Split into Train and Test Data

```r
grad_data <- read.csv("data.csv", sep = ";")
grad_data$Target <- as.factor(grad_data$Target)

set.seed(1234)

ind = sample(1:nrow(grad_data), size = (nrow(grad_data) * 0.8), replace = FALSE)

train = grad_data[ind, ]
test = grad_data[-ind, ]
```

Decision Tree

```r
library(rpart)
library(rpart.plot)
library(caret)

# Fit standard tree
dat_rpart <- rpart(Target ~ . , data = train, method = 'class')
tree_preds <- predict(dat_rpart, newdata = test, type = "class")

confusionMatrix(tree_preds, test$Target)

# Create CP Table
dat_rpart$cptable

# Find minimal cross validation error
dat_min_cp = dat_rpart$cptable[which.min(dat_rpart$cptable[,"xerror"]),"CP"]
dat_pruned <- prune(dat_rpart, cp = dat_min_cp)

tree_preds_pruned <- predict(dat_pruned, newdata = test, type = "class")

confusionMatrix(tree_preds_pruned, test$Target)

# Plot trees
rpart.plot(dat_rpart)
rpart.plot(dat_pruned)
```

Bagging Model

```r
library(randomForest)
# Fit Original Bag Model
bagged_mod <- randomForest(Target ~ ., data = train,
                             mtry = ncol(train) - 1)
# Plot OOB Error
plot(bagged_mod$err.rate[,1], type = 'l', ylab = "OOB Error",
     xlab = "Number of Trees")

bag_preds <- predict(bagged_mod, newdata = test, type = 'class')

confusionMatrix(bag_preds, test$Target)
```

```r
set.seed(7)
# Fit Final Bagging Model
bag_final <- randomForest(Target ~ ., data = train, mtry = p,
                         importance = TRUE, ntree = 25, nodesize = 3, nsplit = 1)

bag_preds <- predict(bag_final, newdata = test, type = 'class', ntree = 25)

confusionMatrix(bag_preds, test$Target)
```

Random Forest

```r
p <- ncol(train) - 1

rf_mod <- randomForest(Target ~ ., data = train, mtry = floor(p/2),
                         importance = TRUE, ntree = 100)
varImpPlot(rf_mod)

plot(rf_mod$err.rate[,1], type = 'l', ylab = "OOB Error", xlab = "Number of Trees")

rf_preds <- predict(rf_mod, newdata = test, type = 'class')

confusionMatrix(rf_preds, test$Target)
```

```r
library(randomForestSRC)

# Define grid values
nodesize_grid <- c(1:5)
nsplit_grid <- c(1:8)

# Create a data frame to store results
results <- expand.grid(nodesize = nodesize_grid, nsplit = nsplit_grid)
results$OOB_Error <- NA

# Loop over combinations
for (i in seq_len(nrow(results))) {
  fit <- rfsrc(Target ~ ., data = train,
               ntree = 25,
               mtry = p,
               nodesize = results$nodesize[i],
               splitrule = "auc",
               nsplit = results$nsplit[i],
               importance = "random")

  results$OOB_Error[i] <- fit$err.rate[fit$ntree]
}

# Find best combination
print(results[order(results$OOB_Error), ])
```

```r
set.seed(7)
# Fit final Random Forest
rf_final <- randomForest(Target ~ ., data = train, mtry = floor(p/2),
                         importance = TRUE, ntree = 25, nodesize = 4, nsplit =8)
# Predictions
rf_preds <- predict(rf_final, newdata = test, type = 'class', ntree = 25)

confusionMatrix(rf_preds, test$Target)
varImpPlot(rf_final)
```

Multinomial Logistic Regression

```r
set.seed(1234)
library(glmnet)

cv_model <- cv.glmnet(x = as.matrix(train[, -37]),
                      y = train$Target,
                      family = "multinomial")

best_lambda <- cv_model$lambda.1se

logit_preds <- predict(cv_model,
                 newx = as.matrix(test[, -37]),
                 type = "class",
                 s = best_lambda)

logit_train <- predict(cv_model,
                       newx = as.matrix(train[,-37]),
                       type = 'class',
                       s = best_lambda)

confusionMatrix(as.factor(as.vector(logit_preds)), test$Target)
```

```
# cv_model_all <- cv.glmnet(x = as.matrix(train[, -c(37:38)]),
#                        y = train$DropoutBinary,
#                        family = "binomial")
#
# best_lambda <- cv_model_all$lambda.1se
#
# logit_preds_all <- predict(cv_model_all,
#                   newx = as.matrix(test[, -c(37:38)]),
#                   type = "class",
#                   s = best_lambda)
#
# confusionMatrix(as.factor(as.vector(logit_preds_all)), test$DropoutBinary)
```

```
# x_train <- train[-37]
# x_test <- test[,-37]
#
# y_train <- train$Target
# y_test <- test$Target
#
# library(e1071)
#
# svm_model <- svm(x = x_train, y = y_train, kernel = "linear", cost = 0.01)
#
# svm_preds <- predict(svm_model, x_test)
#
# confusionMatrix(svm_preds, y_test)
```

```
# svm_tune <- tune(svm, Target ~ ., data = train, kernel = "linear",
#                   ranges = list(cost = c(0.01, 0.1, 1, 2, 5, 7, 10)))
# best_model <- svm_tune$best.model
# summary(best_model)
#
# svm_best_test_pred <- predict(best_model, test, type = 'class')
#
# confusionMatrix(svm_best_test_pred, test$Target)
```

```
# radial_tune <- tune(svm, Target ~ ., data = train, kernel = "radial",
#                   ranges = list(cost = c(0.01, 0.1, 1, 2, 5)))
# best_radial <- radial_tune$best.model
# summary(best_radial)
#
# radial_best_test_pred <- predict(best_radial, test, type = 'class')
#
# confusionMatrix(radial_best_test_pred, test$Target)
```

```
# poly_tune <- tune(svm, Target ~ ., data = train, kernel = "polynomial",
#                   ranges = list(cost = c(0.01, 0.1, 1, 2, 5),
#                                 degree = c(2,3,4)))
# best_poly <- poly_tune$best.model
# summary(best_poly)
#
# poly_best_test_pred <- predict(best_poly, test, type = 'class')
#
# confusionMatrix(poly_best_test_pred, test$Target)
```