

# Configuration Manual

MSc Research Project  
Data Analytics

**Murtaza Saifi**  
Student ID: X18129463

School of Computing  
National College of Ireland

Supervisor: Prof. Vladimir Milosavljevic

**National College of Ireland**  
**MSc Project Submission Sheet**  
**School of Computing**



**Student Name:** MURTAZA SAIFI  
**Student ID:** X1829463  
**Programme:** DATA ANALYTICS **Year:** 2019  
**Module:** MSC RESEARCH PROJECT  
**Lecturer:** Dr. Vladimir Milosavljevic  
**Submission Due Date:** 12/12/2019  
**Project Title:** Implementation of Machine Learning Techniques to Predict Player Performance using Underlying Statistics  
**Word Count:** 1288 **Page Count:** 13

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

**Signature:** .....

**Date:** .....

**PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST**

Attach a completed copy of this sheet to each project (including multiple copies)	<input type="checkbox"/>
<b>Attach a Moodle submission receipt of the online project submission,</b> to each project (including multiple copies).	<input type="checkbox"/>
<b>You must ensure that you retain a HARD COPY of the project,</b> both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	<input type="checkbox"/>

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

<b>Office Use Only</b>	
Signature:	
Date:	
Penalty Applied (if applicable):	

# Configuration Manual

Murtaza Saifi  
Student ID:x18129463  
MSc Research Project in Data Analytics  
11th December 2019

## 1 Introduction

The objective of this manual is to showcase the technical aspect of this project that involves system requirements and programming snippets that have not been covered in the main report. We will initiate with the basic system requirements utilized and discuss the implementation of the methodology.

### 1.1 System Requirement

- Hardware spec
  1. System Manufacturer: Dell Inc.
  2. Operating System: Windows 8.1 Pro 64-bit
  3. Processor: Intel(R) Core (TM) i5-4200U CPU @ 1.60GHZ (4 CPUs), ~2.3GHz
  4. Memory: 6 GB RAM
- Software spec
  1. R
  2. Tableau
  3. Microsoft Excel
  4. Kaggle (Kernel)

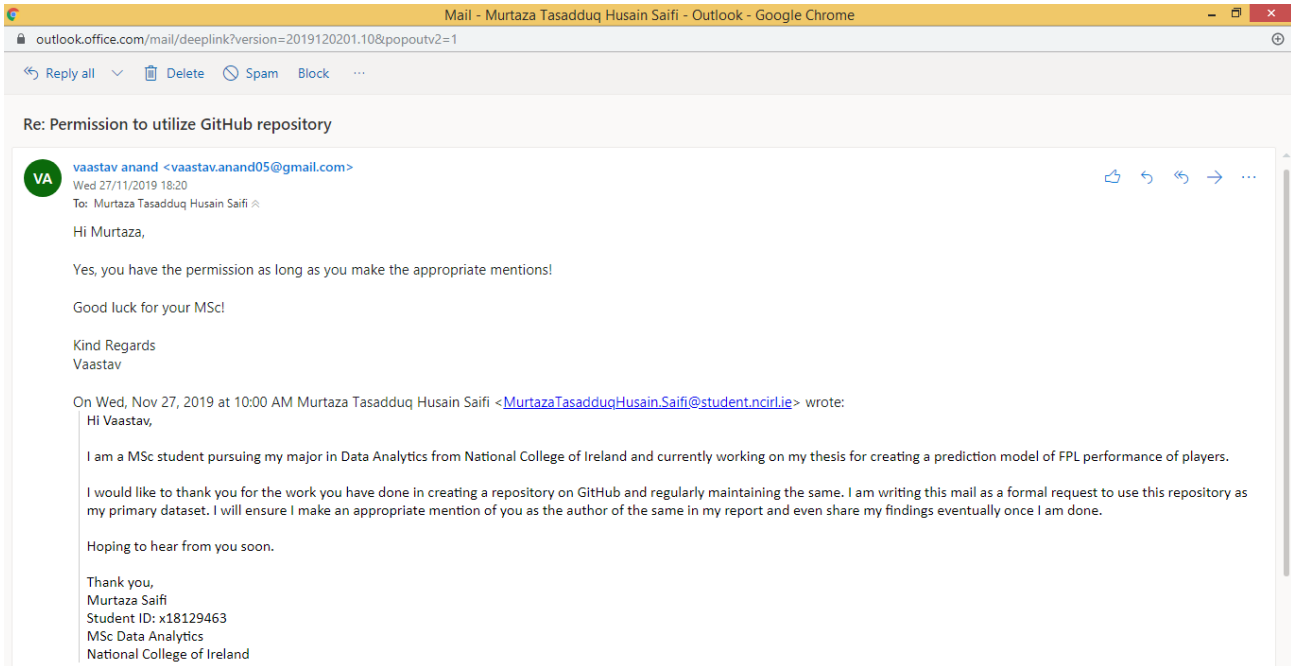
## 2 Project Development

Data preparation is done in multiple stages, between Excel & R studio. The code snapshots have been placed to avoid confusion of any kind.

### 2.1 Data Preparation

We primarily focus on 2 datasets in this study:

1. Fantasy Premier League Dataset: This is a Github repository (available here<sup>1</sup>) that has been active since 2016 and is sharing weekly updates sheets on match-weeks as they are conducted in the season. Permission to use the dataset has been taken via mail and can be observed below in Figure 1.



**Figure 1: Email discussion for permission of FPL Dataset use**

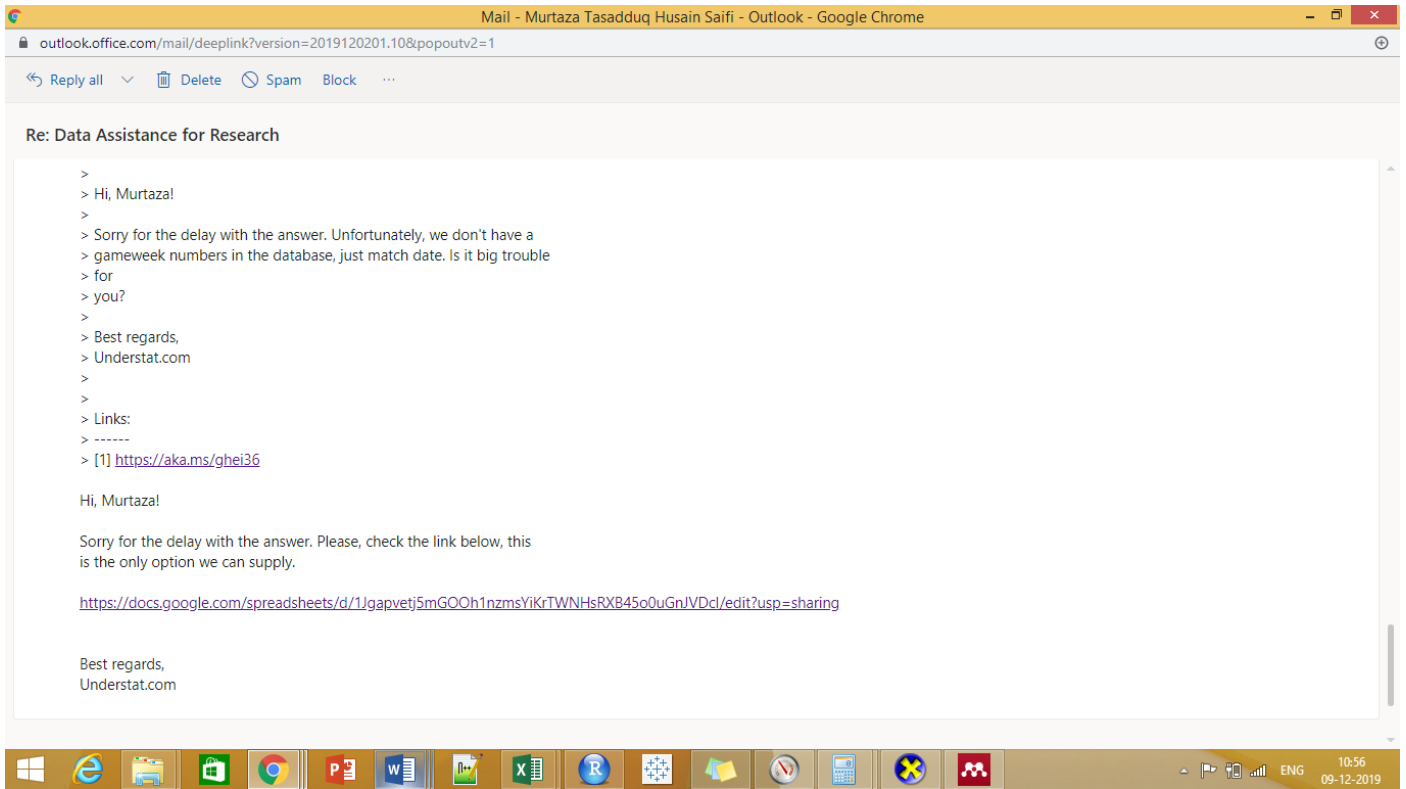
2. Understat.com Dataset: This dataset is obtained from the understat.com team which includes underlying statistics such as Expected Goals (xG) and Expected Assists (xA). The dataset<sup>2</sup> was shared post requesting via mail as can be observed from Figure 2. As we can observe, the communication for both datasets have taken place via the student email address.

---

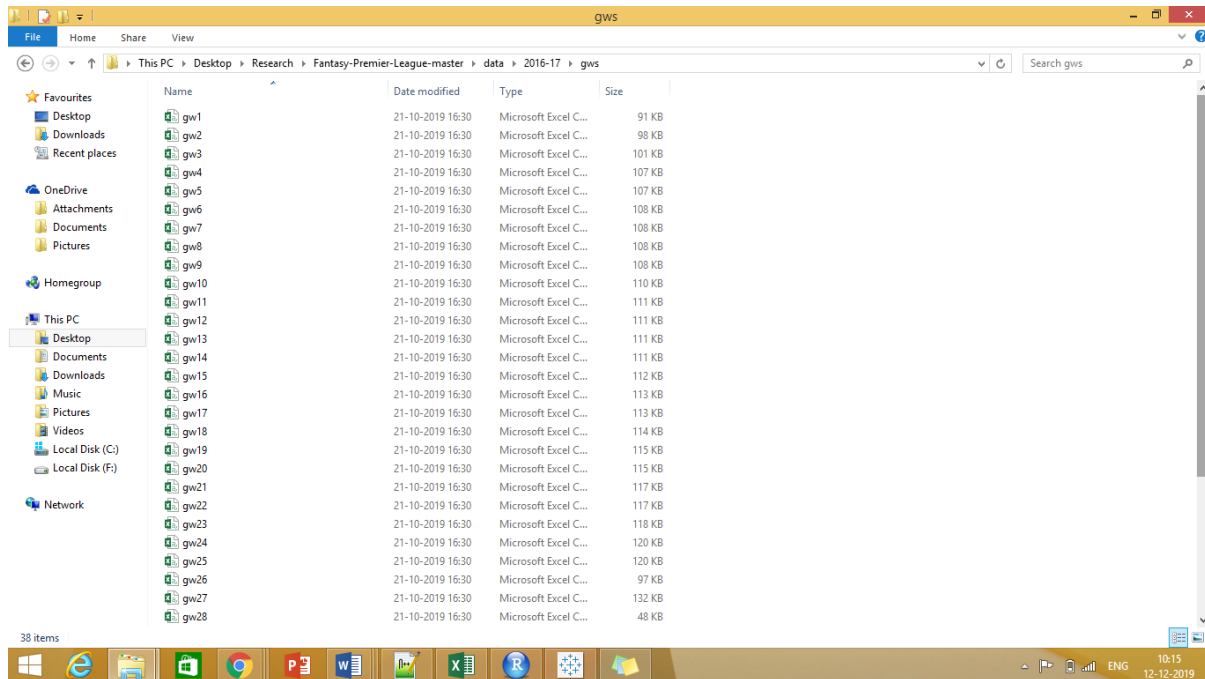
<sup>1</sup> <https://github.com/vaastav/Fantasy-Premier-League>

<sup>2</sup>

<https://docs.google.com/spreadsheets/d/1Jgapvetj5mGOOh1nzmsYiKrTWNHsRXB45o0uGnJVDcl/edit?usp=sharing>



**Figure 2: Mail response from understat.com sharing xG and xA data**



**Figure 3: Gameweek wise datasets**

Each week had its own dataset (which can be seen in Figure 3) which had to be merged to form a combined season dataset. Also, the numerical factor of the team name had to be converted back to its character as the same 20 teams to not play the following season. The code for this can be seen in Figure 4.

```

93 playerdata_s1g37 <- read.csv(s1g37, stringsAsFactors = T)
94 playerdata_s1g38 <- read.csv(s1g38, stringsAsFactors = T)
95
96 ###Merging all gameweek data to form a Seasons dataset
97 gw1and2 <- rbind(playerdata_s1g1,playerdata_s1g2,playerdata_s1g3,playerdata_s1g4,playerdata_s1g5,playerdata_s1g6,
98 playerdata_s1g7,playerdata_s1g8,playerdata_s1g9,playerdata_s1g10,playerdata_s1g11,playerdata_s1g12,
99 playerdata_s1g13,playerdata_s1g14,playerdata_s1g15,playerdata_s1g16,playerdata_s1g17,playerdata_s1g18,
100 playerdata_s1g19,playerdata_s1g20,playerdata_s1g21,playerdata_s1g22,playerdata_s1g23,playerdata_s1g24,
101 playerdata_s1g25,playerdata_s1g26,playerdata_s1g27,playerdata_s1g28,playerdata_s1g29,playerdata_s1g30,
102 playerdata_s1g31,playerdata_s1g32,playerdata_s1g33,playerdata_s1g34,playerdata_s1g35,playerdata_s1g36,
103 playerdata_s1g37,playerdata_s1g38)
104
105 gw1and2[["Season"]] <- "1"
106 gw1and2$opponent_team <- as.factor(gw1and2$opponent_team)
107 gw1and2$opponent_team <- factor(gw1and2$opponent_team, levels = c(1:20), labels=c("Arsenal","Bournemouth","Burnley",
108 "Chelsea","Crystal Palace","Everton",
109 "Hull","Leicester","Liverpool",
110 "Manchester City","Manchester United",
111 "Middlesbrough","Southampton","Stoke",
112 "Sunderland","Swansea","Tottenham",
113 "Watford","West Bromwich Albion",
114 "West Ham"))
115
116 ##write.csv(gw1and2,"Season1CombinedData.csv",row.names = FALSE) ##temporary commented
117
118 rm(list =c("playerdata_s1g1","playerdata_s1g10","playerdata_s1g11","playerdata_s1g12","playerdata_s1g13","playerdata_s1g14"
119 "playerdata_s1g15","playerdata_s1g16","playerdata_s1g17","playerdata_s1g18","playerdata_s1g19","playerdata_s1g2"
120 "playerdata_s1g20","playerdata_s1g21","playerdata_s1g22","playerdata_s1g23","playerdata_s1g24","playerdata_s1g25"
121 "playerdata_s1g26","playerdata_s1g27","playerdata_s1g28","playerdata_s1g29","playerdata_s1g3","playerdata_s1g30"
122 "playerdata_s1g31","playerdata_s1g32","playerdata_s1g33","playerdata_s1g34","playerdata_s1g35","playerdata_s1g36"
123 "playerdata_s1g37","playerdata_s1g38","playerdata_s1g4","playerdata_s1g5","playerdata_s1g6","playerdata_s1g7"
124 "playerdata_s1g8","playerdata_s1g9"))
125
126 rm(list =c("s1g1","s1g10","s1g11","s1g12","s1g13","s1g14"
127 "s1g15","s1g16","s1g17","s1g18","s1g19","s1g2"
128 "s1g20","s1g21","s1g22","s1g23","s1g24","s1g25"
129

```

Figure 4: Merging Gameweek datasets

```

432 ### Season3
433 raw_players_s3 <- "2018-19/players_raw.csv"
434
435 raw_players_data_s3 <- read.csv(raw_players_s3, stringsAsFactors = T)
436
437
438 raw_players_data_s3[["Player_Name"]] <- paste(raw_players_data_s3$first_name,raw_players_data_s3$second_name,sep = "_")
439
440 raw_players_data_s3$Player_Name <- iconv(raw_players_data_s3$Player_Name, from="UTF-8", to="LATIN1")
441
442
443 raw_players_data_s3$team <- as.factor(raw_players_data_s3$team)
444 raw_players_data_s3$element_type <- as.factor(raw_players_data_s3$element_type)
445
446
447 raw_players_data_s3$team <- factor(raw_players_data_s3$team, levels = c(1:20), labels=c("Arsenal","Bournemouth","Brighton","Burnley",
448 "Cardiff","Chelsea","Crystal Palace","Everton",
449 "Fulham","Huddersfield","Leicester","Liverpool",
450 "Manchester City","Manchester United",
451 "Newcastle United","Southampton",
452 "Tottenham","Watford","West Ham",
453 "Wolverhampton Wanderers"))
454
455 raw_players_data_s3$element_type <- factor(raw_players_data_s3$element_type, levels = c(1:4), labels = c("GK","DEF","MID","FWD"))
456
457 raw_players_data_s3 <- raw_players_data_s3[,c(59,1,2,3:58)] ##Bringing player name as column 1
458 raw_players_data_s3 <- raw_players_data_s3[order(raw_players_data_s3$Player_Name),] ##Arranging in alphabetical order
459
460
461 ##### Matching Team Name and Position with Player Name from DB2 to DB1 #####
462
463 ##Season 1
464 gw1and2$Team <- raw_players_data_s3$team[match(gw1and2$name,raw_players_data_s3$Player_Name)]
465 gw1and2$Position <- raw_players_data_s3$element_type[match(gw1and2$name,raw_players_data_s3$Player_Name)]
466

```

Figure 5: Matching Player Name from raw file to obtain Position

We then worked on the raw player dataset to obtain player position from the file. This would be done by matching player name and team. Hence, we had to make a naming conversion from 'UTF-8' to 'LATIN1' to handle the special characters.

Once the data from Dataset 1 had been processed, we initiated our work on dataset 2. Similar to working on the raw player file, we were going to match multiple parameters to pull the underlying stats of Expected Goals (xG) and Expected Assists (xA). These parameters were: Player Name, Player Team, Player Opponent and Match Date. After a first round of attempting a match we found a lot of missing values in the combined dataset. On further inspection, it was observed that certain games had a difference of 1 day in their date and also

the dataset provided by understat held player names as the popularly known names or nicknames while FPL had their official names. Hence, we had to pull the names from the two datasets and find a workaround on Excel (shown is Figure 6).

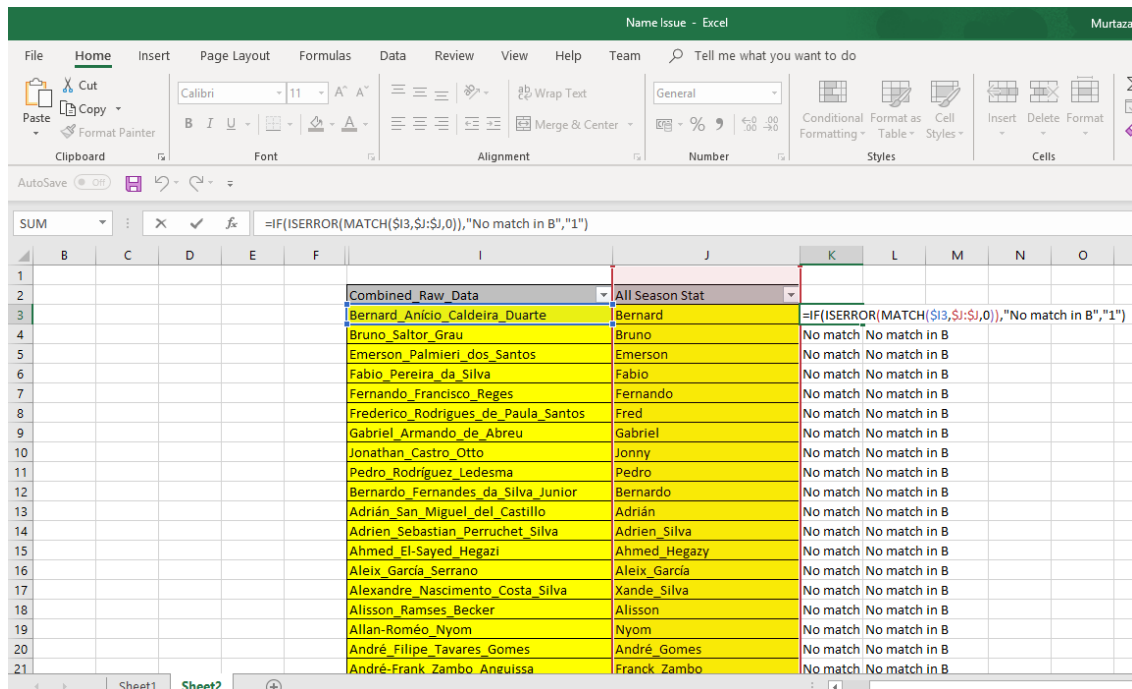


Figure 6: Player Name mismatch analysis in Excel

There were around 85 players with name mismatches which had to be then manually substituted by using the sub() function in RStudio.

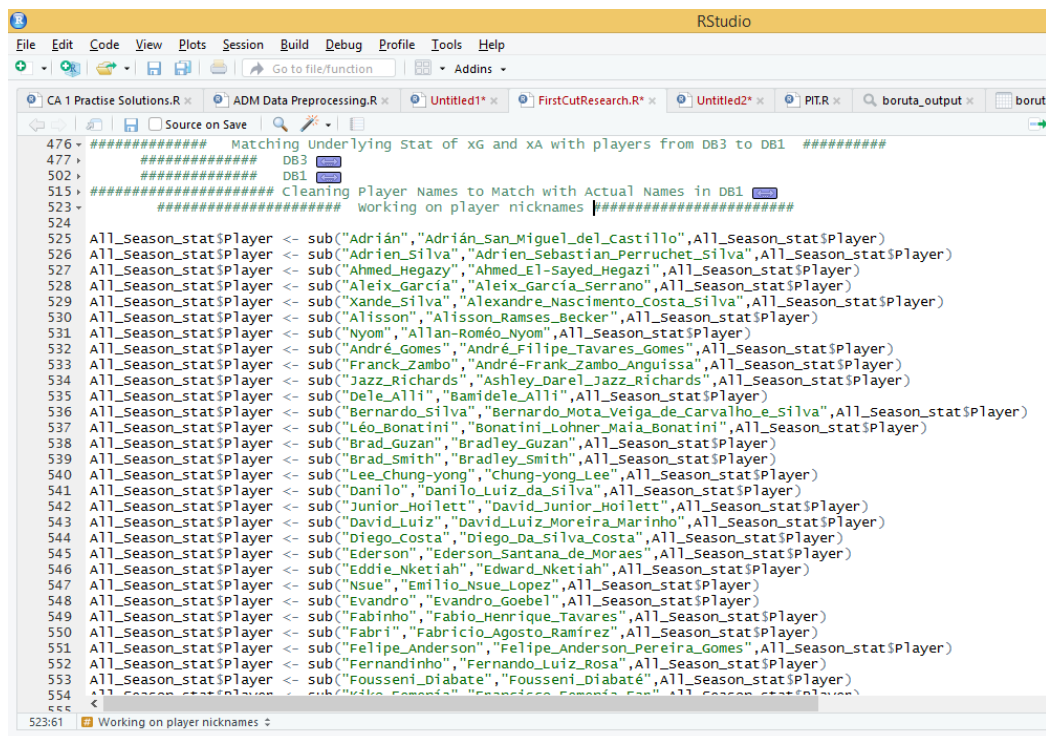
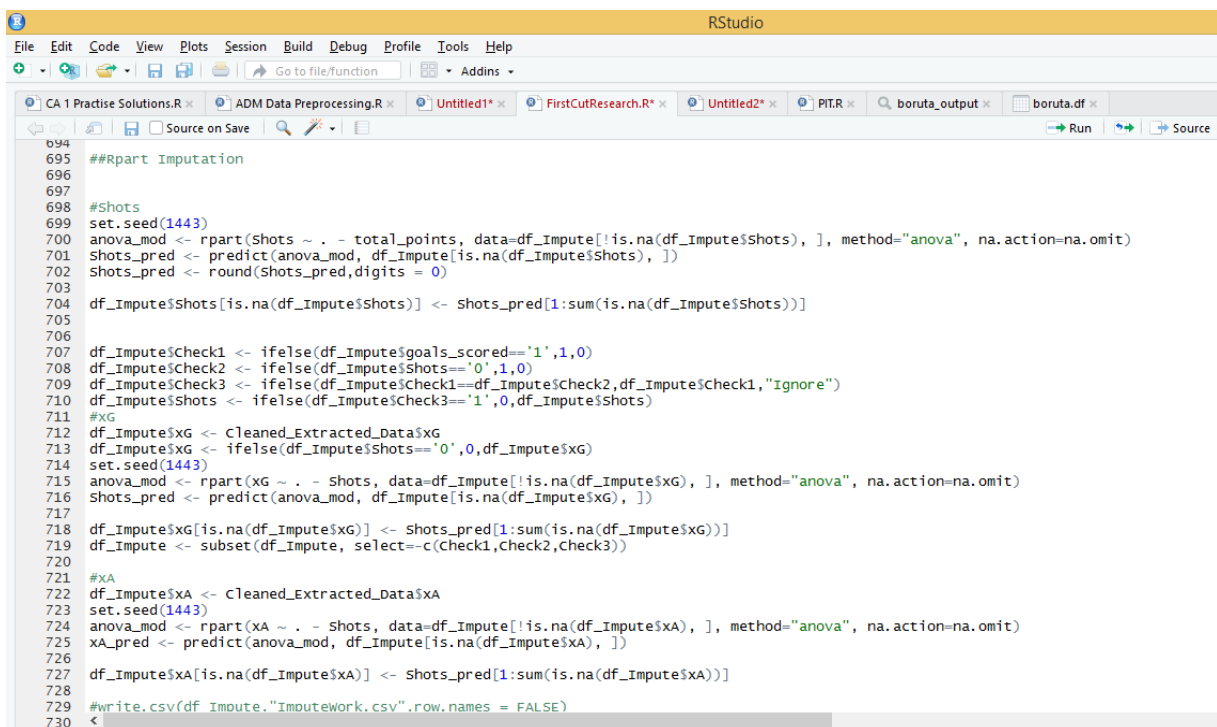


Figure 7: Player Name mismatch handling in RStudio

This was followed by data cleaning where duplicate rows and unwanted columns were removed. Player value did not have a decimal value and was hence divided by 10. Also, upon merging 2 columns highlighting key passes shared the same data but did not match in values. This was handled by assigning a new column which held the maximum of the two on comparison. A new parameter called player form was introduced as it was not present in the dataset. As per FPL, the player form is the average of the points earned in the last 30 days (roughly 4 gameweeks). Missing values in xG, xA and shots were observed after merging datasets. This was handled by assigning 0 value to players who did not play a game on that day. As xG and xA are sum of the probabilities of shots being converted to goals and key passes being possible assists respectively. A value of 1 was placed to any remaining player with missing data in shots if he had scored a goal. The remaining values were then imputed by prediction using the ANOVA method and the rpart function which can be observed in Figure 8.

## 2.2 Data Transformation and Feature Selection



```

694
695 ##Rpart Imputation
696
697
698 #Shots
699 set.seed(1443)
700 anova_mod <- rpart(Shots ~ . - total_points, data=df_Impute[!is.na(df_Impute$Shots), ], method="anova", na.action=na.omit)
701 Shots_pred <- predict(anova_mod, df_Impute[is.na(df_Impute$Shots), ])
702 Shots_pred <- round(Shots_pred,digits = 0)
703
704 df_Impute$Shots[is.na(df_Impute$Shots)] <- Shots_pred[1:sum(is.na(df_Impute$Shots))]
705
706
707 df_Impute$check1 <- ifelse(df_Impute$goals_scored=='1',1,0)
708 df_Impute$check2 <- ifelse(df_Impute$Shots=='0',1,0)
709 df_Impute$check3 <- ifelse(df_Impute$check1==df_Impute$check2,df_Impute$check1,"Ignore")
710 df_Impute$Shots <- ifelse(df_Impute$check3=='1',0,df_Impute$Shots)
711 #xG
712 df_Impute$xG <- Cleaned_Extracted_Data$xG
713 df_Impute$xG <- ifelse(df_Impute$Shots=='0',0,df_Impute$xG)
714 set.seed(1443)
715 anova_mod <- rpart(xG ~ . - Shots, data=df_Impute[!is.na(df_Impute$xG), ], method="anova", na.action=na.omit)
716 Shots_pred <- predict(anova_mod, df_Impute[is.na(df_Impute$xG), ])
717
718 df_Impute$xG[is.na(df_Impute$xG)] <- Shots_pred[1:sum(is.na(df_Impute$xG))]
719 df_Impute <- subset(df_Impute, select=c(check1,check2,check3))
720
721 #xA
722 df_Impute$xA <- Cleaned_Extracted_Data$xA
723 set.seed(1443)
724 anova_mod <- rpart(xA ~ . - Shots, data=df_Impute[!is.na(df_Impute$xA), ], method="anova", na.action=na.omit)
725 xA_pred <- predict(anova_mod, df_Impute[is.na(df_Impute$xA), ])
726
727 df_Impute$xA[is.na(df_Impute$xA)] <- Shots_pred[1:sum(is.na(df_Impute$xA))]
728
729 #write.csv(df_Impute,"ImputeWork.csv",row.names = FALSE)
730 <

```

Figure 8: Handling missing values

Once the missing data had been taken care of, we worked on utilizing the in-game statistics in our dataset. As we cannot use these values to predict that particular entry, we make a summation of all its previous values and shift it to the next week entry where it acts as historical data. This transformation can be observed in Figure 9.



```

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function Addins
CA 1 Practise Solutions.R x ADM Data Preprocessing.R x Untitled1* x FirstCutResearch.R x Untitled2* x PIT.R x boruta_output x
Source on Save
907 df2$xA.lag1 = ifelse(df2$id == 1, 0, df2$xA.lag1)
908 df2$Shots.lag1 = ifelse(df2$id == 1, 0, df2$Shots.lag1)
909
910 df2 = df2 %>%
911   group_by(Player, Team, SeasonNo) %>%
912   mutate(Assist.sum = cumsum(assist.lag1)) %>%
913   mutate(goals_conceded.sum = cumsum(goals_conceded.lag1)) %>%
914   mutate(bonus.sum = cumsum(bonus.lag1)) %>%
915   mutate(Key_Passes.sum = cumsum(Key_Passes.lag1)) %>%
916   mutate(clean_sheets.sum = cumsum(clean_sheets.lag1)) %>%
917   mutate(own_goals.sum = cumsum(own_goals.lag1)) %>%
918   mutate(yellow_cards.sum = cumsum(yellow_cards.lag1)) %>%
919   mutate(red_cards.sum = cumsum(red_cards.lag1)) %>%
920   mutate(winning_goals.sum = cumsum(winning_goals.lag1)) %>%
921   mutate(attempted_passes.sum = cumsum(attempted_passes.lag1)) %>%
922   mutate(big_chances_created.sum = cumsum(big_chances_created.lag1)) %>%
923   mutate(big_chances_missed.sum = cumsum(big_chances_missed.lag1)) %>%
924   mutate(clearances_blocks_interceptions.sum = cumsum(clearances_blocks_interceptions.lag1)) %>%
925   mutate(completed_passes.sum = cumsum(completed_passes.lag1)) %>%
926   mutate(dribbles.sum = cumsum(dribbles.lag1)) %>%
927   mutate(errors_leading_to_goal.sum = cumsum(errors_leading_to_goal.lag1)) %>%
928   mutate(errors_leading_to_goal_attempt.sum = cumsum(errors_leading_to_goal_attempt.lag1)) %>%
929   mutate(fouls.sum = cumsum(fouls.lag1)) %>%
930   mutate(offside.sum = cumsum(offside.lag1)) %>%
931   mutate(open_play_crosses.sum = cumsum(open_play_crosses.lag1)) %>%
932   mutate(penalties_conceded.sum = cumsum(penalties_conceded.lag1)) %>%
933   mutate(penalties_missed.sum = cumsum(penalties_missed.lag1)) %>%
934   mutate(penalties_saved.sum = cumsum(penalties_saved.lag1)) %>%
935   mutate(recoveries.sum = cumsum(recoveries.lag1)) %>%
936   mutate(saves.sum = cumsum(saves.lag1)) %>%
937   mutate(tackled.sum = cumsum(tackled.lag1)) %>%
938   mutate(tackles.sum = cumsum(tackles.lag1)) %>%
939   mutate(target_missed.sum = cumsum(target_missed.lag1)) %>%
940   mutate(goals_scored.sum = cumsum(goals_scored.lag1)) %>%
941   mutate(Shots.sum = cumsum(Shots.lag1))
942
943

```

**Figure 9: Summation of in-game statistics used as historic data**

The dataset had been completely processed and ready to use for modelling, but we had to ensure we did appropriate feature selection and hence applied Boruta algorithm on around 50 attributes which rejected only “penalties missed” and accepted all the other parameters. Below is the plot of the analysis. As we can see in Figure 11, the underlying statistics of FPL i.e, Influence, Threat, Creativity and ICT index are all important parameters. While xG and xA have a relatively low importance, we hold on to them and remove the 9 least important variables including the rejected variable which can be seen in figure 10.

```

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function Addins
CA 1 Practise Solutions.R x ADM Data Preprocessing.R x Untitled1* x FirstCutResearch.R x Untitled2* x PIT.R x boruta_output x
Source on Save
1065 lz <- lapply(1:ncol(boruta_output[["ImpHistory"]]), function(i)
1066   boruta_output[["ImpHistory"]][is.finite(boruta_output[["ImpHistory"]][i,1]),i])
1067
1068 names(lz) <- colnames(boruta_output[["ImpHistory"]])
1069 labels <- sort(sapply(lz, median))
1070 axis(side = 1, las=2, labels = names(labels),
1071       at = 1:ncol(boruta_output[["ImpHistory"]]), cex.axis = 0.7)
1072
1073 boruta.df <- attStats(boruta_output)
1074
1075
1076
1077 ##Removing penalties missed as suggested by Boruta
1078 df5 <- subset(df5, select = -(penalties_missed.sum))
1079
1080 df5 <- read.csv("df5.csv", stringsAsFactors = T)
1081
1082 df5$xG
1083 ##Removing Least important attributes as per Boruta
1084 df5 <- read.csv("df5.csv", stringsAsFactors = T)
1085 |
1086 df5 <- subset(df5, select = -(red_cards.sum))
1087 df5 <- subset(df5, select = -(SeasonNo))
1088 df5 <- subset(df5, select = -(penalties_saved.sum))
1089 df5 <- subset(df5, select = -(penalties_conceded.sum))
1090 df5 <- subset(df5, select = -(own_goals.sum))
1091 df5 <- subset(df5, select = -(winning_goals.sum))
1092 df5 <- subset(df5, select = -(errors_leading_to_goal.sum))
1093 df5 <- subset(df5, select = -(errors_leading_to_goal_attempt.sum))
1094
1095 - #####Building Training Models#####
1096 ##Training and Testing
1097 set.seed(1443)
1098 index <- createDataPartition(df5$status, p = 0.7, list = FALSE)
1099 train_data <- df5[index, ]
1100 <
1085:1 Feature Selection
Console

```

**Figure 10: Removal of attributes based on Boruta Algorithm**

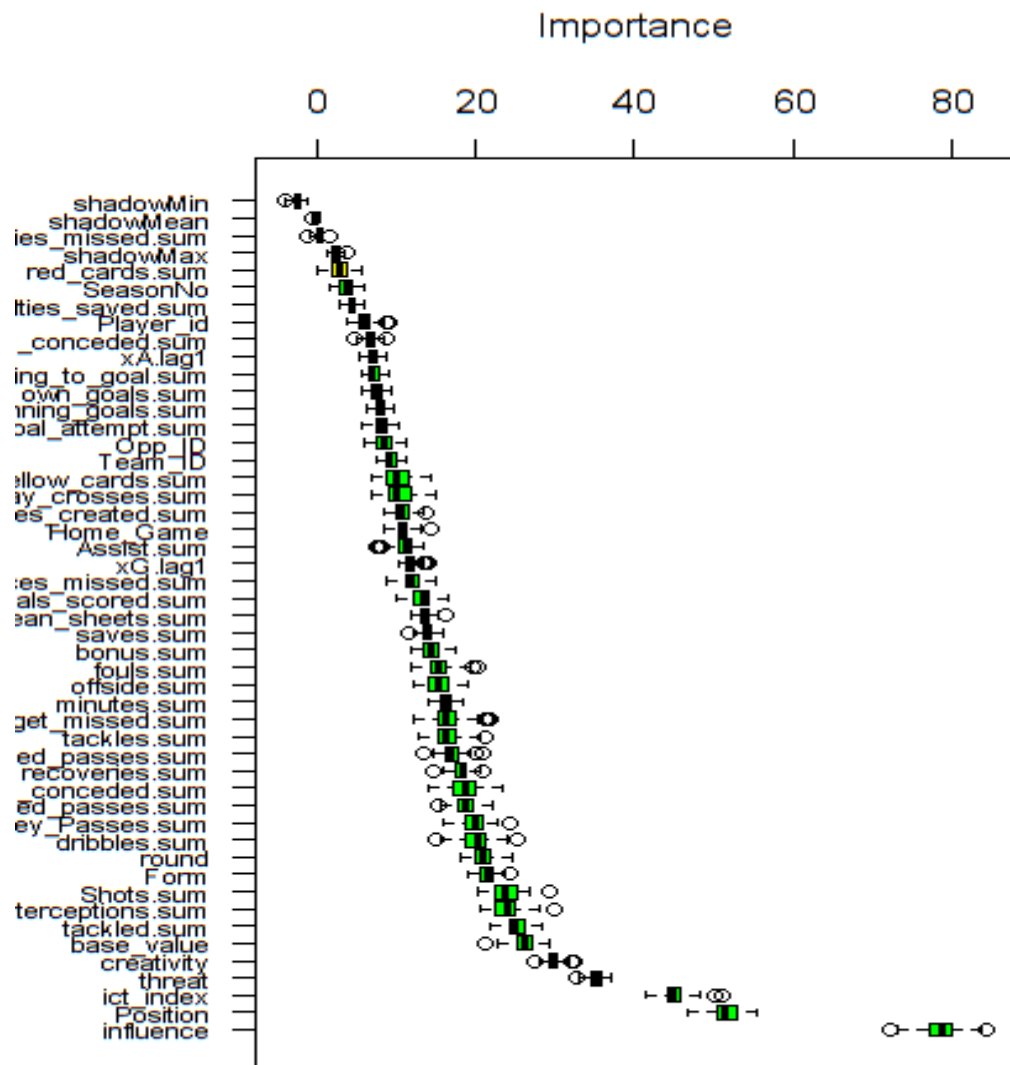


Figure 11: Boruta Algorithm plot

### 3 Modelling

We have applied 2 cases of Random Forest and XGBoost with and without underlying stats. Also 4 different types of sampling have been done to handle class imbalance. Validation of training data sets have been done by k-fold validation.

## Sampling

```
#####Building Training Models#####
1095  ###Training and Testing
1096  set.seed(1443)
1097  index <- createDataPartition(df5$Status, p = 0.7, list = FALSE)
1098  train_data <- df5[index, ]
1099  test_data <- df5[-index, ]
1100  df6 <- df5

df6 <- subset(df6,select= -(xG))
df6 <- subset(df6,select= -(xA))

set.seed(1443)
index <- createDataPartition(df5$Status, p = 0.7, list = FALSE)
train_data <- df6[index, ]
test_data <- df6[-index, ]

###Handling Class Imbalance

#over sampling
data_balanced_over <- ovun.sample(Status ~ ., data = train_data, method = "over",N = 60940)$data ##N = no. of rows
table(data_balanced_over$Status)

#Under sampling
data_balanced_under <- ovun.sample(Status ~ ., data = train_data, method = "under", N = 9922, seed = 1443)$data
table(data_balanced_under$Status)

#Combination of undersampling and oversampling
data_balanced_both <- ovun.sample(Status ~ ., data = train_data, method = "both", p=0.5,
table(data_balanced_both$Status)

##Using Rose function for handling class imbalance
data_balanced_rose <- ROSE(Status ~ ., data = train_data, seed = 1)$data
table(data.rose$Status)
```

## Random Forest

```
#####RANDOM FOREST #####

###Creating Training and Testing Data

set.seed(1443)
index <- createDataPartition(df5$Status, p = 0.7, list = FALSE)
train_data <- df5[index, ]
test_data <- df5[-index, ]

table(train_data$Status) ##Clear case of Class Imbalance

###Handling Class Imbalance

#over sampling
data_balanced_over <- ovun.sample(Status ~ ., data = train_data, method = "over",N = 60886)$data ##N = no. of rows i
table(data_balanced_over$Status)

#Under Sampling
data_balanced_under <- ovun.sample(Status ~ ., data = train_data, method = "under", N = 9976, seed = 1443)$data
table(data_balanced_under$Status)

#Combination of undersampling and oversampling
data_balanced_both <- ovun.sample(Status ~ ., data = train_data, method = "both", p=0.5,
table(data_balanced_both$Status)

##Using Rose function for handling class imbalance
data.rose <- ROSE(Status ~ ., data = train_data, seed = 1)$data
table(data.rose$Status)

####Cross Validation
```

```

####Cross validation

set.seed(1443)
inTrain_over = createDataPartition(data_balanced_over$Status, p = 0.05, list = F)
inTrain_under = createDataPartition(data_balanced_under$Status, p = 0.05, list = F)
inTrain_both = createDataPartition(data_balanced_both$Status, p = 0.05, list = F)
inTrain_rose = createDataPartition(data.rose$Status, p = 0.05, list = F)

crossv = train_data[-inTrain_over, ]
crossv = train_data[-inTrain_under, ]
crossv = train_data[-inTrain_both, ]
crossv = train_data[-inTrain_rose, ]

training2over = data_balanced_over[inTrain_over, ]
training2under = data_balanced_under[inTrain_under, ]
training2both = data_balanced_both[inTrain_both, ]
training2rose = data.rose[inTrain_rose, ]

####Training validated Models
modover = suppressMessages(
  train(Status ~ ., method = "rf", data = training2over,
        trControl = trainControl(method = "cv"), number = 25)
)
modover$finalModel

modunder = suppressMessages(
  train(Status ~ ., method = "rf", data = training2under,
        trControl = trainControl(method = "cv"), number = 25)
)
modunder$finalModel

modboth = suppressMessages(
<
####Training validated Models
modover = suppressMessages(
  train(Status ~ ., method = "rf", data = training2over,
        trControl = trainControl(method = "cv"), number = 25)
)
modover$finalModel

modunder = suppressMessages(
  train(Status ~ ., method = "rf", data = training2under,
        trControl = trainControl(method = "cv"), number = 25)
)
modunder$finalModel

modboth = suppressMessages(
  train(Status ~ ., method = "rf", data = training2both,
        trControl = trainControl(method = "cv"), number = 25)
)
modboth$finalModel

modrose = suppressMessages(
  train(Status ~ ., method = "rf", data = training2rose,
        trControl = trainControl(method = "cv"), number = 25)
)
modrose$finalModel

```

# XGBoost

```
#####XGBOOST#####  
new_train_over <- model.matrix(~ . + 0, data = data_balanced_over[, 1:37])  ###Conversion to Matrix required for XGB  
new_train_under <- model.matrix(~ . + 0, data = data_balanced_under[, 1:37])  ###Conversion to Matrix required for XGB  
new_train_both <- model.matrix(~ . + 0, data = data_balanced_both[, 1:37])  ###Conversion to Matrix required for XGB  
new_train_rose <- model.matrix(~ . + 0, data = data.rose[, 1:37])  ###Conversion to Matrix required for XGBoost  
  
new_test <- model.matrix(~ . + 0, data = test_data[, 1:37])  
  
xgb_train_over <- xgb.DMatrix(data = new_train_over, label = data_balanced_over$Status)  ###Preparing Matrices  
xgb_train_under <- xgb.DMatrix(data = new_train_under, label = data_balanced_under$Status)  ###Preparing Matrices  
xgb_train_both <- xgb.DMatrix(data = new_train_both, label = data_balanced_both$Status)  ###Preparing Matrices  
xgb_train_rose <- xgb.DMatrix(data = new_train_rose, label = data.rose$Status)  ###Preparing Matrices  
xgb_test <- xgb.DMatrix(data = new_test, label = test_data$Status)  
  
# Set parameters(default)  
params <- list(booster = "gbtree", objective = "multi:softprob", num_class = 6, eval_metric = "mlogloss")  
  
# Applying 10 folds for cross-validation  
xgbcv_over <- xgb.cv(params = params, data = xgb_train_over, nrounds = 100, nfold = 10, showsd = TRUE,  
  stratified = TRUE, print_every_n = 10, early_stop_round = 20, maximize = FALSE, prediction = TRUE)  
xgbcv_under <- xgb.cv(params = params, data = xgb_train_under, nrounds = 100, nfold = 10, showsd = TRUE,  
  stratified = TRUE, print_every_n = 10, early_stop_round = 20, maximize = FALSE, prediction = TRUE)  
xgbcv_both <- xgb.cv(params = params, data = xgb_train_both, nrounds = 100, nfold = 10, showsd = TRUE,  
  stratified = TRUE, print_every_n = 10, early_stop_round = 20, maximize = FALSE, prediction = TRUE)  
xgbcv_rose <- xgb.cv(params = params, data = xgb_train_rose, nrounds = 100, nfold = 10, showsd = TRUE,  
  stratified = TRUE, print_every_n = 10, early_stop_round = 20, maximize = FALSE, prediction = TRUE)  
  
##Prediction and Confusion Matrix of Oversampling Training  
OOFP_prediction_over <- data.frame(xgbcv_over$pred) %>%  
  mutate(max_prob = max.col(., ties.method = "last"),  
    label = data_balanced_over$Status + 1)
```

```
##Prediction and Confusion Matrix of Oversampling Training  
OOFP_prediction_over <- data.frame(xgbcv_over$pred) %>%  
  mutate(max_prob = max.col(., ties.method = "last"),  
    label = data_balanced_over$Status + 1)  
  
confusionMatrix(factor(OOFP_prediction_over$max_prob),  
  factor(OOFP_prediction_over$label),  
  mode = "everything")  
  
##Prediction and Confusion Matrix of Undersampling Training  
OOFP_prediction_under <- data.frame(xgbcv_under$pred) %>%  
  mutate(max_prob = max.col(., ties.method = "last"),  
    label = data_balanced_under$Status + 1)  
  
confusionMatrix(factor(OOFP_prediction_under$max_prob),  
  factor(OOFP_prediction_under$label),  
  mode = "everything")  
  
##Prediction and Confusion Matrix of Both Training  
OOFP_prediction_both <- data.frame(xgbcv_both$pred) %>%  
  mutate(max_prob = max.col(., ties.method = "last"),  
    label = data_balanced_both$Status + 1)  
  
confusionMatrix(factor(OOFP_prediction_both$max_prob),  
  factor(OOFP_prediction_both$label),  
  mode = "everything")  
  
##Prediction and Confusion Matrix of Rose Training  
OOFP_prediction_rose <- data.frame(xgbcv_rose$pred) %>%  
  mutate(max_prob = max.col(., ties.method = "last"),  
    label = data.rose$Status + 1)  
  
confusionMatrix(factor(OOFP_prediction_rose$max_prob),  
  factor(OOFP_prediction_rose$label)).
```