Kaggle Project

Kevin Mei

May 1, 2018

This Kaggle project is about predicting house price using decision tree and random forest tree. First, the dataset has to be cleaned by either replacing all NULL values with the sample mean or omit all NULL values. Second, the two tree models are being trained and tested with processed data. Third, using the Mean Absolute Error(MAE) and Percent Error(MAPE) to detemine the accruatcy of the results generated from two models.

Phase1: Loading libraries

```
suppressMessages(library(tidyverse)) # utility functions
suppressMessages(library(rpart)) # for regression trees
suppressMessages(library(randomForest)) # for random forests
suppressMessages(library(readr))#read data file
suppressMessages(library(modelr))#resampling data
suppressMessages(library(caret))#varImp graph
suppressMessages(library(DT))#datatable
suppressMessages(library(ggplot2))#ggplot graph
suppressMessages(library(MLmetrics))#mean absolute percent error
suppressMessages(library(ggmap))#compute elevation
suppressMessages(library(sf))
suppressMessages(library(raster))
suppressMessages(library(leaflet))#leaflet map
suppressMessages(library(leaflet.extras))
```

Phase2: Cleaning Data

```
df <- read.csv("C:/Users/JM020/Desktop/melb_data.csv",stringsAsFactors=FALSE)
#rank missing entries
valName = vector()
misFeq = vector()
misPct = vector()
for(i in 1:length(colnames(df))){
  valName[i]=colnames(df)[i]
  misFeq[i]=sum(is.na(df[,i]))
  misPct[i] = misFeq[i]/nrow(df)*100
}
MisEntrie = data.frame(valName,misFeq,misPct)
MisEntrie = MisEntrie[with(MisEntrie,order(misFeq,decreasing = T)),]
MisEntrie

##           valName misFeq       misPct
## 16   BuildingArea  10634 57.806044792
## 17      YearBuilt   9438 51.304631442
```

```
## 15       Landsize   4793 26.054577082
## 14            Car   3576 19.439008480
## 13       Bathroom   3471 18.868232224
## 12       Bedroom2   3469 18.857360296
## 19      Lattitude   3332 18.112633181
## 20     Longtitude   3332 18.112633181
## 10       Distance      1  0.005435964
## 11       Postcode      1  0.005435964
## 22 Propertycount      1  0.005435964
## 1           Index      0  0.000000000
## 2          Suburb      0  0.000000000
## 3         Address      0  0.000000000
## 4           Rooms      0  0.000000000
## 5            Type      0  0.000000000
## 6           Price      0  0.000000000
## 7          Method      0  0.000000000
## 8         SellerG      0  0.000000000
## 9            Date      0  0.000000000
## 18    CouncilArea      0  0.000000000
## 21     Regionname      0  0.000000000
```

Cleaning Method 1: Omiting NULL values

```
#actual code starts here
df <- read.csv("C:/Users/JM020/Desktop/melb_data.csv",stringsAsFactors=FALSE)
old = dim(df)
df = na.omit(df)
new = dim(df)
(new/old)#omitting 37.13% of data

## [1] 0.3712764 1.0000000
```

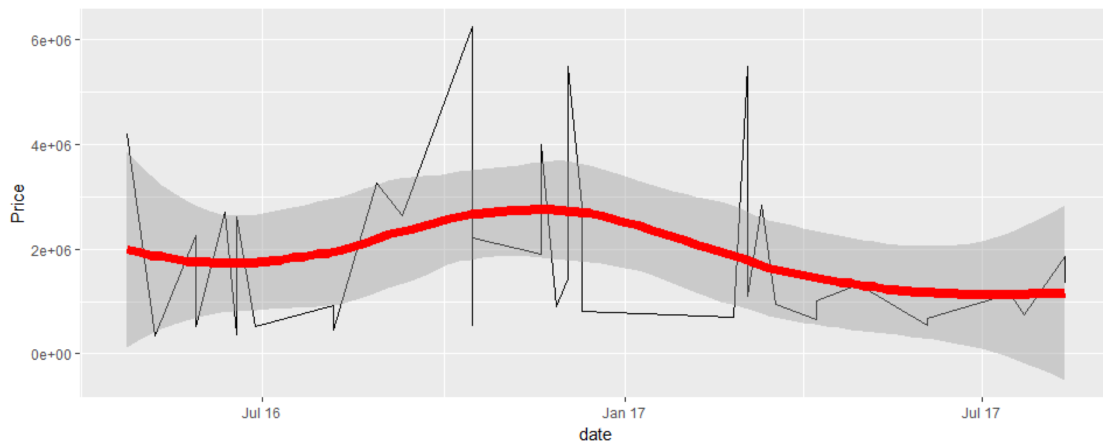
Cleaning Method 2:Replacing NULL values with sample mean

```
df2 = read.csv("C:/Users/JM020/Desktop/melb_data.csv",stringsAsFactors=FALSE)
#replacing na with mean to see improvement
df2$Bathroom[is.na(df2$Bathroom)] <- round(mean(df$Bathroom))
df2$Landsize[is.na(df2$Landsize)| df2$Landsize==0] <-
mean(df$Landsize)#landsize should not be zero
df2$BuildingArea[is.na(df2$BuildingArea)| df2$BuildingArea==0] <-
mean(df$BuildingArea)#building area should not be zero
df2$YearBuilt[is.na(df2$YearBuilt)] <- round(mean(df$YearBuilt))
df2$Lattitude[is.na(df2$Lattitude)] <- mean(df$Lattitude)
df2$Longtitude[is.na(df2$Longtitude)] <- mean(df$Longtitude)
```

Phase3: Data Visualization

```
DD =as.Date(df$Date,"%d/%m/%Y")#split the date in DD/MM/YYYY format


Date = data.frame(date = DD,
```

```r
                 day = as.numeric(format(DD, format = "%d")),
                 month = as.numeric(format(DD, format = "%m")),
                 year = as.numeric(format(DD, format = "%y")))

Price = round(df$Price,digits = 0)
PDchart = data.frame(Price,Date,df$Suburb)

#datatable(PDchart)#table that sorts by different features(Date,Price,Suburb)

PD2016 = PDchart[which(PDchart$year==16),]#2016 Suburb price chart
PD2017 = PDchart[which(PDchart$year==17),]#2017 Suburb price chart
PD2016N =PD2016[c(-2,-3,-5)]#Price vs Suburb
PD2017N =PD2017[c(-2,-3,-5)]

Avg16 = aggregate(PD2016N[, 1], list(PD2016N$df.Suburb), mean)#calculate the
price mean
Avg17 = aggregate(PD2017N[, 1], list(PD2017N$df.Suburb), mean)

X1 = head(Avg16[order(Avg16$x,decreasing = T),],100)#Top 100 house price in
avg in 2016
X2 = head(Avg17[order(Avg17$x,decreasing = T),],100)#Top 100 house price in
avg in 2017

X = cbind(X1,X2)
#datatable(X,colnames = c("Most Luxury/cheapest suburb houses in 2016",
"Cost($)","Most Luxury/cheapest suburb houses in 2017", "Cost($)"))

Toorak = PDchart[PDchart$df.Suburb=="Toorak",]#Targeting suburb "Toorak"

Toorak$date = as.Date(Toorak$date,"%Y%m%d")#Extracting dates related to
Toorak

base =ggplot(Toorak, aes(x=date, y=Price)) + geom_line()

base+ scale_x_date(date_labels = "%b %y")+geom_smooth(stat =
"smooth",position = "identity", color = "red",size =3 , formula = y ~ x,se =
TRUE, na.rm = FALSE,inherit.aes = T,method = "loess")#Price fluctuation and
trend from 2016 to 2017 in term of months
```



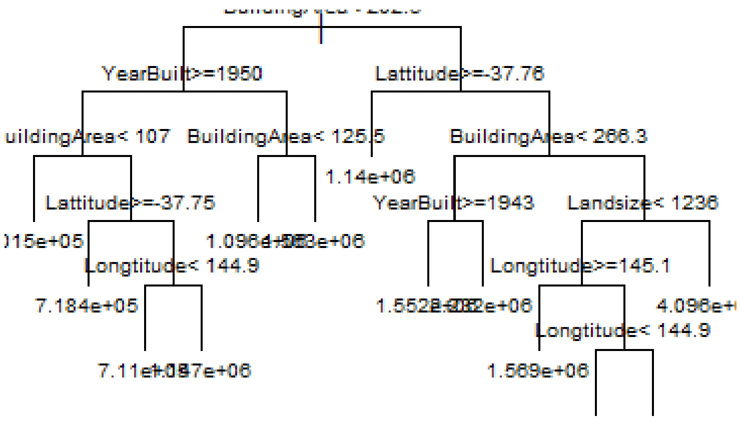

```
Geo = data.frame(df$Suburb,df$Price,df$Longtitude,df$Lattitude)

#Geographic map based on longtitude and latitude
leaflet(Geo) %>% setView(lng = Geo$df.Longtitude[1], lat =
Geo$df.Lattitude[1], zoom = 12)%>% addTiles() %>%
addProviderTiles(providers$CartoDB.DarkMatter) %>%
  addWebGLHeatmap(lng=~Geo$df.Longtitude, lat=~Geo$df.Lattitude, intensity =
~Geo$df.Price, size=100)
```

Phase4:Implementing regression tree models

```
#Producing a new feature based on the old features(longtitude and latitude)
#The objective is to improve model effciency
AUS <- getData('alt', country ="AUS")
GIS <-data.frame(Geo$df.Longtitude,Geo$df.Lattitude)
Elevation <-cbind(GIS, alt = extract(AUS, GIS))[3]
Geo <-data.frame(df$Suburb,df$Price,df$Longtitude,df$Lattitude, Elevation)
df$Elevation <- Geo$alt

which(is.na(df$Elevation))#found 1 NA value by using a package to compute new
feature

## [1] 3518

df <- na.omit(df)
sum(is.na(df))#omit NA value

## [1] 0
```

Model 1: Decision Tree Model

```
#using recursive partitioning and regression tree func(rpart) to create a
decision tree and predict outcomes with categorical data
fit <- rpart(Price ~ Rooms + Bathroom + Landsize + BuildingArea +
             YearBuilt + Lattitude + Longtitude, data = df)
plot(fit, uniform=T)#regression tree
text(fit, cex=.6)#lable the weights
```



Decision Tree Model Accuracy

```r
#Now we want to see how good is our model "fit"
#error=actual-predicted
#the absolute value of the mean of the predictions errors tells us how much our model is off
#namely Mean Absolute Error(MAE)

mae(model = fit, data = df)

## [1] 280307.5

MAPE(y_pred = predict(fit, df), y_true = df$Price)

## [1] 0.2921115

#In average house price prediction is off by $280307.1 or 29.2% comparing to the actual prices
```

Implementing the same model with boostrapping

```r
#taking only partial dataset to predict and the rest(validation data) to train the model
#Preventing overfitting and underfitting the data

set.seed(5132018)#Reproducing results

#50% of the data is used to train the model
#Another 50% of the data is used to test the mode
```

```r
splitData <- resample_partition(df, c(test = .5, train = .5))
#The MAE score is the lowest when test =0.1 and train =0.9 but the model
could be overfitted and unable to adopt new data
#The second lowest score comes from (.5,.5), so it was selected

#Dimension of the splited samples
lapply(splitData,dim)

## $test
## [1] 3414   23
##
## $train
## [1] 3415   23

#using the validation data to create a new model "fit2"
fit2 <- rpart(Price ~ Rooms + Bathroom + Landsize + BuildingArea +
             YearBuilt + Lattitude + Longtitude, data = splitData$train)

#error diff in new model
mae(model = fit2, data = splitData$test)

## [1] 290146.9

MAPE(y_pred = predict(fit2, df), y_true = df$Price)

## [1] 0.2935233

#As a result, the new model(fit2) is slightly higher mae than the old
model(fit)

#Considering the possiblity of overfitting and underfitting the validation
data
#trying to manipulate the height or depth of the tree
#Decision tree with height over 10 levels =>overfitting data and couldn't
adopt new data
#Tree with low levels =>underfitting data and couldn't predit anything

#creating a function with all possible tree detphs using the maxdepth
parameter
get_mae <- function(maxdepth, target, predictors, training_data,
testing_data){

    # turn the predictors & target into a formula to pass to rpart()
    predictors <- paste(predictors, collapse="+")#create a predictor list
    formula <- as.formula(paste(target,"~",predictors,sep = ""))#extract a
formula from a list

    # build our model
    #add the control and rport.control to mannually fix a maxdepth
    model <- rpart(formula, data = training_data,
                   control = rpart.control(maxdepth = maxdepth))
```

```
    # get the mae
    mae <- mae(model, testing_data)#compute the score
    return(mae)
}

# target & predictors to feed into our formula
target <- "Price"
predictors <-  c("Rooms","Bathroom","Landsize","BuildingArea",
                 "YearBuilt","Lattitude","Longtitude")

# get the MAE for maxdepths between 1 & 10
for(i in 1:10){
    mae<- get_mae(maxdepth = i, target = target, predictors = predictors,
                  training_data = splitData$train, testing_data =
splitData$test)
    print(glue::glue("Maxdepth: ",i,"\t MAE: ",mae))
}

## Maxdepth: 1   MAE: 417255.679525158
## Maxdepth: 2   MAE: 361506.571110851
## Maxdepth: 3   MAE: 319825.840600034
## Maxdepth: 4   MAE: 303433.084346229
## Maxdepth: 5   MAE: 290146.899234969
## Maxdepth: 6   MAE: 290146.899234969
## Maxdepth: 7   MAE: 290146.899234969
## Maxdepth: 8   MAE: 290146.899234969
## Maxdepth: 9   MAE: 290146.899234969
## Maxdepth: 10  MAE: 290146.899234969

#The MAE score stays constant at depth level 5
#This means that the model works the best or reaches constant state by tuning
the depth parameter to 5 or above
```

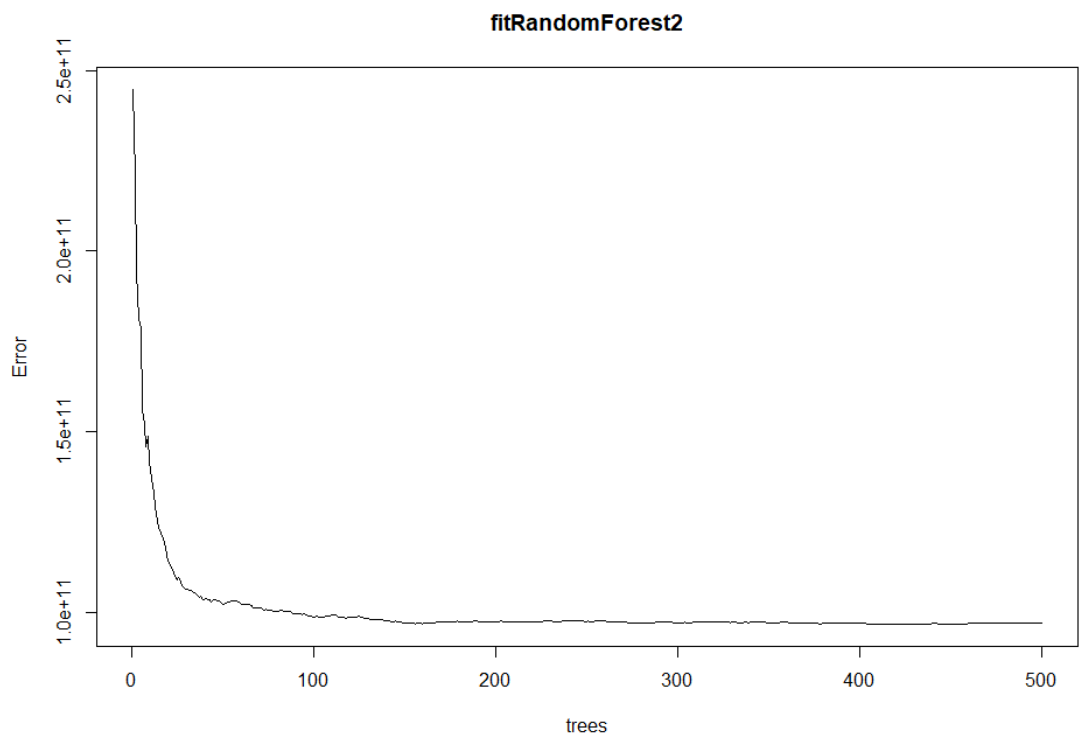
Model 2: Random Forest Tree Model

```
#fitting a model using random forest
splitData <- resample_partition(df, c(test = .5, train = .5))

# fit a random forest model to the training set
fitRandomForest <- randomForest(Price ~ Rooms + Bathroom + Landsize +
BuildingArea +
             YearBuilt + Lattitude + Longtitude ,data =
splitData$train,importance =T)
fitRandomForest2 <- randomForest(Price ~ Rooms + Bathroom + Landsize +
BuildingArea +
             YearBuilt + Lattitude + Longtitude + Elevation,data =
splitData$train,importance =T)
```

Random Forest Tree Model Accuracy and plots

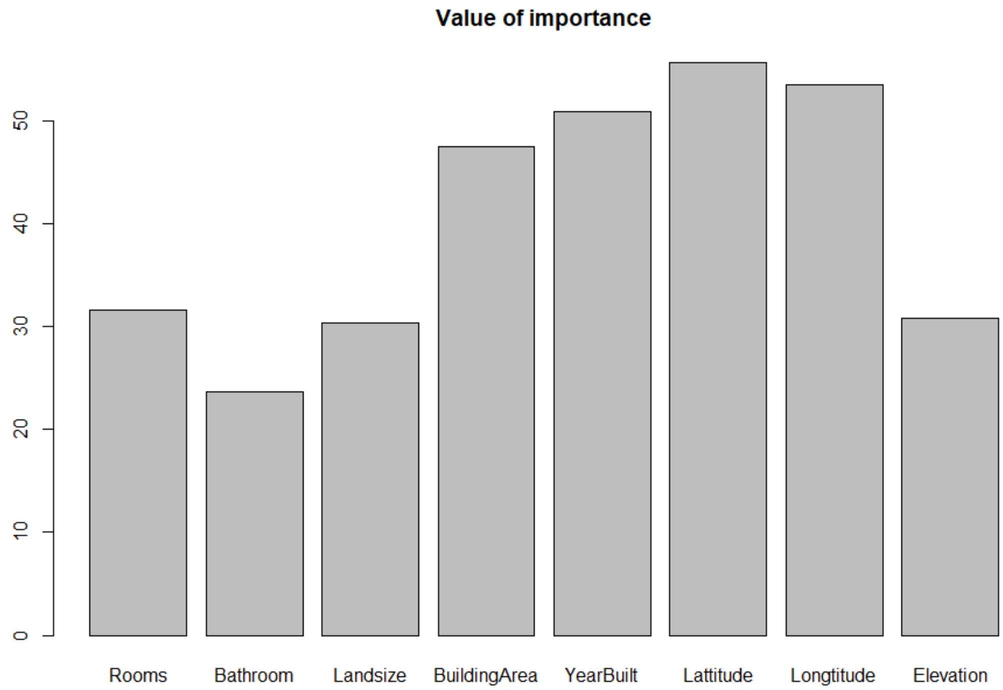
```
# get the mean average error for our new model, based on our test data
mae(model = fitRandomForest, data = splitData$test)
```

```
## [1] 187245.2

MAPE(y_pred = predict(fitRandomForest, df), y_true = df$Price)

## [1] 0.1305653

#Prediction errors reduced from 30% to 13%
#If 90% of the data is used to train the model, the errors are reduced 7%

mae(model = fitRandomForest2, data = splitData$test)

## [1] 184334.5

MAPE(y_pred = predict(fitRandomForest2, df), y_true = df$Price)

## [1] 0.128222

#By adding a new feature Elevation, the model has improved its error rate
from 13.3% to 12.9%

plot(fitRandomForest2)#As the forest gets larger, the errors are lower
```

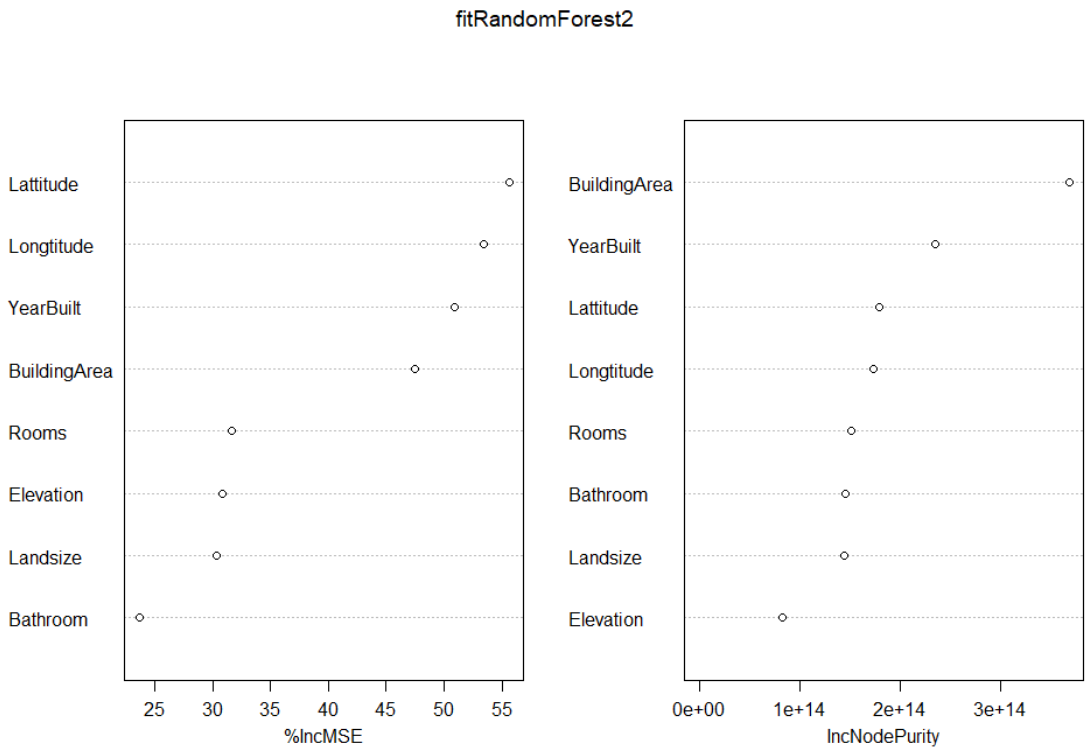



```
ImpVal = varImp(fitRandomForest2)#Showing the importance of each feature

predictors <-  c("Rooms","Bathroom","Landsize","BuildingArea",
                 "YearBuilt","Lattitude","Longtitude","Elevation")
```

```r
barplot(ImpVal$Overall,main = "Value of 
importance",horiz=F,names.arg=predictors)#Overall level of importances
```




```r
varImpPlot(fitRandomForest2)#plot the value of importances of each variable
```




```
#%INcMSE = mean decrease in accruacy and IncNodePurity = mean decrease in MSE

#Higher ranking on the %INcMSE plot means that if that feature is randomly
permuted, its impacts/changes to the predictions is higher.

#IncNodePurity shows that if a feature is dropped from the model, how much it
would changes the result in MAE

#Compute predictions while tuning parameters mtry and ntree
old_MAE = MAE(y_pred = predict(fitRandomForest, newdata = df),y_true =
(df$Price))
cat("MAE score before tuning mtry and ntree is",old_MAE)

## MAE score before tuning mtry and ntree is 138370.7

RF2 = randomForest(Price ~ Rooms + Bathroom + Landsize + BuildingArea +
             YearBuilt + Lattitude + Longtitude ,data =
splitData$train,mtry=8,ntree = 800,importance =T)

## Warning in randomForest.default(m, y, ...): invalid mtry: reset to within
## valid range

new_MAE = MAE(y_pred = predict(RF2, newdata = df),y_true = (df$Price))
cat("MAE score after tuning mtry and ntree is",new_MAE)

## MAE score after tuning mtry and ntree is 129512.1

MAPE(y_pred = predict(RF2, newdata = df), y_true = df$Price)
```

```
## [1] 0.1198585
#Pecent error improves from 13% to 12.2%
#Using the best model so far(random forest tree model) to predict house
prices on data fills with sample means
MAE(y_pred = predict(RF2, newdata = df2),y_true = (df2$Price))
## [1] 289935.1
MAPE(y_pred = predict(RF2, newdata = df2), y_true = df2$Price)
## [1] 0.3511846
#Replacing NULL values with sample means does not perform better than
omitting NULL values
```

Phase5: Conclusion

The best model by far is the random forest tree with tuned parameters. The next step is to train the model with more data and to make house price estimation with new data.The ultimate goal of this kaggle project is to have a trained model with error rate under 10%. So that, it can be used to predict the future market trend of house prices for a perticular district or suburb in Australia.

Phase6: Future Implementation

1. Heatmap or Intensity labels on top of leaflet world map by the house price level
2. Produce more new features to improve the error rates like Elevation
3. Implement ensemble model by boosting three or more regression tree models

Phase7: Reference

This project is inspired by Rachael Tatman