Final_STAA556

R code Appendix

6/30/2019

```
knitr::opts_chunk$set(echo = TRUE)
set.seed(143)
library(readxl)
library(tidyverse)
```

```
## Registered S3 methods overwritten by 'ggplot2':
##   method         from
##   [.quosures     rlang
##   c.quosures     rlang
##   print.quosures rlang
```

```
## -- Attaching packages ---------------------------------------------------------------------------- tidy
```

```
## v ggplot2 3.1.1     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.1
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## -- Conflicts ------------------------------------------------------------------------------- tidyverse_
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(ellipsis)
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
library(e1071)
library(ggplot2)
library(gridExtra)
```

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

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

```
library(caTools)
library(FactoMineR)
library(Hotelling)
```

```
## Loading required package: corpcor
```

```
library(car)
```

```
## Loading required package: carData
```

```
##
## Attaching package: 'car'
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```
## The following object is masked from 'package:purrr':
##
##     some
```

```
library(corrplot)
data<- read_excel("PVYREIMS_data.xlsx")
# Check the structure of data frame
sum(is.na(data))
```

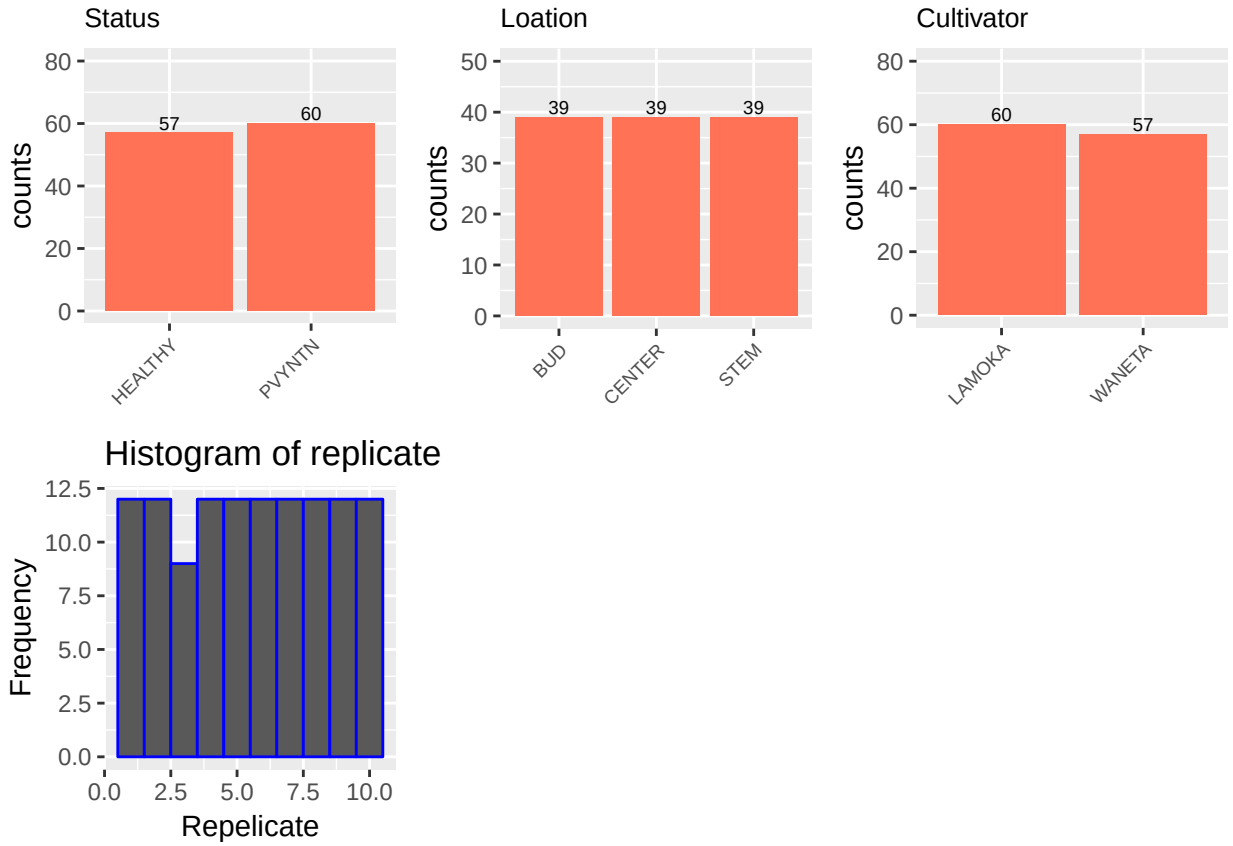
```
## [1] 0
```

```
# load the data
#checking dimensions of data
dim(data)
```

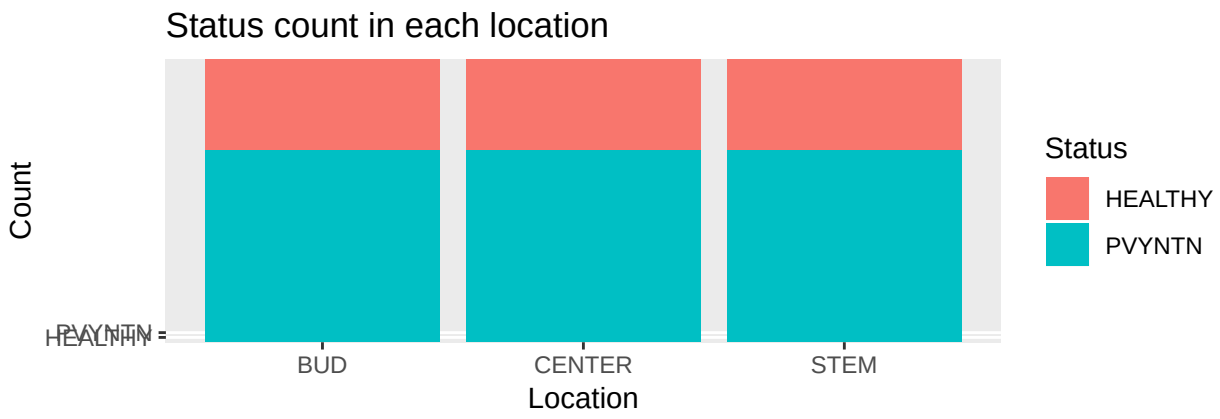
```
## [1] 117 907
```

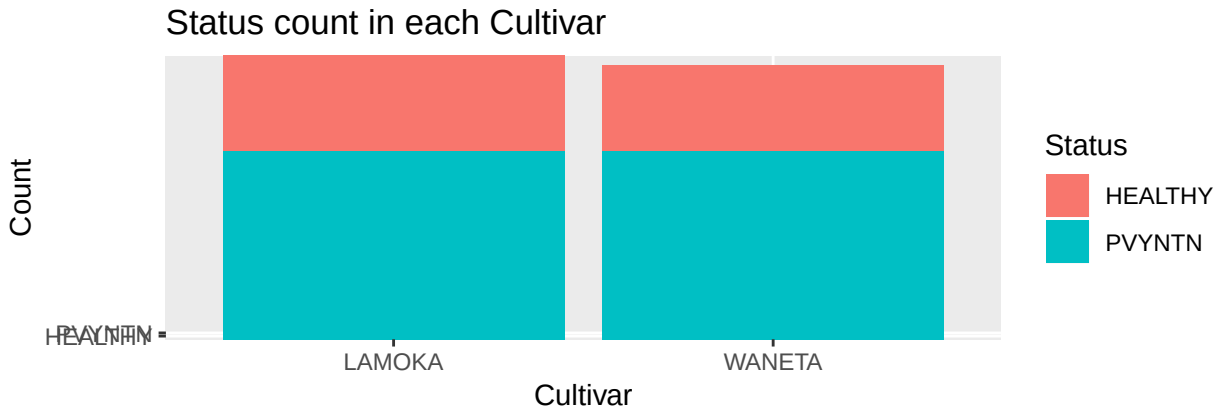
```
#The first step in any machine learning workflows is exploratory data analysis (EDA)
# Exploratory Data Analysis

p1 <- data %>%group_by(Status) %>%
     summarise(counts = n()) %>%
  ggplot(aes(x = as.factor(Status), y = counts)) +
  geom_bar(stat = 'identity',
  fill = "coral1")+ ggtitle("Status") +
  geom_text(aes(label=counts), size = 2.5,
    position=position_dodge(width=0.2),
vjust=-0.25) + theme(plot.title = element_text(size =10),
                     axis.text.x = element_text(size =7,
angle = 45, hjust = 1),axis.title.x=element_blank()) +
  scale_y_continuous(limits = c(0,80))

p2<-data %>% group_by(Location) %>% summarise(counts = n()) %>%
  ggplot(aes(x = as.factor(Location), y = counts)) + geom_bar(stat = 'identity',
  fill = "coral1")+ ggtitle("Loation") +
  geom_text(aes(label=counts), size = 2.5,
  position=position_dodge(width=0.2),vjust=-0.25) +
  theme(plot.title = element_text(size =10),
        axis.text.x = element_text(size =7,
angle = 45, hjust = 1),axis.title.x=element_blank()) +
  scale_y_continuous(limits = c(0,50))

p3<-data %>% group_by(Cultivar) %>% summarise(counts = n()) %>%
  ggplot(aes(x = as.factor(Cultivar), y = counts)) + geom_bar(stat = 'identity',
  fill = "coral1")+
```

```
  ggtitle("Cultivator") +geom_text(aes(label=counts), size = 2.5,
  position=position_dodge(width=0.2), vjust=-0.25) +
  theme(plot.title = element_text(size =10),axis.text.x = element_text(size =7,
angle = 45, hjust = 1),axis.title.x=element_blank()) +
  scale_y_continuous(limits = c(0,80))

p4<-ggplot(data=data, aes(x=data$Replicate)) + geom_histogram(binwidth = 1,
color="blue") + labs(title="Histogram of replicate",x= "Repelicate", y= "Frequency")

grid.arrange(p1, p2, p3,p4,nrow = 2, ncol = 3)
```

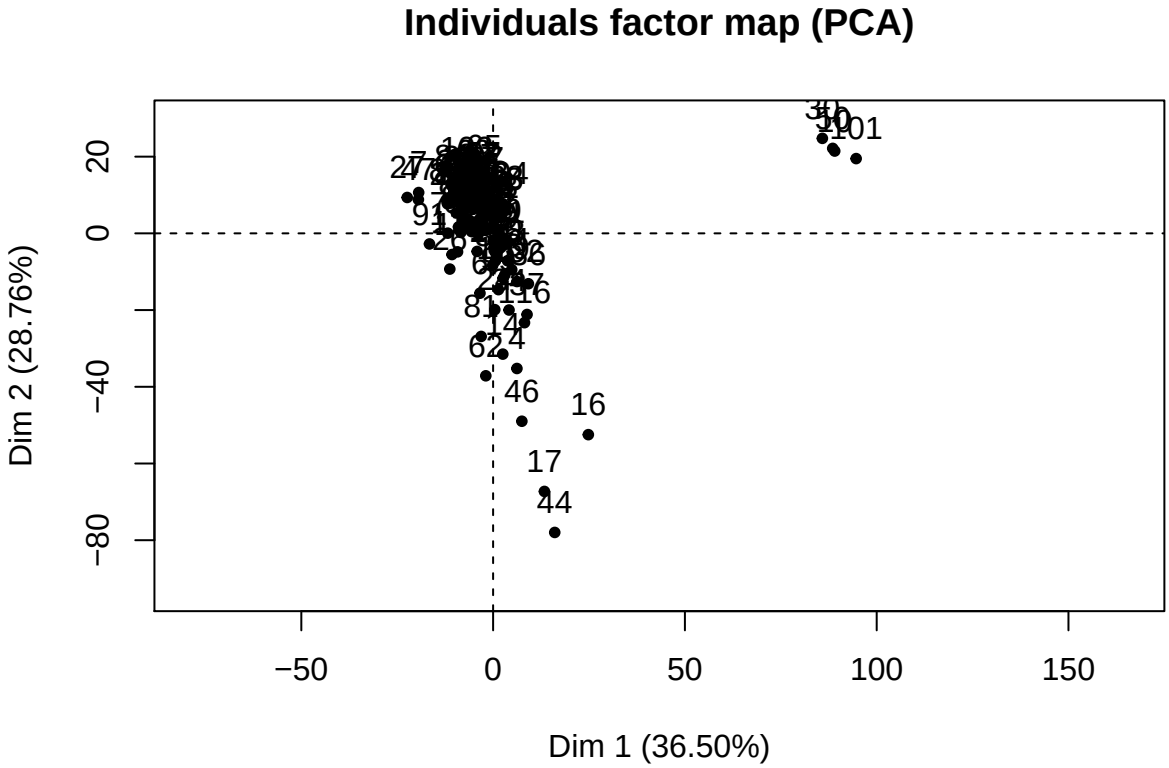


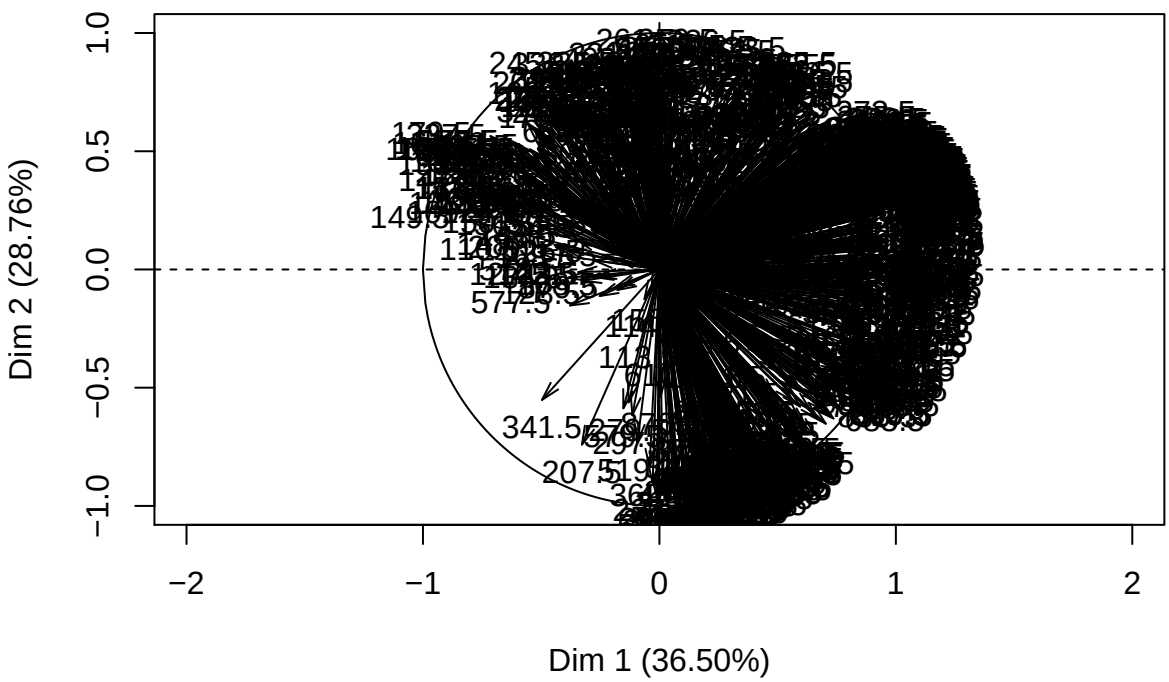

```
p5<-ggplot(data =data,mapping = aes(x =Location,y=Status,fill = Status)) +
  geom_col() +labs(x = "Location", y = "Count",title = "Status count in each location")

p6<-ggplot(data =data,mapping = aes(x =Cultivar,y=Status,fill = Status)) +
  geom_col() +labs(x = "Cultivar", y = "Count",title = "Status count in each Cultivar")
grid.arrange(p5,p6)
```



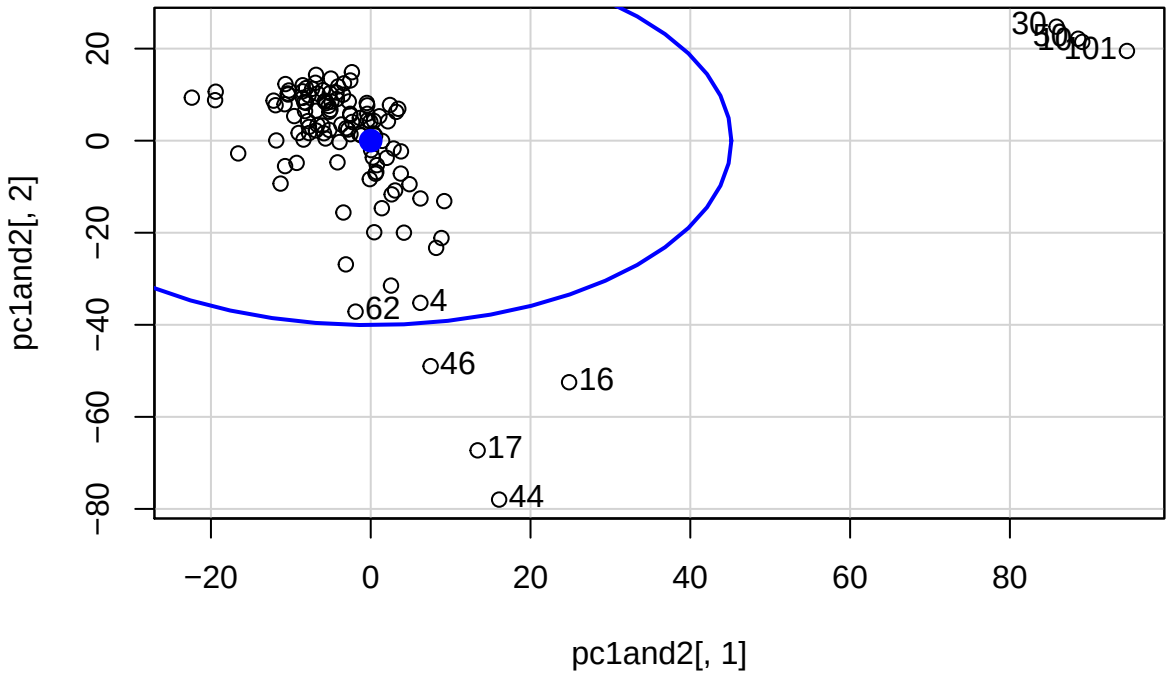



```
# Inspect correlation matrix
#datMy.scale<- scale(data[8:ncol(data)],center=TRUE,scale=TRUE);
#scale all the features (from feature 2 bacause feature 1 is the predictor output)

#corMatMy <- cor(datMy.scale)
#compute the correlation matrix

#corrplot(corMatMy, order = "hclust")
#visualize the matrix, clustering features by correlation index.
pca<-PCA(data[8:907], scale.unit = TRUE, ncp = 900)
```






```
pc1and2<- pca$ind$coord[,1:2] #first and second principal components

dataEllipse(pc1and2[,1], pc1and2[,2], levels = .95, id=list(method="mahal", n=10,
        cex=1, col=carPalette()[1], location="lr"))
```




```
# plot pc1 and 2 with the hotelling 95% confidence region

#Remove observations outside the 95% confidence region: 10, 16, 17, 30, 44, 46, 50, 101
dataset1 <- data[-c(10,16,17,30,44,46,50),]

pcaReduced <- PCA(dataset1[8:339], scale.unit = TRUE, ncp=500, graph=FALSE)

# take to remove lineID,sampleID and file column from the data
dataset<-subset(dataset1,select = -c(1:3))

# Encoding categorical data
dataset$Location = factor(dataset$Location,levels = c('BUD', 'CENTER', 'STEM'),labels = c(1, 2, 3))
dataset$Status = factor(dataset$Status,levels = c('HEALTHY', 'PVYNTN'),labels = c(0, 1))
dataset$Cultivar=factor(dataset$Cultivar,levels = c("LAMOKA","WANETA"),labels = c(0,1))

#specifying outcome variable as factor
dataset$Status<-as.factor(dataset$Status)

index <- createDataPartition(dataset$Status, p = 0.8, list = FALSE)
train_data <- dataset[index, ]
test_data  <- dataset[-index, ]
table(train_data$Status)
```

```
##
##  0  1
## 42 47
```

```
table(test_data$Status)
```

```
##
##  0  1
## 10 11
```

```r
train_d<-subset(train_data,select = -c(1,2))
test_d<-subset(test_data,select = -c(1,2))
target <- 'Status'
predictors <- setdiff(names(train_d), target)
# normalize data
train_nom <- preProcess(train_d[ ,predictors], method=c("center", "scale"))
train_scaled <- predict(train_nom, train_d)
train_scaled$Status <- train_d$Status

#After inspecting the matrix we will set the correlation threshold at 0.90.
# Remove highly correlated predictors
train_corr <- cor(train_scaled[ ,predictors])
train_high_corr_v <- findCorrelation(train_corr, cutoff=.90)
train_low_corr <- train_scaled[ ,-c(train_high_corr_v)]
train_cleaned <- train_low_corr
rm(train_low_corr)
print(paste0("Number of more than 90% correlated columns : ", length(train_high_corr_v)))
```

```
## [1] "Number of more than 90% correlated columns : 421"
```

```r
dim(train_cleaned)
```

```
## [1]  89 481
```

```r
# normalize data
test_cleaned <- predict(train_nom, test_d)
test_cleaned <- test_cleaned[,names(train_cleaned)]
# 5 fold cross validation
ctrl <-trainControl(method="cv",number=5)
```

#Location

```r
#
# take to remove lineID,sampleID and file column from the data
dataset2<-subset(dataset1,select = -c(1:4))

# Encoding categorical data
dataset2$Status = factor(dataset2$Status,levels = c("HEALTHY", "PVYNTN"),labels = c(0, 1))

#specifying outcome variable as factor
center_stem <- subset(dataset2,dataset2$Location=="CENTER" | dataset2$Location=="STEM")

bud_stem <- subset(dataset2,dataset2$Location=="BUD" | dataset2$Location=="STEM")

bud_center <- subset(dataset2,dataset2$Location=="BUD" | dataset2$Location=="CENTER")

# 5 fold cross validation
ctrl <-trainControl(method="cv",number=5)
```

#(i) bud/stem

```r
# Effect of location on prediction

index1 <- createDataPartition(bud_stem$Status, p = 0.85, list = FALSE)
train_bs1 <- bud_stem[index1, ]
test_bs1  <- bud_stem[-index1, ]
target <- 'Status'
```

```
train_bs<-subset(train_bs1,select = -c(1,3))
test_bs<-subset(test_bs1,select=-c(1,3))
predictors <- setdiff(names(train_bs), target)
# normalize data
train_nom_bs <- preProcess(train_bs[ ,predictors], method = c("center", "scale"))
train_scaled_bs <- predict(train_nom_bs, train_bs)
train_scaled_bs$Status <- train_bs$Status

#After inspecting the matrix we will set the correlation threshold at 0.90.
# Remove highly correlated predictors
train_corr <- cor(train_scaled_bs[ ,predictors])
train_high_corr_v <- findCorrelation(train_corr, cutoff = .90)
train_cleaned_bs <- train_scaled_bs[ , -c(train_high_corr_v)]

print(paste0("Number of more than 90% correlated columns : ", length(train_high_corr_v)))
```

```
## [1] "Number of more than 90% correlated columns : 614"
```

```
dim(train_cleaned_bs)
```

```
## [1]  62 287
```

```
# normalize data
test_cleaned_bs <- predict(train_nom_bs, test_bs)
test_cleaned_bs <- test_cleaned_bs[ ,names(train_cleaned_bs)]
# 5 fold cross validation
ctrl <-trainControl(method="cv",number=5)
```

(b)bs training

```
# SVM model linear
fit.linear.full<-caret::train(Status~., data = train_cleaned_bs, method = "svmLinear",
                              metric="Accuracy", trControl=ctrl)
# Predict test data set
y_svmlinear = predict(fit.linear.full, newdata = test_cleaned_bs[-1])
cm_linear.b<-confusionMatrix(y_svmlinear,test_cleaned_bs$Status)
cm_linear.b
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction 0 1
##          0 2 1
##          1 3 4
##
##                Accuracy : 0.6
##                  95% CI : (0.2624, 0.8784)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.3770
##
##                   Kappa : 0.2
##
##  Mcnemar's Test P-Value : 0.6171
```

```
##
##             Sensitivity : 0.4000
##             Specificity : 0.8000
##          Pos Pred Value : 0.6667
##          Neg Pred Value : 0.5714
##              Prevalence : 0.5000
##          Detection Rate : 0.2000
##    Detection Prevalence : 0.3000
##       Balanced Accuracy : 0.6000
##
##        'Positive' Class : 0
##
```

```
 cm.linear.b<-round( cm_linear.b$overall["Accuracy"], 3)
# SVM model Radial

fit.radial.full<-caret::train(Status~.,data=train_cleaned_bs,method = "svmRadial",
      metric="Accuracy",trControl=ctrl)
# Predict test data set
y_svmRedial = predict(fit.radial.full, newdata = test_cleaned_bs[-1])
cm.radial.b<-round(confusionMatrix(y_svmRedial,test_cleaned_bs$Status)$overall["Accuracy"],3)

# svm Poly model
fit.poly.full<- caret::train(Status~.,data=train_cleaned_bs,
        method = "svmPoly",metric="Accuracy",trControl=ctrl)
# Predict test data set
y_svmPoly = predict(fit.poly.full, newdata = test_cleaned_bs[-1])
cm.poly.b<-round(confusionMatrix(y_svmPoly,test_cleaned_bs$Status)$overall["Accuracy"],3)

# LogitBoost model
fit.logit.full<- caret::train(Status~.,data=train_cleaned_bs,
      method = "LogitBoost",metric="Accuracy",trControl=ctrl)
# Predict test data set
y_logit = predict(fit.logit.full, newdata = test_cleaned_bs[-1])
cm.logit.b<-round(confusionMatrix(y_logit,test_cleaned_bs$Status)$overall["Accuracy"],3)

# Random forest model
fit.rf.full<- caret::train(Status~.,data=train_cleaned_bs,method = "rf",
                           metric="Accuracy",trControl=ctrl)
# Predict test data set
y_rf = predict(fit.rf.full, newdata = test_cleaned_bs[-1])
cm.rf.b<-round(confusionMatrix(y_rf,test_cleaned_bs$Status)$overall["Accuracy"],3)

# LDA model
fit.lda.full <- caret::train(Status~.,data=train_cleaned_bs, method = "lda",
                             metric="Accuracy", trControl=ctrl)
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
# Predict test data set
y_lda = predict(fit.lda.full, newdata = test_cleaned_bs[-1])
cm_lda<-confusionMatrix(y_lda,test_cleaned_bs$Status)
cm_lda
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction 0 1
##          0 2 2
##          1 3 3
##
##                Accuracy : 0.5
##                  95% CI : (0.1871, 0.8129)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.623
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : 1.000
##
##             Sensitivity : 0.4
##             Specificity : 0.6
##          Pos Pred Value : 0.5
##          Neg Pred Value : 0.5
##              Prevalence : 0.5
##          Detection Rate : 0.2
##    Detection Prevalence : 0.4
##       Balanced Accuracy : 0.5
##
##        'Positive' Class : 0
##
```

```
cm.lda.b<-round(cm_lda$overall["Accuracy"],3)
#XgBoost Model
fit.xgboost.full<-caret::train(Status~.,data=train_cleaned_bs,
              method = "xgbTree",metric="Accuracy",trControl=ctrl)
# Predict test data set
y_xgboost = predict(fit.xgboost.full, newdata = test_cleaned_bs[-1])
cm.xgb.b<-round(confusionMatrix(y_xgboost,test_cleaned_bs$Status)$overall["Accuracy"],3)
```

#(ii)bud/center

```
index2 <- createDataPartition(bud_center$Status, p = 0.85, list = FALSE)
train_bc1 <- bud_center[index2, ]
test_bc1  <- bud_center[-index2, ]
table(train_bc1$Status)
```

```
##
##  0  1
## 31 33
```

```
table(test_bc1$Status)
```

```
##
## 0 1
## 5 5
```

```
train_bc <- subset(train_bc1,select = -c(1,3))
test_bc <- subset(test_bc1,select = -c(1,3))
target <- 'Status'
predictors <- setdiff(names(train_bc), target)
# normalize data
train_nom <- preProcess(train_bc[ ,predictors], method = c("center", "scale"))
train_scaled <- predict(train_nom, train_bc)
train_scaled$Status <- train_bc$Status

#After inspecting the matrix we will set the correlation threshold at 0.90.
# Remove highly correlated predictors
train_corr <- cor(train_scaled[ ,predictors])
train_high_corr_bc <- findCorrelation(train_corr, cutoff = .90)
train_low_corr <- train_scaled[ ,-c(train_high_corr_bc)]
train_cleaned_bc <- train_low_corr
train_cleaned_bc$Status <- train_bc$Status
rm(train_low_corr)
print(paste0("Number of more than 90% correlated columns : ", length(train_high_corr_bc)))
```

```
## [1] "Number of more than 90% correlated columns : 448"
```

```
# normalize data
test_cleaned_bc <- predict(train_nom, test_bc)
test_cleaned_bc <- test_cleaned_bc[ , names(train_cleaned_bc)]
test_cleaned_bc$Status <- test_bc$Status
```

(b)Training BC

```
# SVM model linear
fit.linear.full<-caret::train(Status~., data = train_cleaned_bc, method = "svmLinear",
                              metric="Accuracy", trControl = ctrl)
# Predict test data set
y_svmlinear = predict(fit.linear.full, newdata = test_cleaned_bc)
cm_linear.c<-confusionMatrix(y_svmlinear, test_cleaned_bc$Status)
cm_linear.c
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction 0 1
##          0 5 1
##          1 0 4
##
##                Accuracy : 0.9
##                  95% CI : (0.555, 0.9975)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.01074
```

```
##
##                  Kappa : 0.8
##
##  Mcnemar's Test P-Value : 1.00000
##
##             Sensitivity : 1.0000
##             Specificity : 0.8000
##          Pos Pred Value : 0.8333
##          Neg Pred Value : 1.0000
##              Prevalence : 0.5000
##          Detection Rate : 0.5000
##    Detection Prevalence : 0.6000
##       Balanced Accuracy : 0.9000
##
##        'Positive' Class : 0
##
```

```
cm.linear.c<-round(cm_linear.c$overall["Accuracy"], 3)
# SVM model Radial

fit.radial.full<-caret::train(Status~.,data=train_cleaned_bc,method = "svmRadial",
           metric="Accuracy",trControl=ctrl)
# Predict test data set
y_svmRedial = predict(fit.radial.full, newdata = test_cleaned_bc)
cm_radial<-confusionMatrix(y_svmRedial,test_cleaned_bc$Status)
cm_radial
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction 0 1
##          0 4 2
##          1 1 3
##
##                Accuracy : 0.7
##                  95% CI : (0.3475, 0.9333)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.1719
##
##                   Kappa : 0.4
##
##  Mcnemar's Test P-Value : 1.0000
##
##             Sensitivity : 0.8000
##             Specificity : 0.6000
##          Pos Pred Value : 0.6667
##          Neg Pred Value : 0.7500
##              Prevalence : 0.5000
##          Detection Rate : 0.4000
##    Detection Prevalence : 0.6000
##       Balanced Accuracy : 0.7000
##
##        'Positive' Class : 0
##
```

```
cm.radial.c<-round(cm_radial$overall["Accuracy"],3)
# svm Poly model
fit.poly.full<- caret::train(Status~.,data=train_cleaned_bc,method = "svmPoly",
             metric="Accuracy",trControl=ctrl)
# Predict test data set
y_svmPoly = predict(fit.poly.full, newdata = test_cleaned_bc)
cm.poly.c<-round(confusionMatrix(y_svmPoly,test_cleaned_bc$Status)$overall["Accuracy"],3)

# LogitBoost model
fit.logit.full<- caret::train(Status~.,data=train_cleaned_bc,
            method = "LogitBoost", metric="Accuracy",trControl=ctrl)
# Predict test data set
y_logit = predict(fit.logit.full, newdata = test_cleaned_bc)
cm.logit.c<-round(confusionMatrix(y_logit,test_cleaned_bc$Status)$overall["Accuracy"],3)

# Random forest model
fit.rf.full<- caret::train(Status~.,data=train_cleaned_bc,method = "rf",metric="Accuracy",
                           trControl=ctrl)
# Predict test data set
y_rf = predict(fit.rf.full, newdata = test_cleaned_bc)
cm.rf.c<-round(confusionMatrix(y_rf,test_cleaned_bc$Status)$overall["Accuracy"],3)

# LDA model
fit.lda.full <- caret::train(Status~.,data=train_cleaned_bc, method = "lda",
                      metric="Accuracy",  trControl=ctrl)
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
# Predict test data set
y_lda = predict(fit.lda.full, newdata = test_cleaned_bc)
cm.lda.c<-round(confusionMatrix(y_lda,test_cleaned_bc$Status)$overall["Accuracy"],3)

#XgBoost Model
fit.xgboost.full<-caret::train(Status~.,data=train_cleaned_bc,
                method = "xgbTree",metric="Accuracy",trControl=ctrl)
# Predict test data set
y_xgboost = predict(fit.xgboost.full, newdata = test_cleaned_bc)
cm.xgb.c<-round(confusionMatrix(y_xgboost,test_cleaned_bc$Status)$overall["Accuracy"],3)
```

#(iii)Center/Stem

```
index3 <- createDataPartition(center_stem$Status, p = 0.85, list = FALSE)
train_cs1 <- center_stem[index3, ]
test_cs1  <- center_stem[-index3, ]
target<-'Status'
```

```
train_cs<-subset(train_cs1,select = -c(1,3))
test_cs<-subset(test_cs1,select=-c(1,3))
predictors <- setdiff(names(train_cs), target)
# normalize data
train_nom_cs <- preProcess(train_cs[ ,predictors], method = c("center", "scale"))
train_scaled_cs <- predict(train_nom_cs, train_cs)
train_scaled_cs$Status <- train_cs$Status

#After inspecting the matrix we will set the correlation threshold at 0.90.
# Remove highly correlated predictors
train_corr_cs <- cor(train_scaled_cs[ ,predictors])
train_high_corr_cs <- findCorrelation(train_corr_cs, cutoff = .90)
train_cleaned_cs <- train_scaled_cs[ , -c(train_high_corr_cs)]
train_cleaned_cs$Status <- train_cs$Status

print(paste0("Number of more than 90% correlated columns : ", length(train_high_corr_cs)))
```

```
## [1] "Number of more than 90% correlated columns : 612"
```

```
dim(train_cleaned_cs)
```

```
## [1]  63 290
```

```
# normalize data
test_cleaned_cs <- predict(train_nom_cs, test_cs)
test_cleaned_cs <- test_cleaned_cs[ ,names(train_cleaned_cs)]
test_cleaned_cs$Status <- test_cs$Status
```

#(b)cs training

```
# SVM model linear
fit.linear.full<-caret::train(Status~., data = train_cleaned_cs, method = "svmLinear",
                              metric="Accuracy", trControl=ctrl)
# Predict test data set
y_svmlinear = predict(fit.linear.full, newdata = test_cleaned_cs)
cm_linear<-confusionMatrix(y_svmlinear,test_cleaned_cs$Status)
cm_linear
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction 0 1
##          0 3 3
##          1 2 3
##
##                Accuracy : 0.5455
##                  95% CI : (0.2338, 0.8325)
##     No Information Rate : 0.5455
##     P-Value [Acc > NIR] : 0.6214
##
##                   Kappa : 0.0984
##
##  Mcnemar's Test P-Value : 1.0000
##
##             Sensitivity : 0.6000
##             Specificity : 0.5000
```

```
##          Pos Pred Value : 0.5000
##          Neg Pred Value : 0.6000
##              Prevalence : 0.4545
##          Detection Rate : 0.2727
##    Detection Prevalence : 0.5455
##       Balanced Accuracy : 0.5500
##
##        'Positive' Class : 0
##
```

```
cm.linear.s  <-round(cm_linear$overall["Accuracy"], 3)

# SVM model Radial

fit.radial.full<-caret::train(Status~.,data=train_cleaned_cs,method = "svmRadial",
                   metric="Accuracy",trControl=ctrl)
# Predict test data set
y_svmRedial = predict(fit.radial.full, newdata = test_cleaned_cs)
cm.radial_s<-confusionMatrix(y_svmRedial,test_cleaned_cs$Status)
cm.radial_s
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction 0 1
##          0 4 2
##          1 1 4
##
##                Accuracy : 0.7273
##                  95% CI : (0.3903, 0.9398)
##     No Information Rate : 0.5455
##     P-Value [Acc > NIR] : 0.1829
##
##                   Kappa : 0.459
##
##  Mcnemar's Test P-Value : 1.0000
##
##             Sensitivity : 0.8000
##             Specificity : 0.6667
##          Pos Pred Value : 0.6667
##          Neg Pred Value : 0.8000
##              Prevalence : 0.4545
##          Detection Rate : 0.3636
##    Detection Prevalence : 0.5455
##       Balanced Accuracy : 0.7333
##
##        'Positive' Class : 0
##
```

```
cm.radial.s<-round(cm.radial_s$overall["Accuracy"],3)
# svm Poly model
fit.poly.full<- caret::train(Status~.,data=train_cleaned_cs,
                     method = "svmPoly",
                     metric="Accuracy",trControl=ctrl)
# Predict test data set
```

```r
y_svmPoly = predict(fit.poly.full, newdata = test_cleaned_cs)
cm.poly.s<-round(confusionMatrix(y_svmPoly,test_cleaned_cs$Status)$overall["Accuracy"],3)

# LogitBoost model
fit.logit.full<- caret::train(Status~.,data=train_cleaned_cs,
          method = "LogitBoost",metric="Accuracy",trControl=ctrl)
# Predict test data set
y_logit = predict(fit.logit.full, newdata = test_cleaned_cs)
cm_log<-confusionMatrix(y_logit,test_cleaned_cs$Status)
cm_log
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction 0 1
##          0 4 1
##          1 1 5
##
##                Accuracy : 0.8182
##                  95% CI : (0.4822, 0.9772)
##     No Information Rate : 0.5455
##     P-Value [Acc > NIR] : 0.0615
##
##                   Kappa : 0.6333
##
##  Mcnemar's Test P-Value : 1.0000
##
##             Sensitivity : 0.8000
##             Specificity : 0.8333
##          Pos Pred Value : 0.8000
##          Neg Pred Value : 0.8333
##              Prevalence : 0.4545
##          Detection Rate : 0.3636
##    Detection Prevalence : 0.4545
##       Balanced Accuracy : 0.8167
##
##        'Positive' Class : 0
##
```

```r
cm.logit.s   <-round(cm_log$overall["Accuracy"],3)

# Random forest model
fit.rf.full<- caret::train(Status~.,data=train_cleaned_cs,method = "rf",
                           metric="Accuracy",trControl=ctrl)
# Predict test data set
y_rf = predict(fit.rf.full, newdata = test_cleaned_cs)
cm_rf<-confusionMatrix(y_rf,test_cleaned_cs$Status)
cm_rf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction 0 1
##          0 3 2
```

```
##          1 2 4
##
##                Accuracy : 0.6364
##                  95% CI : (0.3079, 0.8907)
##     No Information Rate : 0.5455
##     P-Value [Acc > NIR] : 0.3853
##
##                   Kappa : 0.2667
##
##  Mcnemar's Test P-Value : 1.0000
##
##             Sensitivity : 0.6000
##             Specificity : 0.6667
##          Pos Pred Value : 0.6000
##          Neg Pred Value : 0.6667
##              Prevalence : 0.4545
##          Detection Rate : 0.2727
##    Detection Prevalence : 0.4545
##       Balanced Accuracy : 0.6333
##
##        'Positive' Class : 0
##
```

```
cm.rf.s  <-round(cm_rf$overall["Accuracy"],3)

# LDA model
fit.lda.full <- caret::train(Status~.,data=train_cleaned_cs, method = "lda",
                             metric="Accuracy", trControl=ctrl)
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
# Predict test data set
y_lda = predict(fit.lda.full, newdata = test_cleaned_cs)
cm_lda<-confusionMatrix(y_lda,test_cleaned_cs$Status)
cm_lda
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction 0 1
##          0 3 1
##          1 2 5
##
##                Accuracy : 0.7273
##                  95% CI : (0.3903, 0.9398)
```

```
##     No Information Rate : 0.5455
##     P-Value [Acc > NIR] : 0.1829
##
##                   Kappa : 0.4407
##
##  Mcnemar's Test P-Value : 1.0000
##
##             Sensitivity : 0.6000
##             Specificity : 0.8333
##          Pos Pred Value : 0.7500
##          Neg Pred Value : 0.7143
##              Prevalence : 0.4545
##          Detection Rate : 0.2727
##    Detection Prevalence : 0.3636
##       Balanced Accuracy : 0.7167
##
##        'Positive' Class : 0
##
```

```
cm.lda.s<-round(cm_lda$overall["Accuracy"], 3)

#XgBoost Model
fit.xgboost.full<-caret::train(Status~.,data=train_cleaned_cs,
            method = "xgbTree",metric="Accuracy",trControl=ctrl)
# Predict test data set
y_xgboost = predict(fit.xgboost.full, newdata = test_cleaned_cs)
cm_xg<-confusionMatrix(y_xgboost,test_cleaned_cs$Status)
cm_xg
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction 0 1
##          0 4 2
##          1 1 4
##
##                Accuracy : 0.7273
##                  95% CI : (0.3903, 0.9398)
##     No Information Rate : 0.5455
##     P-Value [Acc > NIR] : 0.1829
##
##                   Kappa : 0.459
##
##  Mcnemar's Test P-Value : 1.0000
##
##             Sensitivity : 0.8000
##             Specificity : 0.6667
##          Pos Pred Value : 0.6667
##          Neg Pred Value : 0.8000
##              Prevalence : 0.4545
##          Detection Rate : 0.3636
##    Detection Prevalence : 0.5455
##       Balanced Accuracy : 0.7333
##
##        'Positive' Class : 0
```

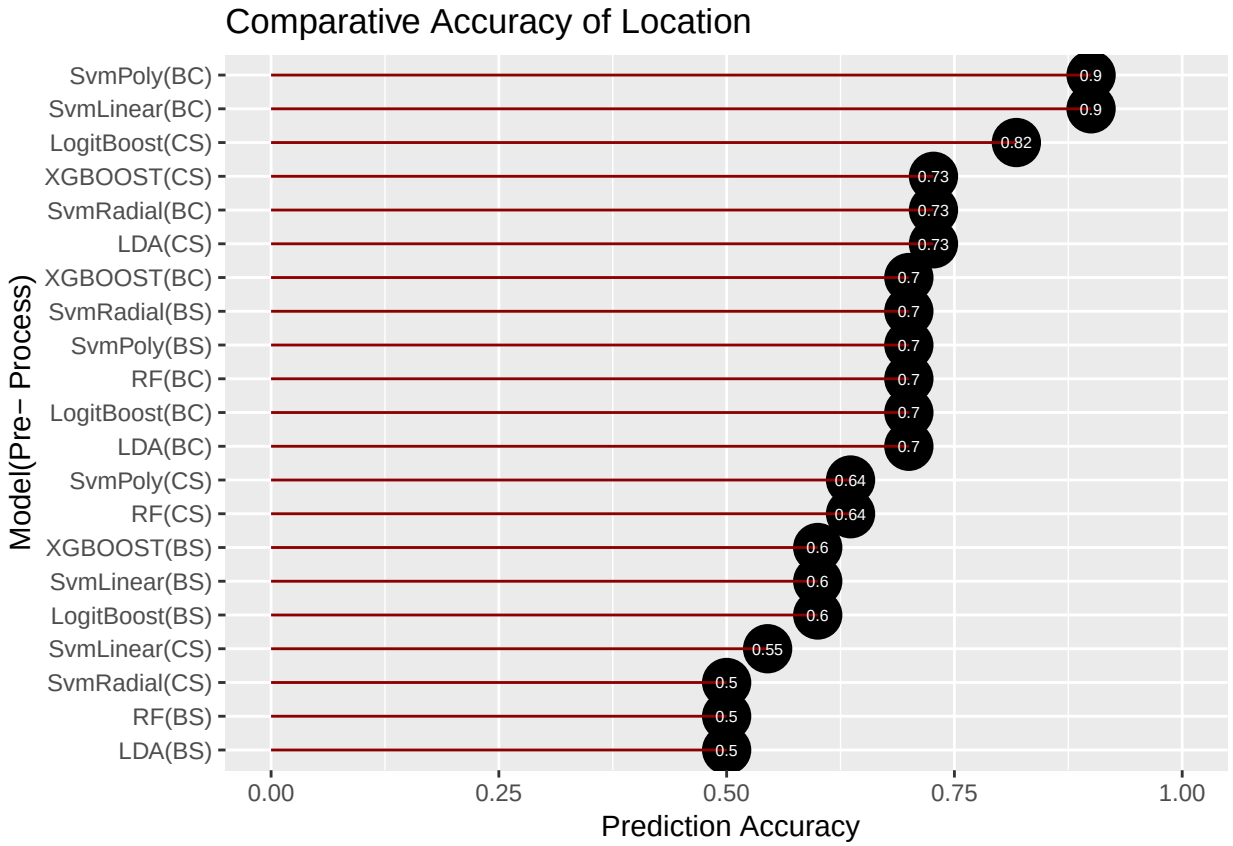
```
##
cm.xgb.s<-round(cm_xg$overall["Accuracy"],3)
```

#Tuber location or combination predictive table

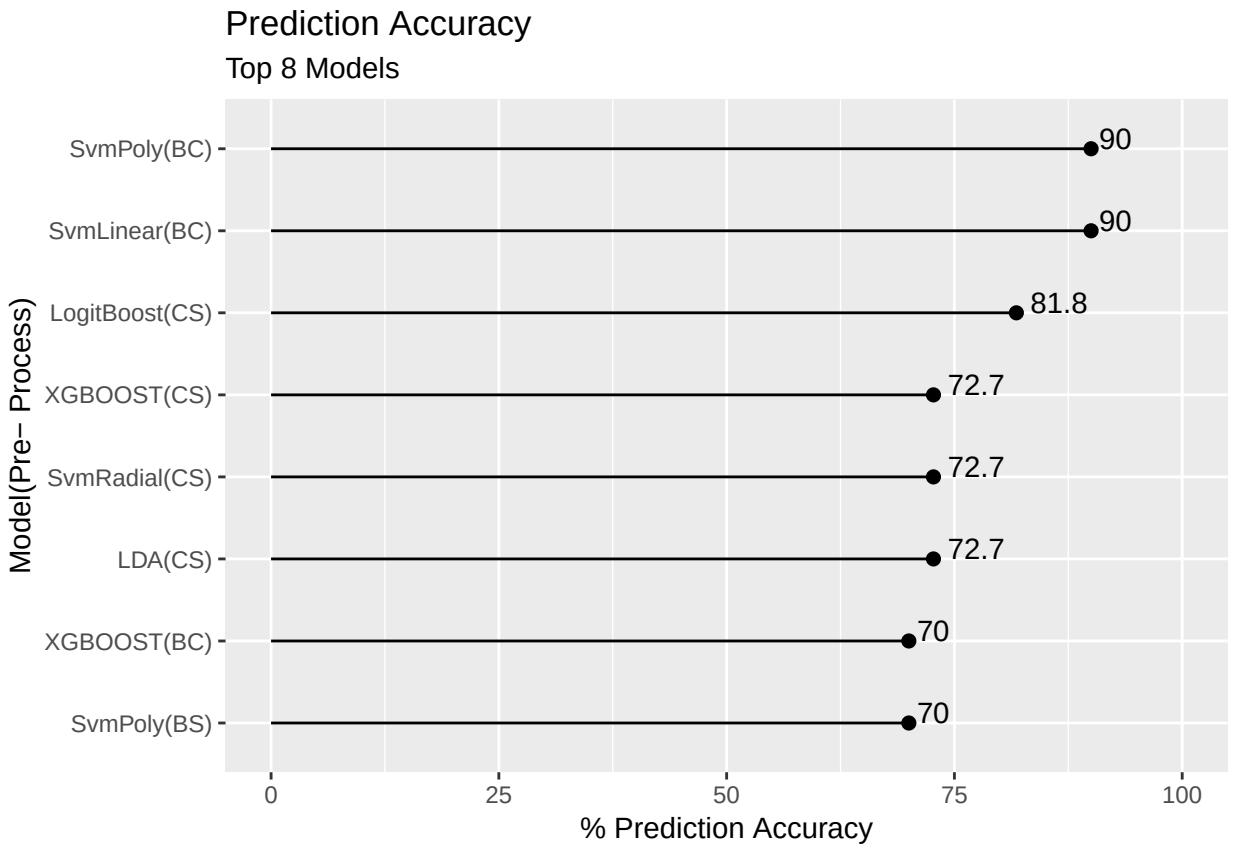
```
  # i) Training tuber location without Cultivar
  locat<-data.frame(Model=c("SvmPoly","XGBOOST", "LogitBoost", "SvmLinear","RF",
                    "SvmRadial", "LDA"),
  CS=c(cm.poly.s,cm.xgb.s,cm.logit.s,cm.linear.s,cm.rf.s,cm.radial.s,cm.lda.s),
   BS=c(cm.poly.b,cm.xgb.b,cm.logit.b,cm.linear.b,cm.rf.b,cm.radial.b,cm.lda.b),
   BC=c(cm.poly.c,cm.xgb.c,cm.logit.c,cm.linear.c,cm.rf.c,cm.radial.c,cm.lda.c))
locat
```

```
##        Model    CS  BS  BC
## 1    SvmPoly 0.636 0.7 0.9
## 2    XGBOOST 0.727 0.6 0.7
## 3 LogitBoost 0.818 0.6 0.7
## 4  SvmLinear 0.545 0.6 0.9
## 5         RF 0.636 0.5 0.7
## 6  SvmRadial 0.727 0.5 0.7
## 7        LDA 0.727 0.5 0.7
```

```
Location_compare<-data.frame(Model=c("SvmPoly(CS)","SvmPoly(BS)","SvmPoly(BC)",
    "XGBOOST(CS)","XGBOOST(BS)","XGBOOST(BC)","LogitBoost(CS)","LogitBoost(BS)",
  "LogitBoost(BC)","SvmLinear(CS)","SvmLinear(BS)","SvmLinear(BC)" ,"RF(CS)",
  "RF(BS)","RF(BC)" ,"SvmRadial(CS)", "SvmRadial(BS)","SvmRadial(BC)",
  "LDA(CS)","LDA(BS)","LDA(BC)"),
            Accuracy=c(cm.poly.s,cm.poly.b,cm.poly.c,cm.xgb.s,cm.xgb.b,
                  cm.xgb.c,cm.logit.s,cm.logit.b,cm.logit.c,cm.linear.s,
                  cm.linear.b,cm.linear.c,cm.rf.s,cm.rf.b,cm.rf.c,
  cm.radial.b,cm.radial.c,cm.radial.s,cm.lda.s,cm.lda.b, cm.lda.c ))

ggplot(Location_compare,aes(x=reorder(Model,Accuracy), y=Accuracy,
                            label=round(Accuracy,2))) +
geom_point(stat = "identity",fill="darkred",size=8) +
geom_segment(aes(y=0,x=Model,yend=Accuracy,xend=Model),color="darkred") +
geom_text(color="white", size=2) +
labs(x = "Model(Pre- Process)", y = " Prediction Accuracy") +
ggtitle('Comparative Accuracy of Location ') +
ylim(0, 1) +coord_flip()
```




```
Location_compare<-data.frame(Model=c("SvmPoly(CS)","SvmPoly(BS)","SvmPoly(BC)",
     "XGBOOST(CS)","XGBOOST(BS)","XGBOOST(BC)","LogitBoost(CS)","LogitBoost(BS)",
  "LogitBoost(BC)","SvmLinear(CS)","SvmLinear(BS)","SvmLinear(BC)" ,"RF(CS)",
  "RF(BS)","RF(BC)" ,"SvmRadial(CS)", "SvmRadial(BS)","SvmRadial(BC)",
  "LDA(CS)","LDA(BS)","LDA(BC)"),
    Accuracy=c(cm.poly.s*100,cm.poly.b*100,cm.poly.c*100,cm.xgb.s*100,cm.xgb.b*100,
    cm.xgb.c*100,cm.logit.s*100,cm.logit.b*100,cm.logit.c*100,cm.linear.s*100,
    cm.linear.b*100,cm.linear.c*100,cm.rf.s*100,cm.rf.b*100,cm.rf.c*100,
    cm.radial.s*100,cm.radial.b*100, cm.radial.c*100,cm.lda.s*100,cm.lda.b*100,
    cm.lda.c*100))


df<-Location_compare[with(Location_compare,order(Accuracy,decreasing = T)),]
df_8<-df[1:8,]

ggplot(df_8,aes(x = reorder(Model, Accuracy), y=Accuracy,
                label=paste0(round(Accuracy,0),"%"))) +
  geom_point(fill ="orange",stat = "identity", size = 2) +
 geom_segment(aes(y=0,x=Model,yend=Accuracy,xend=Model),color="black") +
  geom_text(aes(label = Accuracy), hjust = -.25,vjust=0)  +
 labs(x = "Model(Pre- Process)", y = "% Prediction Accuracy") +
  ggtitle("Prediction Accuracy",
          subtitle= "Top 8 Models") +
   ylim(0, 100) +
  coord_flip()
```



#1.Full Model

#(i)let try added Cultivator

```
tr_set<-subset(train_data,select = c(1))
train_cleaned1<-data.frame(tr_set,train_cleaned)
##
te_set<-subset(test_data,select = c(1))
test_cleaned1<-data.frame(te_set,test_cleaned)

# SVM model linear
fit.linear.full<-caret::train(Status~.,data = train_cleaned1,method = "svmLinear",
                              metric="Accuracy",trControl=ctrl)
# Predict test data set
y_svmlinear = predict(fit.linear.full, newdata = test_cleaned1[-2])
cm.linear.cul<-round(confusionMatrix(y_svmlinear,test_cleaned$Status)$overall["Accuracy"],3)

# SVM model Radial

fit.radial.full<-caret::train(Status~.,data=train_cleaned1,
                  method = "svmRadial",metric="Accuracy",trControl=ctrl)
# Predict test data set
y_svmRedial = predict(fit.radial.full, newdata = test_cleaned1[-2])
cm.radial.cul<-round(confusionMatrix(y_svmRedial,test_cleaned$Status)$overall["Accuracy"],3)

# svm Poly model
fit.poly.full<- caret::train(Status~.,data=train_cleaned1,
                              method = "svmPoly",
```

```
                              metric="Accuracy",trControl=ctrl)
# Predict test data set
y_svmPoly = predict(fit.poly.full, newdata = test_cleaned1[-2])
cm.poly.cul<-round(confusionMatrix(y_svmPoly,test_cleaned$Status)$overall["Accuracy"],3)

# LogitBoost model
fit.logit.full<- caret::train(Status~.,data=train_cleaned1,
                              method = "LogitBoost",metric="Accuracy",
                              trControl=ctrl)

# Predict test data set
y_logit = predict(fit.logit.full, newdata = test_cleaned1[-2])
cm.logit.cul<-round(confusionMatrix(y_logit,test_cleaned$Status)$overall["Accuracy"],3)
# Random forest model
fit.rf.full<- caret::train(Status~.,data=train_cleaned1,method = "rf",
                           metric="Accuracy",trControl=ctrl)
# Predict test data set
y_rf = predict(fit.rf.full, newdata = test_cleaned1[-2])
cm.rf.cul<-round(confusionMatrix(y_rf,test_cleaned$Status)$overall["Accuracy"],3)

# LDA model
fit.lda.full<- caret::train(Status~.,data=train_cleaned1,method = "lda",
                            metric="Accuracy",trControl=ctrl)
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
# Predict test data set
y_lda = predict(fit.lda.full, newdata = test_cleaned1[-2])
cm.lda.cul<-round(confusionMatrix(y_lda,test_cleaned$Status)$overall["Accuracy"],3)

#XgBoost Model
fit.xgboost.full<-caret::train(Status~.,data=train_cleaned1,
                               method ="xgbTree",metric="Accuracy",trControl=ctrl)
# Predict test data set
y_xgboost = predict(fit.xgboost.full, newdata = test_cleaned1[-2])
cm.xgb.cut<-round(confusionMatrix(y_xgboost,test_cleaned$Status)$overall["Accuracy"],3)
```

#(ii)Let try added Location only

```
tr_set<-subset(train_data,select = c(2))
train_cleaned1<-data.frame(tr_set,train_cleaned)
##
te_set<-subset(test_data,select = c(2))
test_cleaned1<-data.frame(te_set,test_cleaned)
# SVM model linear
```

```
fit.linear.full<-caret::train(Status~.,data = train_cleaned1,method = "svmLinear",
                              metric="Accuracy",trControl=ctrl)
# Predict test data set
y_svmlinear = predict(fit.linear.full, newdata = test_cleaned1[-2])
cm.linear.loc<-round(confusionMatrix(y_svmlinear,test_cleaned$Status)$overall["Accuracy"],3)

# SVM model Radial

fit.radial.full<-caret::train(Status~.,data=train_cleaned1,method = "svmRadial",
                              metric="Accuracy",trControl=ctrl)
# Predict test data set
y_svmRedial = predict(fit.radial.full, newdata = test_cleaned1[-2])
cm.radial.loc<-round(confusionMatrix(y_svmRedial,test_cleaned$Status)$overall["Accuracy"],3)

# svm Poly model
fit.poly.full<- caret::train(Status~.,data=train_cleaned1,method = "svmPoly",
                             metric="Accuracy",trControl=ctrl)
# Predict test data set
y_svmPoly = predict(fit.poly.full, newdata = test_cleaned1[-2])
cm.poly.loc<-round(confusionMatrix(y_svmPoly,test_cleaned$Status)$overall["Accuracy"],3)

# LogitBoost model
fit.logit.full<- caret::train(Status~.,data=train_cleaned1,
            method = "LogitBoost",metric="Accuracy",trControl=ctrl)

# Predict test data set
y_logit = predict(fit.logit.full, newdata = test_cleaned1[-2])
cm.logit.loc<-round(confusionMatrix(y_logit,test_cleaned$Status)$overall["Accuracy"],3)
# Random forest model

fit.rf.full<- caret::train(Status~.,data=train_cleaned1,method = "rf",
                           metric="Accuracy",trControl=ctrl)
# Predict test data set
y_rf = predict(fit.rf.full, newdata = test_cleaned1[-2])
cm.rf.loc<-round(confusionMatrix(y_rf,test_cleaned$Status)$overall["Accuracy"],3)

# LDA model
fit.lda.full<- caret::train(Status~.,data=train_cleaned1,method = "lda",
                            metric="Accuracy",trControl=ctrl)
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
# Predict test data set
y_lda = predict(fit.lda.full, newdata = test_cleaned1[-2])
```

```
cm.lda.loc<-round(confusionMatrix(y_lda,test_cleaned$Status)$overall["Accuracy"],3)

#XgBoost Model
fit.xgboost.full<-caret::train(Status~.,data=train_cleaned1,
              method = "xgbTree",metric="Accuracy",trControl=ctrl)
# Predict test data set
y_xgboost = predict(fit.xgboost.full, newdata = test_cleaned1[-2])
cm.xgb.loc<-round(confusionMatrix(y_xgboost,test_cleaned$Status)$overall["Accuracy"],3)
```

#(iii) Removal of Cultivar & Location

```
knitr::opts_chunk$set(echo = TRUE)

# SVM model linear

fit.linear.full<-caret::train(Status~.,data = train_cleaned,
        method = "svmLinear",metric="Accuracy",trControl=ctrl)
# Predict test data set
y_svmlinear = predict(fit.linear.full, newdata = test_cleaned[-1])
cm.linear.full<-round(confusionMatrix(y_svmlinear,test_cleaned$Status)$overall["Accuracy"],3)
# SVM model Radial

fit.radial.full<-caret::train(Status~.,data=train_cleaned,
            method = "svmRadial",metric="Accuracy",trControl=ctrl)
# Predict test data set
y_svmRedial = predict(fit.radial.full, newdata = test_cleaned[-1])
cm.svmradial.full<-round(confusionMatrix(y_svmRedial,test_cleaned$Status)$overall["Accuracy"],3)
# svm Poly model
fit.poly.full<- caret::train(Status~.,data=train_cleaned,method = "svmPoly",
                             metric="Accuracy",trControl=ctrl)
# Predict test data set
y_svmPoly = predict(fit.poly.full, newdata = test_cleaned[-1])
cm.svmpoly.full<-round(confusionMatrix(y_svmPoly,test_cleaned$Status)$overall["Accuracy"],3)
# LogitBoost model
fit.logit.full<- caret::train(Status~.,data=train_cleaned,
        method = "LogitBoost",metric="Accuracy",trControl=ctrl)

# Predict test data set
y_logit = predict(fit.logit.full, newdata = test_cleaned[-1])
cm.logit.full<-round(confusionMatrix(y_logit,test_cleaned$Status)$overall["Accuracy"],3)
# Random forest model

fit.rf.full<- caret::train(Status~.,data=train_cleaned,method = "rf",
                           metric="Accuracy",trControl=ctrl)
# Predict test data set
y_rf = predict(fit.rf.full, newdata = test_cleaned[-1])
cm.rf.full<-round(confusionMatrix(y_rf,test_cleaned$Status)$overall["Accuracy"],3)

# LDA model
fit.lda.full<- caret::train(Status~.,data=train_cleaned,method = "lda",
                            metric="Accuracy",trControl=ctrl)
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
# Predict test data set
y_lda = predict(fit.lda.full, newdata = test_cleaned[-1])
cm.lda.full<-round(confusionMatrix(y_lda,test_cleaned$Status)$overall["Accuracy"],3)

#XgBoost Model
fit.xgboost.full<-caret::train(Status~.,data=train_cleaned,
        method = "xgbTree",metric="Accuracy",trControl=ctrl)
# Predict test data set
y_xgboost = predict(fit.xgboost.full, newdata = test_cleaned[-1])
cm.xgboost.full<-round(confusionMatrix(y_xgboost,test_cleaned$Status)$overall["Accuracy"],3)
```

Full Model

#(iii)Combination of Location,Cultivator

```
tr_set<-subset(train_data,select = c(1,2))
train_cleaned1<-data.frame(tr_set,train_cleaned)

##
te_set<-subset(test_data,select = c(1,2))
test_cleaned1<-data.frame(te_set,test_cleaned)

# SVM model linear
fit.linear.full<-caret::train(Status~.,data = train_cleaned1,method = "svmLinear",
                              metric="Accuracy",trControl=ctrl)
# Predict test data set
y_svmlinear = predict(fit.linear.full, newdata = test_cleaned1[-3])
cm.linear_both<-confusionMatrix(y_svmlinear,test_cleaned$Status)
cm.linear_both
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction 0 1
##          0 9 3
##          1 1 8
##
##                Accuracy : 0.8095
##                  95% CI : (0.5809, 0.9455)
##     No Information Rate : 0.5238
##     P-Value [Acc > NIR] : 0.006689
##
##                   Kappa : 0.6216
```

```
##
##  Mcnemar's Test P-Value : 0.617075
##
##             Sensitivity : 0.9000
##             Specificity : 0.7273
##          Pos Pred Value : 0.7500
##          Neg Pred Value : 0.8889
##              Prevalence : 0.4762
##          Detection Rate : 0.4286
##    Detection Prevalence : 0.5714
##       Balanced Accuracy : 0.8136
##
##        'Positive' Class : 0
##
```

```
cm.linear.both<-round(cm.linear_both$overall["Accuracy"],3)

# SVM model Radial

fit.radial.full<-caret::train(Status~.,data=train_cleaned1,
                              method = "svmRadial",metric="Accuracy",trControl=ctrl)
# Predict test data set
y_svmRedial = predict(fit.radial.full, newdata = test_cleaned1[-3])
cm.radial.both<-round(confusionMatrix(y_svmRedial,
                    test_cleaned$Status)$overall["Accuracy"],3)

# svm Poly model
fit.poly.full<- caret::train(Status~.,data=train_cleaned1,method = "svmPoly",
                             metric="Accuracy",trControl=ctrl)
# Predict test data set
y_svmPoly = predict(fit.poly.full, newdata = test_cleaned1[-3])
cm.poly.both<-round(confusionMatrix(y_svmPoly,test_cleaned$Status)$overall["Accuracy"],3)

# LogitBoost model
fit.logit.full<- caret::train(Status~.,data=train_cleaned1,
      method = "LogitBoost",metric="Accuracy",trControl=ctrl)
# Predict test data set
y_logit = predict(fit.logit.full, newdata = test_cleaned1[-3])
cm.logit.both<-round(confusionMatrix(y_logit,test_cleaned$Status)$overall["Accuracy"],3)
# Random forest model

fit.rf.full<- caret::train(Status~.,data=train_cleaned1,method = "rf",
                           metric="Accuracy",trControl=ctrl)
# Predict test data set
y_rf = predict(fit.rf.full, newdata = test_cleaned1[-3])
cm.rf.both<-round(confusionMatrix(y_rf,test_cleaned$Status)$overall["Accuracy"],3)

# LDA model
fit.lda.full<- caret::train(Status~.,data=train_cleaned1,method = "lda",
                            metric="Accuracy",trControl=ctrl)
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
# Predict test data set
y_lda = predict(fit.lda.full, newdata = test_cleaned1[-3])
cm.lda_both<-confusionMatrix(y_lda,test_cleaned$Status)
cm.lda_both
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction 0 1
##          0 9 2
##          1 1 9
##
##                Accuracy : 0.8571
##                  95% CI : (0.6366, 0.9695)
##     No Information Rate : 0.5238
##     P-Value [Acc > NIR] : 0.001511
##
##                   Kappa : 0.7149
##
##  Mcnemar's Test P-Value : 1.000000
##
##             Sensitivity : 0.9000
##             Specificity : 0.8182
##          Pos Pred Value : 0.8182
##          Neg Pred Value : 0.9000
##              Prevalence : 0.4762
##          Detection Rate : 0.4286
##    Detection Prevalence : 0.5238
##       Balanced Accuracy : 0.8591
##
##        'Positive' Class : 0
##
```

```
cm.ld.both<-round(cm.lda_both$overall["Accuracy"],3)

#XgBoost Model
fit.xgboost.full<-caret::train(Status~.,data=train_cleaned1,
        method = "xgbTree",metric="Accuracy",trControl=ctrl)
# Predict test data set
y_xgboost = predict(fit.xgboost.full, newdata = test_cleaned1[-3])
cm.xgb.both<-round(confusionMatrix(y_xgboost,test_cleaned$Status)$overall["Accuracy"],3)
```

```
result_compare<-data.frame(Model=c("SvmPoly","XGBOOST", "LogitBoost",
             "SvmLinear","RF","SvmRadial", "LDA"),
  "subset location "=c(cm.poly.loc,cm.xgb.loc,cm.logit.loc,cm.linear.loc,
                       cm.rf.loc,cm.radial.loc,cm.lda.loc),
  "subset cultivar"=c(cm.poly.cul,cm.xgb.cut,cm.logit.cul,cm.linear.cul,
```

```
                      cm.rf.cul,cm.radial.cul,cm.lda.cul),
  Both=c(cm.poly.both,cm.xgb.both,cm.logit.both,cm.linear.both,cm.rf.both,
                    cm.radial.both,cm.ld.both),
  Neither=c(cm.svmpoly.full,cm.xgboost.full,cm.logit.full,
              cm.linear.full,cm.rf.full,cm.svmradial.full,cm.lda.full))
result_compare
```

```
##        Model subset.location. subset.cultivar  Both Neither
## 1    SvmPoly            0.762           0.667 0.810   0.810
## 2    XGBOOST            0.524           0.571 0.571   0.524
## 3 LogitBoost            0.571           0.524 0.571   0.524
## 4  SvmLinear            0.810           0.810 0.810   0.810
## 5         RF            0.571           0.571 0.571   0.524
## 6  SvmRadial            0.619           0.619 0.571   0.619
## 7        LDA            0.857           0.810 0.857   0.857
```

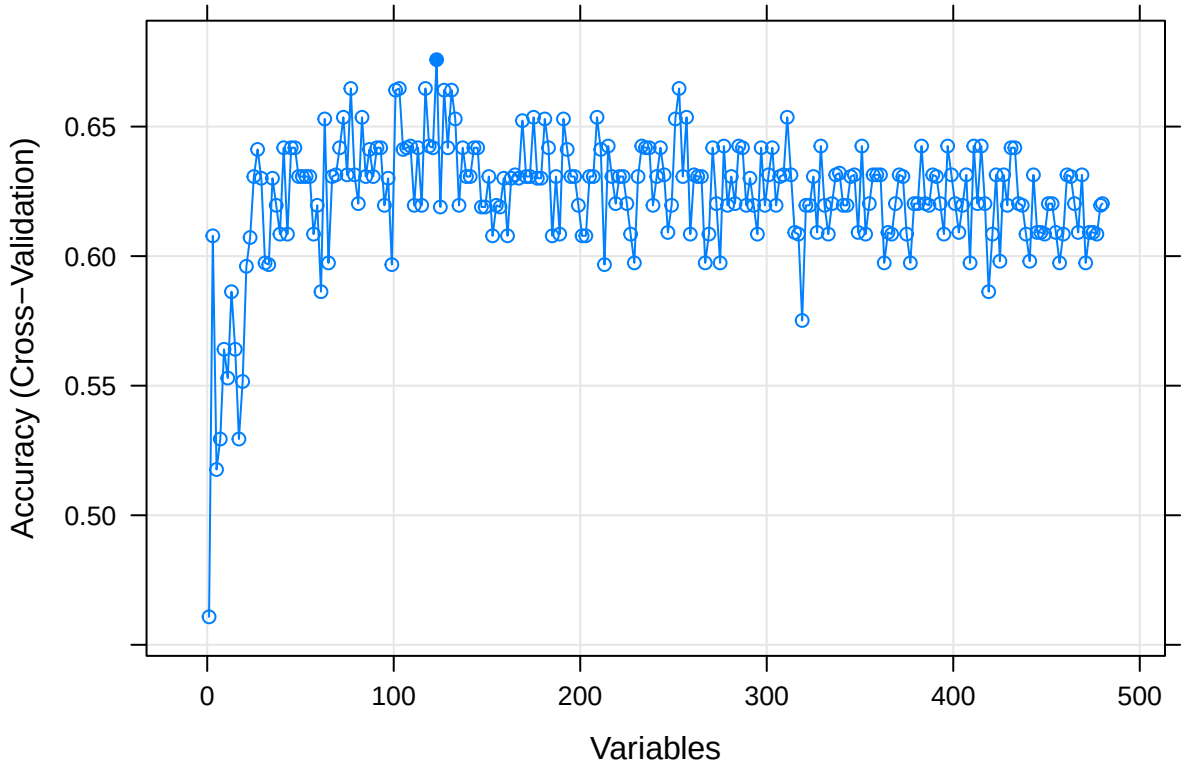
#Finally,we making a full final model set of all predictors include cultivator & #location for future prediction.

#(2)Feature Selection using Recursive Feature Elimination

```
set.seed(134)
control <- rfeControl(functions=rfFuncs, method="cv", number=5,verbose = FALSE)
x<-train_cleaned[,2:481]
y<-train_cleaned$Status

results <- rfe(x,y ,sizes = seq(1,ncol(x),2) ,rfeControl=control)
predictors(results)
```

```
##   [1] "530.5" "261.5" "714.5" "681.5" "515.5" "823.5" "103.5" "191.5"
##   [9] "180.5" "736.5" "779.5" "239.5" "433.5" "335.5" "958.5" "338.5"
##  [17] "203.5" "263.5" "588.5" "986.5" "343.5" "364.5" "781.5" "116.5"
##  [25] "161.5" "645.5" "408.5" "432.5" "259.5" "926.5" "426.5" "661.5"
##  [33] "929.5" "477.5" "251.5" "747.5" "701.5" "456.5" "169.5" "533.5"
##  [41] "655.5" "972.5" "731.5" "250.5" "885.5" "412.5" "283.5" "553.5"
##  [49] "279.5" "640.5" "496.5" "999.5" "244.5" "805.5" "101.5" "811.5"
##  [57] "183.5" "827.5" "402.5" "617.5" "435.5" "417.5" "809.5" "575.5"
##  [65] "373.5" "346.5" "954.5" "328.5" "341.5" "269.5" "485.5" "599.5"
##  [73] "887.5" "396.5" "627.5" "273.5" "974.5" "330.5" "506.5" "469.5"
##  [81] "715.5" "514.5" "258.5" "126.5" "730.5" "902.5" "825.5" "534.5"
##  [89] "987.5" "938.5" "465.5" "495.5" "451.5" "394.5" "751.5" "852.5"
##  [97] "837.5" "311.5" "923.5" "201.5" "223.5" "131.5" "748.5" "326.5"
## [105] "836.5" "507.5" "612.5" "597.5" "676.5" "186.5" "689.5" "296.5"
## [113] "483.5" "255.5" "353.5" "386.5" "569.5" "471.5" "686.5" "762.5"
## [121] "778.5" "900.5" "866.5"
```

```
#The predictors function can be used to get a text string of variable
#names that were picked in the final model.
plot(results, type=c("g", "o"))
```




```
var.imp.cols.train <-c(predictors(results),"Status")
var.imp.cols.test <- c(predictors(results),"Status")
training <- train_cleaned[,c(var.imp.cols.train)]

testing <-test_cleaned[,c(var.imp.cols.test)]


dim(testing)
```

```
## [1]  21 124
```

#(i)Machine Learning with final set of feature variables

```
# Model
# SVM model Linear

fit.linear.fs<-caret::train(Status~.,data = training,method = "svmLinear",
                            metric="Accuracy",trControl=ctrl)
# Predict test data set
y_svmlinear = predict(fit.linear.fs,newdata = testing[-124])
cm.linear_fs<-confusionMatrix(y_svmlinear,test_data$Status)
cm.linear_fs
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction 0 1
##          0 7 2
##          1 3 9
##
##                Accuracy : 0.7619
```

```
##                  95% CI : (0.5283, 0.9178)
##     No Information Rate : 0.5238
##     P-Value [Acc > NIR] : 0.02269
##
##                   Kappa : 0.5205
##
##  Mcnemar's Test P-Value : 1.00000
##
##             Sensitivity : 0.7000
##             Specificity : 0.8182
##          Pos Pred Value : 0.7778
##          Neg Pred Value : 0.7500
##              Prevalence : 0.4762
##          Detection Rate : 0.3333
##    Detection Prevalence : 0.4286
##       Balanced Accuracy : 0.7591
##
##        'Positive' Class : 0
##
```

```
cm.svmlinear.fs<-round(cm.linear_fs$overall["Accuracy"],3)

# SVM model Radial

fit.rad.fs<-caret::train(Status~.,data=training,method = "svmRadial",
                         metric="Accuracy",trControl=ctrl)
# Predict test data set
y_svmRedial = predict(fit.rad.fs, newdata = testing[-124])
cm.svmradial.fs<-round(confusionMatrix(y_svmRedial,test_data$Status)$overall["Accuracy"],3)
cm.svmradial.fs
```

```
## Accuracy
##    0.619
```

```
# svm Poly model

fit.poly.fs<-caret::train(Status~.,data=training,method = "svmPoly",
                          metric="Accuracy",trControl=ctrl)
# Predict test data set
y_svmPoly = predict(fit.poly.fs, newdata = testing[-124])
cm.svmpoly.fs<-round(confusionMatrix(y_svmPoly,test_data$Status)$overall["Accuracy"],3)
cm.svmpoly.fs
```

```
## Accuracy
##    0.714
```

```
# LogitBoost model

fit.logit.fs<- caret::train(Status~.,data=training,method="LogitBoost",
                            metric="Accuracy",trControl=ctrl)
# Predict test data set
y_logit = predict(fit.logit.fs, newdata = testing[-124])
cm.logit.fs<-round(confusionMatrix(y_logit,test_data$Status)$overall["Accuracy"],3)
cm.logit.fs
```

```
## Accuracy
```

```
##    0.524
```

```
# LDA model
fit.lda.fs<-caret::train(Status~.,data=training,method = "lda",
                         metric="Accuracy" ,trControl=ctrl)
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
# Predict test data set
y_lda = predict(fit.lda.fs, newdata = testing[-124])
cm.lda.fs<-round(confusionMatrix(y_lda,testing$Status)$overall["Accuracy"],3)
cm.lda.fs
```

```
## Accuracy
##    0.524
```

```
# Random forest model

fit.rf.fs<- caret::train(Status~.,data=training,method="rf",metric="Accuracy",
                         trControl=ctrl)
# Predict test data set
y_rf = predict(fit.rf.fs, newdata = testing[-124])
cm.rf.fs<-round(confusionMatrix(y_rf,test_cleaned$Status)$overall["Accuracy"],3)
cm.rf.fs
```

```
## Accuracy
##    0.619
```

```
#  run penalised LDA for the model
fit.pda.fs<- caret::train(Status~.,data=training,method="pda",metric="Accuracy",
                          trControl=ctrl)
```

```
## Warning: predictions failed for Fold1: lambda=0e+00 Error in mindist[l] <- ndist[l] :
##   NAs are not allowed in subscripted assignments

## Warning: predictions failed for Fold2: lambda=0e+00 Error in mindist[l] <- ndist[l] :
##   NAs are not allowed in subscripted assignments

## Warning: predictions failed for Fold3: lambda=0e+00 Error in mindist[l] <- ndist[l] :
##   NAs are not allowed in subscripted assignments

## Warning: predictions failed for Fold4: lambda=0e+00 Error in mindist[l] <- ndist[l] :
##   NAs are not allowed in subscripted assignments

## Warning: predictions failed for Fold5: lambda=0e+00 Error in mindist[l] <- ndist[l] :
##   NAs are not allowed in subscripted assignments

## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info =
## trainInfo, : There were missing values in resampled performance measures.

## Warning in train.default(x, y, weights = w, ...): missing values found in
```

```
## aggregated results
```

```
# Predict test data set
y_pda = predict(fit.pda.fs, newdata = testing[-124])
cm.pda_fs<-confusionMatrix(y_pda,test_cleaned$Status)
cm.pda_fs
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction 0 1
##          0 7 2
##          1 3 9
##
##                Accuracy : 0.7619
##                  95% CI : (0.5283, 0.9178)
##     No Information Rate : 0.5238
##     P-Value [Acc > NIR] : 0.02269
##
##                   Kappa : 0.5205
##
##  Mcnemar's Test P-Value : 1.00000
##
##             Sensitivity : 0.7000
##             Specificity : 0.8182
##          Pos Pred Value : 0.7778
##          Neg Pred Value : 0.7500
##              Prevalence : 0.4762
##          Detection Rate : 0.3333
##    Detection Prevalence : 0.4286
##       Balanced Accuracy : 0.7591
##
##        'Positive' Class : 0
##
```

```
cm.pda.fs<-round(cm.pda_fs$overall["Accuracy"],3)

# xgboost

fit.xgboost.fs<- caret::train(Status~.,data=training,
                              method = "xgbTree" ,metric="Accuracy",
                              trControl=ctrl,verbose = FALSE)
# Predict test data set
y_xgboost.fs = predict(fit.xgboost.fs, newdata = testing[-124])
cm.xgboost.fs<-round(confusionMatrix(y_xgboost.fs,test_cleaned$Status)$overall["Accuracy"],3)
cm.xgboost.fs
```

```
## Accuracy
##    0.667
```

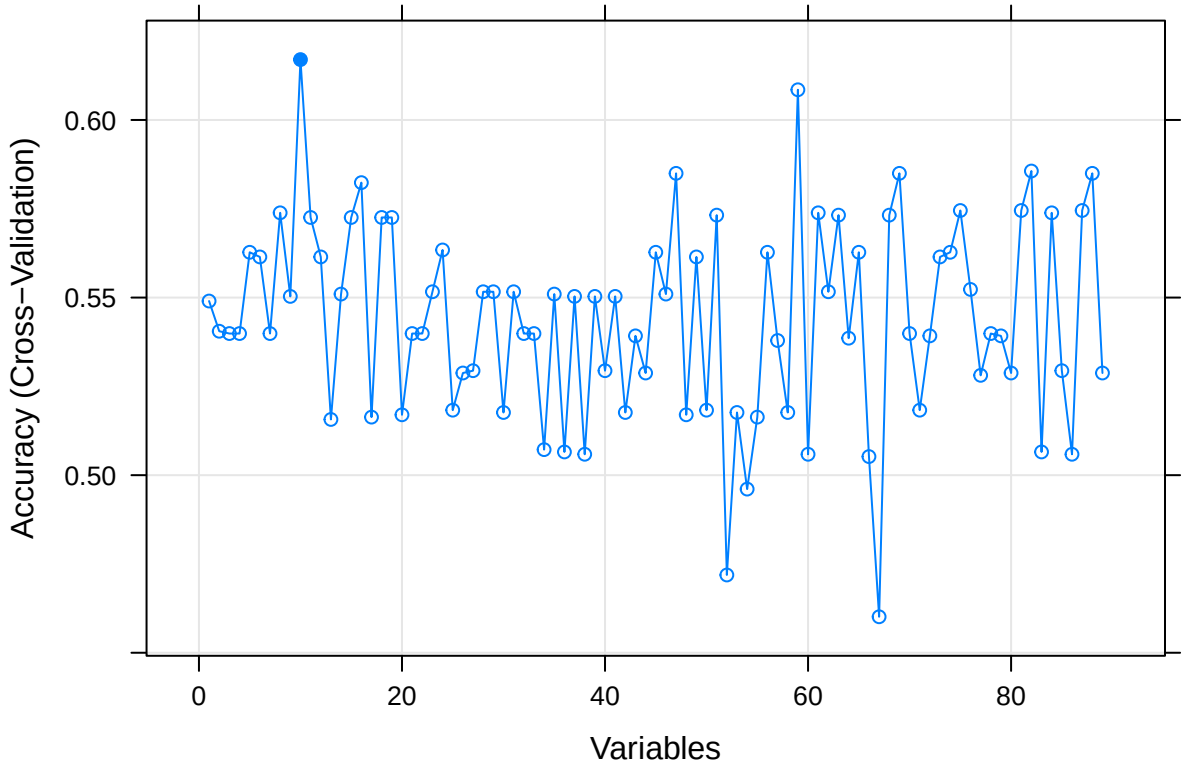
#(ii) PCA follow Feature Selection method

```
PCA <- prcomp(train_cleaned[-1], retx=TRUE, center=TRUE, scale = TRUE)
predictors <- PCA$x
Status  <- train_cleaned$Status
training2  <- data.frame(Status, predictors)
```

```
#testing
predictors <- predict(PCA, test_cleaned)  #rotated data using the same PCA
predictors <- predictors
Status  <- test_cleaned$Status
testing2  <- data.frame(Status, predictors)
control <- rfeControl(functions=rfFuncs, method="cv", number=5,verbose = FALSE)
x<-training2[,2:90]
y<-training2$Status

results <- rfe(x,y ,sizes = seq(1,89) ,rfeControl=control)
predictors(results)
```

```
##  [1] "PC15" "PC2"  "PC10" "PC78" "PC29" "PC9"  "PC24" "PC7"  "PC62" "PC50"
```

```
#The predictors function can be used to get a text string of variable names
#that were picked in the final model.
plot(results, type=c("g", "o"))
```




```
var.imp.cols.train <-c(predictors(results),"Status")
var.imp.cols.test <- c(predictors(results),"Status")
training_set <- training2[,c(var.imp.cols.train)]

testing_set <-testing2[,c(var.imp.cols.test)]

# SVM model linear
# 5 fold cross validation
ctrl <-trainControl(method="cv",number=5)
fit.linear.pca<-caret::train(Status~.,data = training_set,method = "svmLinear",
                             metric="Accuracy", trControl=ctrl)
# Predict test data set
```

```r
y_svmlinear = predict(fit.linear.pca, newdata = testing_set[-16])
cm.linear.pca_fs<-confusionMatrix(y_svmlinear,testing_set$Status)
cm.linear.pca_fs
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0  9  1
##          1  1 10
##
##                Accuracy : 0.9048
##                  95% CI : (0.6962, 0.9883)
##     No Information Rate : 0.5238
##     P-Value [Acc > NIR] : 0.0002453
##
##                   Kappa : 0.8091
##
##  Mcnemar's Test P-Value : 1.0000000
##
##             Sensitivity : 0.9000
##             Specificity : 0.9091
##          Pos Pred Value : 0.9000
##          Neg Pred Value : 0.9091
##              Prevalence : 0.4762
##          Detection Rate : 0.4286
##    Detection Prevalence : 0.4762
##       Balanced Accuracy : 0.9045
##
##        'Positive' Class : 0
##
```

```r
cm.fs.linear.pca<-round(cm.linear.pca_fs$overall["Accuracy"],3)
###

# SVM model Radial
fit.rad.pca<-caret::train(Status~.,data=training_set,method = "svmRadial" ,
                          metric="Accuracy" ,trControl=ctrl)
# Predict test data set
y_svmRedial = predict(fit.rad.pca, newdata = testing_set[-16])
cm.fs.radial.pca<-round(confusionMatrix(y_svmRedial,testing_set$Status)$overall["Accuracy"],3)
cm.fs.radial.pca
```

```
## Accuracy
##    0.714
```

```r
# svm poly model
fit.poly.pca<-caret::train(Status~.,data=training_set,method = "svmPoly",
                           metric="Accuracy" ,trControl=ctrl)
# Predict test data set
y_svmpoly = predict(fit.poly.pca, newdata = testing_set[-16])
cm.poly_pca_fs<-confusionMatrix(y_svmpoly,testing_set$Status)
cm.poly_pca_fs
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction 0 1
##          0 8 2
##          1 2 9
##
##                Accuracy : 0.8095
##                  95% CI : (0.5809, 0.9455)
##     No Information Rate : 0.5238
##     P-Value [Acc > NIR] : 0.006689
##
##                   Kappa : 0.6182
##
##  Mcnemar's Test P-Value : 1.000000
##
##             Sensitivity : 0.8000
##             Specificity : 0.8182
##          Pos Pred Value : 0.8000
##          Neg Pred Value : 0.8182
##              Prevalence : 0.4762
##          Detection Rate : 0.3810
##    Detection Prevalence : 0.4762
##       Balanced Accuracy : 0.8091
##
##        'Positive' Class : 0
##
```

```
cm.fs.poly.pca<-round(cm.poly_pca_fs$overall["Accuracy"],3)

# logitBoost model
fit.logit.pca<-caret::train(Status~.,data=training_set,method = "LogitBoost",
                            metric="Accuracy" ,trControl=ctrl)
# Predict test data set
y_logit = predict(fit.logit.pca, newdata = testing_set[-16])
cm.fs.logit.pca<-round(confusionMatrix(y_logit,testing_set$Status)$overall["Accuracy"],3)
cm.fs.logit.pca
```

```
## Accuracy
##    0.714
```

```
# Random forest model
fit.rf.pca<-caret::train(Status~.,data=training_set,method = "rf",metric="Accuracy",
                         trControl=ctrl)
# Predict test data set
y_rf = predict(fit.rf.pca, newdata = testing_set[-16])
cm.rf_pca_fs<-confusionMatrix(y_rf,testing_set$Status)
cm.rf_pca_fs
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction 0 1
##          0 9 3
##          1 1 8
##
```

```
##                Accuracy : 0.8095
##                  95% CI : (0.5809, 0.9455)
##     No Information Rate : 0.5238
##     P-Value [Acc > NIR] : 0.006689
##
##                   Kappa : 0.6216
##
##  Mcnemar's Test P-Value : 0.617075
##
##             Sensitivity : 0.9000
##             Specificity : 0.7273
##          Pos Pred Value : 0.7500
##          Neg Pred Value : 0.8889
##              Prevalence : 0.4762
##          Detection Rate : 0.4286
##    Detection Prevalence : 0.5714
##       Balanced Accuracy : 0.8136
##
##        'Positive' Class : 0
##
```

```
cm.fs.rf.pca<-round(cm.rf_pca_fs$overall["Accuracy"],3)

# LDA model
fit.lda.pca<-caret::train(Status~.,data=training_set,method = "lda",
                          metric="Accuracy",trControl=ctrl)
# Predict test data set
y_lda = predict(fit.lda.pca, newdata = testing_set[-16])
cm.fs.lda.pca<-round(confusionMatrix(y_lda,testing_set$Status)$overall["Accuracy"],3)
cm.fs.lda.pca
```

```
## Accuracy
##     0.81
```

```
# stochastic gradient boosting machine
fit.xgboost.pca<-caret::train(Status~.,data=training_set,method = "xgbTree" ,
                        metric="Accuracy" ,trControl=ctrl,verbose = FALSE)
# Predict test data set
y_xgboost = predict(fit.xgboost.pca, newdata = testing_set[-16])
cm.xgboost_pca_fs<-confusionMatrix(y_xgboost,testing_set$Status)
cm.xgboost_pca_fs
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 10  3
##          1  0  8
##
##                Accuracy : 0.8571
##                  95% CI : (0.6366, 0.9695)
##     No Information Rate : 0.5238
##     P-Value [Acc > NIR] : 0.001511
##
##                   Kappa : 0.7175
```

```
##
##  Mcnemar's Test P-Value : 0.248213
##
##             Sensitivity : 1.0000
##             Specificity : 0.7273
##          Pos Pred Value : 0.7692
##          Neg Pred Value : 1.0000
##              Prevalence : 0.4762
##          Detection Rate : 0.4762
##    Detection Prevalence : 0.6190
##       Balanced Accuracy : 0.8636
##
##        'Positive' Class : 0
##
```

```
cm.fs.xgboost.pca<-round(cm.xgboost_pca_fs$overall["Accuracy"],3)
##
```

#3.PCA reduction method

```
knitr::opts_chunk$set(echo = TRUE)

#2. Apply PCA dimentional reduction

pca = preProcess(x =train_cleaned[-2] , method = 'pca', pcaComp = 2)
training_set = predict(pca, train_cleaned)
training_set = training_set[c(2, 3, 1)]
testing_set = predict(pca, test_cleaned)
testing_set = testing_set[c(2, 3, 1)]

# SVM model linear
# 5 fold cross validation
ctrl <-trainControl(method="cv",number=5)
fit.linear.pca<-caret::train(Status~.,data = training_set,method = "svmLinear",
                             metric="Accuracy" ,trControl=ctrl)
# Predict test data set
y_svmlinear = predict(fit.linear.pca, newdata = testing_set[-3])
cm.svmlinear.pca<-round(confusionMatrix(y_svmlinear,testing_set$Status)$overall["Accuracy"],3)
# SVM model Radial
fit.rad.pca<-caret::train(Status~.,data=training_set,method = "svmRadial" ,
                          metric="Accuracy" ,trControl=ctrl)
# Predict test data set
y_svmRedial = predict(fit.rad.pca, newdata = testing_set[-3])
cm.svmradial.pca<-round(confusionMatrix(y_svmRedial,testing_set$Status)$overall["Accuracy"],3)
# svm poly model
fit.poly.pca<-caret::train(Status~.,data=training_set,method = "svmPoly",
                           metric="Accuracy" ,trControl=ctrl)
# Predict test data set
y_svmpoly = predict(fit.poly.pca, newdata = testing_set[-3])
cm.poly_pca<-confusionMatrix(y_svmpoly,testing_set$Status)
cm.poly_pca
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction 0 1
```

```
##          0 7 2
##          1 3 9
##
##                Accuracy : 0.7619
##                  95% CI : (0.5283, 0.9178)
##     No Information Rate : 0.5238
##     P-Value [Acc > NIR] : 0.02269
##
##                   Kappa : 0.5205
##
##  Mcnemar's Test P-Value : 1.00000
##
##             Sensitivity : 0.7000
##             Specificity : 0.8182
##          Pos Pred Value : 0.7778
##          Neg Pred Value : 0.7500
##              Prevalence : 0.4762
##          Detection Rate : 0.3333
##    Detection Prevalence : 0.4286
##       Balanced Accuracy : 0.7591
##
##        'Positive' Class : 0
##
```

```
cm.svmpoly.pca<-round(cm.poly_pca$overall["Accuracy"],3)
# logitBoost model
fit.logit.pca<-caret::train(Status~.,data=training_set,method = "LogitBoost",
                            metric="Accuracy" ,trControl=ctrl)
# Predict test data set
y_logit = predict(fit.logit.pca, newdata = testing_set[-3])
cm.logit.pca<-round(confusionMatrix(y_logit,testing_set$Status)$overall["Accuracy"],3)
# Random forest model
fit.rf.pca<-caret::train(Status~.,data=training_set,method = "rf",
                         metric="Accuracy",trControl=ctrl)
```

```
## note: only 1 unique complexity parameters in default grid. Truncating the grid to 1 .
```

```
# Predict test data set
y_rf = predict(fit.rf.pca, newdata = testing_set[-3])
cm.rf.pca<-round(confusionMatrix(y_rf,testing_set$Status)$overall["Accuracy"],3)
# LDA model
fit.lda.pca<-caret::train(Status~.,data=training_set,method = "lda",
                          metric="Accuracy",trControl=ctrl)
# Predict test data set
y_lda = predict(fit.lda.pca, newdata = testing_set[-3])
cm.lda.pca<-round(confusionMatrix(y_lda,testing_set$Status)$overall["Accuracy"],3)
# stochastic gradient boosting machine
fit.xgboost.pca<-caret::train(Status~.,data=training_set,method = "xgbTree" ,
                              metric="Accuracy" ,trControl=ctrl,verbose = FALSE)
# Predict test data set
y_xgboost = predict(fit.xgboost.pca, newdata = testing_set[-3])
cm.xgboost.pca<-round(confusionMatrix(y_xgboost,testing_set$Status)$overall["Accuracy"],3)
```

```
#Method Comparsion
model.compare<-data.frame(
```

```
    Model=c(  "SvmPoly","XGBOOST","LogitBoost","SvmLinear","RF","SvmRadial",
              "PDA","LDA"),
    Full= c(cm.poly.both,cm.xgb.both,cm.logit.both,cm.linear.both,
            cm.rf.both,cm.radial.both,"NULL",cm.ld.both),
    FS=   c(cm.svmpoly.fs, cm.xgboost.fs,  cm.logit.fs,  cm.svmlinear.fs,
            cm.rf.fs,  cm.svmradial.fs,cm.pda.fs,cm.lda.fs),
  PCA_FS=c(cm.fs.poly.pca,cm.fs.xgboost.pca,cm.fs.logit.pca,cm.fs.linear.pca,
           cm.fs.rf.pca,cm.fs.radial.pca,"n/a",cm.fs.lda.pca),
    PCA=  c(cm.svmpoly.pca,cm.xgboost.pca, cm.logit.pca,cm.svmlinear.pca,
            cm.rf.pca, cm.svmradial.pca, "n/a",cm.lda.pca))


model.compare
```

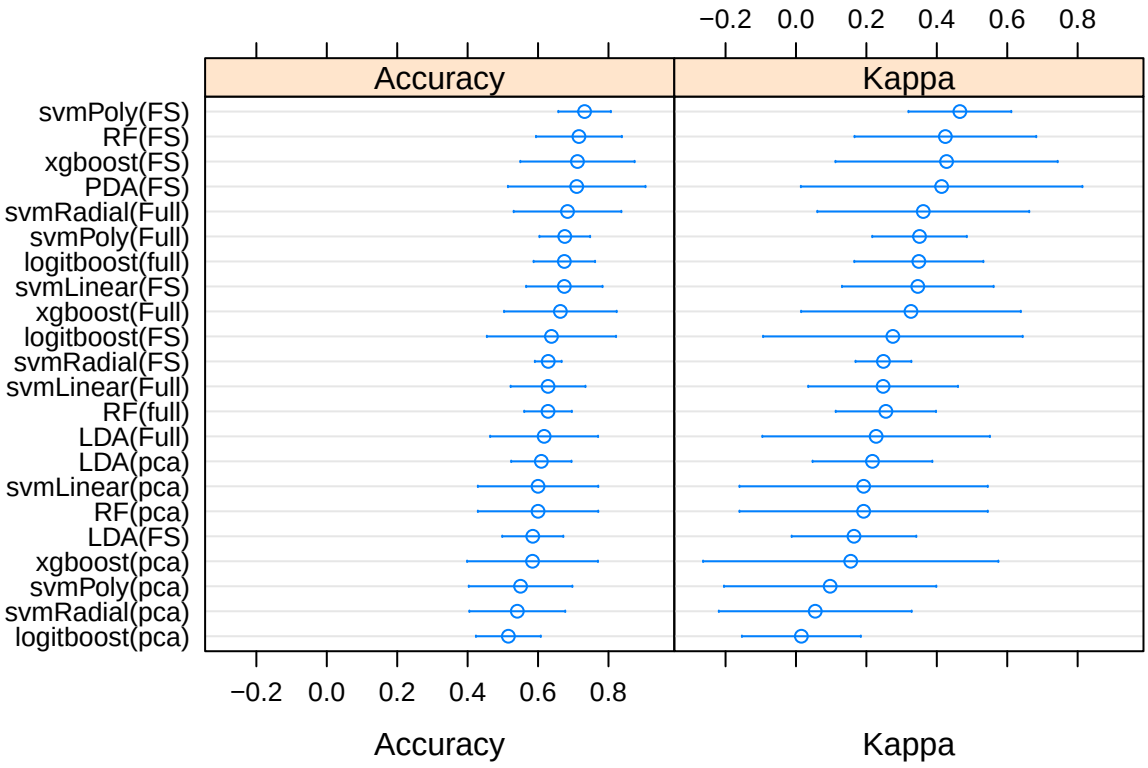
```
##        Model  Full    FS PCA_FS   PCA
## 1    SvmPoly  0.81 0.714   0.81 0.762
## 2    XGBOOST 0.571 0.667  0.857 0.571
## 3 LogitBoost 0.571 0.524  0.714 0.476
## 4  SvmLinear  0.81 0.762  0.905 0.571
## 5         RF 0.571 0.619   0.81 0.619
## 6  SvmRadial 0.571 0.619  0.714 0.619
## 7        PDA  NULL 0.762    n/a   n/a
## 8        LDA 0.857 0.524   0.81 0.571
```

```
results <-resamples(list("svmPoly(FS)"=fit.poly.fs,"svmPoly(Full)"=fit.poly.full,
           "svmPoly(pca)"=fit.poly.pca,"xgboost(FS)"=fit.xgboost.fs,
          "xgboost(Full)"=fit.xgboost.full,"xgboost(pca)"=fit.xgboost.pca,
           "logitboost(FS)"=fit.logit.fs,"logitboost(full)"=fit.logit.full,
         "logitboost(pca)"=fit.logit.pca,"svmLinear(FS)"=fit.linear.fs,
      "svmLinear(Full)"=fit.linear.full,"svmLinear(pca)"=fit.linear.pca,
      "RF(FS)"=fit.rf.fs,"RF(full)"=fit.rf.full,"RF(pca)"=fit.linear.pca,
         "svmRadial(FS)"=fit.rad.fs,"svmRadial(Full)"=fit.radial.full,
    "svmRadial(pca)"=fit.rad.pca,"PDA(FS)"=fit.pda.fs,"LDA(FS)"=fit.lda.fs,
             "LDA(Full)"=fit.lda.full,"LDA(pca)"=fit.lda.pca))

nums<-c(cm.svmpoly.fs,cm.poly.both,cm.svmpoly.pca,cm.fs.poly.pca,
        cm.xgboost.fs,cm.xgb.both,cm.xgboost.pca,cm.fs.xgboost.pca,
        cm.logit.fs,cm.logit.both,cm.logit.pca,cm.fs.logit.pca,
        cm.svmlinear.fs,cm.linear.both,cm.svmlinear.pca,cm.fs.linear.pca,
        cm.rf.fs,cm.rf.both,cm.rf.pca,cm.fs.rf.pca,
        cm.svmradial.fs,cm.radial.both,cm.svmradial.pca,cm.fs.radial.pca,
        cm.pda.fs,
        cm.lda.fs,cm.ld.both,cm.lda.pca,cm.fs.lda.pca)

model_compare<-data.frame(Model=c("svmPoly(FS)","svmPoly(Full)","svmPoly(pca)",
    "svmPoly(PCA_FS)","xgboost(FS)","xgboost(Full)","xgboost(pca)",
    "xgboost(PCA_FS","logitboost(FS)","logitboost(full)","logitboost(pca)",
   "logitboost(PCA_FS)", "svmLinear(FS)","svmLinear(Full)","svmLinear(pca)",
     "svmLinear(PCA_FS","RF(FS)","RF(full)","RF(pca)","RF(PCA_FS",
    "svmRadial(FS)","svmRadial(Full)","svmRadial(pca)",
    "svmRadial(PCA_FS","PDA(FS)", "LDA(FS)","LDA(Full)","LDA(pca)",
     "LDA(PCA_FS)"),
        Accuracy=c(nums))
```

```r
model_compare[with(model_compare,order(Accuracy,decreasing = T)),]
```

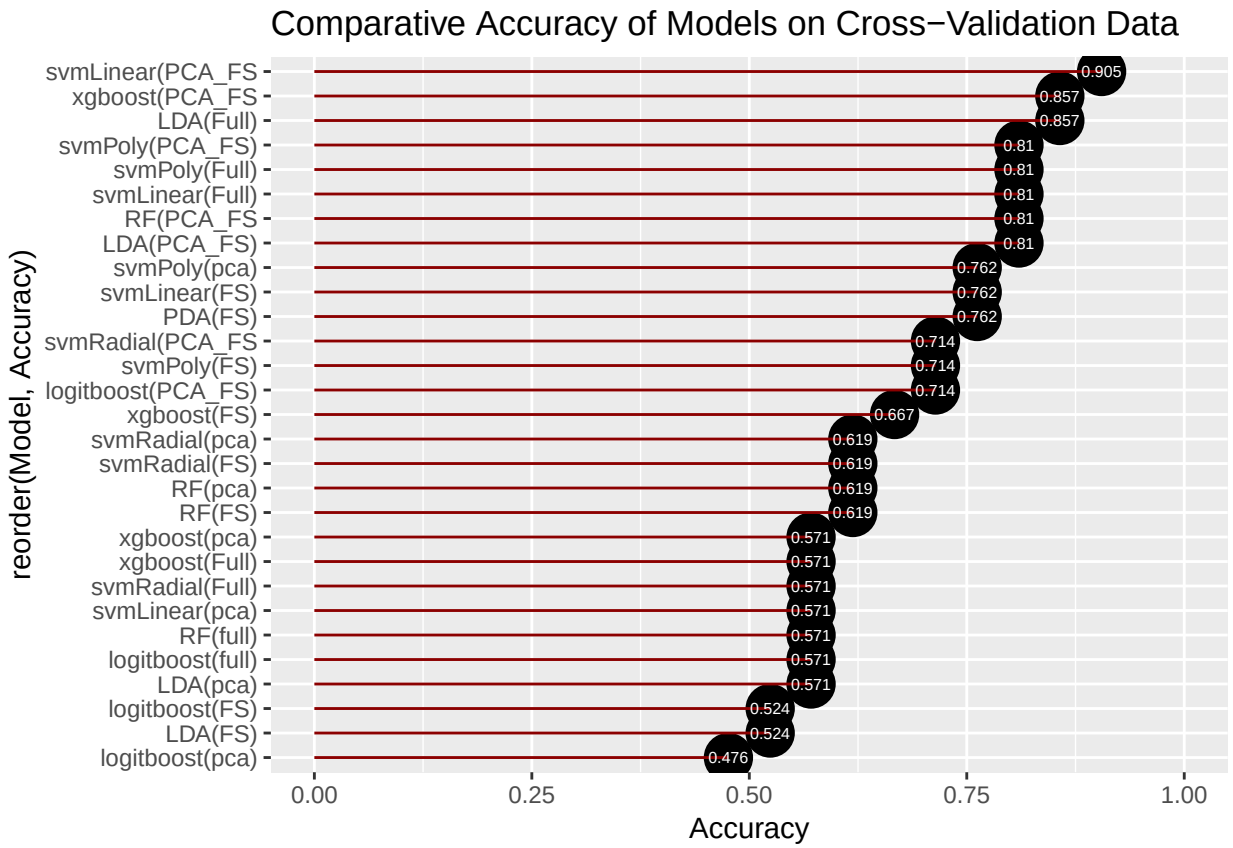
```
##                 Model Accuracy
## 16   svmLinear(PCA_FS    0.905
## 8      xgboost(PCA_FS    0.857
## 27          LDA(Full)    0.857
## 2       svmPoly(Full)    0.810
## 4     svmPoly(PCA_FS)    0.810
## 14    svmLinear(Full)    0.810
## 20          RF(PCA_FS    0.810
## 29        LDA(PCA_FS)    0.810
## 3        svmPoly(pca)    0.762
## 13      svmLinear(FS)    0.762
## 25            PDA(FS)    0.762
## 1         svmPoly(FS)    0.714
## 12 logitboost(PCA_FS)    0.714
## 24   svmRadial(PCA_FS    0.714
## 5         xgboost(FS)    0.667
## 17             RF(FS)    0.619
## 19            RF(pca)    0.619
## 21      svmRadial(FS)    0.619
## 23     svmRadial(pca)    0.619
## 6       xgboost(Full)    0.571
## 7        xgboost(pca)    0.571
## 10   logitboost(full)    0.571
## 15     svmLinear(pca)    0.571
## 18           RF(full)    0.571
## 22    svmRadial(Full)    0.571
## 28           LDA(pca)    0.571
## 9      logitboost(FS)    0.524
## 26            LDA(FS)    0.524
## 11    logitboost(pca)    0.476
```

```r
# Plot results
dotplot(results)
```



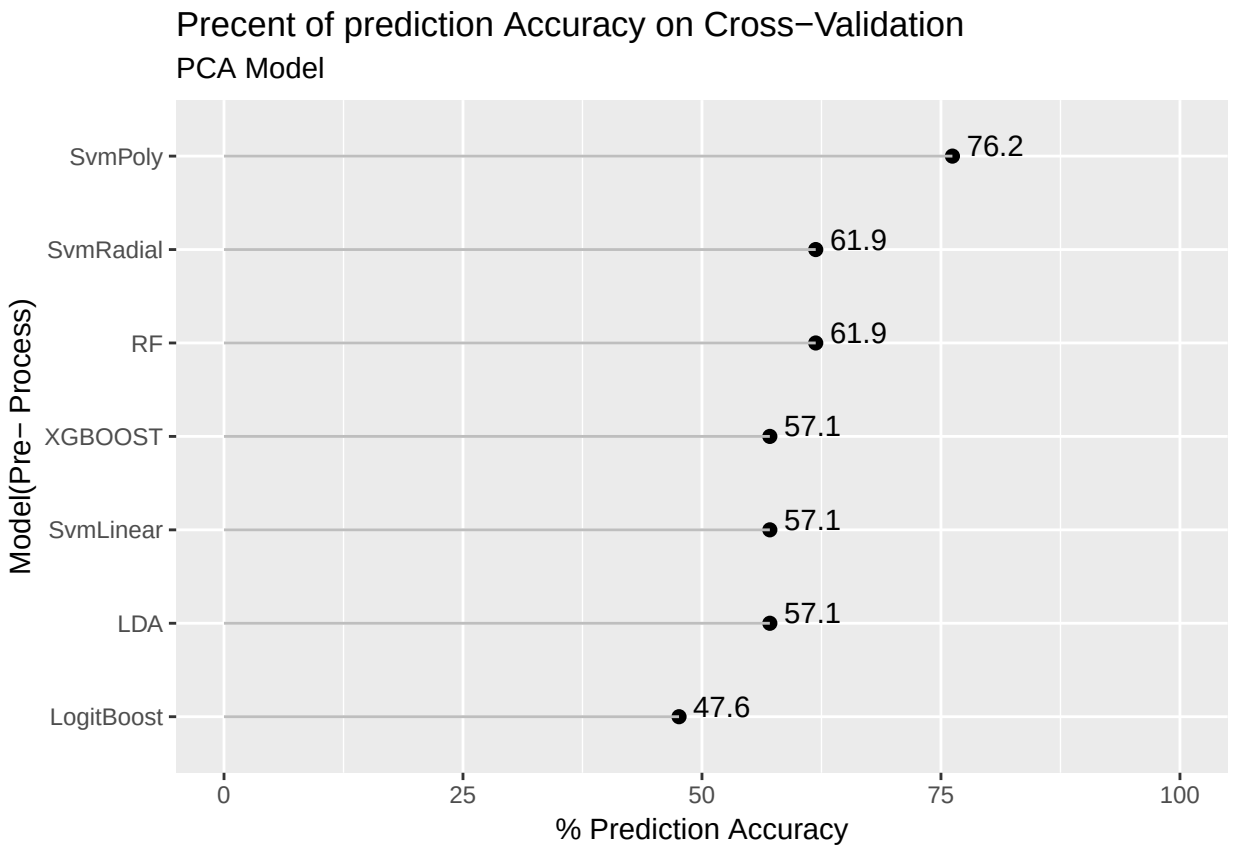
Confidence Level: 0.95

```
ggplot(model_compare,aes(x=reorder(Model,Accuracy), y=Accuracy,
                         label=round(Accuracy,3))) +
  geom_point(stat = "identity",fill="darkred",size=8) +
geom_segment(aes(y=0,x=Model,yend=Accuracy,xend=Model),color="darkred")  +
  geom_text(color="white", size=2)    +
  labs(title = 'Comparative Accuracy of Models on Cross-Validation Data') +
        ylim(0, 1)  +coord_flip()
```

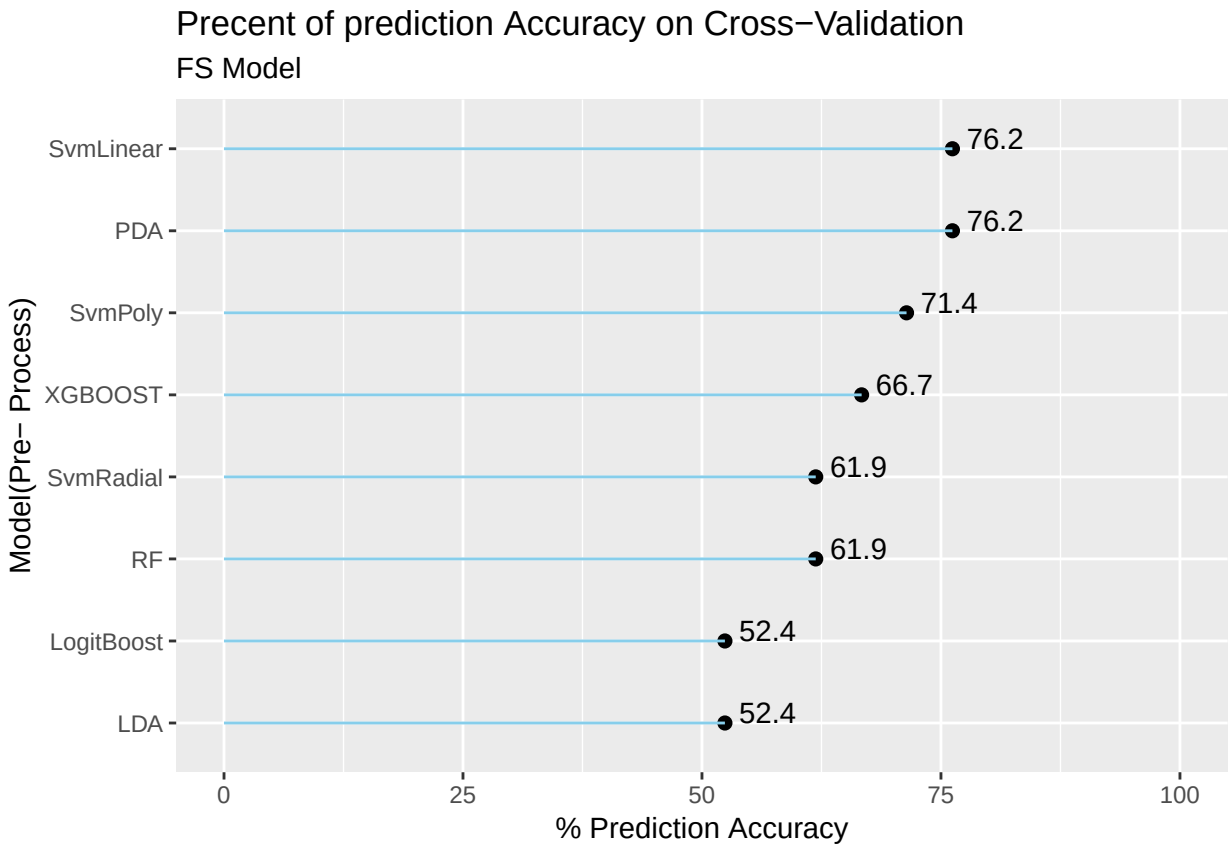



```
model_PCA<-data.frame(Model=c("SvmPoly","XGBOOST","LogitBoost",
                "SvmLinear", "RF",  "SvmRadial","LDA"),
Accuracy=c(cm.svmpoly.pca*100,cm.xgboost.pca*100,cm.logit.pca*100,
           cm.svmlinear.pca*100,cm.rf.pca*100,
           cm.svmradial.pca*100,cm.lda.pca*100))

model_FS<-data.frame(Model=c("SvmPoly","XGBOOST","LogitBoost",
                             "SvmLinear", "RF", "SvmRadial","PDA","LDA"),
Accuracy=c(cm.svmpoly.fs*100,cm.xgboost.fs*100,cm.logit.fs*100,
    cm.svmlinear.fs*100,cm.rf.fs*100,cm.svmradial.fs*100,
    cm.pda.fs*100,cm.lda.fs*100))

model_PCA_FS<-data.frame(Model=c("SvmPoly","XGBOOST","LogitBoost",
            "SvmLinear", "RF", "SvmRadial","LDA"),
Accuracy= c(cm.fs.poly.pca*100,cm.fs.xgboost.pca*100,cm.fs.logit.pca*100,
    cm.fs.linear.pca*100,cm.fs.rf.pca*100,cm.fs.radial.pca*100,
    cm.fs.lda.pca*100))

model_FULL<-data.frame(Model=c("SvmPoly","XGBOOST","LogitBoost",
            "SvmLinear", "RF","SvmRadial","LDA"),
Accuracy=c(cm.poly.both*100,cm.xgb.both*100,cm.logit.both*100,
           cm.linear.both*100,cm.rf.both*100,cm.radial.both*100,
           cm.ld.both*100))

# Plot PCA
ggplot(model_PCA,aes(x=reorder(Model,Accuracy), y=Accuracy,
                     label=paste0(round(Accuracy,0),"%"))) +
```

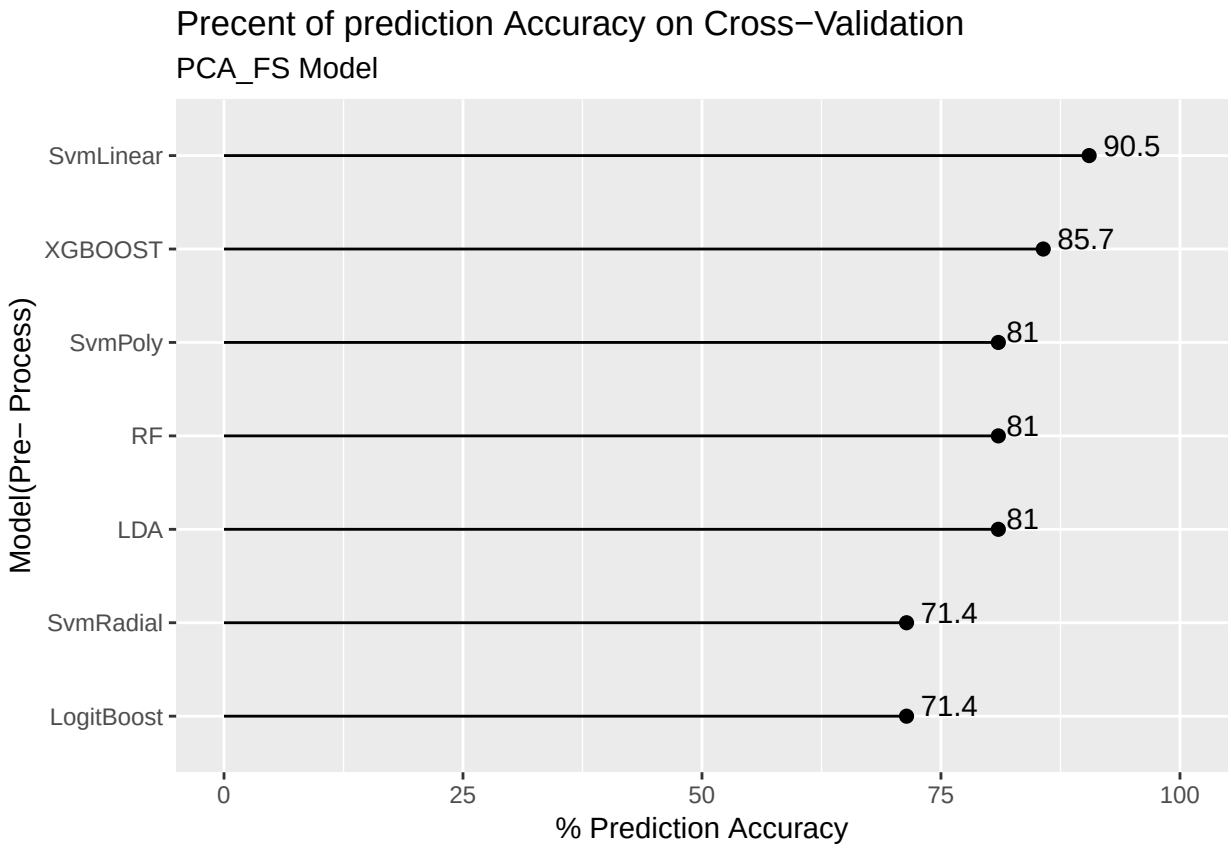
```
  geom_point(stat = "identity",fill= "orange", size = 2) +
 geom_segment(aes(y=0,x=Model,yend=Accuracy,xend=Model),color="gray")  +
 geom_text(aes(label = Accuracy), hjust = -.25,vjust=0)  +
labs(x = "Model(Pre- Process)", y = "% Prediction Accuracy") +
 ggtitle("Precent of prediction Accuracy on Cross-Validation",
         subtitle= "PCA Model") +
       coord_flip() +
       ylim(0, 100)
```



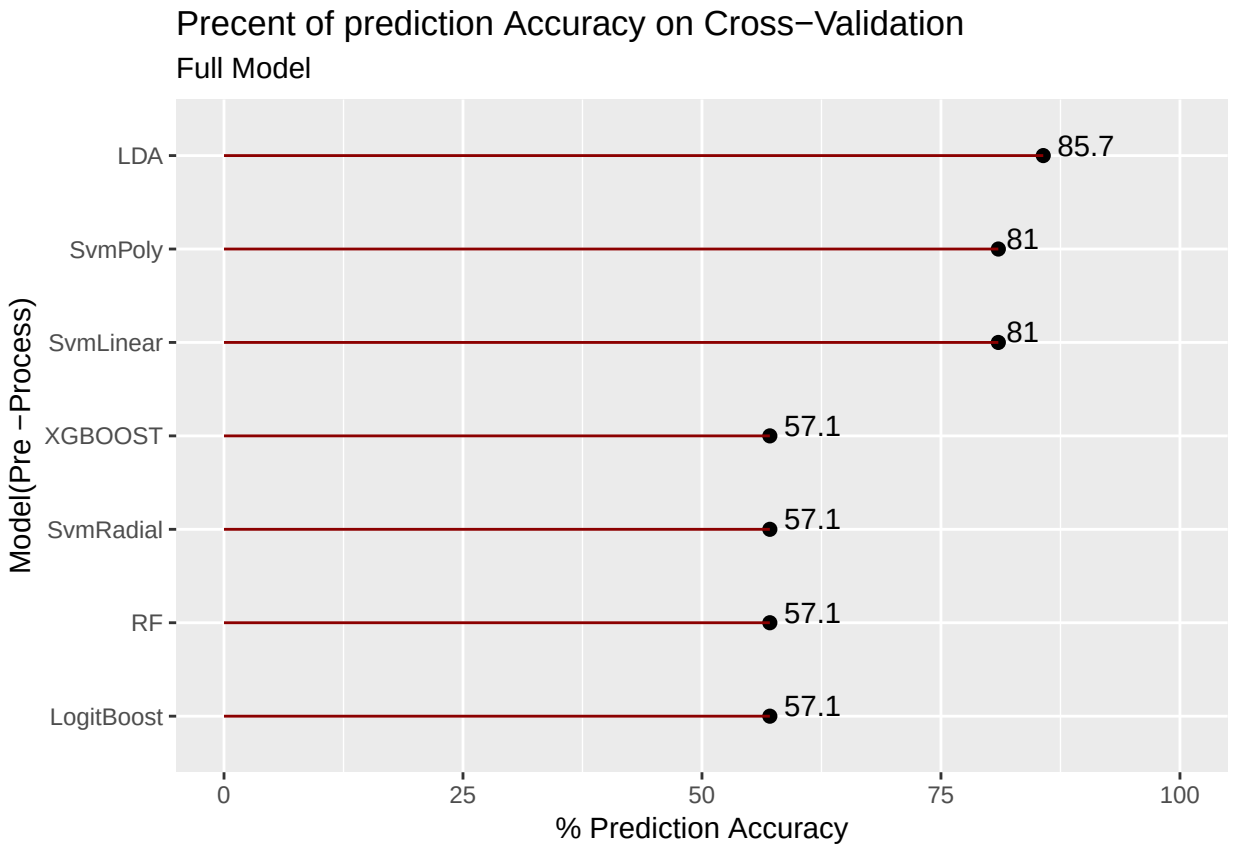

```
#Plot FS
ggplot(model_FS,aes(x=reorder(Model,Accuracy), y=Accuracy,
                    label=paste0(round(Accuracy,0),"%"))) +
   geom_point(fill="blue",stat = "identity", size = 2) +
  geom_segment(aes(y=0,x=Model,yend=Accuracy,xend=Model),color="skyblue") +
  geom_text(aes(label = Accuracy), hjust = -.25,vjust=0)  +
 labs(x = "Model(Pre- Process)", y = "% Prediction Accuracy") +
  ggtitle("Precent of prediction Accuracy on Cross-Validation",
          subtitle= "FS Model") +
   ylim(0, 100) +
  coord_flip()
```




```
# Plot PCA_FS
ggplot(model_PCA_FS,aes(x = reorder(Model, Accuracy), y=Accuracy,
                        label=paste0(round(Accuracy,0),"%"))) +
  geom_point(fill ="orange",stat = "identity", size = 2) +
 geom_segment(aes(y=0,x=Model,yend=Accuracy,xend=Model),color="black") +
  geom_text(aes(label = Accuracy), hjust = -.25,vjust=0)  +
 labs(x = "Model(Pre- Process)", y = "% Prediction Accuracy") +
  ggtitle("Precent of prediction Accuracy on Cross-Validation",
          subtitle= "PCA_FS Model") +
   ylim(0, 100) +
  coord_flip()
```




```
#Plot of Full
ggplot(model_FULL,aes(x=reorder(Model,Accuracy), y=Accuracy,
                      label=paste0(round(Accuracy,0),"%"))) +
geom_point(,stat = "identity",fill="darkred", size = 2) +
geom_segment(aes(y=0,x=Model,yend=Accuracy,xend=Model),color="darkred")  +
geom_text(aes(label = Accuracy), hjust = -.25,vjust=0)  +
 labs(x = "Model(Pre -Process)", y = "% Prediction Accuracy") +
  ggtitle("Precent of prediction Accuracy on Cross-Validation",
          subtitle= "Full Model") +
  ylim(0, 100) +
  coord_flip()
```




```
nums2<-c(cm.svmpoly.fs*100,cm.poly.both*100,cm.svmpoly.pca*100,
         cm.fs.poly.pca*100,cm.xgboost.fs*100,cm.xgb.both*100,
         cm.xgboost.pca*100,cm.fs.xgboost.pca*100,cm.logit.fs*100,
         cm.logit.both*100,cm.logit.pca*100,cm.fs.logit.pca*100,
        cm.svmlinear.fs*100,cm.linear.both*100,cm.svmlinear.pca*100,
        cm.fs.linear.pca*100,cm.rf.fs*100,cm.rf.both*100,cm.rf.pca*100,
        cm.fs.rf.pca*100,cm.svmradial.fs*100,cm.radial.both*100,
        cm.svmradial.pca*100,cm.fs.radial.pca*100,cm.pda.fs*100,
        cm.lda.fs*100,cm.ld.both*100,cm.lda.pca*100,cm.fs.lda.pca*100)

model_compare2<-data.frame(Model=c("svmPoly(FS)","svmPoly(Full)",
                                   "svmPoly(pca)","svmPoly(PCA_FS)"
           ,"xgboost(FS)","xgboost(Full)","xgboost(pca)","xgboost(PCA_FS",
           "logitboost(FS)","logitboost(full)","logitboost(pca)",
           "logitboost(PCA_FS)",
             "svmLinear(FS)","svmLinear(Full)","svmLinear(pca)",
           "svmLinear(PCA_FS","RF(FS)","RF(full)","RF(pca)","RF(PCA_FS",
             "svmRadial(FS)","svmRadial(Full)","svmRadial(pca)",
           "svmRadial(PCA_FS",
           "PDA(FS)", "LDA(FS)","LDA(Full)","LDA(pca)","LDA(PCA_FS)"),
           Accuracy=c(nums2))
df<-model_compare2[with(model_compare2,order(Accuracy,decreasing = T)),]
df_9<-df[1:8,]

 #Plot Top 9 Accuracy prediction model
ggplot(df_9,aes(x=reorder(Model,Accuracy), y=Accuracy,
                    label=paste0(round(Accuracy,0),"%"))) +
```

```
geom_point(fill ="darkred",stat = "identity", size = 2) +
geom_segment(aes(y=0,x=Model,yend=Accuracy,xend=Model),color="darkred")  +
geom_text(aes(label = Accuracy), hjust = -.25)  +
 labs(x = "Model(Pre -Process)", y = "% Prediction Accuracy") +
  ggtitle("Precent of prediction Accuracy ",
          subtitle= "Top 8 Models") +
  ylim(0, 100) +
  coord_flip()
```