—'

title: “Github_Womens Machine learning”

output: html_document

Created for Google Machine learning competition Final place 217/500 participants My graded Log loss score of .40367, winner scored .32245 log loss

Background: On a scale of 0 to 1 predict percentage chance of a team winning a given tournament matchup for all potential matchups

Data: sets utilized will utilize regular season womens game by game stat numbers to predict performance in tournament

Train Data: 2010-2018 womens regular season data as predicting (independent) variables – use tournament performance as dependent variable

Test Data: 2019 regular season data - predict winning percentage chance by all 2016 potential games (64 teams x63 potential opponents/ 2 teams per game)

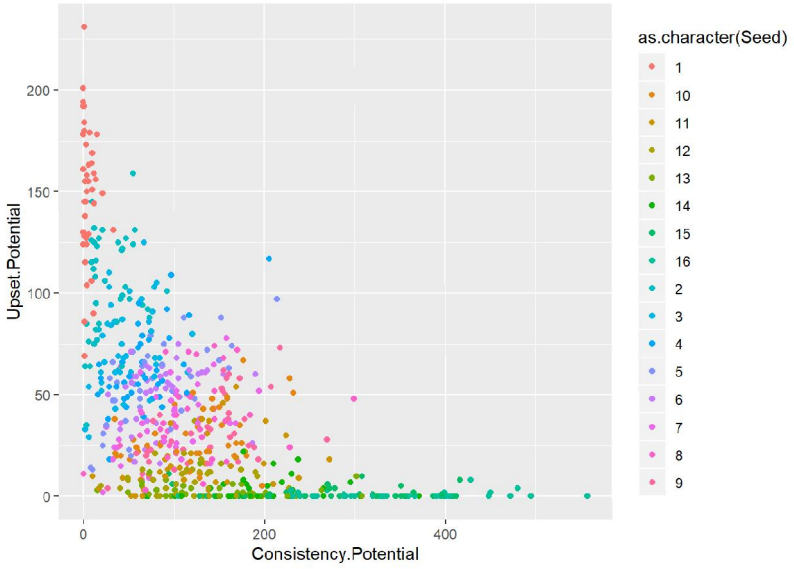
****

Section 1 - data manipulation and feature engineering based on a single teams standalone statistics

****

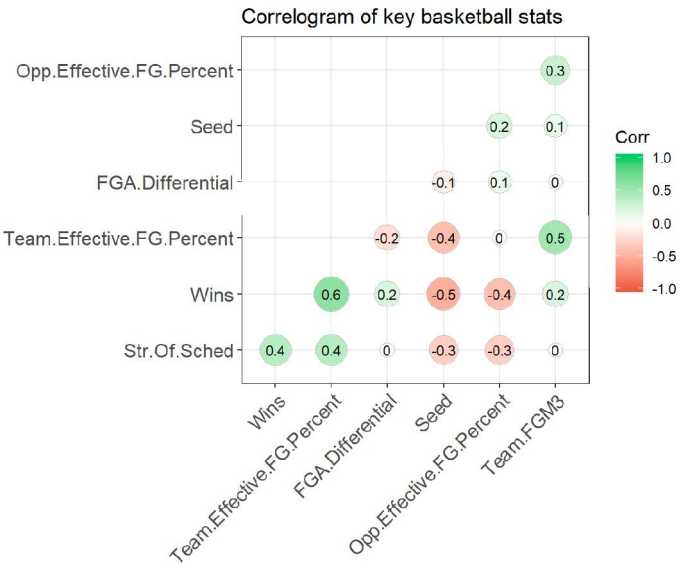
Section 1a. : Strength of schedule data & data manipulation Purpose of this section is to create a strength of schedule measure Will be broken out by upset poential - i.e. how many quality ranked wins a team has Also will take into account consistency - i.e. general quality of losses, are they against good or bad teams

```
#This plot highlights the features we created by looking at head to head wins and losses.  My consistency and upset potentia
l features highly correlate with seeding
qplot(Consistency.Potential,Upset.Potential,colour = as.character(Seed),
      data = CombinedTeamGrouped)
```



Section 1b. : Create features based on advanced basketball statistics Look at things like Field goal pecentage, effective fg %, and field goal differential. In prior year challenge I simply used these features to create a simulated game score based on more advanced stats

```
# Correlation matrix
selection <- CombinedTeamGrouped[,c("Wins","Seed","Str.Of.Sched","Team.Effective.FG.Percent","FGA.Differential","Opp.Effecti
ve.FG.Percent","Team.FGM3")]
corr <- round(cor(selection), 1)
ggcorrplot(corr, hc.order = TRUE,
           type = "lower",
           lab = TRUE,
           lab_size = 3,
           method="circle",
           colors = c("tomato2", "white", "springgreen3"),
           title="Correlogram of key basketball stats",
           ggtheme=theme_bw)
```



****

Section 2 - Preparing data set for predictive model and feature engineering based on head to head statistics

****

Section 2a. : Create data matrix that computes all possible games Compute theoretical game scores (The last feature we will engineer for our data set (Section 1)) Need to create all possible games by doing a join between the same data frames (Section 2)

****

Section 3 - Feature normalization and splitting datasets

Note: some feature engineering is included in this section as well as a way to improve random forest model (was done after initial runs)

****

```
###
# Normalize Features for supervised learning
###

#set NAs = 0
DataMatrixFinal[is.na(DataMatrixFinal)] <- 0

#Normalize
normalize <- function(x) {
  return((x-min(x))/(max(x)-min(x)))
}

DataMatrix.Normalized <- as.data.frame(lapply(DataMatrixFinal[2:ncol(DataMatrixFinal)],normalize))
```

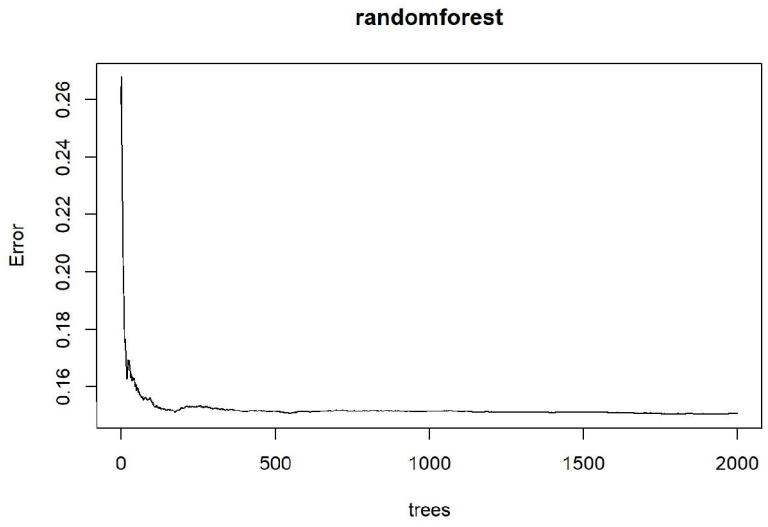
```
###
# Run Random Forest
###


#
#Final Split of dataset
#Train = 2010-2017
#Test = 2018 data (see how model would have done in 2018)
#Test.2019 = 2019 data (i.e. my submission for the competition)
#Note: For actual competition for the final Train data set i incorporated my 2018 data as well in my final algorithm

Train <- DataMatrix.Normalized[DataMatrix.Normalized$Season.x.x != 2019 & DataMatrix.Normalized$Season.x.x != 2018,]
Test <- DataMatrix.Normalized[DataMatrix.Normalized$Season.x.x ==2018,]
Test.2019 <- DataMatrix.Normalized[DataMatrix.Normalized$Season.x.x ==2019,]
DataMatrix.Normalized$Season.x.x <- NULL


# Set seed and run Random Forest model on train data
set.seed(1000)

randomforest <- randomForest(Result ~. -Team.TeamID.x-Team.TeamID.y-Key2,data = Train,ntree=2000,mtry =10,importance=TRUE)
```

```
## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values. Are you sure you want to do regression?
```

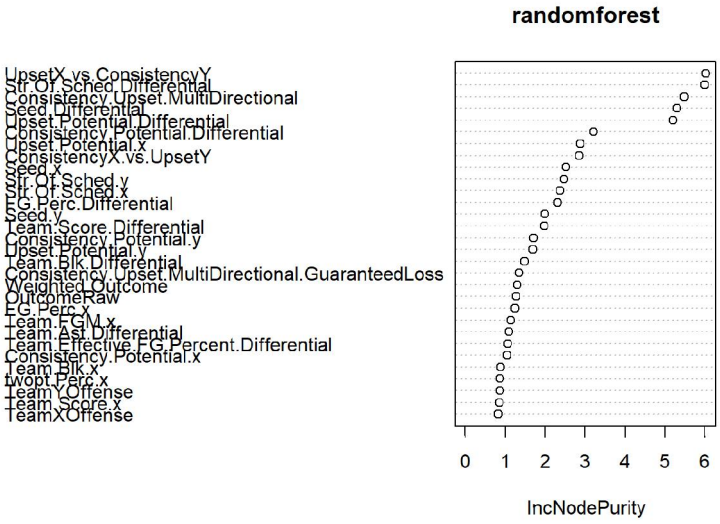
```
randomforest
```

```
##
## Call:
##  randomForest(formula = Result ~ . - Team.TeamID.x - Team.TeamID.y -      Key2, data = Train, ntree = 2000, mtry = 10, im
portance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 2000
## No. of variables tried at each split: 10
##
##           Mean of squared residuals: 0.150625
##                     % Var explained: 39.63
```

```
plot(randomforest) #shows that error rate levels around a little over 500 trees
```



```
# Run training model on test data
pred<-predict(randomforest,Train) #Predictions on Train Set for each Tree
varImpPlot(randomforest,type=2)

#This above visual shows which features were most relevant in our model

Train$pred <- pred

pred<-predict(randomforest,Test) #Predictions on Test Set for each Tree
varImpPlot(randomforest,type=2)
```

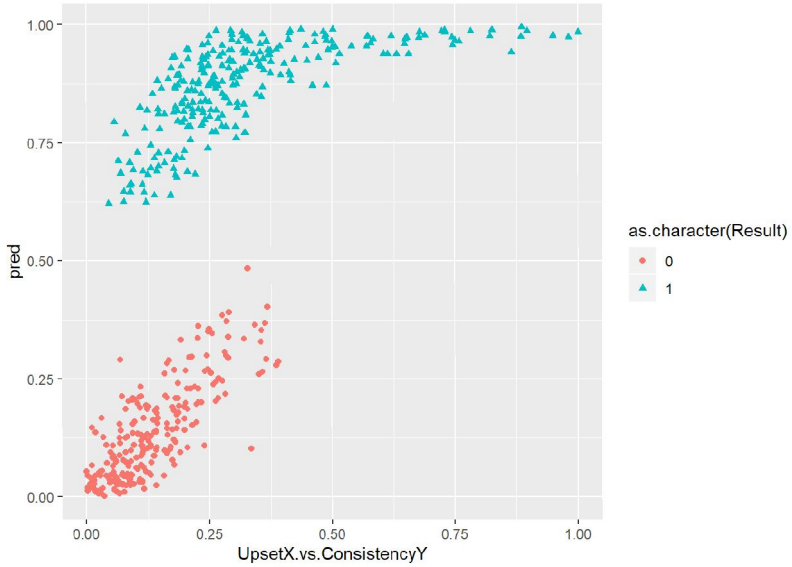



```
Test$pred <- pred

Test$difference.result <- abs(Test$pred - Test$Result)
Test$Log.Loss <- Test$Result*log(Test$pred)+(1-Test$Result)*log(1-Test$pred)
-mean(Test$Log.Loss) #.4534892 log loss in prior year this would have been good enough for top 55 out of 800 submissions
```

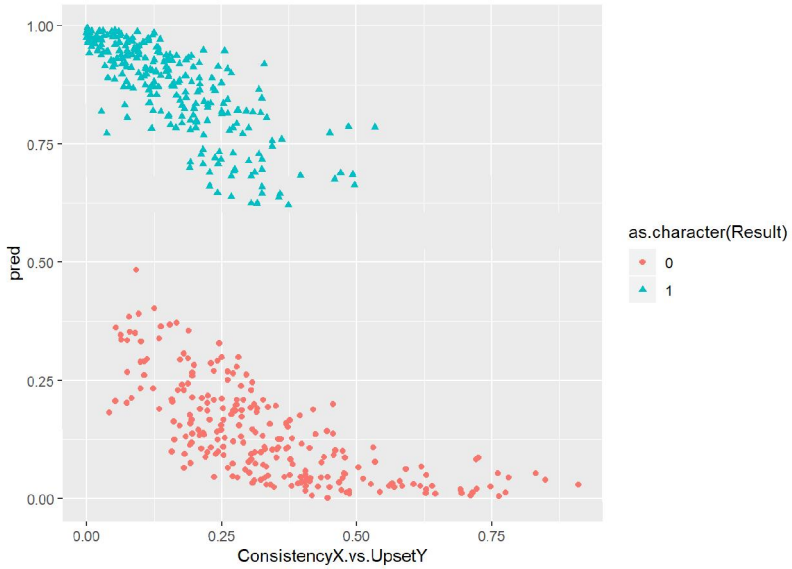
```
## [1] 0.4509685
```

```
#Create 2019 File

pred<-predict(randomforest,Test.2019) #Predictions on Test Set for each Tree
varImpPlot(randomforest,type=2)
Test.2019$pred <- pred

Final <- Test.2019[,c("Key2","pred")]
Final <- Final[order(Final$Key2),]
Final$Key2 <- gsub("-","_",Final$Key2)
setnames(Final,"Key2","ID")
setnames(Final,"pred","Pred")

write.csv(Final,"Womens.ML.Output1_ALL FEATURES.csv")
```

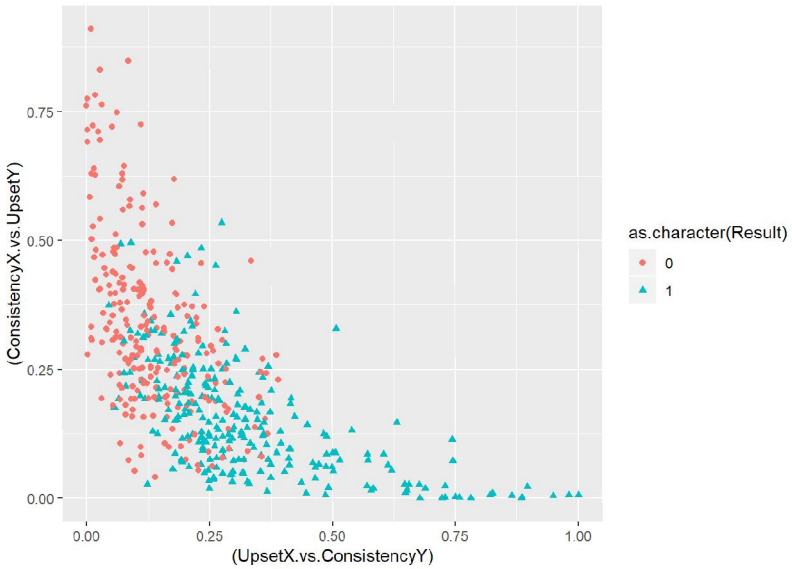
```
#Scatterplot results Random Forest
##These visuals go together
# Proves out that a team with not a lot of ranked wins (upset potential) cannot beat a team that has consistently beaten wor
se competition (consistency)
#Consistency and upset and how it aligns with predictions and actual wis and losses
qplot(UpsetX.vs.ConsistencyY, pred, colour = as.character(Result), shape = as.character(Result),
      data = Train)  # Great visual every team with a value over .5 has won
```



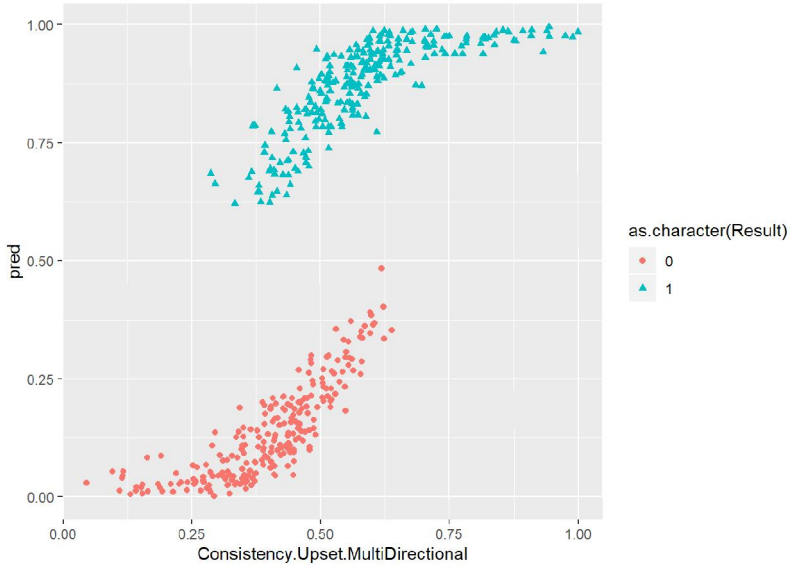

```
qplot(ConsistencyX.vs.UpsetY, pred, colour = as.character(Result), shape = as.character(Result),
      data = Train)  # Great visual every team except 1 with a value over .5 has won
```



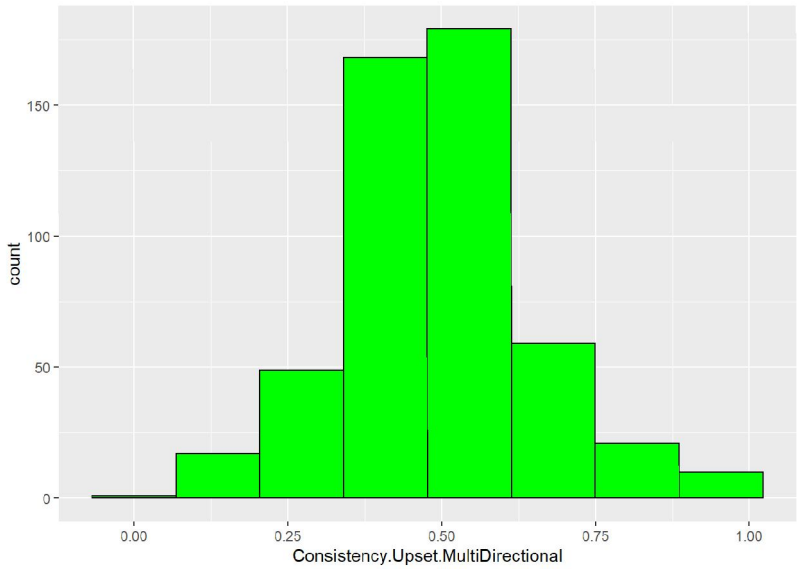

```
qplot((UpsetX.vs.ConsistencyY),(ConsistencyX.vs.UpsetY), colour = as.character(Result), shape = as.character(Result),
      data = Train)
```




```
qplot(Consistency.Upset.MultiDirectional, pred, colour = as.character(Result), shape = as.character(Result),
      data = Train)  # Combining the upset and consistency features together to show a very low score always is loss and ver
y high score always is a win. To digest this feature basically it is saying if a team is really consistent and the other tea
m has no signature wins they wont be able to win that game
```




```
ggplot(Train, aes(x=Consistency.Upset.MultiDirectional)) +
  geom_histogram(colour="black", fill="green",bins = 8) #This analyzes the distribution of the feature about str of schedule
(consistency.upset.multidirectional) that compares teams str of schedule measures against eachother for each given matchup.
```



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.