Predicting Rookie Running Back Grades

By Derrick Edwards

I. Introduction

With the immense rise in passing statistics, altering rules to favor offensive production, and ever increasing player contracts, the NFL running back has become the scapegoat for cutting costs. The average shelf-life for quality running backs (RB's) has been dwindling since the turn of the century as finding a second contract has become a goal instead of a guarantee. Therefore, it is imperative for NFL teams to find talent within the draft to acquire RB's for smaller monetary contracts who are ready to impact the game immediately. Using college stats, we will predict incoming rookie running back PFF player grades for their first season. The goal is to identify cost-effective running backs for NFL teams to acquire in the NFL draft.

II. Background

PFF's player grade is a statistics developed by Pro Football Focus as a method to compare and rank players quickly and easily. PFF player grades are calculated by assigning a value ranging from (-2, 2) to a player on selected plays during the game where 0 is the expected quality of a play (completing their assignment for a snap), -2 is a catastrophic event attributed to the player in question and 2 represents a clutch game breaking moment. We will use a players college's production for their last year of college to predict their first year's PFF player grade.

III. Data Description

The data used for this project was collected from sports-reference.com to collect the players college statistics and NFL draft history, and from pff.com to collect PFF player grades. We collected data ranging from 2010 through 2022 and data from the 2023 incoming draft class for predictions. Sports-Reference offers a library of statistics for nearly every sport and PFF is a cutting edge statistics company set out to develop new statistics centering around American Football.

The multiple data sources were cleaned and concatenated in R, exported to excel and csv files, and finally uploaded to Python and SAS. The predictors include a players Rush Yards, Attempts, Rushing Touchdowns, Receptions, Receiving Yards, Receiving Touchdowns, School and Conference. All predictors were used to predict a players first year of PFF player grades.

IV. Results

The accuracy results for the 6 models revealed that the Random Forest Regression model outperformed the XG Boost Regression model when performing analysis in R and Python. However, the SAS XG Boost Model reported slightly higher scores compared to the SAS Random Forest Model which may be in part to the seed set in each coding language respectively. Python produced the highest performing model utilizing Random Forest Regression.

When analyzing the future predictions for the 2023 draft class, the respective models produced varying PFF player grades suggesting that there is a key predictor not accounted for in this study. For future studies in this area, lagged variables describing a college players entire collegiate career, as well as measurable athletic traits should vastly improve the model. Specifically, the main area for concern in this model lies within small sample sizes and artificially inflated efficiency. Collegiate players with less than 100 carries in a season with high touchdown production received the highest predicted PFF player grades.

V. Conclusion

The Python produced Random Forest Model proved to be the best predictor of an NFL running back's first year PFF player grade relying on collegiate production as predictors. The aforementioned model produced an 88.57% accuracy score at .2 level of significance, while producing the most tempered predictions. There is a lot of room for improvement in this study, even when achieving stronger accuracy scores.

VI. Appendix

A. Random Forest Regression

a. R Code

```
# merge datasets by RB name and year

college_stats <- read.csv("C:/Users/saedw/OneDrive/Desktop/STAT 574 Data Mining/Final Project - RB WAR/college_football_stats.csv")

nfl_draft <- read.csv("C:/Users/saedw/OneDrive/Desktop/STAT 574 Data Mining/Final Project - RB WAR/nfl_draft.csv")


pff_grade <- read.csv("C:/Users/saedw/OneDrive/Desktop/STAT 574 Data Mining/Final Project - RB WAR/nfl_player_grades.csv")

# data cleaning ################################################
## merge datasets together - predictors - college stats #######

library(dplyr)


# future predictions
pred_data <- subset(college_stats, Year==2023)

write.csv(pred_data,"C:/Users/saedw/OneDrive/Desktop/STAT 574 Data Mining/Final Project - RB WAR/RB_pred_data.csv", row.names = FALSE)

# subset nfl_draft and pff_grade to only needed stats
draft_sub <- subset(nfl_draft, select = c(Player, Year, Pos))

pff_sub <- subset(pff_grade, select = c(Player, Year, grades_offense))


# merge college stats and nfl draft - dropping non-matches
drafted_college_stats <- merge(college_stats, draft_sub,
                        by.x = c("Player", "Year"), by.y=c("Player", "Year"),
                        all = FALSE)

# merge drafted_college_stats and pff_grades
yearly_stats <- merge(drafted_college_stats, pff_sub,
                by.x = c("Player", "Year"), by.y = c("Player", "Year"),
                all = FALSE)
```

```
concat_data <- subset(yearly_stats, Pos=="RB")

# convert NA to 0
concat_data[is.na(concat_data)] <- 0


# subset to needed data
final_data <- subset(concat_data, select = c(School, Conf, G, Att, Rsh_Yds,
                                     Rsh_Avg, Rsh_TD, Rec, Rec_Yds, Rec_Avg,
                                     Rec_TD, Plays, grades_offense))


write.csv(final_data,"C:/Users/saedw/OneDrive/Desktop/STAT 574 Data Mining/Final Project - RB
WAR/project_data_og.csv", row.names = FALSE)

### Random Forest Model ##########
library(randomForest)


#SPLITTING DATA INTO 80% TRAINING AND 20% TESTING SETS
set.seed(109283)
sample <- sample(c(TRUE, FALSE), nrow(final_data),
             replace = TRUE, prob = c(0.8, 0.2))
train <- final_data[sample,]
test <- final_data[!sample,]


# build random forest regression model - grades_offense
randfor <- randomForest(grades_offense ~ School + Conf + G + Att + Rsh_Yds +
                 Rsh_Avg + Rsh_TD + Rec + Rec_Yds + Rec_Avg + Rec_TD +
                 Plays, data=train, ntree=150,
                mtry=5, maxnodes=30)


# display variable importance #######
print(importance(randfor,type=2))

# computing prediction accuracy for testing data
p_grades_offense <- predict(randfor, newdata = test)

# accuracy 10,15, 20 store true false values - compute means for accuracy scores

# accuracy within 10%
accuracy10 <- ifelse(abs(test$grades_offense - p_grades_offense)
```

< 0.10*test$grades_offense,1,0)

# accuracy within 15%
accuracy15 <- ifelse(abs(test$grades_offense - p_grades_offense)
< 0.15*test$grades_offense,1,0)
# accuracy within 20%
accuracy20 <- ifelse(abs(test$grades_offense - p_grades_offense)
< 0.20*test$grades_offense,1,0)

# print means of accuracy scores
print("Accuracy Scores - Random Forest")
print(mean(accuracy10))
print(mean(accuracy15))
print(mean(accuracy20))

b. R - Output

```
          IncNodePurity
School        1272.7106
Conf           792.5722
G              523.4567
Att            974.0574
Rsh_Yds       1512.7737
Rsh_Avg       1399.2879
Rsh_TD        1661.0861
Rec           1044.7728
Rec_Yds       1559.1485
Rec_Avg       1291.7112
Rec_TD         831.5700
Plays          960.2410
[1] "Accuracy Scores - Random Forest"
[1] 0.516129
[1] 0.6774194
[1] 0.8387097
```

c. Python Code

```
B. # final project STAT 574
C.
D. import pandas
E. from sklearn.ensemble import RandomForestRegressor
F. from sklearn.model_selection import train_test_split
G.
H. # dataset
I. nfl_data=pandas.read_csv(r'C:/Users/saedw/OneDrive/Desktop/STAT 574 Data
   Mining/Final Project - RB WAR/project_data.csv')
J.
K. nfl_data=nfl_data.drop('School', axis=1)
L. nfl_data=nfl_data.drop('Conf', axis=1)
M.
N. # Random Forest Regression
O. X=nfl_data.iloc[:,0:12].values
```

```
y=nfl_data.iloc[:,12].values

#SPLITTING DATA INTO 80% TRAINING AND 20% TESTING SETS
X_train, X_test, y_train, y_test=train_test_split(X, y, test_size=0.20,
random_state=348644)

#FITTING RANDOM FOREST REGRESSION TREE
rf_reg=RandomForestRegressor(n_estimators=100, random_state=323445,
max_depth=50, max_features=4)
rf_reg.fit(X_train, y_train)

#DISPLAYING VARIABLE IMPORTANCE
from sklearn.ensemble import ExtraTreesClassifier

var_names=pandas.DataFrame(['School_Count','Conf_Count','G','Att','Rsh_Yds
', 'Rsh_Avg','Rsh_TD','Rec','Rec_Yds','Rec_Avg','Rec_TD'
                            ,'Plays'], columns=['var_name'])
loss_reduction=pandas.DataFrame(rf_reg.feature_importances_,
columns=['loss_reduction'])
var_importance=pandas.concat([var_names, loss_reduction], axis=1)
var_importance=var_importance.sort_values("loss_reduction", axis=0,
ascending=False)
print(var_importance)

#COMPUTING PREDICTION ACCURACY FOR TESTING DATA
y_pred=rf_reg.predict(X_test)

ind10=[]
ind15=[]
ind20=[]

for sub1, sub2 in zip(y_pred, y_test):
    ind10.append(1) if abs(sub1-sub2)<0.10*sub2 else ind10.append(0)
    ind15.append(1) if abs(sub1-sub2)<0.15*sub2 else ind15.append(0)
    ind20.append(1) if abs(sub1-sub2)<0.20*sub2 else ind20.append(0)

#accuracy within 10%
accuracy10=sum(ind10)/len(ind10)
print(accuracy10)

#accuracy within 15%
accuracy15=sum(ind15)/len(ind15)
print(accuracy15)

#accuracy within 20%
```

```
FFF.      accuracy20=sum(ind20)/len(ind20)
GGG.      print(accuracy20)
HHH.
```

d. Python Output

```
        var_name  loss_reduction
4        Rsh_Yds        0.109390
6         Rsh_TD        0.107606
8        Rec_Yds        0.105852
9        Rec_Avg        0.100319
5        Rsh_Avg        0.098297
11         Plays        0.080639
0   School_Count        0.080532
3            Att        0.079207
7            Rec        0.075545
10        Rec_TD        0.063123
1     Conf_Count        0.051894
2              G        0.047595
0.6285714285714286
0.7142857142857143
0.8857142857142857
```

e. SAS Code

```
proc import out=nfl_data
file="\\vdi-fileshare01\UEMprofiles\017365554\Desktop\STAT 574\STAT 574
Final\project_data_og.csv"
dbms=csv replace;
run;

proc print data=nfl_data;
run;

/*SPLITTING DATA INTO 80% TRAINING AND 20% TESTING*/
proc surveyselect data=nfl_data rate=0.8 seed=502305
out=nfl_new outall method=srs;
run;

/* random forest regression model */
proc hpforest data=nfl_new seed=109283
maxtrees=60 vars_to_try=4 trainfraction=0.7
maxdepth=50;
target grades_offense/level=interval;
input School Conf/level=nominal;
input G Att Rsh_Yds Rsh_Avg Rsh_TD Rec Rec_Yds Rec_Avg Rec_TD
Plays/level=interval;
partition rolevar=selected(train='1');
save file='\\vdi-fileshare01\UEMprofiles\017365554\Desktop\STAT
574\random_forest_final.bin';
run;
```

```
/*COMPUTING PREDICTED VALUES FOR TESTING DATA*/
data test;
set nfl_new;
if(selected='0');
run;


proc hp4score data=test;
id grades_offense;
score file='\\vdi-fileshare01\UEMprofiles\017365554\Desktop\STAT
574\random_forest_final.bin'
out=predicted;
run;

/*DETERMINING 10%, 15%, AND 20% ACCURACY*/
data accuracy;
set predicted;
if(abs(grades_offense-P_grades_offense)
<0.10*grades_offense)
then ind10=1; else ind10=0;
if(abs(grades_offense-P_grades_offense)
<0.15*grades_offense)
then ind15=1; else ind15=0;
if(abs(grades_offense-P_grades_offense)
<0.20*grades_offense)
then ind20=1; else ind20=0;
run;


proc sql;
 select sum(ind10)/count(*) as accuracy10,
sum(ind15)/count(*) as accuracy15,
 sum(ind20)/count(*) as accuracy20
 from accuracy;
 quit;
```

f. SAS Output

accuracy10	accuracy15	accuracy20
0.470588	0.558824	0.764706

B. XG Boost Regression

a. R Code

```
# XGBoost Regression Model ########

library(xgboost)


xg_data <- subset(final_data, select = c(School, Conf, G, Att, Rsh_Yds,
                                   Rsh_Avg, Rsh_TD, Rec, Rec_Yds, Rec_Avg,
                                   Rec_TD, Plays, grades_offense))

#SPLITTING DATA INTO 80% TRAINING AND 20% TESTING SETS
```

```
set.seed(109283)
sample <- sample(c(TRUE, FALSE), nrow(xg_data),
          replace = TRUE, prob = c(0.8, 0.2))
train <- xg_data[sample,]
test <- xg_data[!sample,]


# numerical value is dependent variable
train.x<- data.matrix(train[-13])
train.y<- data.matrix(train[13])
test.x<- data.matrix(test[-13])
test.y<- data.matrix(test[13])

# fit extreme gradient boosted regression tree
xgb_reg <- xgboost(data = train.x, label = train.y, max.depth=6, eta=0.01,
           subsample=0.8, colsample_bytree=0.5, nrounds=1000,
           objective="reg:linear")


# display feature importance
print(xgb.importance(colnames(train.x), model = xgb_reg))

# compute prediction accuracy for testing data
pred.y <- as.data.frame(predict(xgb_reg, test.x))


# accuracy scores
# 10%
accuracy10 <- ifelse(abs(test.y-pred.y) < 0.10*test.y,1,0)
# 15%
accuracy15 <- ifelse(abs(test.y-pred.y) < 0.15*test.y,1,0)
# 20%
accuracy20 <- ifelse(abs(test.y-pred.y) < 0.20*test.y,1,0)


# print accuracy scores
print(mean(accuracy10))
print(mean(accuracy15))
print(mean(accuracy20))
```

b. R Output

Feature	**Gain**	**Cover**	**Frequency**
<chr>	<dbl>	<dbl>	<dbl>
Rsh_Yds	0.12880323	0.12945271	0.10582604

Rsh_Avg	0.11929768	0.10772122	0.09896934
School	0.11115524	0.10477508	0.15283108
Rec_Avg	0.10655149	0.08594971	0.07615680
Rec_Yds	0.10276505	0.11609675	0.07891673
Rsh_TD	0.09632641	0.10435047	0.07408685
Att	0.07316946	0.07805099	0.11164776
Rec	0.07031693	0.07111633	0.07154254
Conf	0.06645264	0.05496563	0.08258226
Plays	0.05375977	0.06352214	0.05248178
G	0.04057916	0.04375185	0.06309026
Rec_TD	0.03082294	0.04024711	0.03186856

```
> print(mean(accuracy10))
[1] 0.483871
> print(mean(accuracy15))
[1] 0.6129032
> print(mean(accuracy20))
[1] 0.7741935
```

c. Python Codes

```
# xgboost regression

from sklearn.ensemble import GradientBoostingRegressor


#SPLITTING DATA INTO 80% TRAINING AND 20% TESTING SETS
X_train, X_test, y_train, y_test=train_test_split(X, y, test_size=0.20,
random_state=348644)

#FITTING GRADIENT BOOSTED REGRESSION TREE
gbreg_params = {'n_estimators': 1000, 'max_depth': 6, 'learning_rate': 0.01,
'loss': 'squared_error'}
gb_reg=GradientBoostingRegressor(**gbreg_params)
gb_reg.fit(X_train, y_train)

#DISPLAYING VARIABLE IMPORTANCE
var_names=pandas.DataFrame(['School_Count','Conf_Count','G','Att','Rsh_Yds',
'Rsh_Avg','Rsh_TD','Rec','Rec_Yds','Rec_Avg','Rec_TD'
                            ,'Plays'], columns=['var_name'])
loss_reduction=pandas.DataFrame(gb_reg.feature_importances_,
columns=['loss_reduction'])
var_importance=pandas.concat([var_names, loss_reduction], axis=1)
var_importance=var_importance.sort_values("loss_reduction", axis=0,
ascending=False)
print(var_importance)
```

```python
#COMPUTING PREDICTION ACCURACY FOR TESTING DATA
y_pred=gb_reg.predict(X_test)

ind10=[]
ind15=[]
ind20=[]

for sub1, sub2 in zip(y_pred, y_test):
    ind10.append(1) if abs(sub1-sub2)<0.10*sub2 else ind10.append(0)
    ind15.append(1) if abs(sub1-sub2)<0.15*sub2 else ind15.append(0)
    ind20.append(1) if abs(sub1-sub2)<0.20*sub2 else ind20.append(0)

#accuracy within 10%
accuracy10=sum(ind10)/len(ind10)
print(accuracy10)

#accuracy within 15%
accuracy15=sum(ind15)/len(ind15)
print(accuracy15)

#accuracy within 20%
accuracy20=sum(ind20)/len(ind20)
print(accuracy20)
```

d. Python Output

```
        var_name  loss_reduction
4        Rsh_Yds        0.140416
6         Rsh_TD        0.136628
5        Rsh_Avg        0.119844
8        Rec_Yds        0.107342
9        Rec_Avg        0.094537
0   School_Count        0.093963
10        Rec_TD        0.059058
1     Conf_Count        0.058577
11         Plays        0.053608
7            Rec        0.053425
3            Att        0.048302
2              G        0.034299
0.5428571428571428
0.6857142857142857
0.8285714285714286
```

e. SAS Codes

SAS Code

```sas
proc import out=sasuser.nfl_data
```

```
file="\\vdi-fileshare01\UEMprofiles\017365554\Desktop\STAT 574\STAT 574
Final\project_data_og.csv"
dbms=csv replace;
run;


/*Gradient boosted regression model is built
in Enterprise Miner*/



libname hw2q1 "\\vdi-fileshare01\UEMprofiles\017365554\Desktop\STAT 574\STAT
574 Final\Final_XGBoost\Workspaces\EMWS1\emsave";

data accuracy;
set hw2q1.em_save_test;
ind10=(abs(R_grades_offense)<0.10*grades_offense);
ind15=(abs(R_grades_offense)<0.15*grades_offense);
ind20=(abs(R_grades_offense)<0.20*grades_offense);
run;

proc sql;
select sum(ind10)/count(*) as accuracy10,
sum(ind15)/count(*) as accuracy15,
sum(ind20)/count(*) as accuracy20
from accuracy;
quit;
```

f. SAS Output

accuracy10	accuracy15	accuracy20
0.514286	0.628571	0.771429

C. Predictions

a. R Code

```
## Predictions on New Rookie Class ###

# pred_data - 2023 rookie running back class - subset to needed variables

sub_pred <- subset(pred_data, select = c(School, Conf, G, Att, Rsh_Yds,
                                 Rsh_Avg, Rsh_TD, Rec, Rec_Yds, Rec_Avg,
                                 Rec_TD, Plays))

#### random forest predictions ###################################################
################################################################################
pred_2023 <- predict(randfor, newdata = sub_pred)

rb_23_class <- as.data.frame(pred_2023)

#### Merge with player names ####
# create dummy variable for to merge player names
for(i in 1:nrow(rb_23_class)){
```

```
  rb_23_class$row_count[i] <- i
}

# sub pred_data to player names ##
player_pred <- subset(pred_data, select = c(Player))

# create dummy variable for to merge player names
for(i in 1:nrow(player_pred)){
  player_pred$row_count[i] <- i
}

# merge pred with player names by row_count
predictions <- merge(player_pred, rb_23_class, by.x = "row_count",
                by.y = "row_count", all=TRUE)

## XGBoost Predictions ############################################################
##################################################################################
# convert sub_pred to xgb readable matrix
mat_pred <- xgb.DMatrix(as.matrix(sub_pred))

# compute prediction accuracy for testing data
xg_23_pred <- as.data.frame(predict(xgb_reg, mat_pred))

mat_xg_23_pred <- as.data.frame(xg_23_pred)

for(i in 1:nrow(mat_xg_23_pred)){
  mat_xg_23_pred$row_count[i] <- i
}


both_preds <- merge(predictions, mat_xg_23_pred, by.x = "row_count",
                by.y = "row_count", all=TRUE)

both_preds <- both_preds %>% dplyr::rename("Random Forest Prediction"=pred_2023,
                        "XGBoost Prediction"="predict(xgb_reg, mat_pred)")

write.csv(both_preds,"C:/Users/saedw/OneDrive/Desktop/STAT 574 Data Mining/Final Project - RB WAR/r_predictions.csv", row.names = FALSE)
```

b. Python Code

```
# Predictions for 2023 RB Class

# prediction data
```

```python
pred_data=pandas.read_csv(r"C:/Users/saedw/OneDrive/Desktop/STAT 574 Data
Mining/Final Project - RB WAR/RB_pred_data.csv")

player_info=pred_data[['Player']]

# data cleaning - drop unused columns
pred_data=pred_data.drop('Player', axis=1)
pred_data=pred_data.drop('Rk', axis=1)
pred_data=pred_data.drop('Yds', axis=1)
pred_data=pred_data.drop('Avg', axis=1)
pred_data=pred_data.drop('TD', axis=1)
pred_data=pred_data.drop('Year', axis=1)


pred_data.fillna(0, inplace=True)

#compute predictions for new data Random Forest
rf_pred=rf_reg.predict(pred_data)

#compute predictions for new data XGBoost
xg_pred=gb_reg.predict(pred_data)


rf_df=pandas.DataFrame(rf_pred, columns=['Random Forest Prediction'])
xgb_df=pandas.DataFrame(xg_pred, columns=['XGBoost Prediction'])

# create id column for pred df and player info
rf_df["id"] = rf_df.index + 1
xgb_df["id"] = xgb_df.index + 1
player_info["id"] = player_info.index + 1


# merge prediction and player name dataframes
total_pred=pandas.merge(pandas.merge(player_info,rf_df,on='id'),xgb_df,on='id')

total_pred.to_csv (r'C:/Users/saedw/OneDrive/Desktop/STAT 574 Data Mining/Final
Project - RB WAR/pyth_predictions.csv', index = False, header=True)
```

c. R and Python Output

Running Backs	Random Forest – Python	Random Forest – R	XGB – Python	XGB - R

Bijan Robinson	64.10	65.16	65.47	61.25
Jahmyr Gibbs	65.89	65.99	63.63	60.72
Roschon Johnson	64.38	67.68	69.66	67.63
Devon Achane	59.46	57.39	61.77	59.44
Tyjae Spears	66.03	61.56	67.53	71.38
Zach Charbonnet	63.55	60.16	62.76	60.59
Tank Bigsby	58.99	58.12	59.18	53.71
Zach Evans	63.82	66.24	62.24	55.37
Eric Gray	59.65	61.12	62.60	51.89
Kendre Miller	65.56	69.22	65.44	58.33

VII. References

1. “2022 College Football Rushing Stats.” *College Football at Sports-Reference.com*, www.sports-reference.com/cfb/years/2022-rushing.html. Accessed 3 May 2023.

2. “PFF Player Grades.” *PFF*, www.pff.com/grades.

3. “NFL and AFL Draft History | Pro-Football-Reference.com.” *Pro-Football-Reference.com*, 2000, www.pro-football-reference.com/draft/.