DS3 Hackathon Celestial Prediction

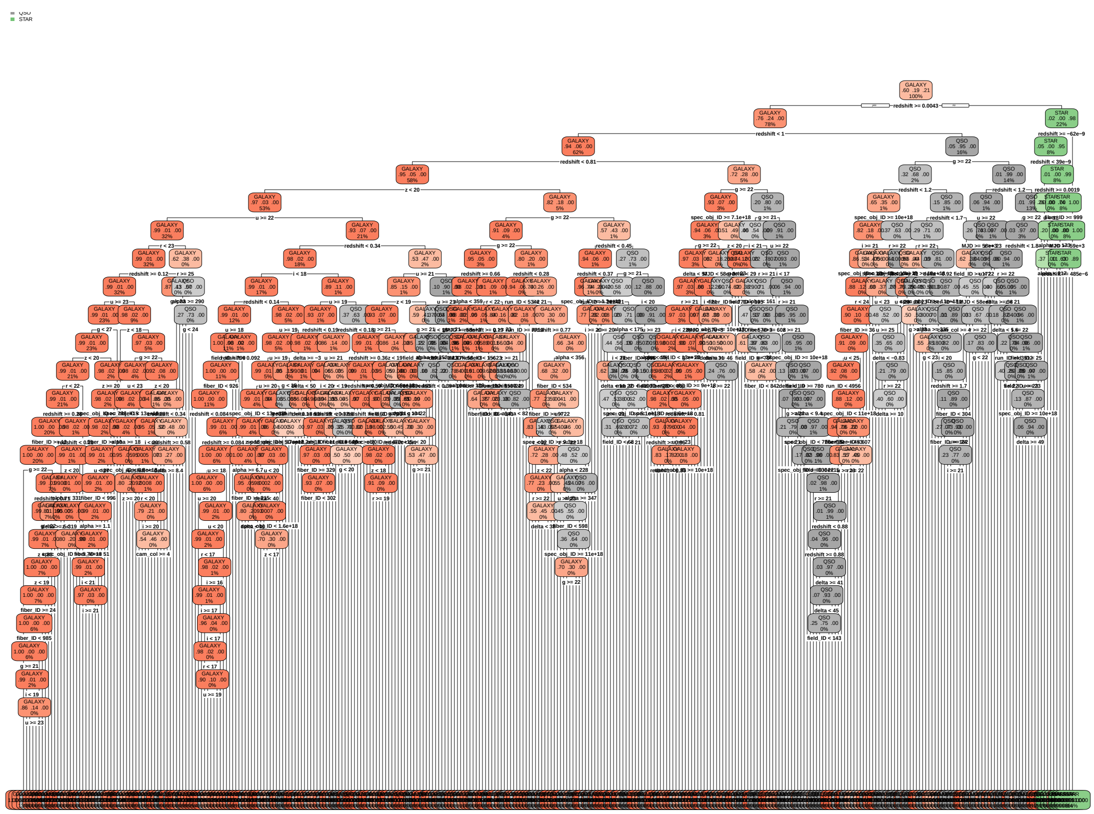
Jeffrey Zhou

Setup packages, load data, and further split the training (labeled) data into a subset for training and a subset for testing

```
library(tidyverse)
library(rpart)
library(rpart.plot)
library(randomForest)
library(gbm)
library(xgboost)
library(kableExtra)
library(ggplot2)
library(dplyr)
library(caret)
library(RCurl)

url_head <- "https://raw.githubusercontent.com/jeffreyz374/DS3-Hackathon-2023/main/celestial/"

tr_url <- paste0(url_head, "celestial_train.csv")
celestial_train <- getURL(tr_url)
celestial_train <- read.csv(text = celestial_train)
celestial_train <- subset(celestial_train, select = -c(id))

te_url <- paste0(url_head, "celestial_test.csv")
celestial_test <- getURL(te_url)
celestial_test <- read.csv(text = celestial_test)

celestial_test_no_ids <- subset(celestial_test, select = -c(id))

set.seed(1234)
train_idx <- sample(1:nrow(celestial_train), round(0.8 * nrow(celestial_train)))
train <- celestial_train[train_idx,]
test <- celestial_train[-train_idx,]
```

The first classifier we will use is the ordinary decision tree

```
celestial_tree <- rpart(class~., data = train, method = "class",
                        control = list(minsplit = 10), minbucket = 3, cp = 0,
                        xval = 10)

rpart.plot(celestial_tree)
```



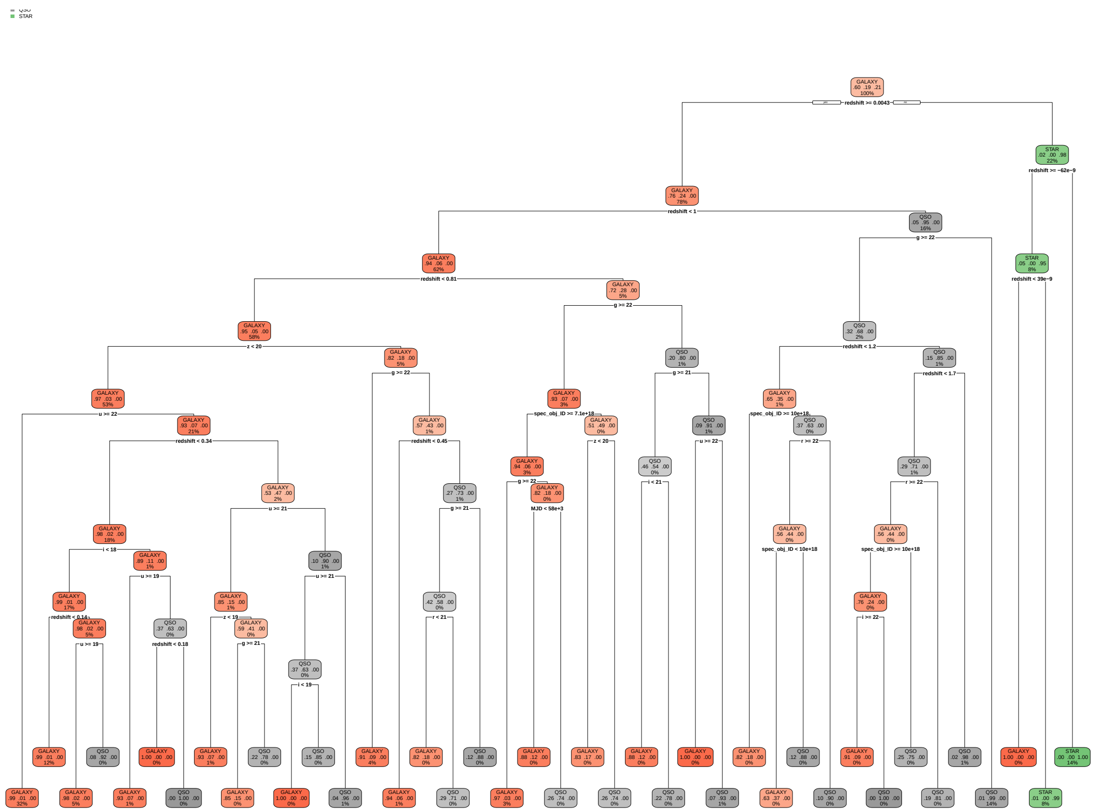
- Since this tree is probably overfitting, we can prune the tree by looking at the complexity parameter:

```
optimalcp <- celestial_tree$cptable[which.min(celestial_tree$cptable[,"xerror"]), "CP"]
optimalcp
```

```
## [1] 0.0002474635
```

- Prune the tree using the best value for the complexity parameter and draw the resulting tree:

```
celestial_tree_prune <- prune(celestial_tree, cp = optimalcp)

rpart.plot(celestial_tree_prune)
```



- Compute the test misclassification error:

```r
celestial_pred <- predict(celestial_tree_prune, test)
celestial_pred <- as.data.frame(celestial_pred)
celestial_pred$class <- ifelse(celestial_pred$"GALAXY" >= 0.5, "GALAXY",
                               ifelse(celestial_pred$"QSO" >= 0.5, "QSO", "STAR"))

confmatrix_table <- table(true = test$class, predicted = celestial_pred$class)
confmatrix_table
```

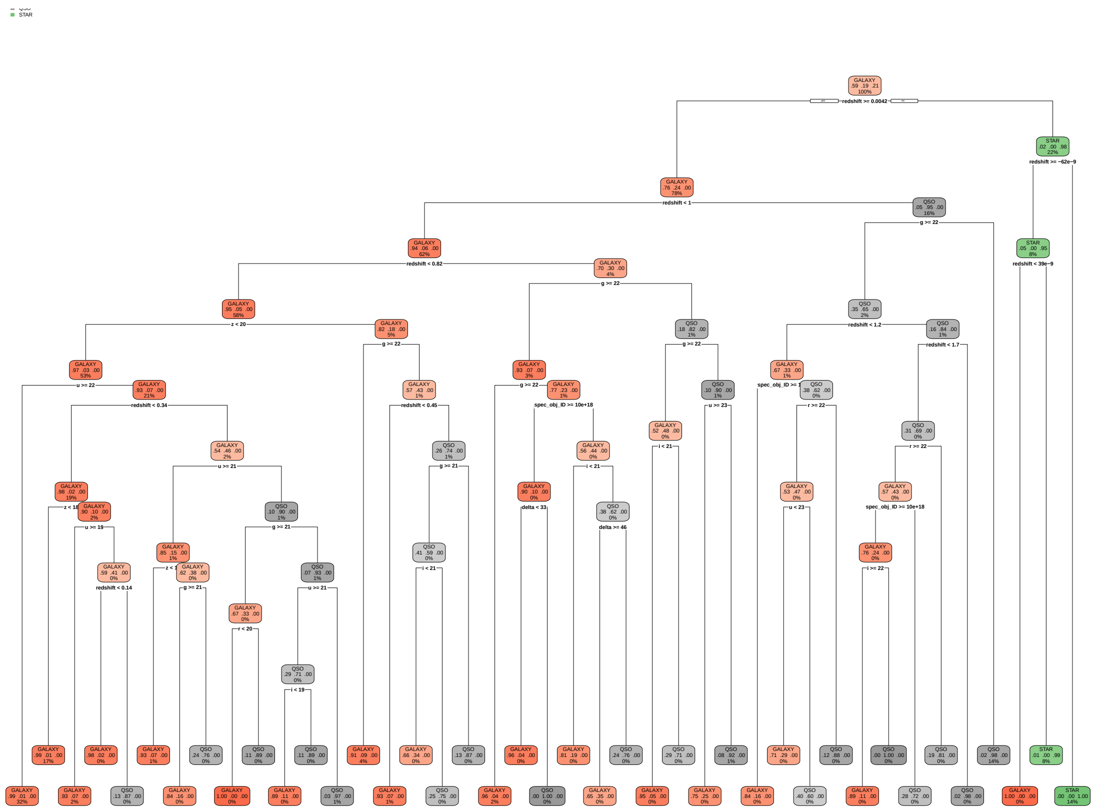
```
##         predicted
## true      GALAXY  QSO STAR
##   GALAXY    5807   77    9
##   QSO        143 1796    0
##   STAR         0    0 2168
```

```r
misclass_err <- (confmatrix_table[1, 2] + confmatrix_table[1, 3] +
                   confmatrix_table[2, 1] + confmatrix_table[2, 3] +
                   confmatrix_table[3, 1] + confmatrix_table[3, 2]) / nrow(test)
misclass_err
```

```
## [1] 0.0229
```

- Fit the tree with the optimal complexity parameter to the full (labeled) data:

```r
celestial_tree <- rpart(class~., data = celestial_train, method = "class",
                        control = list(cp = optimalcp))
rpart.plot(celestial_tree)
```



- Predict on testing data:

```r
celestial_pred_full <- predict(celestial_tree, celestial_test_no_ids)
celestial_pred_full <- as.data.frame(celestial_pred_full)
celestial_pred_full$class <- ifelse(celestial_pred_full$"GALAXY" >= 0.5,
                                    "GALAXY",
                                    ifelse(celestial_pred_full$"QSO" >= 0.5,
                                           "QSO", "STAR"))
```

- Join predictions with ids and export:

```r
x_pruned <- as.data.frame(list(id = celestial_test$id,
                               output = celestial_pred_full$class))

# Uncomment and change the destination directory to export
# write.csv(x_pruned, "~/Downloads/celestialSubmissionPruned.csv", row.names = FALSE)
```

- This classifier ultimately produced an accuracy of 0.9688396

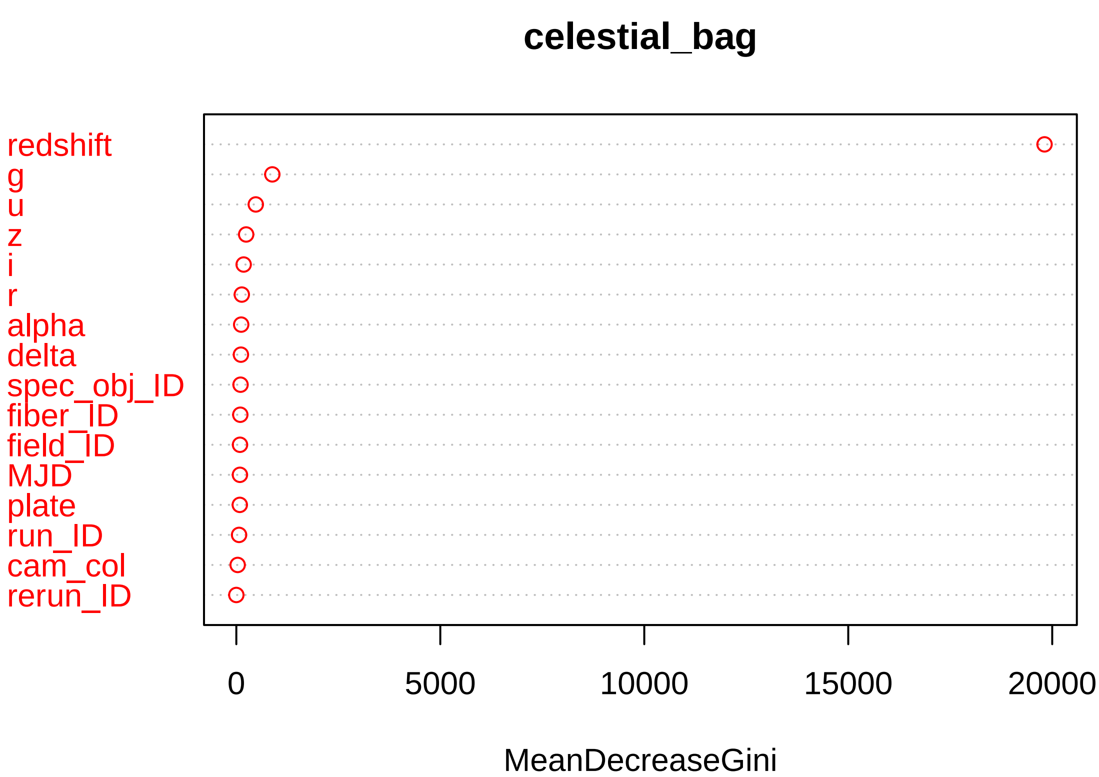
The next classifier we will use is bagging

- We will set `mtry` equal to the number of features (all other parameters at their default values). We will also generate the variable importance plot using `varImpPlot`:

```
celestial_bag <- randomForest(as.factor(class)~., data = train, mtry = 16,
                              na.action = na.omit)

# Error rate
sum(celestial_bag$err.rate[,1])
```

```
## [1] 11.34593
```

```
# Variable importance plot
varImpPlot(celestial_bag, n.var = 16, col = "red")
```



- Now, we will find the test misclassification error:

```
celestial_pred_bag <- predict(celestial_bag, test)
celestial_pred_bag <- as.data.frame(celestial_pred_bag)
celestial_pred_bag$class <- ifelse(celestial_pred_bag$celestial_pred_bag == test$class, 1, 0)

confmatrix_table <- table(true = test$class,
                          predicted = celestial_pred_bag$celestial_pred_bag)
confmatrix_table
```

```
##         predicted
## true      GALAXY  QSO STAR
##   GALAXY   5816   68    9
##   QSO       121 1818    0
##   STAR        1    0 2167
```

```r
misclass_err <- (confmatrix_table[1, 2] + confmatrix_table[1, 3]
                 + confmatrix_table[2, 1] + confmatrix_table[2, 3]
                 + confmatrix_table[3, 1] + confmatrix_table[3, 2]) / nrow(test)
misclass_err
```

```
## [1] 0.0199
```

- This is a slight improvement from a single decision tree. Now, train it on the full data and predict on the test data:

```r
celestial_bag_full <- randomForest(as.factor(class)~., data = celestial_train,
                                   mtry = 16, na.action = na.omit)

celestial_bag_pred_full <- predict(celestial_bag_full, celestial_test_no_ids)
celestial_bag_pred_full <- as.data.frame(celestial_bag_pred_full)
```

- Join predictions with ids and export:

```r
x_bag <- as.data.frame(list(id = celestial_test$id,
                            output = celestial_bag_pred_full$celestial_bag_pred_full))

# Uncomment and change the destination directory to export
# write.csv(x_bag, "~/Downloads/celestialSubmissionBag.csv", row.names = FALSE)
```

- This classifier ultimately produced an accuracy of 0.971105

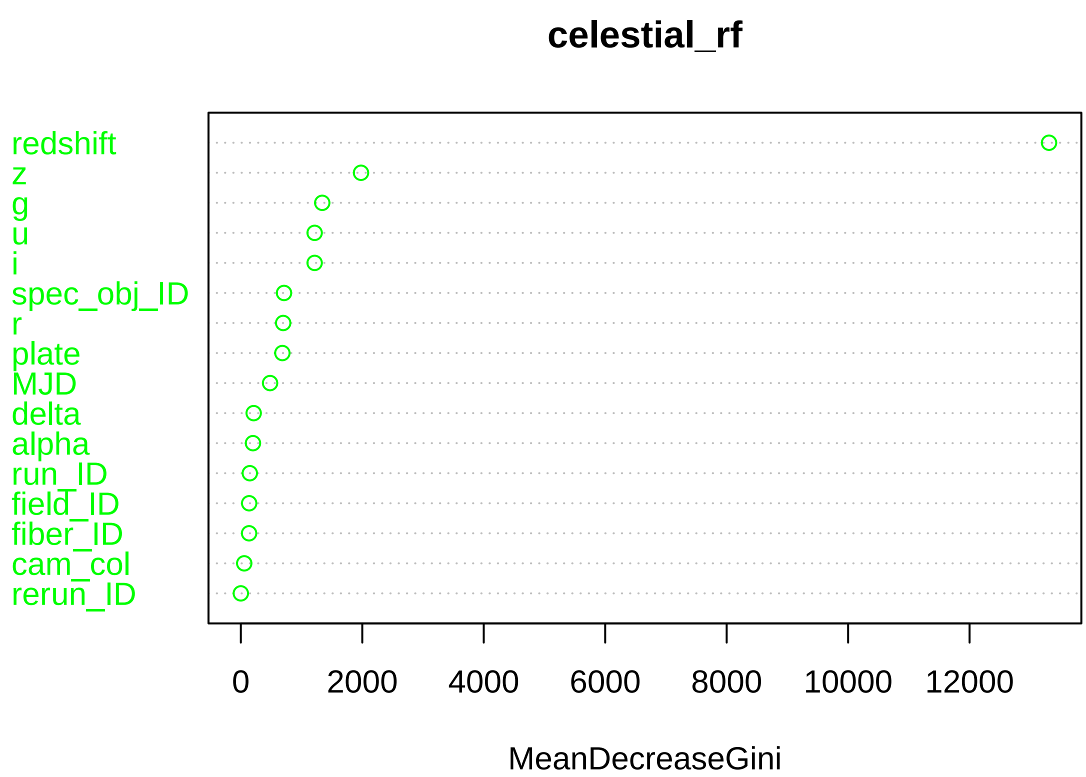
The next classifier we will use is random forest

- This time, we will use `randomForest` with the default parameters. We will again generate the variable importance plot using `varImpPlot`:

```
celestial_rf <- randomForest(as.factor(class)~., data = train, na.action = na.omit)

# Error rate
sum(celestial_rf$err.rate[,1])
```

```
## [1] 11.95631
```

```
# Variable importance plot
varImpPlot(celestial_rf, n.var = 16, col = "green")
```



- Now, we will find the test misclassification error

```
celestial_pred_rf <- predict(celestial_rf, test)
celestial_pred_rf <- as.data.frame(celestial_pred_rf)
celestial_pred_rf$class <- ifelse(celestial_pred_rf$celestial_pred_rf == test$class, 1, 0)

confmatrix_table <- table(true = test$class,
                          predicted = celestial_pred_rf$celestial_pred_rf)
confmatrix_table
```

```
##         predicted
## true     GALAXY  QSO STAR
##   GALAXY   5814   61   18
##   QSO       128 1811    0
##   STAR        1    0 2167
```

```
misclass_err <- (confmatrix_table[1, 2] + confmatrix_table[1, 3] +
                   confmatrix_table[2, 1] + confmatrix_table[2, 3] +
                   confmatrix_table[3, 1] + confmatrix_table[3, 2]) / nrow(test)
misclass_err
```

```
## [1] 0.0208
```

- This is slightly worse than bagging, but we will still train it on the full data, run it on the test data, and export:

```
celestial_rf_full <- randomForest(as.factor(class)~., data = celestial_train,
                                  na.action = na.omit)

celestial_rf_pred_full <- predict(celestial_rf_full, celestial_test_no_ids)
celestial_rf_pred_full <- as.data.frame(celestial_rf_pred_full)

x_rf <- as.data.frame(list(id = celestial_test$id,
                           output = celestial_rf_pred_full$celestial_rf_pred_full))

# Uncomment and change the destination directory to export
# write.csv(x_rf, "~/Downloads/celestialSubmissionRF.csv", row.names = FALSE)
```

- This classifier ultimately produced an accuracy of 0.9711846

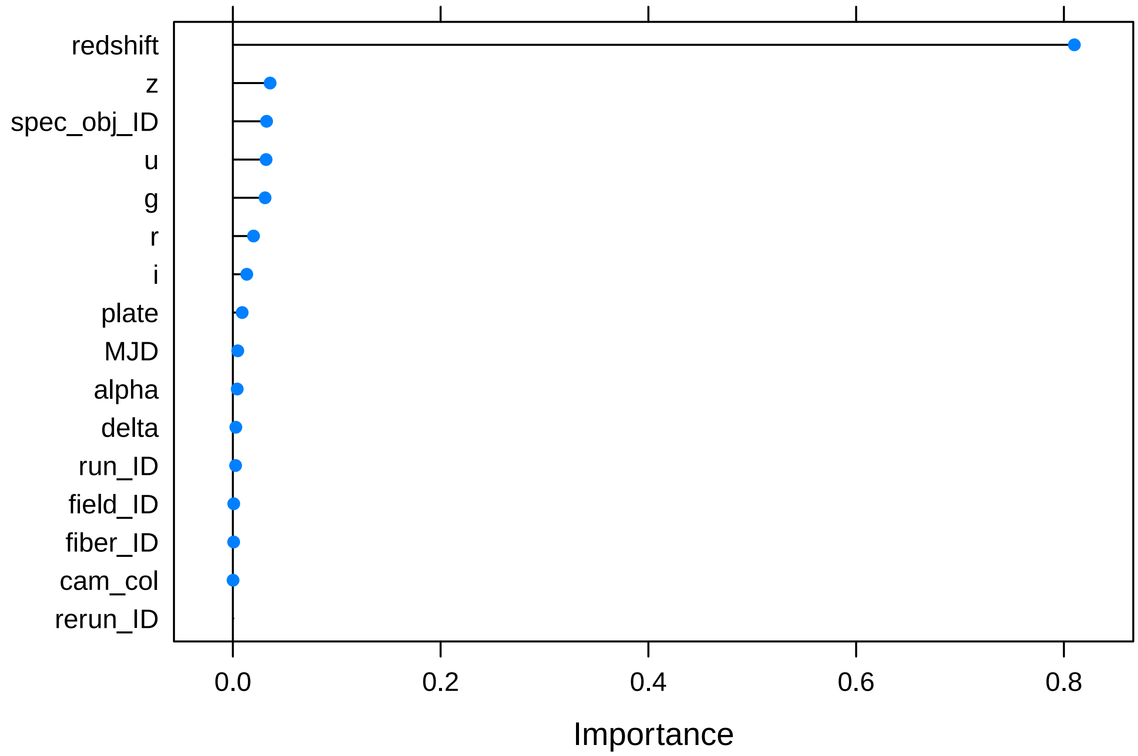
The final classifier we will use is extreme gradient boosting

- Turn the class variable into a numeric variable, initialize a grid search for the best hyperparameters, and train the model on the training data:

```
set.seed(1234)

# Replace categorical variables with numerical placeholders
celestial_train$class_num <- ifelse(celestial_train$class=="GALAXY", 1,
                                    ifelse(celestial_train$class=="QSO",2,0))

# Reassign labeled data, now with newly created numerical variables
celestial_train <- celestial_train %>% select(-c(class))

# Re-split the labeled data into training and testing datasets
train <- sample(1:nrow(celestial_train), floor(nrow(celestial_train) * 0.8))
test <- setdiff(1:nrow(celestial_train), train)
```

- Train extreme gradient boosting model with `xgboost` and perform a grid search for tuning the number of trees and the maximum depth of the tree. Then, we perform 10-fold cross-validation and determine the variable importance:

```
train_control <- trainControl(method = "cv", number = 10, search = "grid")
tune_grid <- expand.grid(max_depth = c(1, 3, 5, 7), nrounds = (1:10) * 50,
                         eta = c(0.01, 0.1, 0.3), gamma = 0, subsample = 1,
                         min_child_weight = 1, colsample_bytree = 0.6)

celestial_xgb <- caret::train(class_num~., data = celestial_train[train,],
                              method = "xgbTree",  trControl = train_control,
                              tuneGrid = tune_grid, verbose = FALSE, verbosity = 0)

# Show variable importance plot
plot(varImp(celestial_xgb, scale = FALSE))
```



- Compute the test MSE:

```
yhat_xgb <- predict(celestial_xgb, newdata = celestial_train[test,])
mean((yhat_xgb - celestial_train[test, "class_num"]) ** 2)
```

```
## [1] 0.0179447
```

- This is an improvement from both random forest and bagging. Now, train it on the full data and predict on the test data:

```
train_control <- trainControl(method = "cv", number = 10, search = "grid")
tune_grid <- expand.grid(max_depth = c(1, 3, 5, 7), nrounds = (1:10) * 50,
                         eta = c(0.01, 0.1, 0.3), gamma = 0, subsample = 1,
                         min_child_weight = 1, colsample_bytree = 0.6)

celestial_xgb_full <- caret::train(class_num~., data = celestial_train,
                                   method = "xgbTree", trControl = train_control,
                                   tuneGrid = tune_grid, verbose = FALSE,
                                   verbosity = 0)

celestial_xgb_pred_full <- predict(celestial_xgb_full, celestial_test_no_ids)
celestial_xgb_pred_full <- as.data.frame(celestial_xgb_pred_full)
```

- Join predictions with ids and export:

```r
output <- case_when(celestial_xgb_pred_full$celestial_xgb_pred_full < 0.5 ~ "STAR",
                    celestial_xgb_pred_full$celestial_xgb_pred_full >= 1.5 ~ "QSO",
                    TRUE ~ "GALAXY")

x_xgb <- as.data.frame(list(id = celestial_test$id, output = output))

# Uncomment and change the destination directory to export
# write.csv(x_xgb, "~/Downloads/celestialSubmissionXGB.csv", row.names = FALSE)
```

- This classifier ultimately produced an accuracy of 0.9715283

Conclusion

All in all, the accuracies we received can be summarized in the below table:

```r
accs <- c("0.9688396", "0.971105", "0.9711846", "0.9715283")
rownames <- c("Pruned Decision Tree", "Bagging", "Random Forest",
              "Extreme Gradient Boosting")
colname <- c("Accuracy")

acc_table <- cbind(accs)
rownames(acc_table) <- rownames
colnames(acc_table) <- colname

acc_table %>%
  knitr::kable(align = c("r"))
```

	Accuracy
Pruned Decision Tree	0.9688396
Bagging	0.971105
Random Forest	0.9711846
Extreme Gradient Boosting	0.9715283