Appendix:

The code is available on https://github.com/AlexLeungZ/SDSC3006-proj. You can follow the instructions from the README.md file. The following codes run on a Jupyter notebook with IRkernel, assume required library are installed.

Project Repository

https://github.com/AlexLeungZ/SDSC3006-proj

Code on Github

https://github.com/AlexLeungZ/SDSC3006-proj/blob/main/code/proj.ipynb

Steel Plates Faults Data Set

https://archive.ics.uci.edu/ml/machine-learning-databases/00198/Faults.NNA

https://archive.ics.uci.edu/ml/machine-learning-databases/00198/Faults27x7_var

Project Code

```
# SDSC3006 Group Project
# Steel Plates Faults Dataset (27 attributes, 1941 instances)


# Loading library
suppressPackageStartupMessages({
    library(repr)
    library(ggplot2)
    library(reshape2)
    library(pROC)
    library(class)
    library(caret)
    library(e1071)
    library(gbm)
    library(kernlab)
    library(randomForest)
})


# Supporting functions
setPlotSize <- function(wRatio, hRatio) {
    options(repr.plot.width = wRatio * repr_option_defaults$repr.plot.width)
    options(repr.plot.height = hRatio * repr_option_defaults$repr.plot.height)
}

plotCM2Heatmap <- function(table) {
    ggplot(data = melt(table), aes(x = Prediction, y = Reference, fill = value)) +
        geom_tile(color = "black") +
        geom_text(aes(label = value)) +
        scale_fill_gradientn(colours = heat.colors(100, rev = TRUE)) +
        coord_fixed() +
        theme_grey(base_size = 14) +
```

```
        theme(axis.text.x = element_text(angle = 315, hjust = 0))
}

plotMultiRoc <- function(prediction, predictor, arrow) {
    set.seed(0)
    auc <- multiclass.roc(prediction, predictor, direction = arrow)
    for (i in 1:length(auc$rocs)) {
        plot.roc(
            auc$rocs[[i]],
            add = (if (i == 1) F else T),
            legacy.axes = T,
            lwd = 2,
            col = sample.int(100)
        )
    }
}

cv <- trainControl(method = "repeatedcv", number = 10)
train.Res <- list()
train.PcaRes <- list()
test.Res <- list()
test.PcaRes <- list()


# Set random seed
seed <- 0
print(seed)


# Load dataset
df <- read.table("Faults.NNA", col.names = as.vector(read.table("Faults27x7_var")$V1))
df.X <- scale(df[1:27])
df.Y <- data.frame(Faults = factor(names(df[28:34])[max.col(df[28:34])]))

setPlotSize(3, 1)
ggplot(data = melt(cor(df.X, df[28:34])), aes(x = Var1, y = Var2, fill = value)) +
    geom_tile(color = "black") +
    scale_fill_gradientn(colours = heat.colors(100, rev = TRUE)) +
    coord_fixed() +
    theme_grey(base_size = 14) +
    theme(axis.text.x = element_text(angle = 315, hjust = 0))

setPlotSize(1, 1)


# PCA Tranformation
df.pca <- prcomp(df.X, scale = TRUE)
df.pcaX <- df.pca$x[, 1:12]

ggplot(cbind(df.pcaX, df.Y), aes(x = PC1, y = PC2, color = Faults)) +
    geom_point() +
    stat_ellipse(level = 0.95, show.legend = F) +
    theme_grey(base_size = 14) +
    theme(legend.position = c(0.2, 0.2))
```

```r
impt <- melt(summary(df.pca)$importance[3, ])
ggplot(cbind(key = 1:nrow(impt), impt), aes(x = key, y = value, group = 1)) +
    geom_line(color = "grey") +
    geom_point(shape = 21, color = "black", fill = "purple", size = 3) +
    theme_grey(base_size = 14) +
    xlab("Principal Component") +
    ylab("Cumulative Variance Explained")


# Split dataset
set.seed(seed)
rand <- sample(nrow(df), nrow(df) * 0.8)

df.train.X <- df.X[rand, ]
df.train.Y <- df.Y[rand, ]
df.train.pcaX <- df.pcaX[rand, ]

df.test.X <- df.X[-rand, ]
df.test.Y <- df.Y[-rand, ]
df.test.pcaX <- df.pcaX[-rand, ]


# Classifiers


# Naive Bayes
# With PCA
set.seed(seed)
gnb.pca <- naiveBayes(df.train.pcaX, df.train.Y)
gnb.train.yPredPca <- predict(gnb.pca, df.train.pcaX)
gnb.test.yPredPca <- predict(gnb.pca, df.test.pcaX)

gnb.train.cmPca <- confusionMatrix(gnb.train.yPredPca, df.train.Y)
gnb.test.cmPca <- confusionMatrix(gnb.test.yPredPca, df.test.Y)
train.PcaRes <- c(train.PcaRes, gnb = gnb.train.cmPca$overall["Accuracy"])
test.PcaRes <- c(test.PcaRes, gnb = gnb.test.cmPca$overall["Accuracy"])

print("Naive Bayes training set with PCA Accuracy")
gnb.train.cmPca$overall["Accuracy"]

print("Naive Bayes testing set with PCA Accuracy")
gnb.test.cmPca$overall["Accuracy"]

# Confusion Matrix Heat map and Area under Curve
plotMultiRoc(gnb.test.yPredPca, as.numeric(df.test.Y), "<")
plotCM2Heatmap(gnb.test.cmPca$table)


# Naive Bayes
# Without PCA
set.seed(seed)
gnb.model <- naiveBayes(df.train.X, df.train.Y)
gnb.train.yPred <- predict(gnb.model, df.train.X)
```

```r
gnb.test.yPred <- predict(gnb.model, df.test.X)

gnb.train.cm <- confusionMatrix(gnb.train.yPred, df.train.Y)
gnb.test.cm <- confusionMatrix(gnb.test.yPred, df.test.Y)
train.Res <- c(train.Res, gnb = gnb.train.cm$overall["Accuracy"])
test.Res <- c(test.Res, gnb = gnb.test.cm$overall["Accuracy"])

print("Naive Bayes training set Accuracy")
gnb.train.cm$overall["Accuracy"]

print("Naive Bayes testing set Accuracy")
gnb.test.cm$overall["Accuracy"]

# Confusion Matrix Heat map and Area under Curve
plotMultiRoc(gnb.test.yPred, as.numeric(df.test.Y), "<")
plotCM2Heatmap(gnb.test.cm$table)


# Multinomial Logistic Regression
# With PCA
set.seed(seed)
mlg.pca <- train(df.train.pcaX, df.train.Y, method = "multinom", trControl = cv, trace = FALSE)
mlg.train.yPredPca <- predict(mlg.pca, df.train.pcaX)
mlg.test.yPredPca <- predict(mlg.pca, df.test.pcaX)

mlg.train.cmPca <- confusionMatrix(mlg.train.yPredPca, df.train.Y)
mlg.test.cmPca <- confusionMatrix(mlg.test.yPredPca, df.test.Y)
train.PcaRes <- c(train.PcaRes, mlg = mlg.train.cmPca$overall["Accuracy"])
test.PcaRes <- c(test.PcaRes, mlg = mlg.test.cmPca$overall["Accuracy"])

print("Multinomial Logistic Regression training set with PCA Accuracy")
mlg.train.cmPca$overall["Accuracy"]

print("Multinomial Logistic Regression testing set with PCA Accuracy")
mlg.test.cmPca$overall["Accuracy"]

mlg.pca
plot(mlg.pca)

# Confusion Matrix Heat map and Area under Curve
plotMultiRoc(mlg.test.yPredPca, as.numeric(df.test.Y), "<")
plotCM2Heatmap(mlg.test.cmPca$table)


# Multinomial Logistic Regression
# Without PCA
set.seed(seed)
mlg.model <- train(df.train.X, df.train.Y, method = "multinom", trControl = cv, trace = FALSE)
mlg.train.yPred <- predict(mlg.model, df.train.X)
mlg.test.yPred <- predict(mlg.model, df.test.X)

mlg.train.cm <- confusionMatrix(mlg.train.yPred, df.train.Y)
mlg.test.cm <- confusionMatrix(mlg.test.yPred, df.test.Y)
train.Res <- c(train.Res, mlg = mlg.train.cm$overall["Accuracy"])
```

```r
test.Res <- c(test.Res, mlg = mlg.test.cm$overall["Accuracy"])

print("Multinomial Logistic Regression training set Accuracy")
mlg.train.cm$overall["Accuracy"]

print("Multinomial Logistic Regression testing set Accuracy")
mlg.test.cm$overall["Accuracy"]

mlg.model
plot(mlg.model)

# Confusion Matrix Heat map and Area under Curve
plotMultiRoc(mlg.test.yPred, as.numeric(df.test.Y), "<")
plotCM2Heatmap(mlg.test.cm$table)


# K Nearest Neighbor
# With PCA
set.seed(seed)
knn.pca <- tune(gknn, df.train.pcaX, df.train.Y, ranges = list(k = 1:10))
knn.train.yPredPca <- predict(knn.pca$best.model, df.train.pcaX)
knn.test.yPredPca <- predict(knn.pca$best.model, df.test.pcaX)

knn.train.cmPca <- confusionMatrix(knn.train.yPredPca, df.train.Y)
knn.test.cmPca <- confusionMatrix(knn.test.yPredPca, df.test.Y)
train.PcaRes <- c(train.PcaRes, knn = knn.train.cmPca$overall["Accuracy"])
test.PcaRes <- c(test.PcaRes, knn = knn.test.cmPca$overall["Accuracy"])

print("K Nearest Neighbor training set with PCA Accuracy")
knn.train.cmPca$overall["Accuracy"]

print("K Nearest Neighbor testing set with PCA Accuracy")
knn.test.cmPca$overall["Accuracy"]

summary(knn.pca)
ggplot(knn.pca$performances, aes(x = k, y = error)) +
    geom_line(color = "darkred") +
    geom_point(shape = 21, color = "black", fill = "black", size = 3) +
    theme_grey(base_size = 14)

# Confusion Matrix Heat map and Area under Curve
plotMultiRoc(knn.test.yPredPca, as.numeric(df.test.Y), "<")
plotCM2Heatmap(knn.test.cmPca$table)


# K Nearest Neighbor
# Without PCA
set.seed(seed)
knn.model <- tune(gknn, df.train.X, df.train.Y, ranges = list(k = 1:10))
knn.train.yPred <- predict(knn.model$best.model, df.train.X)
knn.test.yPred <- predict(knn.model$best.model, df.test.X)

knn.train.cm <- confusionMatrix(knn.train.yPred, df.train.Y)
knn.test.cm <- confusionMatrix(knn.test.yPred, df.test.Y)
```

```r
train.Res <- c(train.Res, knn = knn.train.cm$overall["Accuracy"])
test.Res <- c(test.Res, knn = knn.test.cm$overall["Accuracy"])

print("K Nearest Neighbor training set Accuracy")
knn.train.cm$overall["Accuracy"]

print("K Nearest Neighbor testing set Accuracy")
knn.test.cm$overall["Accuracy"]

summary(knn.model)
ggplot(knn.model$performances, aes(x = k, y = error)) +
    geom_line(color = "darkred") +
    geom_point(shape = 21, color = "black", fill = "black", size = 3) +
    theme_grey(base_size = 14)

# Confusion Matrix Heat map and Area under Curve
plotMultiRoc(knn.test.yPred, as.numeric(df.test.Y), "<")
plotCM2Heatmap(knn.test.cm$table)


# Random Forest
# With PCA
set.seed(seed)
rf.pca <- train(df.train.pcaX, df.train.Y, method = "rf", trControl = cv)
rf.train.yPredPca <- predict(rf.pca, df.train.pcaX)
rf.test.yPredPca <- predict(rf.pca, df.test.pcaX)

rf.train.cmPca <- confusionMatrix(rf.train.yPredPca, df.train.Y)
rf.test.cmPca <- confusionMatrix(rf.test.yPredPca, df.test.Y)
train.PcaRes <- c(train.PcaRes, rf = rf.train.cmPca$overall["Accuracy"])
test.PcaRes <- c(test.PcaRes, rf = rf.test.cmPca$overall["Accuracy"])

print("Random Forest training set with PCA Accuracy")
rf.train.cmPca$overall["Accuracy"]

print("Random Forest testing set with PCA Accuracy")
rf.test.cmPca$overall["Accuracy"]

rf.pca
plot(rf.pca)

# Confusion Matrix Heat map and Area under Curve
plotMultiRoc(rf.test.yPredPca, as.numeric(df.test.Y), "<")
plotCM2Heatmap(rf.test.cmPca$table)


# Random Forest
# Without PCA
set.seed(seed)
rf.model <- train(df.train.X, df.train.Y, method = "rf", trControl = cv)
rf.train.yPred <- predict(rf.model, df.train.X)
rf.test.yPred <- predict(rf.model, df.test.X)

rf.train.cm <- confusionMatrix(rf.train.yPred, df.train.Y)
```

```r
rf.test.cm <- confusionMatrix(rf.test.yPred, df.test.Y)
train.Res <- c(train.Res, rf = rf.train.cm$overall["Accuracy"])
test.Res <- c(test.Res, rf = rf.test.cm$overall["Accuracy"])

print("Random Forest training set Accuracy")
rf.train.cm$overall["Accuracy"]

print("Random Forest testing set Accuracy")
rf.test.cm$overall["Accuracy"]

rf.model
plot(rf.model)

# Confusion Matrix Heat map and Area under Curve
plotMultiRoc(rf.test.yPred, as.numeric(df.test.Y), "<")
plotCM2Heatmap(rf.test.cm$table)

# Boosting
# With PCA
set.seed(seed)
bst.pca <- train(df.train.pcaX, df.train.Y, method = "gbm", trControl = cv, verbose = FALSE)
bst.train.yPredPca <- predict(bst.pca, df.train.pcaX)
bst.test.yPredPca <- predict(bst.pca, df.test.pcaX)

bst.train.cmPca <- confusionMatrix(bst.train.yPredPca, df.train.Y)
bst.test.cmPca <- confusionMatrix(bst.test.yPredPca, df.test.Y)
train.PcaRes <- c(train.PcaRes, bst = bst.train.cmPca$overall["Accuracy"])
test.PcaRes <- c(test.PcaRes, bst = bst.test.cmPca$overall["Accuracy"])

print("Boosting training set with PCA Accuracy")
bst.train.cmPca$overall["Accuracy"]

print("Boosting testing set with PCA Accuracy")
bst.test.cmPca$overall["Accuracy"]

bst.pca
plot(bst.pca)

# Confusion Matrix Heat map and Area under Curve
plotMultiRoc(bst.test.yPredPca, as.numeric(df.test.Y), "<")
plotCM2Heatmap(bst.test.cmPca$table)


# Boosting
# Without PCA
set.seed(seed)
bst.model <- train(df.train.X, df.train.Y, method = "gbm", trControl = cv, verbose = FALSE)
bst.train.yPred <- predict(bst.model, df.train.X)
bst.test.yPred <- predict(bst.model, df.test.X)

bst.train.cm <- confusionMatrix(bst.train.yPred, df.train.Y)
bst.test.cm <- confusionMatrix(bst.test.yPred, df.test.Y)
train.Res <- c(train.Res, bst = bst.train.cm$overall["Accuracy"])
test.Res <- c(test.Res, bst = bst.test.cm$overall["Accuracy"])
```

```r
print("Boosting training set Accuracy")
bst.train.cm$overall["Accuracy"]

print("Boosting testing set Accuracy")
bst.test.cm$overall["Accuracy"]

bst.model
plot(bst.model)

# Confusion Matrix Heat map and Area under Curve
plotMultiRoc(bst.test.yPred, as.numeric(df.test.Y), "<")
plotCM2Heatmap(bst.test.cm$table)


# Learning Vector Quantization
# With PCA
set.seed(seed)
lvq.pca <- train(df.train.pcaX, df.train.Y, method = "lvq", trControl = cv)
lvq.train.yPredPca <- predict(lvq.pca, df.train.pcaX)
lvq.test.yPredPca <- predict(lvq.pca, df.test.pcaX)

lvq.train.cmPca <- confusionMatrix(lvq.train.yPredPca, df.train.Y)
lvq.test.cmPca <- confusionMatrix(lvq.test.yPredPca, df.test.Y)
train.PcaRes <- c(train.PcaRes, lvq = lvq.train.cmPca$overall["Accuracy"])
test.PcaRes <- c(test.PcaRes, lvq = lvq.test.cmPca$overall["Accuracy"])

print("Learning Vector Quantization training set with PCA Accuracy")
lvq.train.cmPca$overall["Accuracy"]

print("Learning Vector Quantization testing set with PCA Accuracy")
lvq.test.cmPca$overall["Accuracy"]

lvq.pca
plot(lvq.pca)

# Confusion Matrix Heat map and Area under Curve
plotMultiRoc(lvq.test.yPredPca, as.numeric(df.test.Y), "<")
plotCM2Heatmap(lvq.test.cmPca$table)

# Learning Vector Quantization
# Without PCA
set.seed(seed)
lvq.model <- train(df.train.X, df.train.Y, method = "lvq", trControl = cv)
lvq.train.yPred <- predict(lvq.model, df.train.X)
lvq.test.yPred <- predict(lvq.model, df.test.X)

lvq.train.cm <- confusionMatrix(lvq.train.yPred, df.train.Y)
lvq.test.cm <- confusionMatrix(lvq.test.yPred, df.test.Y)
train.Res <- c(train.Res, lvq = lvq.train.cm$overall["Accuracy"])
test.Res <- c(test.Res, lvq = lvq.test.cm$overall["Accuracy"])

print("Learning Vector Quantization training set Accuracy")
lvq.train.cm$overall["Accuracy"]
```

```r
print("Learning Vector Quantization testing set Accuracy")
lvq.test.cm$overall["Accuracy"]

lvq.model
plot(lvq.model)

# Confusion Matrix Heat map and Area under Curve
plotMultiRoc(lvq.test.yPred, as.numeric(df.test.Y), "<")
plotCM2Heatmap(lvq.test.cm$table)


# Support Vector Machine
# With PCA
# Linear Kernel
set.seed(seed)
tuning <- expand.grid(C = 2^seq(-5, 5, 1))
svml.pca <- train(df.train.pcaX, df.train.Y, method = "svmLinear", trControl = cv, tuneGrid =
tuning)
svml.train.yPredPca <- predict(svml.pca, df.train.pcaX)
svml.test.yPredPca <- predict(svml.pca, df.test.pcaX)

svml.train.cmPca <- confusionMatrix(svml.train.yPredPca, df.train.Y)
svml.test.cmPca <- confusionMatrix(svml.test.yPredPca, df.test.Y)
train.PcaRes <- c(train.PcaRes, svml = svml.train.cmPca$overall["Accuracy"])
test.PcaRes <- c(test.PcaRes, svml = svml.test.cmPca$overall["Accuracy"])

print("Support Vector Machine Linear Kernel training set with PCA Accuracy")
svml.train.cmPca$overall["Accuracy"]

print("Support Vector Machine Linear Kernel testing set with PCA Accuracy")
svml.test.cmPca$overall["Accuracy"]

svml.pca
plot(svml.pca)

# Confusion Matrix Heat map and Area under Curve
plotMultiRoc(svml.test.yPredPca, as.numeric(df.test.Y), "<")
plotCM2Heatmap(svml.test.cmPca$table)


# Polynomial Kernel
set.seed(seed)
svmp.pca <- train(df.train.pcaX, df.train.Y, method = "svmPoly", trControl = cv)
svmp.train.yPredPca <- predict(svmp.pca, df.train.pcaX)
svmp.test.yPredPca <- predict(svmp.pca, df.test.pcaX)

svmp.train.cmPca <- confusionMatrix(svmp.train.yPredPca, df.train.Y)
svmp.test.cmPca <- confusionMatrix(svmp.test.yPredPca, df.test.Y)
train.PcaRes <- c(train.PcaRes, svmp = svmp.train.cmPca$overall["Accuracy"])
test.PcaRes <- c(test.PcaRes, svmp = svmp.test.cmPca$overall["Accuracy"])

print("Support Vector Machine Polynomial Kernel training set with PCA Accuracy")
svmp.train.cmPca$overall["Accuracy"]
```

```r
print("Support Vector Machine Polynomial Kernel testing set with PCA Accuracy")
svmp.test.cmPca$overall["Accuracy"]

svmp.pca
plot(svmp.pca)

# Confusion Matrix Heat map and Area under Curve
plotMultiRoc(svmp.test.yPredPca, as.numeric(df.test.Y), "<")
plotCM2Heatmap(svmp.test.cmPca$table)


# Radial Kernel
set.seed(seed)
svmr.pca <- train(df.train.pcaX, df.train.Y, method = "svmRadial", trControl = cv)
svmr.train.yPredPca <- predict(svmr.pca, df.train.pcaX)
svmr.test.yPredPca <- predict(svmr.pca, df.test.pcaX)

svmr.train.cmPca <- confusionMatrix(svmr.train.yPredPca, df.train.Y)
svmr.test.cmPca <- confusionMatrix(svmr.test.yPredPca, df.test.Y)
train.PcaRes <- c(train.PcaRes, svmr = svmr.train.cmPca$overall["Accuracy"])
test.PcaRes <- c(test.PcaRes, svmr = svmr.test.cmPca$overall["Accuracy"])

print("Support Vector Machine Radial Kernel training set with PCA Accuracy")
svmr.train.cmPca$overall["Accuracy"]

print("Support Vector Machine Radial Kernel testing set with PCA Accuracy")
svmr.test.cmPca$overall["Accuracy"]

svmr.pca
plot(svmr.pca)

# Confusion Matrix Heat map and Area under Curve
plotMultiRoc(svmr.test.yPredPca, as.numeric(df.test.Y), "<")
plotCM2Heatmap(svmr.test.cmPca$table)


# Support Vector Machine
# Without PCA
# Linear Kernel
set.seed(seed)
tuning <- expand.grid(C = 2^seq(-5, 5, 1))
svml.model <- train(df.train.X, df.train.Y, method = "svmLinear", trControl = cv, tuneGrid =
tuning)
svml.train.yPred <- predict(svml.model, df.train.X)
svml.test.yPred <- predict(svml.model, df.test.X)

svml.train.cm <- confusionMatrix(svml.train.yPred, df.train.Y)
svml.test.cm <- confusionMatrix(svml.test.yPred, df.test.Y)
train.Res <- c(train.Res, svml = svml.train.cm$overall["Accuracy"])
test.Res <- c(test.Res, svml = svml.test.cm$overall["Accuracy"])

print("Support Vector Machine Linear Kernel training set Accuracy")
svml.train.cm$overall["Accuracy"]
```

```r
print("Support Vector Machine Linear Kernel testing set Accuracy")
svml.test.cm$overall["Accuracy"]

svml.model
plot(svml.model)

# Confusion Matrix Heat map and Area under Curve
plotMultiRoc(svml.test.yPred, as.numeric(df.test.Y), "<")
plotCM2Heatmap(svml.test.cm$table)


# Polynomial Kernel
set.seed(seed)
svmp.model <- train(df.train.X, df.train.Y, method = "svmPoly", trControl = cv)
svmp.train.yPred <- predict(svmp.model, df.train.X)
svmp.test.yPred <- predict(svmp.model, df.test.X)

svmp.train.cm <- confusionMatrix(svmp.train.yPred, df.train.Y)
svmp.test.cm <- confusionMatrix(svmp.test.yPred, df.test.Y)
train.Res <- c(train.Res, svmp = svmp.train.cm$overall["Accuracy"])
test.Res <- c(test.Res, svmp = svmp.test.cm$overall["Accuracy"])

print("Support Vector Machine Polynomial Kernel training set Accuracy")
svmp.train.cm$overall["Accuracy"]

print("Support Vector Machine Polynomial Kernel testing set Accuracy")
svmp.test.cm$overall["Accuracy"]

svmp.model
plot(svmp.model)

# Confusion Matrix Heat map and Area under Curve
plotMultiRoc(svmp.test.yPred, as.numeric(df.test.Y), "<")
plotCM2Heatmap(svmp.test.cm$table)


# Radial Kernel
set.seed(seed)
svmr.model <- train(df.train.X, df.train.Y, method = "svmRadial", trControl = cv)
svmr.train.yPred <- predict(svmr.model, df.train.X)
svmr.test.yPred <- predict(svmr.model, df.test.X)

svmr.train.cm <- confusionMatrix(svmr.train.yPred, df.train.Y)
svmr.test.cm <- confusionMatrix(svmr.test.yPred, df.test.Y)
train.Res <- c(train.Res, svmr = svmr.train.cm$overall["Accuracy"])
test.Res <- c(test.Res, svmr = svmr.test.cm$overall["Accuracy"])

print("Support Vector Machine Polynomial Radial training set Accuracy")
svmr.train.cm$overall["Accuracy"]

print("Support Vector Machine Polynomial Radial testing set Accuracy")
svmr.test.cm$overall["Accuracy"]
```

```r
svmr.model
plot(svmr.model)

# Confusion Matrix Heat map and Area under Curve
plotMultiRoc(svmr.test.yPred, as.numeric(df.test.Y), "<")
plotCM2Heatmap(svmr.test.cm$table)


# Conclusion
# Training set with PCA
train.PcaRes <- data.frame(train.PcaRes)
train.PcaRes


# Hightest Accuracy
train.PcaRes[which(train.PcaRes == max(train.PcaRes))]


# Training set without PCA
train.Res <- data.frame(train.Res)
train.Res


# Hightest Accuracy
train.Res[which(train.Res == max(train.Res))]


# Testing set with PCA
test.PcaRes <- data.frame(test.PcaRes)
test.PcaRes


# Hightest Accuracy
test.PcaRes[which(test.PcaRes == max(test.PcaRes))]


# Testing set without PCA
test.Res <- data.frame(test.Res)
test.Res


# Hightest Accuracy
test.Res[which(test.Res == max(test.Res))]
```