Final Project

Ethan, Stephen, William

5/6/2021

```
##Connect

con <- dbConnect(drv=RSQLite::SQLite(), dbname="database.sqlite")

##Getting the tables

tables <- dbListTables(con)
tables <- tables[tables != "sqlite_sequence"]

##Reading in SQL DATA

country = dbReadTable(con, "Country")
league = dbReadTable(con, "League")
matches = dbReadTable(con, "Match")
player = dbReadTable(con, "Player")
player_Attributes = dbReadTable(con, "Player_Attributes")
teams = dbReadTable(con, "Team")
team_attributes = dbReadTable(con, "Team_Attributes")

##Disconnect from Database
dbDisconnect(con)


# Importing dataset made by Stephen
library(readr)
team_ratings_dataset <- read_csv("dataset.csv")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   league = col_character(),
##   date = col_date(format = ""),
##   home_team = col_character(),
##   away_team = col_character(),
##   match_api_id = col_double(),
##   home_team_goal = col_double(),
##   away_team_goal = col_double(),
##   home_team_rating = col_double(),
##   away_team_rating = col_double(),
##   match_result = col_character()
## )
```

Data Source

This data is an SQL database containing information for more than 25,000 matches and 10,000 players from European professional football.

The source of the data originates from:

- http://football-data.mx-api.enetscores.com/ : scores, lineup, team formation and events
- http://www.football-data.co.uk/ : betting odds.
- http://sofifa.com/ : players and teams attributes from EA Sports FIFA games.

This can be found at https://www.kaggle.com/hugomathien/soccer where these are all compiled into one large dataset.

Team Rating Database

```
head(team_ratings_dataset)
```

```
## # A tibble: 6 x 10
##   league     date       home_team      away_team     match_api_id home_team_goal
##   <chr>      <date>     <chr>          <chr>                <dbl>          <dbl>
## 1 France Li~ 2008-08-09 Girondins de ~ SM Caen             483130              2
## 2 France Li~ 2008-08-09 Le Havre AC    OGC Nice            483131              1
## 3 France Li~ 2008-08-10 Olympique Lyo~ Toulouse FC         483133              3
## 4 France Li~ 2008-08-09 AS Monaco      Paris Saint-~       483134              1
## 5 France Li~ 2008-08-09 AS Nancy-Lorr~ LOSC Lille          483135              0
## 6 France Li~ 2008-08-09 FC Sochaux-Mo~ Grenoble Foo~       483137              1
## # ... with 4 more variables: away_team_goal <dbl>, home_team_rating <dbl>,
## #   away_team_rating <dbl>, match_result <chr>
```

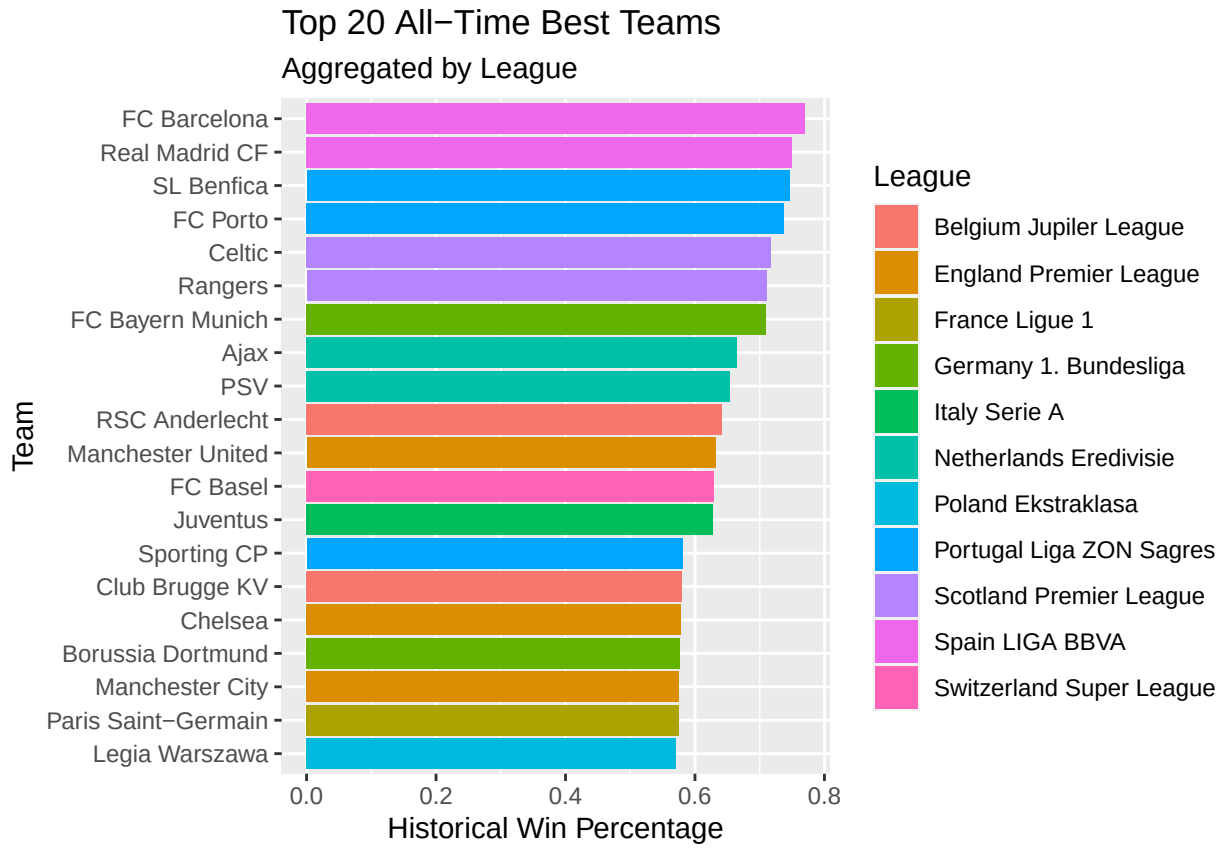
- This is a csv database created by Stephen that calculates the average team rating, derived from the average player ratings of all players on each team.
- The data used to create these values was all from our orginal dataset.
- To save computational time, we simply imported the data set instead of running the code every time to create it. This database was created in R, the code used can be found in the file DataCleanseCode.R

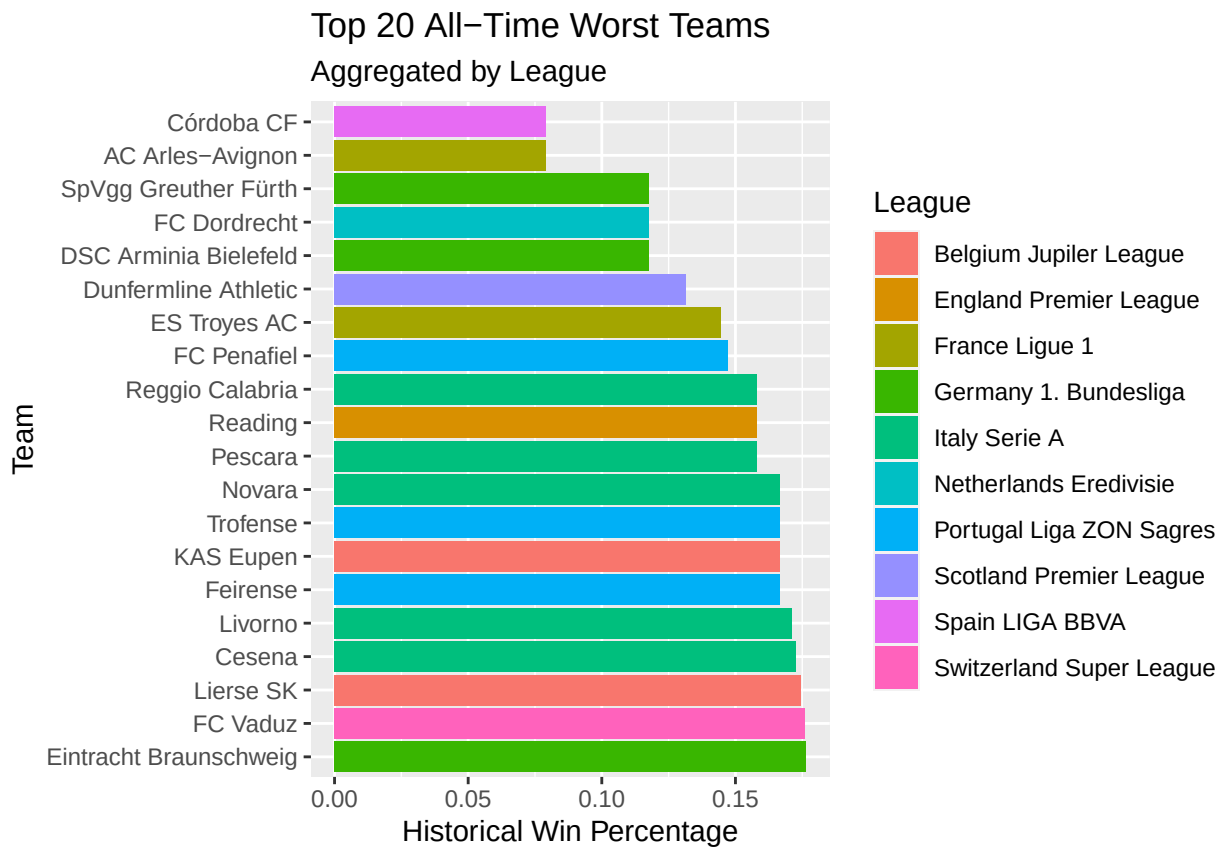
```
## Joining, by = "country_id"
```

```
## `summarise()` has grouped output by 'home_team'. You can override using the `.groups` argument.
```

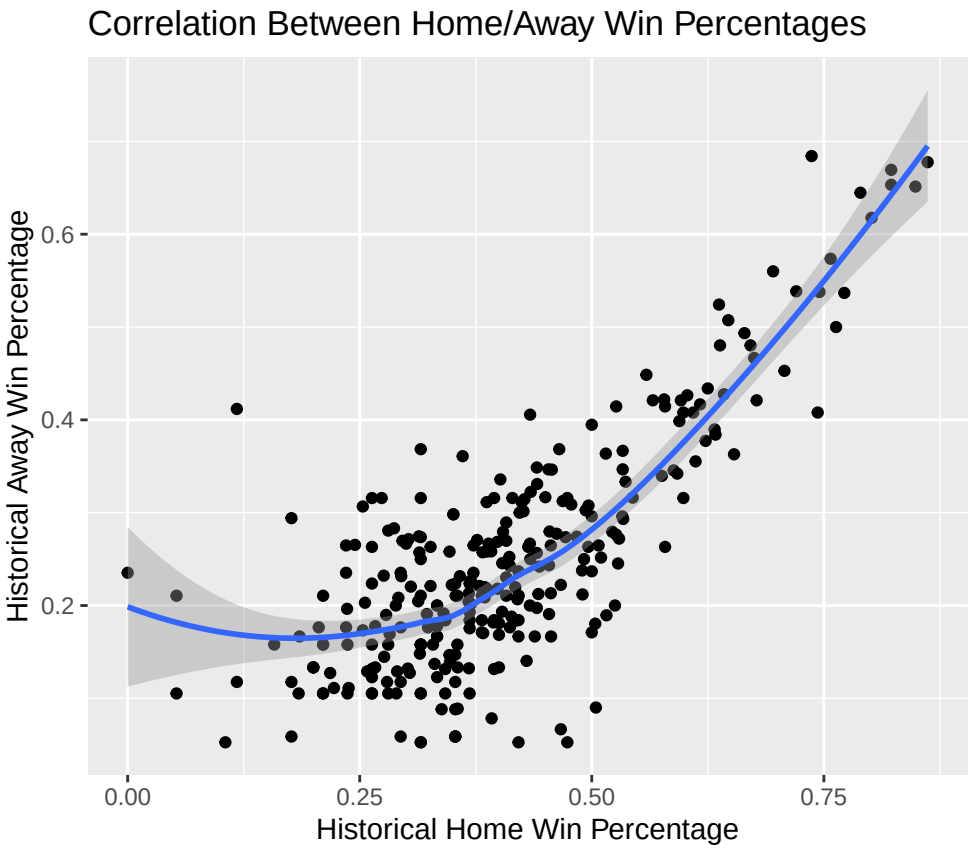
```
## `summarise()` has grouped output by 'away_team'. You can override using the `.groups` argument.
```

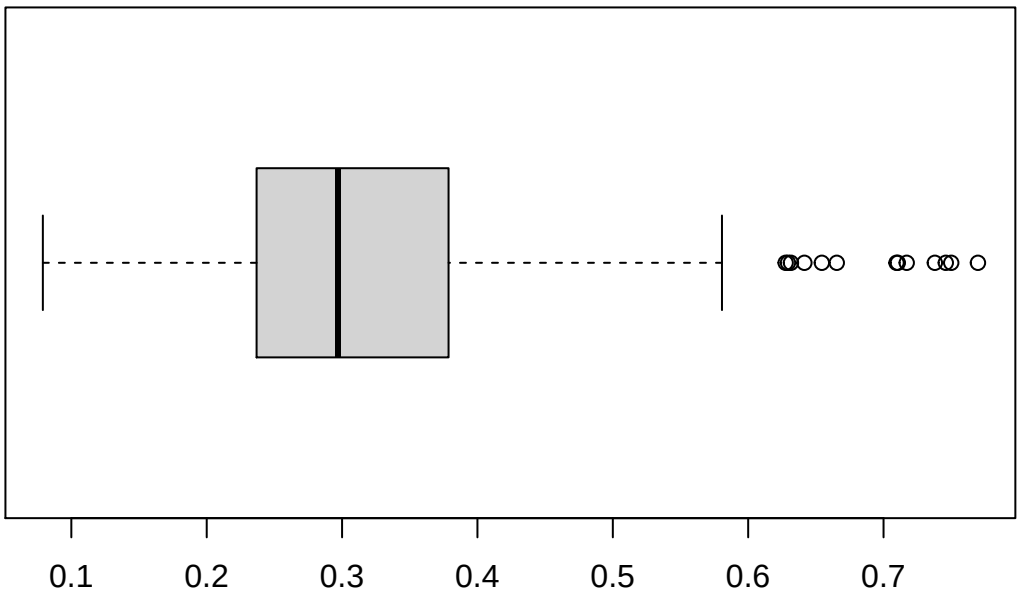
```
## `summarise()` has grouped output by 'team'. You can override using the `.groups` argument.
```

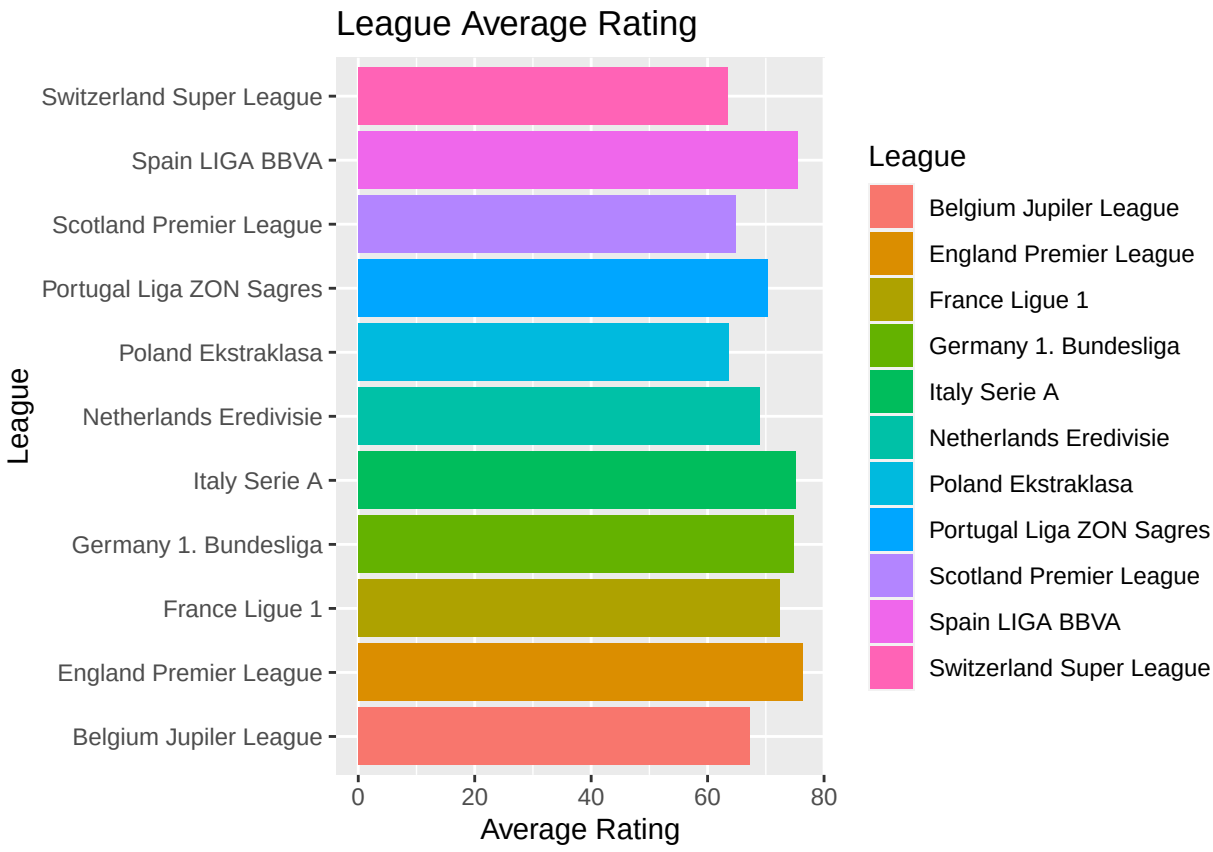





```
## `geom_smooth()` using formula 'y ~ x'
```


Correlation Between Home/Away Win Percentages
Historical Away Win Percentage
0.6
0.4
0.2
0.00
0.25
0.50
0.75
Historical Home Win Percentage


Distribution of Win Probabilities
0.1
0.2
0.3
0.4
0.5
0.6
0.7



Summary Statstics and Visualizations

Here shows some visualizations of the data, the first showing the top 20 teams in terms of win percentage and the second showing the bottom 20 teams in terms of win percentage. This should give us a good idea of what to expect out of these teams by the end of the final project.

This means that the best team, FC Barcelona, should be predicted to do very well and that the worst team, Cordoba CF, should be predicted to do very poorly. If our model does not reflect this, it will be obvious that we have made an error.

In addition to this, we thought it would be interesting to show the correlation between home winning percent and away winning percent. As expected, we see a largely linear line of best fit. This means that a team that wins a lot of home games is likely to also win a lot of away games. However, there were some irregularities at low winning percentage caused by sparse observations and outliers.

Next, we look at the average rating of each league. From the graph, we can see that Spain LIGA BBVA and England Premier League have the highest average ratings, whereas Switzerland Super League and Poland Ekstraklasa have the lowest.

Lastly, we wanted to look at the distribution of win probabilities as you recommended in the Data Analysis. As you can see, there are some very extreme outliers with a very high win percentages, which is something that we really did not expect.

Data Description

```
Data_Description<-read_excel('Data Description.xlsx')
print(Data_Description, n=Inf)
```

```
## # A tibble: 37 x 4
##    `Variable Name`   Description        `Possible Values` `Interpretation (If a~
##    <chr>             <chr>              <chr>             <chr>
##  1 id                ID that represent~ Positive Integers <NA>
##  2 team_fifa_api_id  ID that represent~ Positive Integers <NA>
##  3 away_team_api_id  ID that represent~ Positive integers <NA>
##  4 buildUpPlaySpeed  Represents  the s~ Positive Integer~ Higher number implies~
##  5 buildUpPlaySpeed~ Brackets based of~ Slow, Fast, Bala~ <NA>
##  6 buildUpPlayDribb~ Represents the dr~ Positive Integer~ Higher number implies~
##  7 buildUpPlayDribb~ Brackets based of~ Little, Normal, ~ <NA>
##  8 buildUpPlayPassi~ Representation of~ Positive Integers Higher number implies~
##  9 buildUpPlayPassi~ Brackets based of~ Short, Mixed, Lo~ <NA>
## 10 buildUpPlayPosit~ Represents how th~ Organised, Free ~ Organized implies the~
## 11 chanceCreationPa~ Represents the ab~ Positive Integer~ Higher number implies~
## 12 chanceCreationPa~ Brackets based of~ Safe, Normal, Ri~ <NA>
## 13 chanceCreationCr~ Represents the ab~ Positive Integer~ Higher number implies~
## 14 chanceCreationCr~ Brackets based of~ Little, Normal, ~ <NA>
## 15 chanceCreationSh~ Represents the ab~ Positive Integer~ Higher number implies~
## 16 chanceCreationSh~ Brackets based of~ Little, Normal, ~ <NA>
## 17 chanceCreationPo~ Represents how th~ Organised, Free ~ Organized implies the~
## 18 defencePressure   Represents the pr~ Positive Integer~ Higher number implies~
## 19 defencePressureC~ Brackets based of~ High, Medium, De~ High means full press~
## 20 defenceAggression Represents how ag~ Positive Integer~ Higher number implies~
## 21 defenceAggressio~ Brackets based of~ Press, Double, C~ Press implies more ag~
## 22 defenceTeamWidth  Represents how wi~ Positive Integer~ Higher number implies~
## 23 defenceTeamWidth~ Brackets based of~ Narrow, Normal, ~ <NA>
## 24 defenceDefenderL~ Represents playst~ Cover, Offside T~ Cover is the standard~
## 25 season            Shows the soccer ~ 20XX/20YY         <NA>
## 26 match_id          ID representing t~ Positive Integers <NA>
## 27 league_id         ID representing t~ Positive Integers <NA>
## 28 date              Shows the date of~ Year-Month-Day    <NA>
## 29 home_team_api_id  ID representing t~ Positive Integers <NA>
## 30 home_team_goal    Number of goals t~ Positive Integers <NA>
## 31 away_team_goal    Number of goals t~ Positive Integers <NA>
## 32 match_score       Outcome of game f~ Win, Tie, Loss    <NA>
## 33 home_team_name    Full name of the ~ Characters        <NA>
## 34 away_team_name    Full name of the ~ Characters        <NA>
## 35 home_team_name_S  Shortened name of~ Three letter cha~ <NA>
## 36 away_team_name_S  Shortened name of~ Three letter cha~ <NA>
## 37 country           Country in which ~ Characters        <NA>
```

- This is a xlsx that William created in excel that had a description of all of the variables that was made from scratch.
- For clarity we also attached the xlsx document to our submission

---

In order to help with some of the code for data cleaning, we referenced:

- https://www.kaggle.com/abharg16/predicting-epl-team-season-win-percentages/data

```
team_ratings <- read_csv("team_ratings.csv")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   league = col_character(),
##   date = col_date(format = ""),
##   home_team = col_character(),
##   away_team = col_character(),
##   match_api_id = col_double(),
##   home_team_goal = col_double(),
##   away_team_goal = col_double(),
##   home_team_rating = col_double(),
##   away_team_rating = col_double(),
##   match_result = col_character()
## )
```

```
team_win_streaks <- read_csv("team_win_streaks.csv")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   match_api_id = col_double(),
##   home_win_streak = col_double(),
##   away_win_streak = col_double()
## )
```

```
                    ### COMBINE DATASETS ###
Dataset <- Dataset %>%
  left_join(team_ratings %>% select(match_api_id, home_team_rating, away_team_rating),
            by = "match_api_id")

Dataset <- bind_cols(Dataset %>% arrange(match_api_id),
                     team_win_streaks %>% arrange(match_api_id)) %>%
  rename(match_api_id = match_api_id...30) %>%
  select(-match_api_id...42)
```

```
## New names:
## * match_api_id -> match_api_id...30
## * match_api_id -> match_api_id...42
```

```
Dataset <- Dataset %>%
  relocate(match_score)

                    ### FEATURE ENGINEERING FOR THINGS THAT WONT WORK IN RECIPE
## DROP MISSING MATCHES
Dataset <- Dataset %>%
```

```
  drop_na()

## CONVERT CHARACTERS TO FACTORS
Dataset <- Dataset %>%
  mutate_if(is.character, as.factor) %>%
  mutate(match_score = as.factor(match_score))

## REMOVE UNWANTED FEATURES
Dataset <- Dataset %>%
  select(-id, -team_fifa_api_id, -away_team_api_id, -date.x, -season, -match_id, -league_id, -date.y,
         -match_api_id, -home_team_api_id, -home_team_name, -away_team_name, -home_team_api_id,
         -home_team_name_S, -away_team_name_S, -country, -home_team_goal, -away_team_goal,
         -buildUpPlayDribbling)
```

XGBoost Model

```
                    ### BUILDING MODEL ###
## SPLITTING DATA
set.seed(1)
soccer_split <- initial_split(Dataset, strata = match_score)
soccer_train <- training(soccer_split)
soccer_test <- testing(soccer_split)

## MODEL SPECIFICATIONS
xgb_model <- boost_tree(
  trees = 1000,
  tree_depth = tune(), min_n = tune(),
  loss_reduction = tune(),
  sample_size = tune(), mtry = tune(),
  learn_rate = tune(),
) %>%
  set_engine("xgboost") %>%
  set_mode("classification")

xgb_model
```

```
## Boosted Tree Model Specification (classification)
##
## Main Arguments:
##   mtry = tune()
##   trees = 1000
##   min_n = tune()
##   tree_depth = tune()
##   learn_rate = tune()
##   loss_reduction = tune()
##   sample_size = tune()
##
## Computational engine: xgboost
```

```r
## GRID SPECIFICATIONS
xgb_grid <- grid_latin_hypercube(
  tree_depth(),
  min_n(),
  loss_reduction(),
  sample_size = sample_prop(),
  finalize(mtry(), soccer_train),
  learn_rate(),
  size = 30
)

kable(head(xgb_grid))
```

tree_depth	min_n	loss_reduction	sample_size	mtry	learn_rate
5	7	0.00e+00	0.9707212	7	0.0208823
12	32	3.13e-05	0.9114690	22	0.0001123
6	10	0.00e+00	0.8960829	16	0.0000000
6	3	0.00e+00	0.6865963	10	0.0915586
2	4	0.00e+00	0.5484144	15	0.0000010
14	5	1.00e-07	0.5735252	21	0.0000352

```r
## SETTING UP WORKFLOW
xgb_workflow <- workflow() %>%
  add_formula(match_score ~ .) %>%
  add_model(xgb_model)

xgb_workflow
```

```
## == Workflow ====================================================================
## Preprocessor: Formula
## Model: boost_tree()
##
## -- Preprocessor ----------------------------------------------------------------
## match_score ~ .
##
## -- Model -----------------------------------------------------------------------
## Boosted Tree Model Specification (classification)
##
## Main Arguments:
##   mtry = tune()
##   trees = 1000
##   min_n = tune()
##   tree_depth = tune()
##   learn_rate = tune()
##   loss_reduction = tune()
##   sample_size = tune()
##
## Computational engine: xgboost
```

```r
## FOLDS
set.seed(1)
vb_folds <- vfold_cv(soccer_train, strata = match_score)

vb_folds
```

```
## #  10-fold cross-validation using stratification
## # A tibble: 10 x 2
##    splits             id
##    <list>             <chr>
##  1 <split [3841/428]> Fold01
##  2 <split [3841/428]> Fold02
##  3 <split [3842/427]> Fold03
##  4 <split [3842/427]> Fold04
##  5 <split [3842/427]> Fold05
##  6 <split [3842/427]> Fold06
##  7 <split [3842/427]> Fold07
##  8 <split [3842/427]> Fold08
##  9 <split [3843/426]> Fold09
## 10 <split [3844/425]> Fold10
```

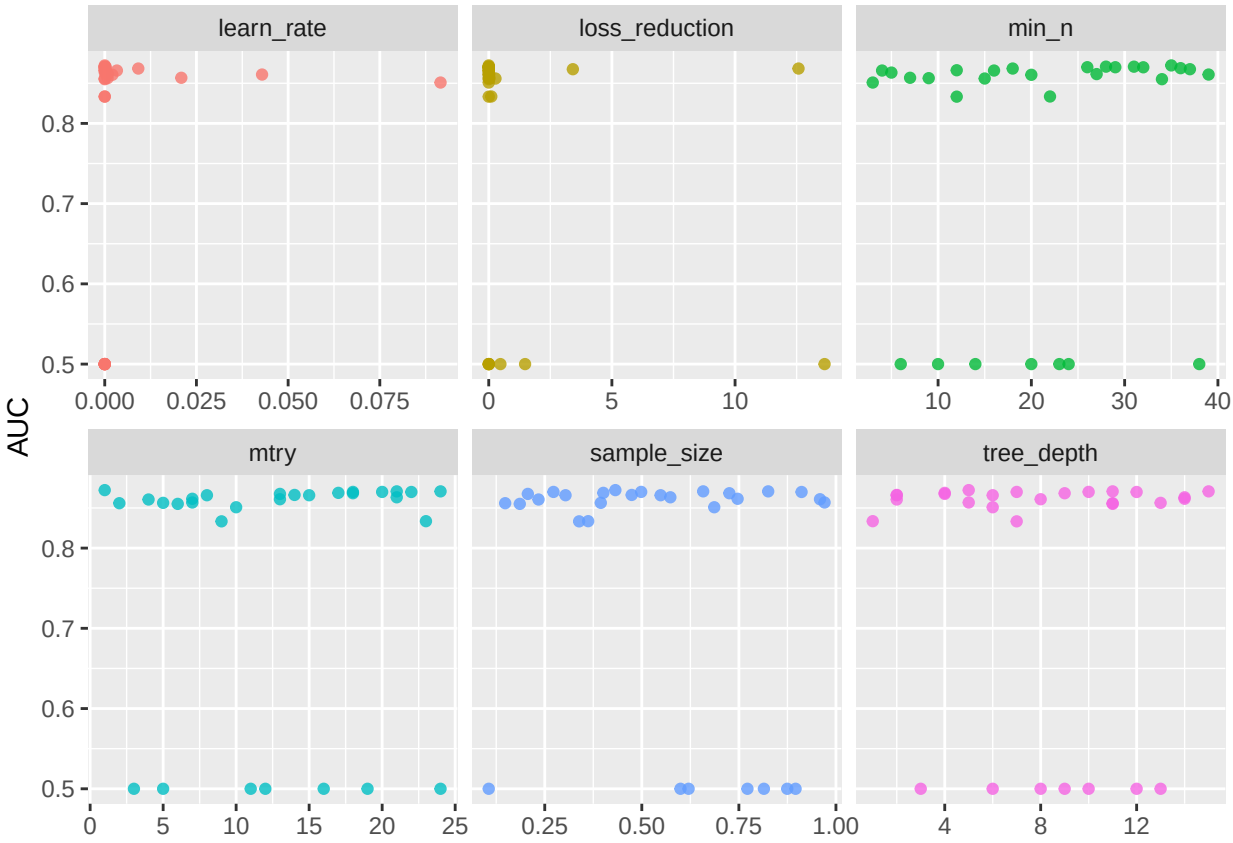
```r
## PRAY FOR MY PC
doParallel::registerDoParallel()

## TUNE GRID
set.seed(1)
xgb_results <- tune_grid(
  xgb_workflow,
  resamples = vb_folds,
  grid = xgb_grid,
  control = control_grid(save_pred = TRUE)
)

view(xgb_results$.notes)

xgb_results
```

```
## # Tuning results
## # 10-fold cross-validation using stratification
## # A tibble: 10 x 5
##    splits            id     .metrics          .notes          .predictions
##    <list>            <chr>  <list>            <list>          <list>
##  1 <split [3841/428~ Fold01 <tibble [60 x 1~ <tibble [0 x ~ <tibble [12,840 x 1~
##  2 <split [3841/428~ Fold02 <tibble [60 x 1~ <tibble [0 x ~ <tibble [12,840 x 1~
##  3 <split [3842/427~ Fold03 <tibble [60 x 1~ <tibble [0 x ~ <tibble [12,810 x 1~
##  4 <split [3842/427~ Fold04 <tibble [60 x 1~ <tibble [0 x ~ <tibble [12,810 x 1~
##  5 <split [3842/427~ Fold05 <tibble [60 x 1~ <tibble [0 x ~ <tibble [12,810 x 1~
##  6 <split [3842/427~ Fold06 <tibble [60 x 1~ <tibble [0 x ~ <tibble [12,810 x 1~
##  7 <split [3842/427~ Fold07 <tibble [60 x 1~ <tibble [0 x ~ <tibble [12,810 x 1~
##  8 <split [3842/427~ Fold08 <tibble [60 x 1~ <tibble [0 x ~ <tibble [12,810 x 1~
##  9 <split [3843/426~ Fold09 <tibble [60 x 1~ <tibble [0 x ~ <tibble [12,780 x 1~
## 10 <split [3844/425~ Fold10 <tibble [60 x 1~ <tibble [0 x ~ <tibble [12,750 x 1~
```

```
xgb_results %>% collect_metrics()
```

```
## # A tibble: 60 x 12
##     mtry min_n tree_depth learn_rate loss_reduction sample_size .metric
##    <int> <int>      <int>      <dbl>          <dbl>       <dbl> <chr>
##  1     7     7          5   2.09e- 2 0.000000000956       0.971 accuracy
##  2     7     7          5   2.09e- 2 0.000000000956       0.971 roc_auc
##  3    22    32         12   1.12e- 4 0.0000313            0.911 accuracy
##  4    22    32         12   1.12e- 4 0.0000313            0.911 roc_auc
##  5    16    10          6   7.31e-10 0.00000000750        0.896 accuracy
##  6    16    10          6   7.31e-10 0.00000000750        0.896 roc_auc
##  7    10     3          6   9.16e- 2 0.000000000157       0.687 accuracy
##  8    10     3          6   9.16e- 2 0.000000000157       0.687 roc_auc
##  9    15     4          2   9.56e- 7 0.00000000307        0.548 accuracy
## 10    15     4          2   9.56e- 7 0.00000000307        0.548 roc_auc
## # ... with 50 more rows, and 5 more variables: .estimator <chr>, mean <dbl>,
## #   n <int>, std_err <dbl>, .config <chr>
```

```
## VISUALIZING ACCURACY METRICS
xgb_results %>%
  collect_metrics() %>%
  filter(.metric == "roc_auc") %>%
  select(mean, mtry:sample_size) %>%
  pivot_longer(mtry:sample_size,
               values_to = "value",
               names_to = "parameter"
  ) %>%
  ggplot(aes(value, mean, color = parameter)) +
  geom_point(alpha = 0.8, show.legend = FALSE) +
  facet_wrap(~parameter, scales = "free_x") +
  labs(x = NULL, y = "AUC")
```




```
## BEST PERFORMING SET OF PARAMETERS
show_best(xgb_results, "accuracy")
```

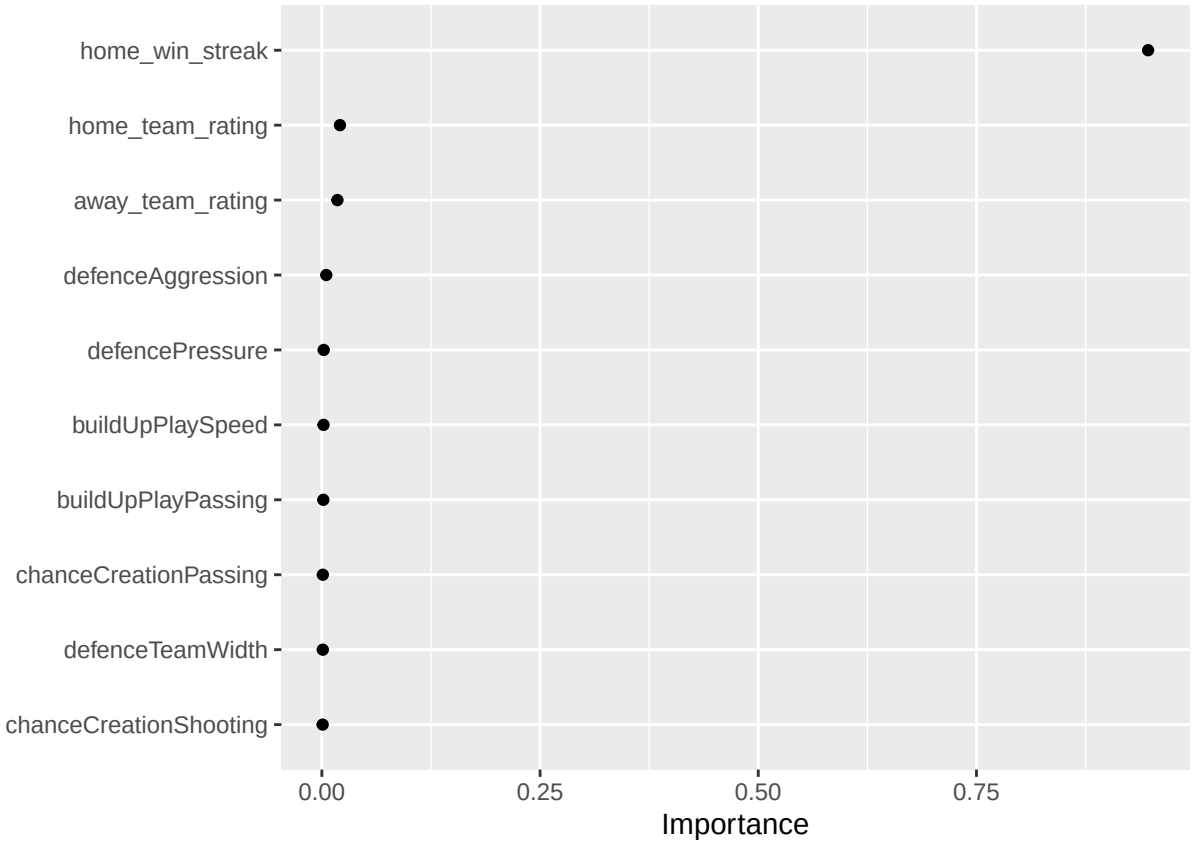
```
## # A tibble: 5 x 12
##    mtry min_n tree_depth  learn_rate loss_reduction sample_size .metric
##   <int> <int>      <int>       <dbl>          <dbl>       <dbl> <chr>
## 1     1    35          5 0.00000826    0.000253           0.432 accuracy
## 2    24    31         15 0.000000158   0.000000485        0.825 accuracy
## 3    21    28         11 0.0000155     0.0000154          0.658 accuracy
## 4    20    29          7 0.00000608    0.0000000421       0.272 accuracy
## 5    22    32         12 0.000112      0.0000313          0.911 accuracy
## # ... with 5 more variables: .estimator <chr>, mean <dbl>, n <int>,
## #   std_err <dbl>, .config <chr>
```

```
## SELECT BEST ACCURACY MEASUREMENT
best_accuracy <- select_best(xgb_results, "accuracy")
best_accuracy
```

```
## # A tibble: 1 x 7
##    mtry min_n tree_depth learn_rate loss_reduction sample_size .config
##   <int> <int>      <int>      <dbl>          <dbl>       <dbl> <chr>
## 1     1    35          5 0.00000826       0.000253       0.432 Preprocessor1_Mo~
```

```
## FINALIZE WORKFLOW
xgb_final <- finalize_workflow(
  xgb_workflow,
  best_accuracy
)

## VIP PLOT
xgb_final %>%
  fit(data = soccer_train) %>%
  pull_workflow_fit() %>%
  vip(geom = "point")
```

```
## [01:41:33] WARNING: amalgamation/../src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evalu
```




```
## FIT TO TRAINING DATA
final_results <- last_fit(xgb_final, soccer_split)
collect_metrics(final_results)
```

```
## # A tibble: 2 x 4
##   .metric  .estimator .estimate .config
##   <chr>    <chr>          <dbl> <chr>
## 1 accuracy multiclass     0.770 Preprocessor1_Model1
## 2 roc_auc  hand_till      0.874 Preprocessor1_Model1
```

```
#### STUFF TO SAVE
saveRDS(xgb_results, "./xgb_results.rds")
saveRDS(xgb_final, "./xgb_final.rds")
```

Final Summary of Project

In the end, we got a 76.8% accuracy, which for the scope of the project certainly isn't too bad. We believe the model was slightly overfit, but given the amount of time that this project has taken (much more than we expected), we chose to continue with this model. If we had more time, we would work on trying to make this model even better and maybe even do some more fine tuning on additional parameters.