Machine Learning Assignment-1

Chandana Dasari (18231267)

1. Why R ?

Among the many available open source Machine Learning packages i have chosen "R" for this particular Assignment. My Reasons for chosing R includes :

1. R's main focus is on statistic analysis of data which is most imporatant for the early stages of Machine learners to get a good knowledge on the data first.

2. It includes a treasure of libraries and packages for Data related projects. For example : caret, caTools, class, gmodels, ggplot.

3. R is very simple to use and understand. It is readable and concise. R's tools and techniques made data manipulations required for Preprocessing easy.

5. R packages includes the Pre-Modeling, Modeling, and Post-Modeling stages and can also be used for model validation and data visualisation.

R is one among the most competative tools for Data Analysis.

2. Data Preprocessing

Pre-Processing is an important step in any model building, it includes the data cleaning, transformation, dealing with NA's, Feature extraction and selection,.

For this assignment - The supplied dataset observations are seperated by tab and features are seperated by new lines. And there are no Labels assigned to the data.

Pre processing steps carried out for this data is :

1. Reading the data using read.csv by using the separator as tab

2.Transposing the data to get the features column wise.

3. Assigning the proper feature names as lables to the data

4.Transforming the data to a data frame for the convinience

After the above preprocessing steps the the data is in a nice tabular format - where each row is an observation and each column indicates a feature.

Now this data can be directly supplied to Machine Learning packages to process.

```
# Importing data
data_cleaning1 <- read.csv(file = "text.txt", header = FALSE, sep = "\t", dec
= ".")
```

```r
# trasnposing and column naming the data as part of preprocessing the data
data <- t(data_cleaning1)
colnames(data) <- c("Age", "Blood_Pressure", "BMI", "Plasma_level",
"Autoimmune_Disease", "Adverse_events", "Drug_in_serum", "Liver_function",
"Activity_test", "Secondary_test")

# Converting data into data frame for convinience
Data <- data.frame(data, stringsAsFactors = F)
Data <- Data[,c("Autoimmune_Disease","Age", "Blood_Pressure", "BMI",
"Plasma_level", "Adverse_events", "Drug_in_serum", "Liver_function",
"Activity_test", "Secondary_test")]

#converting data to num type
Data$Autoimmune_Disease <- ifelse(Data$Autoimmune_Disease == "positive", 1,
0)
Data <- as.data.frame(sapply(Data, as.numeric))
```

Data Normalization :

1. Normalization is a data preparation technique which brings all the features to same scale without losing the information.

2. The insight behind Normalization is all features are equally important with respect to the output and one feature should not dominate over other features. for Example: If only some features has broad values(i.e High variance), this feature highly effects the model performance and the effect of other featres will be ignored. To avoid this problem and to give equal imporatnce to all the features we apply Scaling/Normalization on our data.

3. Scaling improves interpretability of the results. 4.This helps in modelling the data correctly.

I have applied 0/1 or Min-Max Normalization on this data.

A Min-Max Normalization can be done using the following equation:

$X_{norm} = (X-X_{min})/(X_{max}-X_{min})$

In this approach, the data is scaled to a fixed range - 0 to 1.

```r
library("class")
library("caret")

## Loading required package: lattice

## Loading required package: ggplot2

library("caTools")
# Normalizing the data - Min-Max(0/1) Normalization
Normalize <- function(x){
  return ((x-min(x,na.rm = T))/(max(x,na.rm = T)-min(x, na.rm = T)))
}
```

```
Data_Norm <- apply(Data[ ,-1],2,Normalize)
Data_N <- Data
Data_N[ ,2:10] = Data_Norm
```

Traditional Data split includes test and train data - in which Train data is used to train the model and testing the performance of the model using the test data. Presence of any outliers highly impacts this traditional Train-test split. The train test split goes as follows.

That is the reason we used the Cross validation method. Which briefly explained below

```
indices <- createDataPartition(y = Data$Autoimmune_Disease,p = .75,list =
FALSE)
training <- Data[ indices,]
testing  <- Data[-indices,]
```

3.Classification Models

I have chosen K-Nearest Neighours(knn) and Support Vector Machines (svm) for my analysis.

What is K Nearest Neighbour ?

Knn works with the insight of If two samples are close to each other in space, they should be close to each other in general.

Knn is a lazy learning algorithm - when the training data is fed to the system, It stores all the independent features and lables in 'n' dimensional space and when it faces test data it stores this test data in the same n-dimensional space as the training data and searches for k nearest neighbours based on the distance measures. (Ex: Euclidean, Manhattan, chebychev) For each test observation it goes through entire training set to predict the label. That is the reason it is called Lazy learning algorithm. After we get K nearest neighbors labels, we take simple majority of these K nearest neighbours and assign it's label to test observation.

What is Support Vector Machine ?

Support Vector Machine(SVM) is a supervised machine learning algorithm which can be used for both classification or regression problems. SVM is a generalization of Maximum margin classifier. Here we are using svm as an Classification Algorithm. In this algorithm, we plot each data item as a point in n-dimensional space (where n is the number of features) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the optimal seperating hyper-plane that seperates the different classes in the training data. It works well with the clear margin of seperation and effective in high dimensional spaces

10-fold Cross Validation

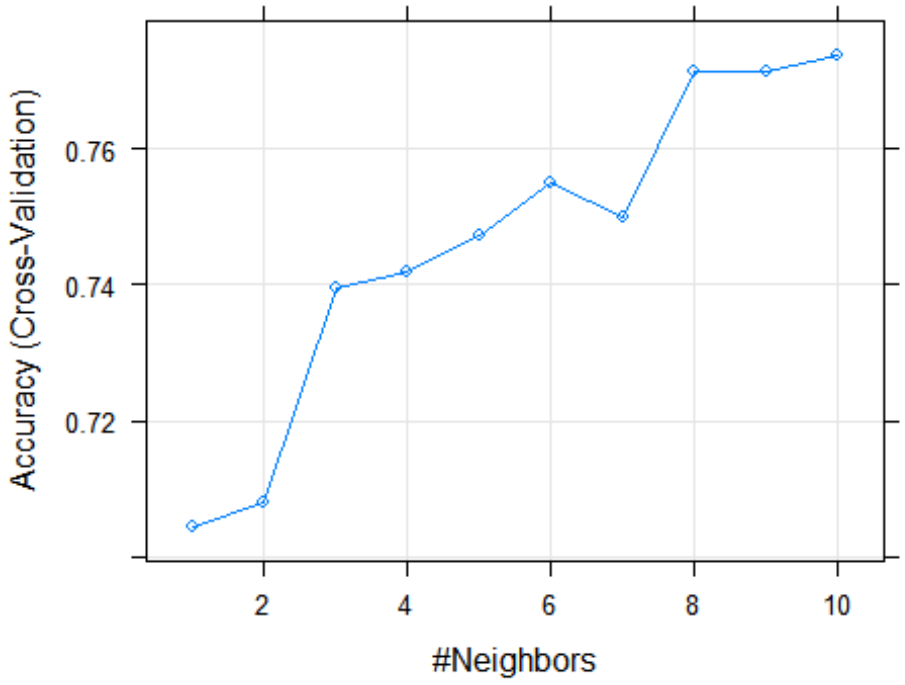
Cross-validation is a technique to estimate the performance of a machine learning model. compared to the traditional train-test split which get effected highly with the bias in the data, presence of outlier and out of sample error effect. On other hand Cross validation overcomes the effect of Data Bias and huge effect of out of sample error by creating

multiple train and test sets (from the k-folds). This method is used for estimating the out of sample error for the model.

This procedure has a single parameter 'k' that refers to the number of groups/folds that a given data sample is to be split into. if k = 10, it will be 10 fold cross validation. The general procedure for k-fold cross validation is as follows: 1.Shuffle the dataset randomly. 2.Split the dataset into k groups 3.For each unique group: Take the group as a hold out or test data set Take the remaining groups as a training data set 4.Fit a model on the training set and evaluate it on the test set 5.Get the evaluation score and discard the model

10 fold cross validation implementation for Knn

```
#Transforming the dependent variables as factors
Data_N$Autoimmune_Disease = as.factor(Data_N$Autoimmune_Disease)
levels(Data_N$Autoimmune_Disease) <-
make.names(levels(factor(Data_N$Autoimmune_Disease)))
ControlParameters <- trainControl(method  = "cv", number  = 10,
                          savePredictions = TRUE, classProbs = T)
Model_knn <- train(Autoimmune_Disease ~ ., data       = Data_N,
             method     = "knn", tuneGrid   = expand.grid(k = 1:10),
             trControl  = ControlParameters, metric     = "Accuracy")
plot(Model_knn)
```



The accuracy for knn = 77.1% with k(number of neighbours) value 9.

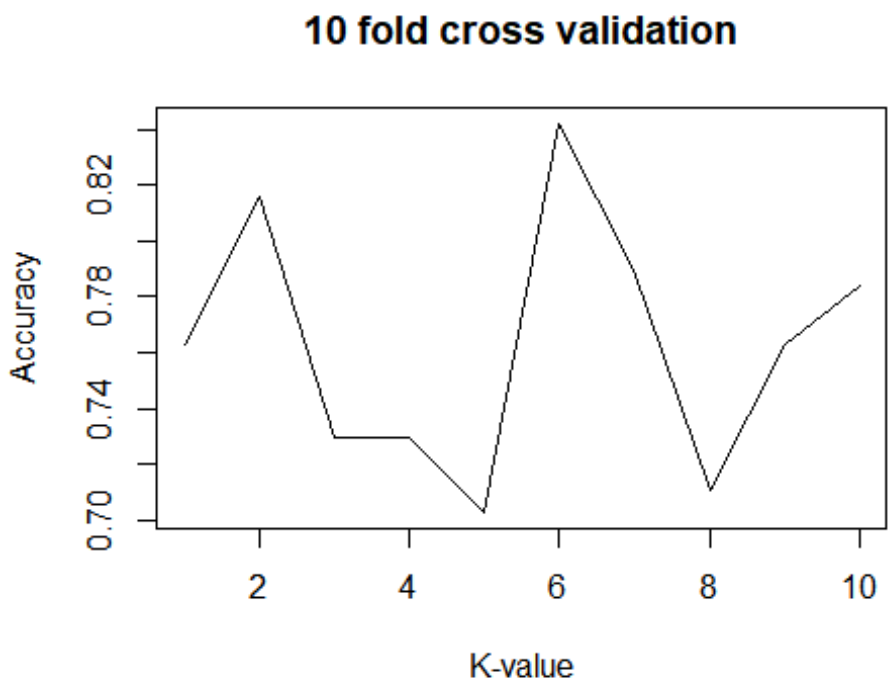
10 fold cross validation implementation for SVM

```
library(e1071)
folds = createFolds(Data_N$Autoimmune_Disease, k = 10)

cv_svm = lapply(folds,function(x){
  training_fold = Data_N[-x, ]
  test_fold = Data_N[x, ]
  classifier_svm = svm( formula = Autoimmune_Disease ~. , data =
training_fold,
                        type = 'C-classification', kernal = 'radial')
  y_pred = predict(classifier_svm, newdata = test_fold[-1])
  cm= table(test_fold[,1], y_pred)
accuracy = (cm[1,1] + cm[2,2])/(cm[1,1]+cm[1,2]+cm[2,1]+cm[2,2])
return(accuracy)
    })
accuracy_svm = mean(as.numeric(cv_svm))
print(paste("Average Accuracy :", accuracy_svm))

## [1] "Average Accuracy : 0.763015647226174"

plot(c(1:10),cv_svm, type = "l", xlab = "K-value",ylab = "Accuracy" ,main
="10 fold cross validation")
```



Accuracy for Svm using 10- fold cross validation is 76.57%

4.Results

The accuracies for the above two classifiers by using 10-fold cross validation is stated below Knn classifiers accuracy = 77.1% SVM classifiers accuracy = 76.57%

There is not much difference with respect to the performance of these classifiers on this particluar data. Comparatively Knn has performed better on this data. As the data has only 376 samples and 9 features(Independent), we can't notice the huge difference.

References

1. Machine learning Studio Module Reference
2. R for Data Science By Hadley Wickham
3. https://www.datacamp.com/
4. Lecture Notes
5. https://medium.com/data-science-group-iitr