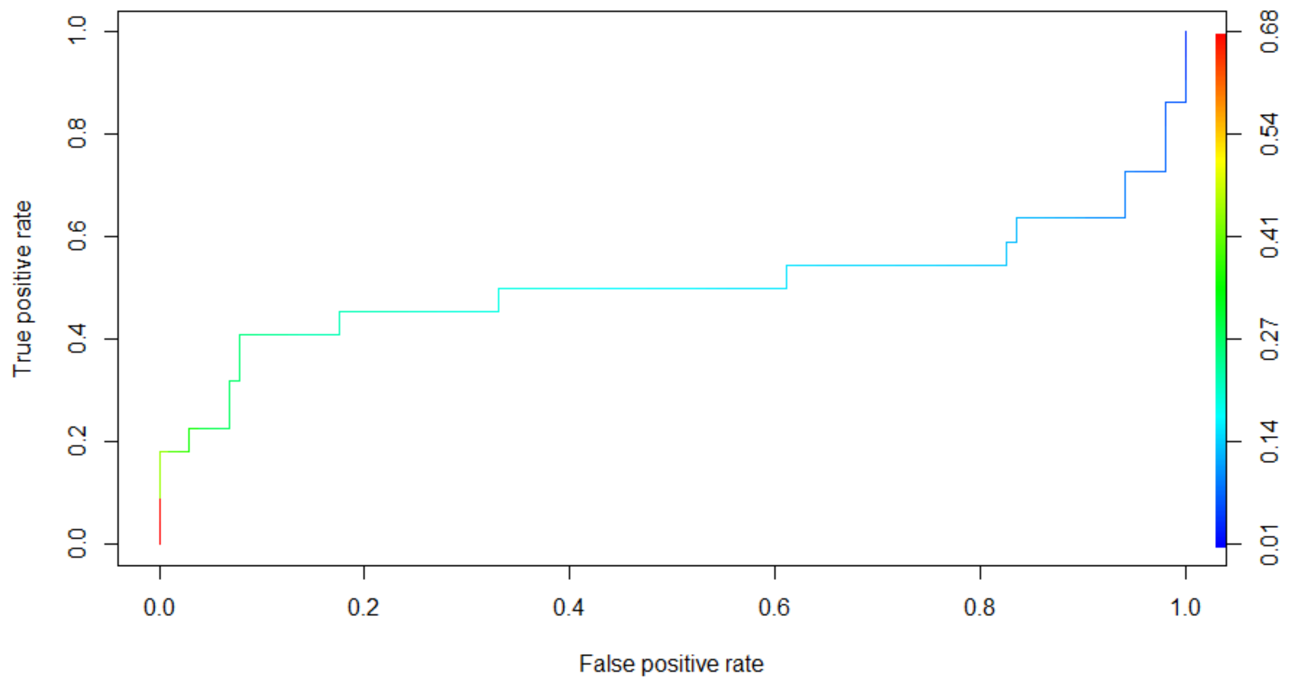
LDA	Actual	
Predicted	0	1
0	103	21
1	0	1

Accuracy : 0.832

Sensitivity : 1

Specificity : 0.04545

The number of subjects that have now been categorised as not having the seizures, despite having them is now 16.8%, higher than before. The ROC curve for LDA is as



below:

```
#LDA fitting
library(MASS)
library(caret)
lda.fit = lda(y~., data = train)

##Training error
lda.train = predict(lda.fit,train)
lda.class.train = lda.train$class
tr.lda.table = table(lda.class.train, train$y)
lda.train.error = mean(lda.class.train != train$y)

##prediction accuracy
lda.pred = predict(lda.fit,test)
lda.class = lda.pred$class
cm.lda = confusionMatrix(lda.class, as.factor(test$y))
print(cm.lda)
```

2. Classification Tree: Decision Tree

Decision trees are very easy to interpret and can be used for both regression and classification, and hence can be used here. Decision trees require very less data preparation, performs well with large data sets and closely mimics human decision making. The classification tree using all the variables is shows below in the figure.

The training error for this data is ***2.4%*** and the test error is ***12%.*** The number of subjects who had and seizures but were falsely identified are ***12%.*** The specificity and sensitivity are 0.96116 and 0.5 respectively. The confusion matrix for the decision tree

```
#Classification Tree CHANGE TO FACTORS
library(tree)
library(ISLR)
set.seed(3)
train$y = as.factor(train$y)
test$y = as.factor(test$y)
tree.ep = tree(y~., data = train)
##Training Error
pred.tree.train = predict(tree.ep, data = train, type = "class")
table(pred.tree.train, train$y)
plot(tree.ep)
text(tree.ep, pretty = 0)
tree.train.error = mean(pred.tree.train !=train$y)
##Test Error
tree.pred.test = predict(tree.ep, test, type = "class")
cm.tree = confusionMatrix(tree.pred.test, test$y)
```

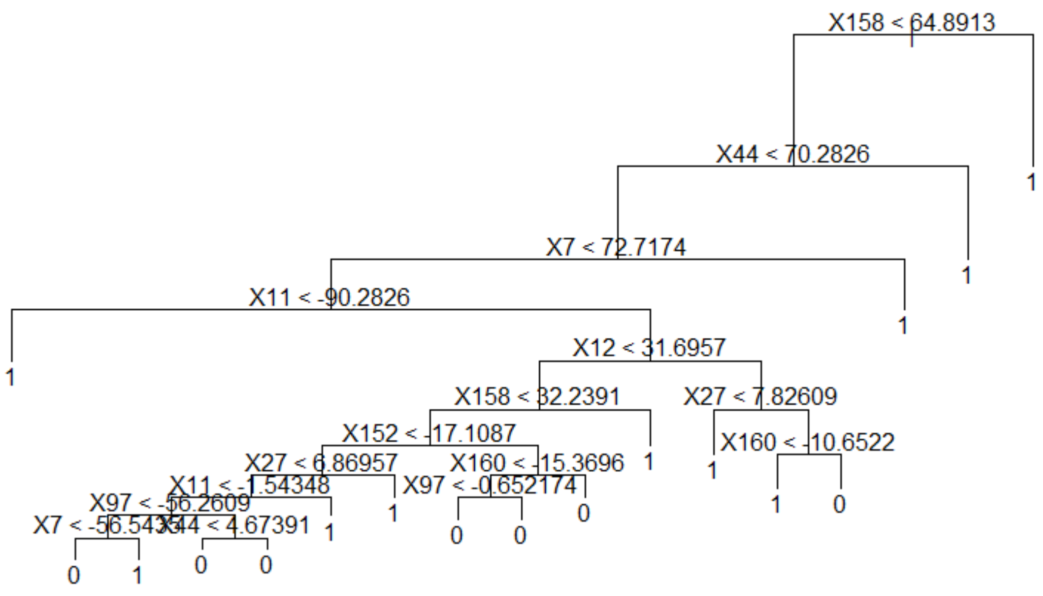
is shown below.

	Actual	
Predicted	0	1
0	99	11
1	4	11

Using the selected data with limited variables, the training error was ***4.53%*** and the test error was ***9.6%***. The subjects that had actual 1 but predicted 0 were around ***2.4%***.

The sensitivity and specificity were around 0.912 and 0.863 respectively. The decision tree and the confusion matrix are shown below

	Actual	
Predicted	0	1
0	94	3
1	9	19



3. Classification Tree: Random Forests

The training error for Random Forests using training data set is 100%. The test error calculated was ***4.8%***. The Specificity and sensitivity are 0.99 and 0.68 respectively. The subjects that were actually 1 but classified as 0 are around ***4%.*** The confusion matrix is given below:

	Actual	
Predicted	0	1
0	102	5
1	1	17

```
library(randomForest)
# random forests (m = sqrt(p))
rf.tree = randomForest(y~., data = train, mtry = sqrt(178), ntree = 100,
importance = TRUE)
pred.rf = predict(rf.tree, newdata = test)
cm.lda = confusionMatrix(pred.rf, as.factor(test$y))
print(cm.lda)
```

4. Classification Tree: Bagging

The accuracy again for the training data is 100%. Using the test data, the error was again ***4.8%*** but the subjects that were actually 1 but predicted 0 went up to ***4.8%.*** The sensitivity and specificity are 1 and 0.72 respectively. The confusion matrix for bagging is given below:

```
# Bagging
bag.tree = randomForest(y~., data = train, mtry = 178, ntree = 100,
importance = TRUE)
pred.bag = predict(rf.tree, newdata = test)
cm.bag = confusionMatrix(pred.bag, as.factor(test$y))
print(cm.bag)
```

	Actual	
Predicted	0	1
0	102	6
1	0	17

5. K-Nearest Neighbours

KNN is one of the simplest classification algorithm and is one of the most used algorithms. It is non parametric and is generally used with datasets that have multiple classes for classification. In this project, we use KNN with 10 fold CV to select the best value of k (nearest neighbours). Using k = 4, the error rate is 14.4% and the specficity is 0.18. Also, the subjects with Actual class 1 but predicted class 0 are 18. The confusion matrix for KNN with k = 4 is

	Actual	
Predicted	0	1
0	103	18
1	0	4

After subset selection, using the new data set, the test error is 13.6%. The confusion matrix is:

```
library(class)
train.x = train[-179]
test.x = test[-179]
train.y = train$y
train.y = as.factor(train$y)
test.y = as.factor(test$y)
data$y = as.factor(data$y)
# ##Finding best K for K Nearest Neighbors
trControl = trainControl(method = "cv", number = 10)
fit = train(y~., method = "knn", tuneGrid = expand.grid(k=1:11), trControl =
trControl, metric = "Accuracy", data = data) ##Change to test data
# ##Predicting with KNN
pred.knn = knn(train.x, test.x, train.y, k = 4)
cm.knn = confusionMatrix(pred.knn, test.y)
print(cm.knn)
```

	Actual	
Predicted	0	1
0	103	17
1	0	5

6. Neural Networks

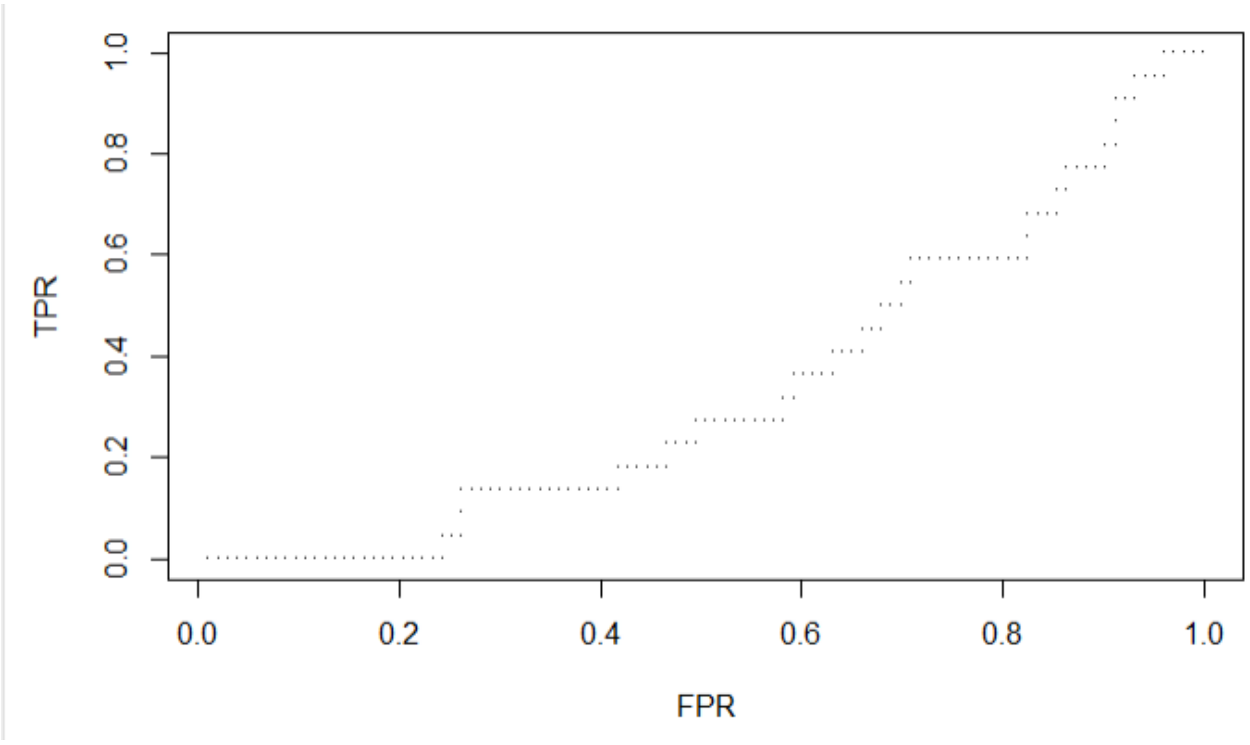
Neural network algorithms are loosely based on the human brain and are used to recognise patterns. They interpret sensory data through a kind of machine perception, labelling or clustering raw input. They patterns recognised are numerical. Neural networks help us cluster and classify. They also help group unlabelled data according to similarities among the example inputs.

The library "Neural Net" was used in R to model the data. The first layer of this network after receiving the raw input, processes it and passes it on to the next layer. Thus, this model trains itself from the data (with outcomes already available) and optimised the weights for better classification where the outcome is unavailable.

After running neuralnet for the whole data, the error is ***4.8%***. The number of subjects with Actual class 1 but predicted class 1 are 5 (4%). The confusion matrix is given by,

	Actual	
Predicted	0	1
0	102	5
1	1	17

	Actual	
Predicted	0	1
0	90	10
1	13	12



The Neural network for this data is shown below:

Running Neural Net on the selected data after subset selection, the error rate is ***18.4%***. The number of subjects actually in Class 1 but with predicted class 0 is 10. The confusion matrix is:

```
library(neuralnet)
 tt = cbind(train[-179], class.# Bagging
bag.tree = randomForest(y~., data = train, mtry = 178, ntree = 100,
importance = TRUE)
pred.bag = predict(rf.tree, newdata = test)
cm.bag = confusionMatrix(pred.bag, as.factor(test$y))
print(cm.bag)ind(train$y))
 nn = names(train[-179])
 nnn = names(train)
```

7. Naive Bayes Classifier

The Naive Bayes Classifier technique is based on the so-called Bayesian theorem and is particularly suited when the dimensionality of the inputs is high. We first apply it

```
library(e1071)
model = naiveBayes(y~., data = train)
p1 <- predict(model,train)
tab1 <- table(p1, train$y)
print(tab1)
1-sum(diag(tab1))/sum(tab1)
p2 <- predict(model,test)
tab2 <- table(p2, test$y)
tab2
1-sum(diag(tab2))/sum(tab2)
```

to the training data and use the model on the testing data in the next step.

	Actual	
Predicted	0	1
0	91	5
1	12	17

This data tells us that 86.4% of the subjects have been classified correctly and 13.6% have been classified incorrectly.

8. Support Vector Machines

SVMs are supervised learning models that have associated learning algorithms to analyze data used for classification and regression. With a set of available training data, the SVM training algorithm builds a model that assigns new examples to one category or the other, thus making it a non-probabilistic binary linear classifier.

8.1. Support Vector Classifiers

The training error rate for the whole data without subset selection was ***6.4%*** and the test error rate was ***5.6%***. The sensitivity and specificity are 1 and 0.68 respectively.

The percentage of subjects that classified as 0 despite being 1 was ***5.6%*.** The confusion matrix is given below:

	Actual	
Predicted	0	1
0	103	7
1	0	15

For the data after subset selection, the test error was ***8.8%*.** The sensitivity and specificity are 1 and 0.5 respectively. The confusion matrix is:

	Actual	
Predicted	0	1
0	103	11
1	0	11

```
train$y = as.factor(train$y)
test$y = as.factor(test$y)
library(e1071)
svm.fit = svm(y~., data = train, cost = 0.1, degree = 1)
svmpredict = predict(svm.fit, test, type = "response")
cm.svm = confusionMatrix(svmpredict, test$y)
print(cm.svm)
```

8.1.2. Radial SVM

On the whole data, the accuracy was found out to be ***17.6*%**. The sensitivity was calculated to be 1.. Also, The number of subjects in class 1 but predicted as class 0 is 22 (17.6%) The confusion matrix is

	Actual	
Predicted	0	1
0	103	22
1	0	0

Running SVM on the data after subset selection the error rate was the same but the specificity was found out to be 0.9545 and sensitivity to be 0.9514. The number of subjects in Class 1 but classified as 0 is only 1 (0.8%). The confusion matrix for this is :

	Actual	
Predicted	0	1
0	98	1
1	5	21

```
train$y = as.factor(train$y)
test$y = as.factor(test$y)
library(e1071)
svm.radial = svm(y~., data = train, kernel = "radial", cost = 1, gamma =
0.5)
svmpredict = predict(svm.radial, test, type = "response")
cm.svm = confusionMatrix(svmpredict, test$y)
print(cm.svm)
```

8.1.3. Polynomial SVM

Using the whole data without subset selection, Polynomial SVM with degree = 2 and cost = 10, the test error is **6.4%.** The sensitivity and specificity is 1 and 0.63 respectively. The number of subjects with Actual class 1 but predicted class 0 is 8 (6.4%). The confusion matrix is:

	Actual	
Predicted	0	1
0	103	8
1	0	14

Now using the data after subset selection with the same variables, the accuracy is 95.2% . The subjects with Class 0 but predicted class 1 are 6. The confusion matrix for this is:

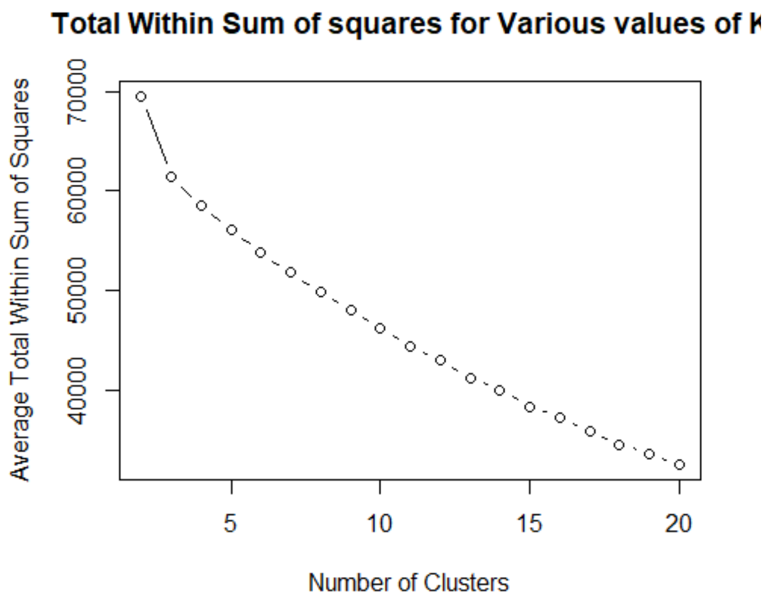
	Actual	
Predicted	0	1
0	103	6
1	0	16

```
train$y = as.factor(train$y)
test$y = as.factor(test$y)
library(e1071)
svm.poly = svm(y~., data = train, kernel = "polynomial", cost = 10,
degree = 2)
svmpred.poly = predict(svm.poly, test, type = "response")
cm.poly = confusionMatrix(svmpred.poly, test$y)
print(cm.poly)
```

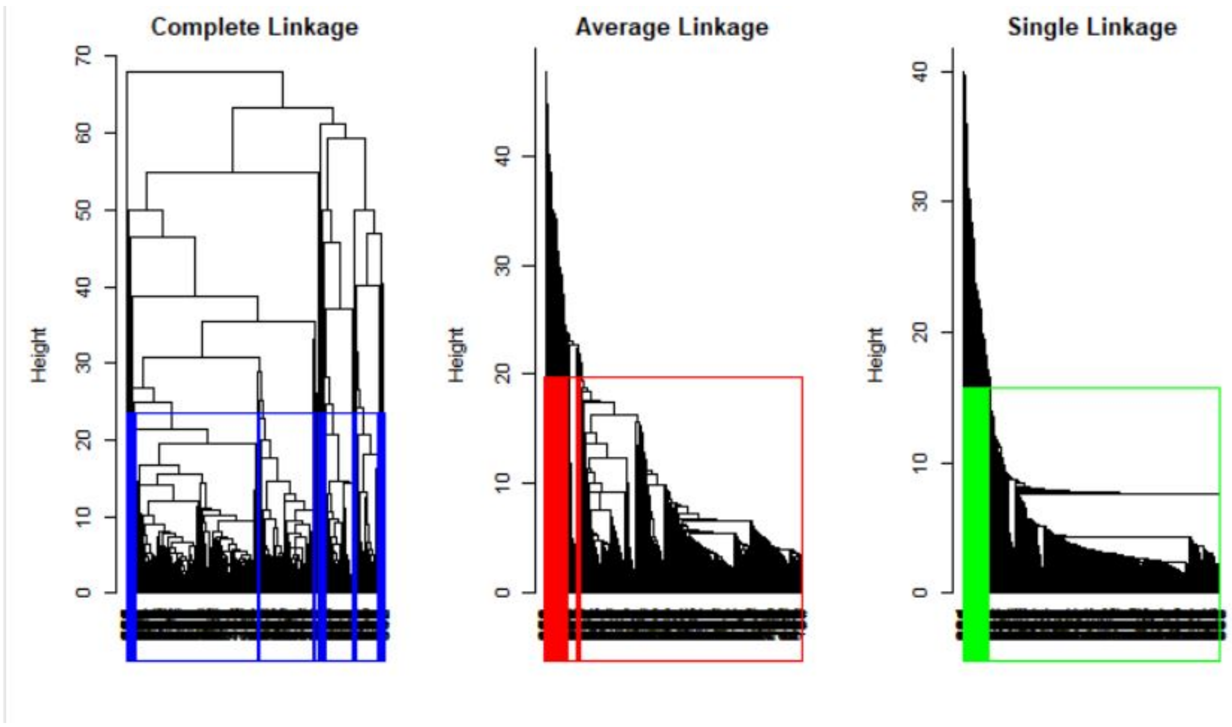
Unsupervised Learning

K Means Clustering

K means clustering is a type of unsupervised learning technique in which we divide a data set into K distinct, non-overlapping clusters(no observation belongs to more than one cluster). It is used when we have data without defined categories or simply unlabelled data. The main aim is to have minimum "within-cluster-variation".The number of clusters is denoted by K. The algorithm performs iterations to assign each data point to one of K clusters whose centroid is closest(Euclidean distance). Data points with similar features are clustered under same group.



Hierarchical Clustering:
In k-means clustering we have to specify the number of clusters K. Hierarchical clustering is an alternative approach to k-means clustering which doesn't require specifying the value of K. In addition, hierarchical clustering has an added advantage over K-means clustering in which it produces a tree-based representation of the observations, called a Dendrogram.



Summary

For entire data set and after subset selection,

Method	Testing Error (%)
Logistic	9.6
LDA	12.8
Decision Tree	12
KNN	14.4
Random Forests	4.8
Bagging	4.8
SVC	5.6
SVM Radial	17.6
SM Polynomial	6.4
Neural Networks	4.8

Method	Testing Error (%)
Logistic	9.6
LDA	16.8
Decision Tree	9.6
KNN	13.6
SVC	8.8
SVM Radial	4.8
SM Polynomial	4.8
Neural Networks	18.4

Comparisons

From all the methods used here, the one with the best test error is Random Forests, Bagging and Neural Networks. SVM Radial is the worst performed method with test error rate of 17.6%. False negative rate has also been computed for all these methods as it is necessary in this case. AS mentioned before, it might be okay to incorrectly classify subjects without Epilepsy as the ones with Epilepsy but highly dangerous to wrongly predict subjects with Epilepsy as without. Although the test error for Random Forests is the same as bagging, the False Negative value is lower for Random Forests (4% vs 4.8%).

One reason LDA has high error rate is because of the assumption. The data set needs to have constant variance for LDA to hold. The general trend is - the binary classification methods perform better than other methods. Apart from this, the data was also analysed as multi class (which is not shown in the report) and had very high error rates and also because, in a medical diagnosis it only valid if the subject has Epilepsy or not.

Executive Summary

Epilepsy is the fourth most common neurological disorder and affects people of all ages .Epilepsy means the same thing as seizure disorders. Seizures can be different for each person. Different epilepsies are due to many different underlying causes. The causes can be complex, and sometimes hard to identify. Thus, diagnosing Epilepsy requires several tests and studies to ensure the symptoms and sensations, are indeed, the result of epilepsy.The most common and effective method to diagnose epilepsy is EEG.

In this project we observed the data collected from EEG test of 500 patients across 23 seconds time , 178 predictor variables and used this data to classify whether a particular patient would experience a seizure of not.For this, we have performed various data analysis techniques and compared their results to obtain a prediction model with maximum accuracy of prediction and least false negative rate.This project can be helpful in finding the most important predictors which can help in classifying different patients who could have a chance of experiencing epileptic seizure. The approach adopted in this project can be further extrapolated to similar problem statements in medical sciences and human behavioural sciences ; by taking the current medical conditions of the patients' as input data for future predictions.

Literature Review

Topic 1: Epileptic Seizure Detection: A Deep Learning Approach :
Authors: Ramy Hussein , Hamid Palangi , Rabab Ward , and Z. Jane Wang
In this work, they used deep learning-based approach that automatically detects epileptic seizures using EEG signals. Specifically, to reveal the correlation between successive data samples, the time-series EEG data were first segmented into a sequence of nonoverlapping epochs. Second, Long Short-Term Memory (LSTM) network were used to learn the high-level representations of the normal and the seizure EEG patterns. Third, these representations were fed into Softmax function for training and classification. The results on a well-known benchmark clinical dataset demonstrate the superiority of the proposed approach over the existing state of-the-art methods.

Topic 2: Cloud-based Deep Learning of Big EEG Data for Epileptic Seizure Prediction
Authors: Mohammad Parsa Hosseini, Hamid Soltanian Zadeh, Kost Elisevich, Dario Pompil
In this work, they used a dimensionality reduction technique to increase the classification accuracy as well as to decrease the communication bandwidth and computation time. They followed a deep learning approach for which an anto-encoder is trained in two steps for unsupervised feature extraction and classification. A cloud computing solution is proposed for real time analysis of Big EEG data.

Gap in Literature

After thoroughly going through the literature it was found that the recognition of epileptic seizure with the EEG signals was done using deep learning techniques. We have used Logistic Regression, LDA, KNN, Bagging, Random Forests, Support Vector Machines and other machine learning techniques such as Neural Networks using R. The previous work was mostly done using python instead of R. In this project, we have used a package called **NeuralNet** for implementing neural networks.

Conclusion

The main goal of this project was to classify one of the most common neurological disease that can result in unprovoked seizures. This project deals with diagnosing Epilepsy that can help in early identification using the readings from EEG (Electroencephalography). EEG is one of the most common and one of the most successive process to diagnose Epilepsy. This has been a very common problem (Heart disease and other medical problems) over the last couple of decades.

This problem in particular is a classification problem and the final goal is to classify subject according to their Epileptic condition with most accuracy and the least False negative rate.This report goes through a lot of classification machine learning algorithms to get the desired results based on the standard performance parameters.To simplify the problem, this multiclass classification problem was transformed to binary classification problem. The best model with all these parameters in mind was Random Forests, including lower false negative rate.

A few Unsupervised methods were also used to better understand the problem. Subset selection was also used to better understand the important variables, but the test error was higher for almost in all the cases than the original data with all predictors. Neural Networks was one method which was expected to give better results, but, the test error was one of the highest obtained. This might be because of the size of the dataset. Neural Networks do no perform as well as Bayesian approaches on small datasets. Finally, the important steps can be summarized as:

- Cleaned and Analyzed the data followed by standardizing.
- Finding error rates for different methods and the False Negative rate
- Choosing the best model based on the above two parameters.
- The model with the best accuracy and lease false negative rate is the best of all.

Future Scope

Next step in this report is increasing prediction accuracy while still keeping a low false negative rate. More data will really be helpful in this approach. Another option is to generate artificial observations using SMTOE. Enough observations can be help ful to train the data better and hence perform better on any test data available later on. Enough observations can also help us use multiclass - classification techniques to predict the other classes. Finally, Neural Networks can also be improved by using more hidden layers to train the model better.

References

[1]. James, G., Witten, D., Hastie, T., Tibshirani, R., 2013 "An Introduction to Statistical Learning with Applications in R", Springer.

[2]. UCI: https://archive.ics.uci.edu/ml/datasets/Epileptic+Seizure+Recognition

[3]. https://www.datascience.com/blog

[4]. https://www.epilepsysociety.org.uk/epileptic-seizures#.XAsBInRKg2w

[5].https://rpubs.com

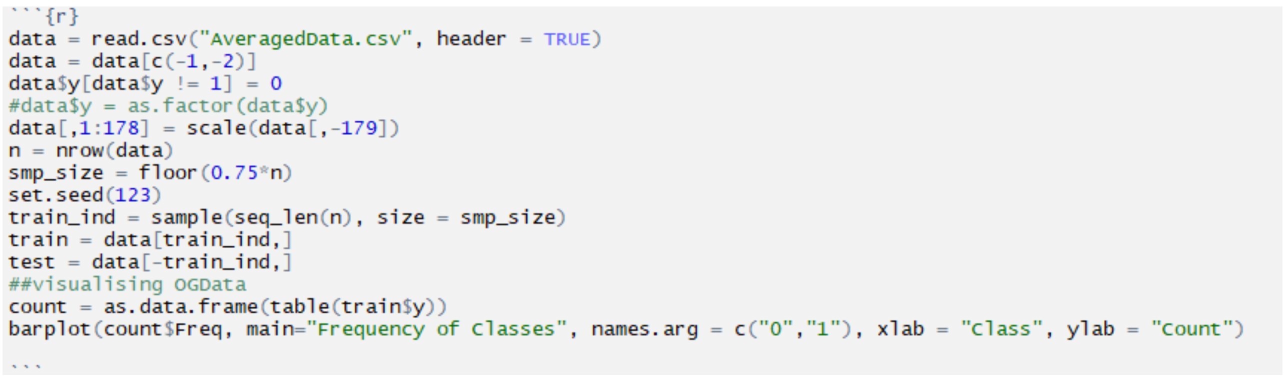
[6]. https://machinelearningmastery.com/linear-discriminant-analysis-for-machine-learning

[7].http://epileptologie-bonn.de/cms/front_content.php?idcat=193&lang=3&changelang=3

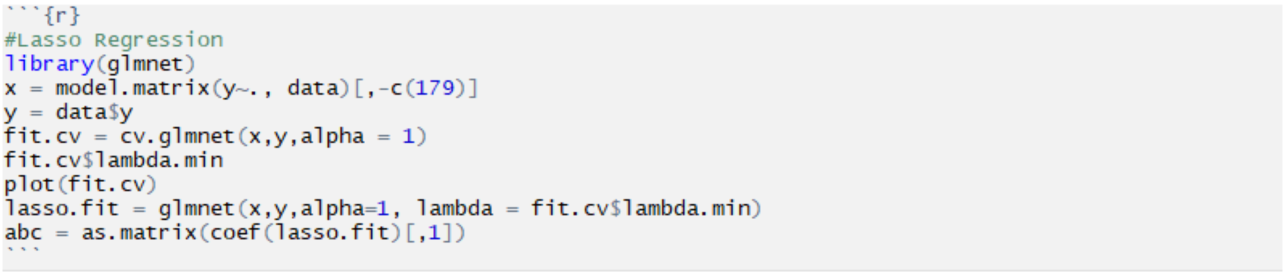
[8]. https://www.rdocumentation.org

[9].Definitions and Notes: https://uc-r.github.io/

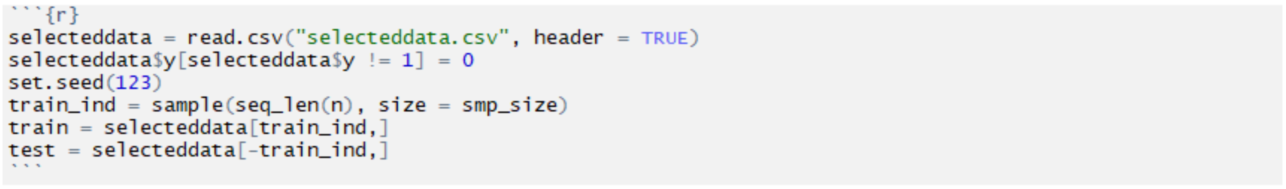
Appendix



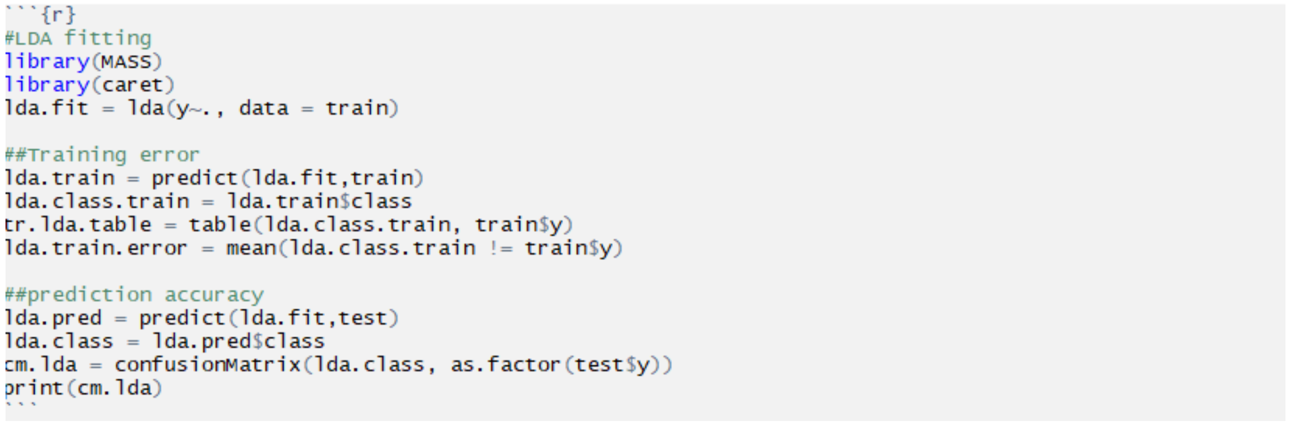
~~~
```{r}
data = read.csv("AveragedData.csv", header = TRUE)
data = data[c(-1,-2)]
data$y[data$y != 1] = 0
#data$y = as.factor(data$y)
data[,1:178] = scale(data[,-179])
n = nrow(data)
smp_size = floor(0.75*n)
set.seed(123)
train_ind = sample(seq_len(n), size = smp_size)
train = data[train_ind,]
test = data[-train_ind,]
##visualising OGData
count = as.data.frame(table(train$y))
barplot(count$Freq, main="Frequency of Classes", names.arg = c("0","1"), xlab = "Class", ylab = "Count")

```
~~~



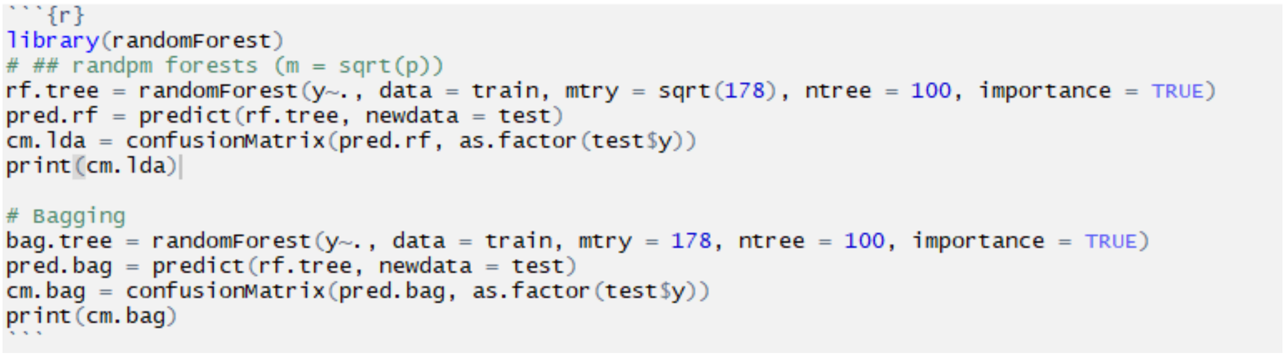
~~~
```{r}
#Lasso Regression
library(glmnet)
x = model.matrix(y~., data)[,-c(179)]
y = data$y
fit.cv = cv.glmnet(x,y,alpha = 1)
fit.cv$lambda.min
plot(fit.cv)
lasso.fit = glmnet(x,y,alpha=1, lambda = fit.cv$lambda.min)
abc = as.matrix(coef(lasso.fit)[,1])
```
~~~



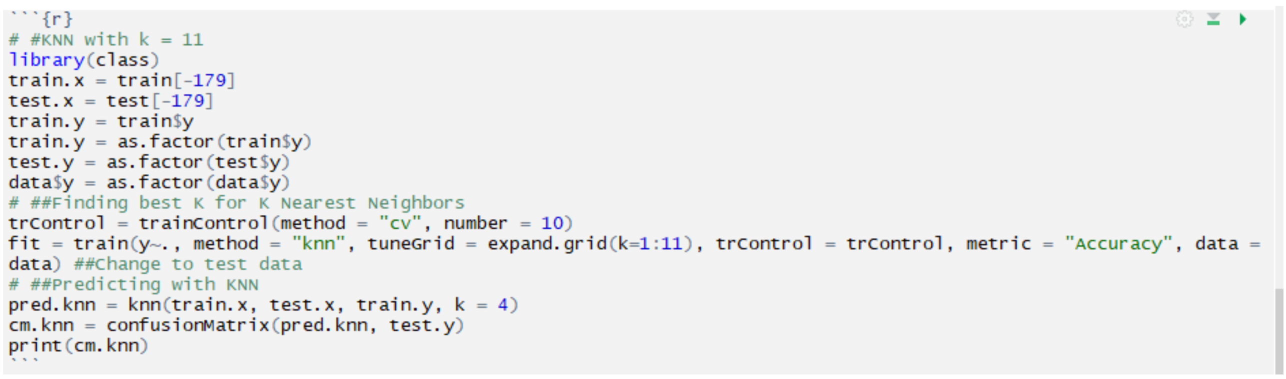
~~~
```{r}
selecteddata = read.csv("selecteddata.csv", header = TRUE)
selecteddata$y[selecteddata$y != 1] = 0
set.seed(123)
train_ind = sample(seq_len(n), size = smp_size)
train = selecteddata[train_ind,]
test = selecteddata[-train_ind,]
```
~~~



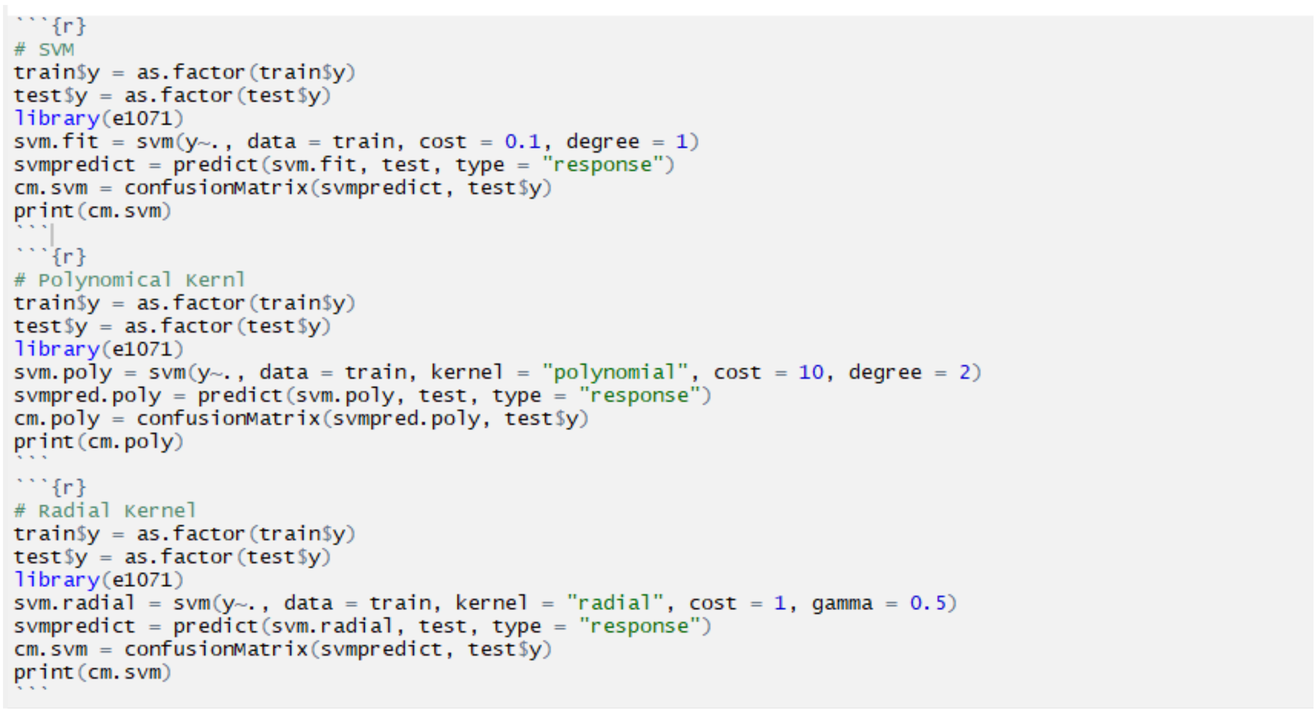
~~~
```{r}
#LDA fitting
library(MASS)
library(caret)
lda.fit = lda(y~., data = train)

##Training error
lda.train = predict(lda.fit,train)
lda.class.train = lda.train$class
tr.lda.table = table(lda.class.train, train$y)
lda.train.error = mean(lda.class.train != train$y)

##prediction accuracy
lda.pred = predict(lda.fit,test)
lda.class = lda.pred$class
cm.lda = confusionMatrix(lda.class, as.factor(test$y))
print(cm.lda)
```
~~~



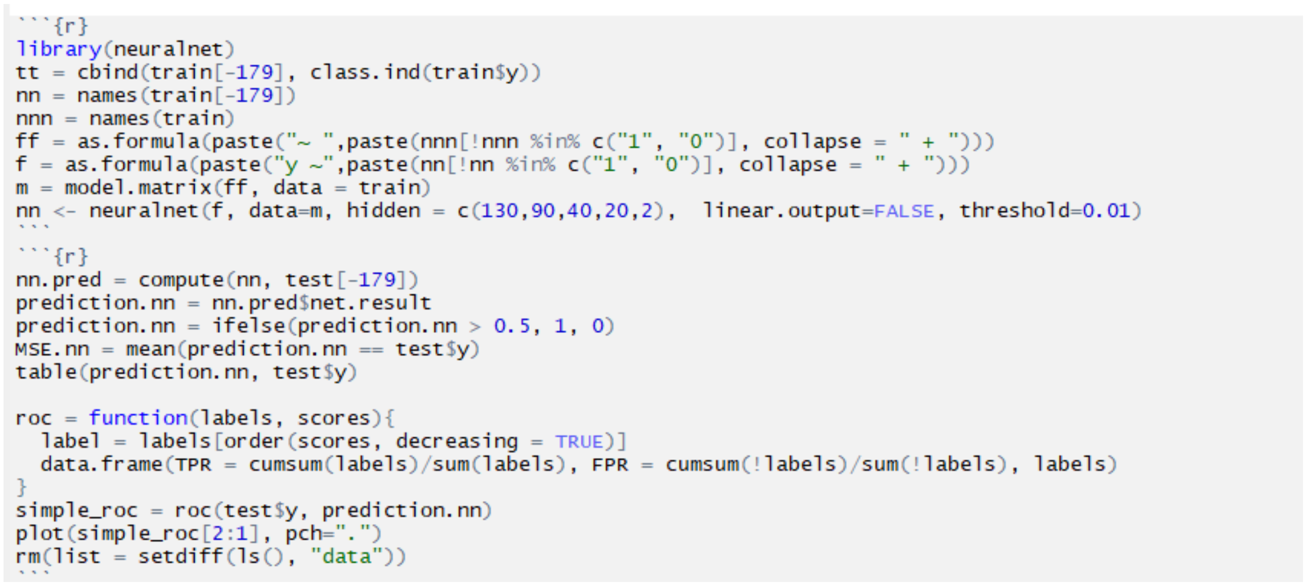
~~~
```{r}
library(randomForest)
# ## randpm forests (m = sqrt(p))
rf.tree = randomForest(y~., data = train, mtry = sqrt(178), ntree = 100, importance = TRUE)
pred.rf = predict(rf.tree, newdata = test)
cm.lda = confusionMatrix(pred.rf, as.factor(test$y))
print(cm.lda)

# Bagging
bag.tree = randomForest(y~., data = train, mtry = 178, ntree = 100, importance = TRUE)
pred.bag = predict(rf.tree, newdata = test)
cm.bag = confusionMatrix(pred.bag, as.factor(test$y))
print(cm.bag)
```
~~~



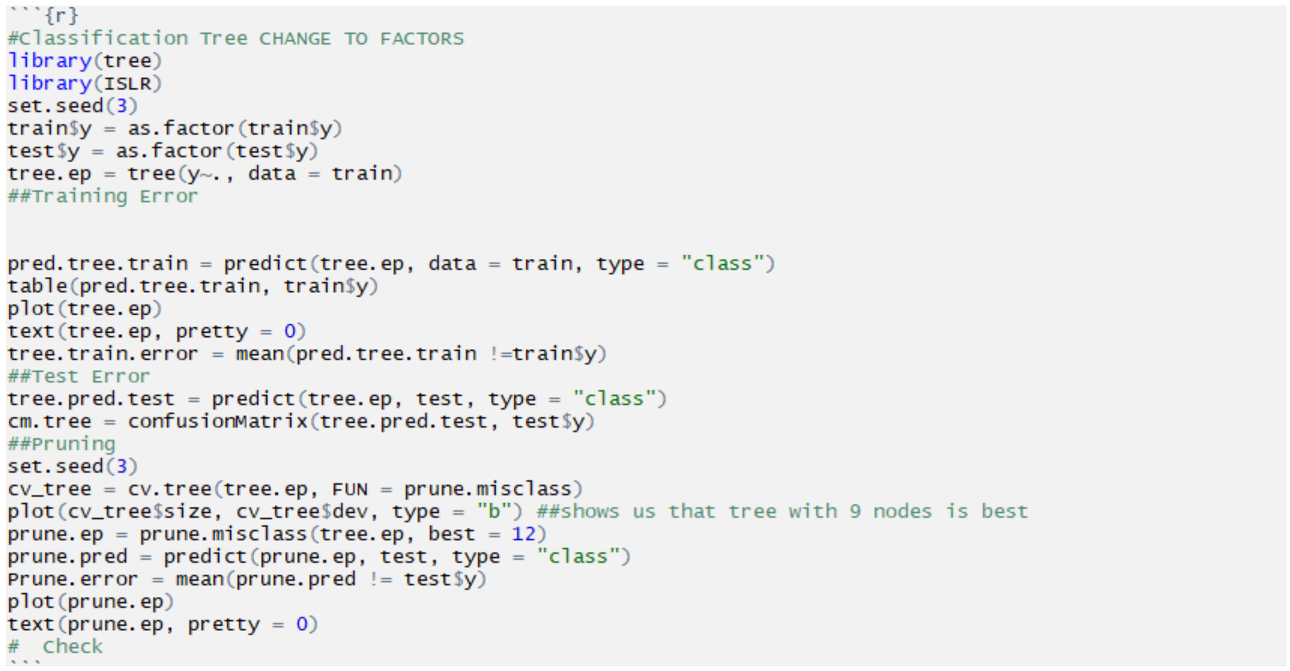
~~~
```{r}
# #KNN with k = 11
library(class)
train.x = train[-179]
test.x = test[-179]
train.y = train$y
train.y = as.factor(train$y)
test.y = as.factor(test$y)
data$y = as.factor(data$y)
# ##Finding best K for K Nearest Neighbors
trControl = trainControl(method = "cv", number = 10)
fit = train(y~., method = "knn", tuneGrid = expand.grid(k=1:11), trControl = trControl, metric = "Accuracy", data =
data) ##Change to test data
# ##Predicting with KNN
pred.knn = knn(train.x, test.x, train.y, k = 4)
cm.knn = confusionMatrix(pred.knn, test.y)
print(cm.knn)
```
~~~



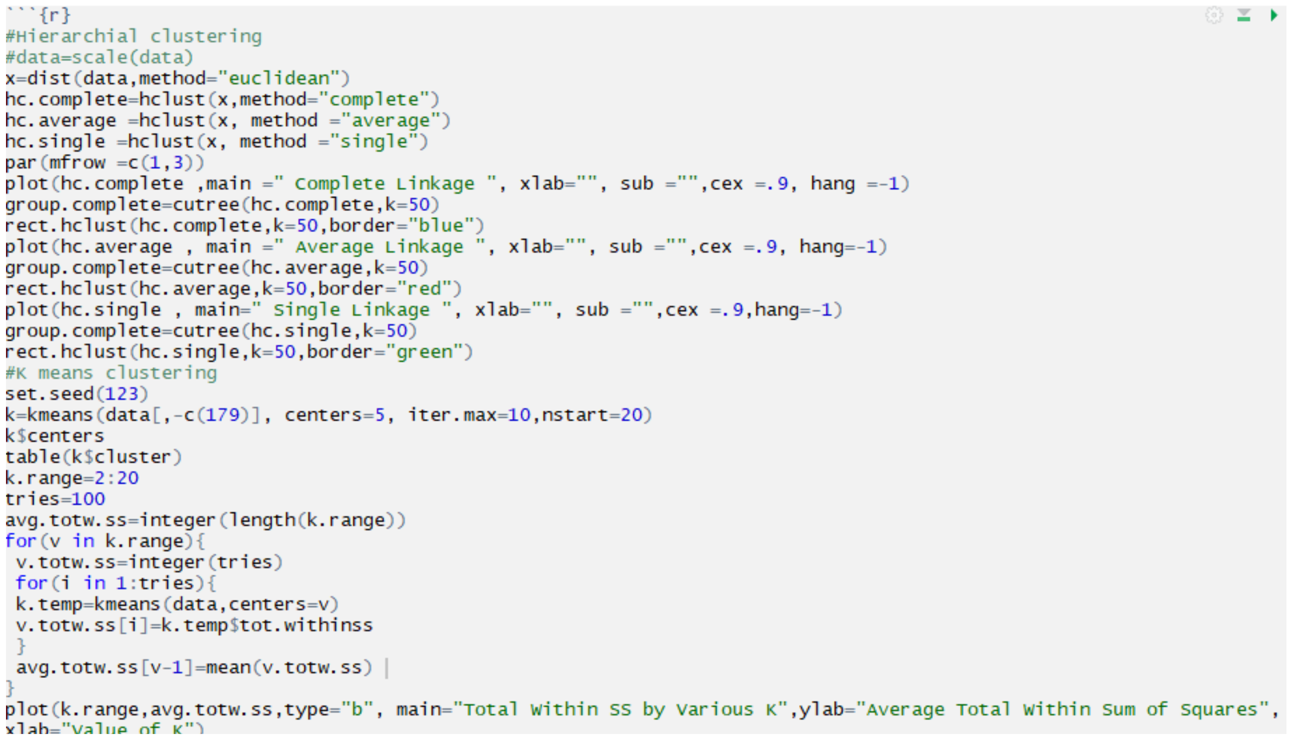
~~~
```{r}
# SVM
train$y = as.factor(train$y)
test$y = as.factor(test$y)
library(e1071)
svm.fit = svm(y~., data = train, cost = 0.1, degree = 1)
svmpredict = predict(svm.fit, test, type = "response")
cm.svm = confusionMatrix(svmpredict, test$y)
print(cm.svm)
```
```{r}
# Polynomical Kernl
train$y = as.factor(train$y)
test$y = as.factor(test$y)
library(e1071)
svm.poly = svm(y~., data = train, kernel = "polynomial", cost = 10, degree = 2)
svmpred.poly = predict(svm.poly, test, type = "response")
cm.poly = confusionMatrix(svmpred.poly, test$y)
print(cm.poly)
```
```{r}
# Radial Kernel
train$y = as.factor(train$y)
test$y = as.factor(test$y)
library(e1071)
svm.radial = svm(y~., data = train, kernel = "radial", cost = 1, gamma = 0.5)
svmpredict = predict(svm.radial, test, type = "response")
cm.svm = confusionMatrix(svmpredict, test$y)
print(cm.svm)
```
~~~



~~~
```{r}
library(neuralnet)
tt = cbind(train[-179], class.ind(train$y))
nn = names(train[-179])
nnn = names(train)
ff = as.formula(paste("~ ",paste(nnn[!nnn %in% c("1", "0")], collapse = " + ")))
f = as.formula(paste("y ~",paste(nn[!nn %in% c("1", "0")], collapse = " + ")))
m = model.matrix(ff, data = train)
nn <- neuralnet(f, data=m, hidden = c(130,90,40,20,2),  linear.output=FALSE, threshold=0.01)
```
```{r}
nn.pred = compute(nn, test[-179])
prediction.nn = nn.pred$net.result
prediction.nn = ifelse(prediction.nn > 0.5, 1, 0)
MSE.nn = mean(prediction.nn == test$y)
table(prediction.nn, test$y)

roc = function(labels, scores){
  label = labels[order(scores, decreasing = TRUE)]
  data.frame(TPR = cumsum(labels)/sum(labels), FPR = cumsum(!labels)/sum(!labels), labels)
}
simple_roc = roc(test$y, prediction.nn)
plot(simple_roc[2:1], pch=".")
rm(list = setdiff(ls(), "data"))
```
~~~



~~~
```{r}
#Classification Tree CHANGE TO FACTORS
library(tree)
library(ISLR)
set.seed(3)
train$y = as.factor(train$y)
test$y = as.factor(test$y)
tree.ep = tree(y~., data = train)
##Training Error


pred.tree.train = predict(tree.ep, data = train, type = "class")
table(pred.tree.train, train$y)
plot(tree.ep)
text(tree.ep, pretty = 0)
tree.train.error = mean(pred.tree.train !=train$y)
##Test Error
tree.pred.test = predict(tree.ep, test, type = "class")
cm.tree = confusionMatrix(tree.pred.test, test$y)
##Pruning
set.seed(3)
cv_tree = cv.tree(tree.ep, FUN = prune.misclass)
plot(cv_tree$size, cv_tree$dev, type = "b") ##shows us that tree with 9 nodes is best
prune.ep = prune.misclass(tree.ep, best = 12)
prune.pred = predict(prune.ep, test, type = "class")
Prune.error = mean(prune.pred != test$y)
plot(prune.ep)
text(prune.ep, pretty = 0)
#  Check
```
~~~



~~~
```{r}
#Hierarchial clustering
#data=scale(data)
x=dist(data,method="euclidean")
hc.complete=hclust(x,method="complete")
hc.average =hclust(x, method ="average")
hc.single =hclust(x, method ="single")
par(mfrow =c(1,3))
plot(hc.complete ,main =" Complete Linkage ", xlab="", sub ="",cex =.9, hang =-1)
group.complete=cutree(hc.complete,k=50)
rect.hclust(hc.complete,k=50,border="blue")
plot(hc.average , main =" Average Linkage ", xlab="", sub ="",cex =.9, hang=-1)
group.complete=cutree(hc.average,k=50)
rect.hclust(hc.average,k=50,border="red")
plot(hc.single , main=" Single Linkage ", xlab="", sub ="",cex =.9,hang=-1)
group.complete=cutree(hc.single,k=50)
rect.hclust(hc.single,k=50,border="green")
#K means clustering
set.seed(123)
k=kmeans(data[,-c(179)], centers=5, iter.max=10,nstart=20)
k$centers
table(k$cluster)
k.range=2:20
tries=100
avg.totw.ss=integer(length(k.range))
for(v in k.range){
 v.totw.ss=integer(tries)
 for(i in 1:tries){
 k.temp=kmeans(data,centers=v)
 v.totw.ss[i]=k.temp$tot.withinss
 }
 avg.totw.ss[v-1]=mean(v.totw.ss)
}
plot(k.range,avg.totw.ss,type="b", main="Total Within SS by Various K",ylab="Average Total Within Sum of Squares",
xlab="Value of K")
~~~