1. What models or machines did you attempt to fit?

I attempted to fit a bootstrap aggregation model and a random forest regression model. For the bagging model, I use 150 bootstrapped samples to build the bagged model. I also specified coob to be TRUE to obtain the estimated OOB error. By using rpart.control() function, each tree grew deep, resulting trees with high variance and low bias so that we can reduce variance of the final model.

"bag = bagging(

formula = Y ~.,

data = df,

nbagg = 150,

coob = TRUE,

control = rpart.control(minsplit = 2, cp = 0)

)"

to fit the model. When I try to fit a random forest regression model, I use the following code:

"regr <- randomForest(x = x_train,

y = y_train,

maxnodes = 10,

ntree = 10)".

After tuning the model, I use

"tuned_regr <- randomForest(x = x_train,

y = y_train,

maxnodes = 70,

ntree = 1100)"

because the grid search produces the best maxnodes equal to 70 and ntree equal to 1100.

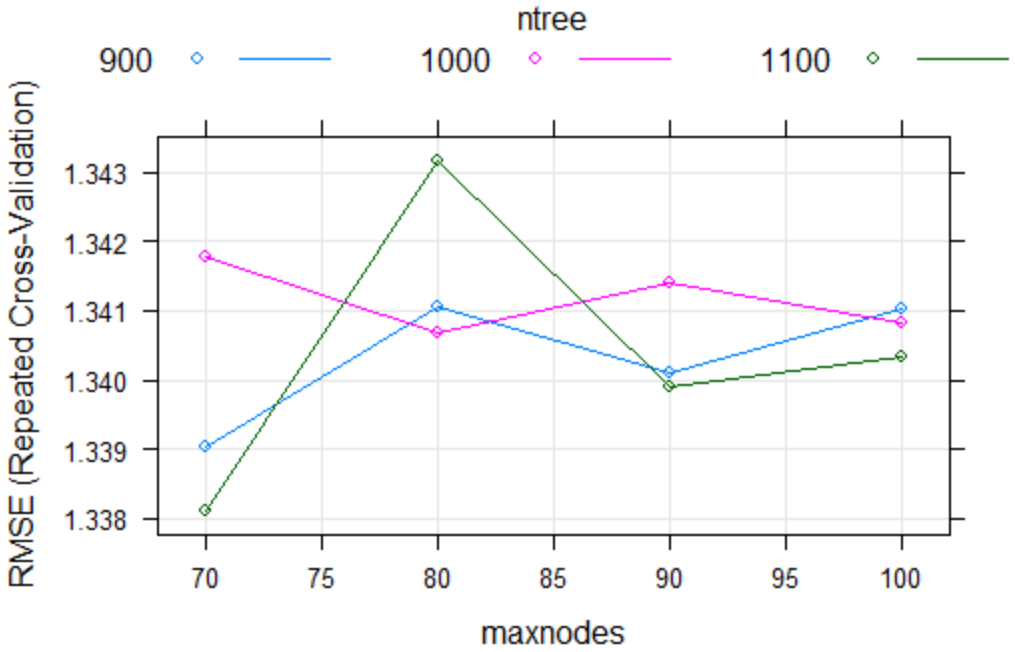
2. What process(es) did you use to evaluate and compare models and to select your final model?

For the bagging model, I chose to apply a 5-fold cv to assess the performance of the model. It produced and stored RMSE, R-squared, and MAE as the criteria to evaluate and compare performances. These three metrics can tell us how well the model performed on previously unseen data. The output of the model gave the OOB estimated RMSE amount to 1.33. For the random forest model after tuning, I use

sqrt(tuned_regr$mse[which.min(tuned_regr$mse)]) to find RMSE of best model, which amount to 1.32. Although the results are relatively close, I chose the random forest as the final model based on a smaller RMSE.

3. Did you tune any methods?

To improve the predictive power of the random forest model, I tune the hyperparameters of the algorithm by tuning the parameters ntrees and maxnodes. I built a custom Random Forest model to obtain the best set of parameters for our model and compare the output for various combination of the parameters. I set the grid search parameter using "control <- trainControl(method = "repeatedcv", number = 10, repeats = 3, search = 'grid')", outlined the grid of parameters by "tunegrid <- expand.grid(.maxnodes = c(70,80,90,100), .ntree = c(900,1000,1100))"and trained the model by "rf_gridsearch <- train(x=x_train_, y=y_train_, method = customRF, metric = metric, tuneGrid = tunegrid, trControl = control)". I visualized the impact of tuned parameters on RMSE, which shows that the model seemed to perform best when maxnodes is 70 and ntrees 1100. The plot is given below. After finding the best parameters for the model, the model was tuned.




4. What was your chosen prediction machine?

My chosen prediction machine is the random forest model. The code that produced all predicted values is as below:

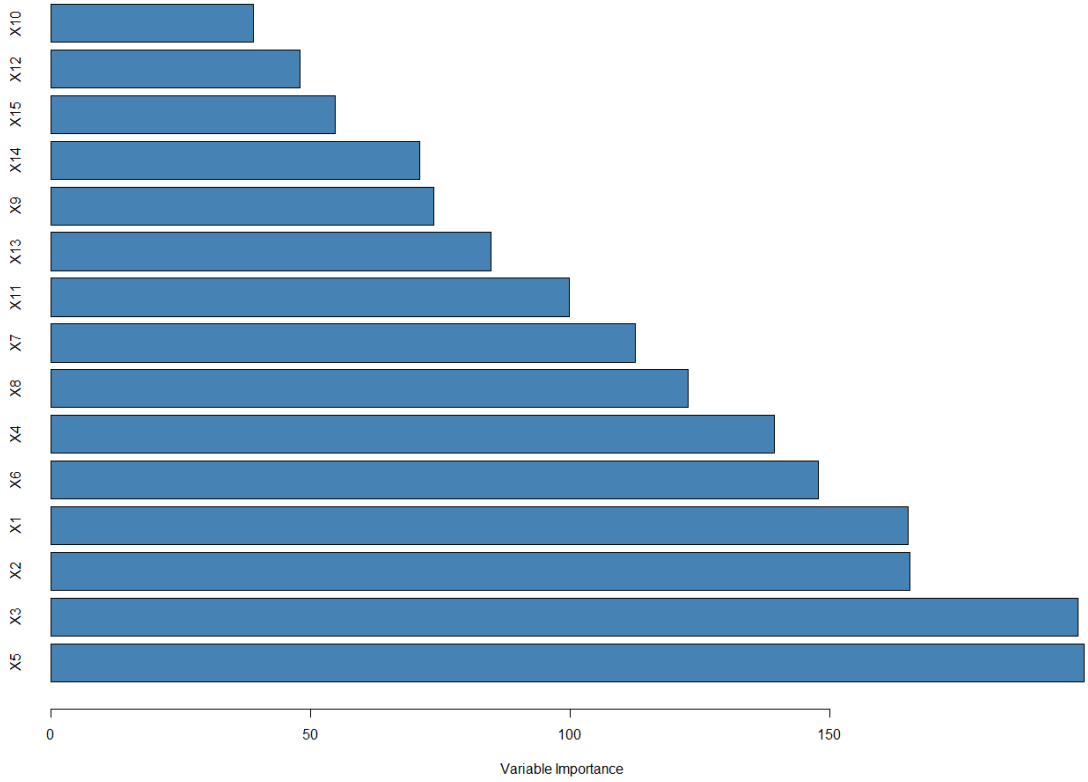
```
############################################################


# Second try: random forest

data <- read.csv("project2/Data2021_final.csv")

data

x <- data %>%
  select(X1:X15)
y <- data$Y

index <- createDataPartition(y, p = 0.75, list = F)
x_train <- x[ index, ]
x_test <- x[-index, ]
y_train <- y[index]
y_test <- y[-index]

regr <- randomForest(x = x_train,
                     y = y_train,
                     maxnodes = 10,
                     ntree = 10)

predictions <- predict(regr, x_test)

result <- x_test
result['Y'] <- y_test
result['prediction'] <- predictions
```

```
head(result)

# Import library for visualization
library(ggplot2)

# Build scatterplot
ggplot(  ) +
  geom_point( aes(x = x_test$X1, y = y_test, color = 'red', alpha = 0.5) ) +
  geom_point( aes(x = x_test$X1 , y = predictions, color = 'blue',  alpha = 0.5)) +
  labs(x = "X1", y = "Y", color = "", alpha = 'Transperency') +
  scale_color_manual(labels = c( "Predicted", "Real"), values = c("blue", "red"))

library(Metrics)
print(paste0('MAE: ', mae(y_test,predictions)))
print(paste0('MSE: ' ,caret::postResample(predictions , y_test)['RMSE']^2 ))
print(paste0('R2: ' ,caret::postResample(predictions , y_test)['Rsquared'] ))



N=500 #length(X_train)
x_train_ = x_train[1:N , ]
y_train_ = y_train[1:N]

seed <- 301325351
metric <- 'RMSE'

customRF <- list(type = "Regression", library = "randomForest", loop = NULL)
```

```
customRF$parameters <- data.frame(parameter = c("maxnodes", "ntree"), class = rep("numeric", 2), label = c("maxnodes", "ntree"))

customRF$grid <- function(x, y, len = NULL, search = "grid") {}

customRF$fit <- function(x, y, wts, param, lev, last, weights, classProbs, ...) {
  randomForest(x, y, maxnodes = param$maxnodes, ntree=param$ntree, ...)
}

customRF$predict <- function(modelFit, newdata, preProc = NULL, submodels = NULL)
  predict(modelFit, newdata)
customRF$prob <- function(modelFit, newdata, preProc = NULL, submodels = NULL)
  predict(modelFit, newdata, type = "prob")
customRF$sort <- function(x) x[order(x[,1]),]
customRF$levels <- function(x) x$classes

control <- trainControl(method = "repeatedcv",
                        number = 10,
                        repeats = 3,
                        search = 'grid')

tunegrid <- expand.grid(.maxnodes = c(70,80,90,100),
                        .ntree = c(900,1000,1100))
set.seed(seed)


rf_gridsearch <- train(x=x_train_,
                       y=y_train_,
                       method = customRF,
```

```
                    metric = metric,
                    tuneGrid = tunegrid,
                    trControl = control)

plot(rf_gridsearch)

rf_gridsearch$bestTune # maxnodes = 70; ntree = 1100

tuned_regr <- randomForest(x = x_train,
                           y = y_train,
                           maxnodes = 70,
                           ntree = 1100)
tuned_predictions <- predict(tuned_regr, x_test)

tuned_result <- x_test
tuned_result['Y'] <- y_test
tuned_result['prediction'] <- tuned_predictions
head(tuned_result)

predictions <- predict(tuned_regr,newdata = new)

write.table(predictions,"project2/results_rf2.csv",sep = ",", row.names = F, col.names = F)

tuned_regr
which.min(tuned_regr$mse) # 462
sqrt(tuned_regr$mse[which.min(tuned_regr$mse)])

################################################
```

5. (optional) List the variables that you believe are important.




By calculating the total reduction in the residual sum of squares, I used a bar plot to visualize the importance of the predictor variables. The larger the value, the more important the predictor. By using the varImp() function from the caret library, I created a plot for the fitted bagged model which showed that X5 and X3 seemed to be the most important variable and X10 the least one.