DATA698 Final Research Project

Christian Thieme

12/10/2021

## Abstract

The world of fantasy football analytics has been growing in popularity each year. This research attempted to build a model that can predict “breakout” wide receivers in the NFL with 80% accuracy. Random Forest, KNN, XGBoost, and SVM models were built to model this scenario. No models met the above threshold, however, the XGBoost and SVM model proved directional, and further investigation indicates that an ensemble model may provide more accuracy given more computing power. As a pioneer in this field, this research will act as a stepping stone for additional research.

## Introduction

Sports analytics has grown exponentially in the last decade. The success of “Moneyball” type analytics in baseball has led to many other sports attempting to find an edge using analytics. Sports teams around the country, both professional and collegiate, employ full-time analysts to try and squeeze out any ounce of improvement for their teams that they can. Another area in the sports arena that has grown tremendously over the last 10 years are fans involvement in “fantasy” leagues. These leagues allow the individual to “draft” players on to their team and then are given points each week based on the performance of the players in the team. The fantasy football following is the largest of all the fantasy sports. While these leagues may sound silly to some, there are millions of people who play and there is **big** money involved. The NFL’s biggest prize is $1,000,000 (Subramanian, 2014).

More and more people are turning from using their gut, to really diving into the analytics of players and teams (Koss, 2020). In a special kind of league called a dynasty league, participants are allowed to keep their players year after year. That means that a major way to improve your standings each year is to draft the right rookies on to your team as players transition from college to the NFL. One of the positions that can have a huge impact on a dynasty team as a rookie is that of the wide receiver.

Each year, dozens of sports analysts make predictions about who the “break out” wide receivers will be based on having watched their college careers. Often the analysts get the obvious players right, but each year there are also surprises and disappointments. Is there a more quantitative way to determine who these “breakout” players will be?

Answering that question is the purpose of this research project. This project will seek to build a predictive model using a wide receiver’s collegiate and NFL combine statistics to accurately predict 80% of the wide receivers that will average 10 fantasy points per game.

## Literature Review

There is little to no academic research in the way of determining “breakout” players in the NFL as it relates to NFL Fantasy leagues. Because having an accurate predictive model in this arena can be extremely lucrative, those who have managed to build such models have done so privately and either charge money to use it, or make money by using it themselves.

While there are articles that discuss what age a player will be when they “break out”, they are talked about in an extremely high level fashion without describing any specifics as to the data, the methods, or the model. In fact, most articles won’t share more than an r-squared value. Further, when articles do share an r-squared value, it hints that more simple regression type models are being used as opposed to more leading edge, advanced algorithms.

Besides the fact that keeping these models as proprietary property can be “good business”, we can also attribute some of the lack of research to the fact that the NFL has historically had the least amount of data available to the public (Lindsey, 2016). The league has only recently found value in tracking and sharing more data, which has led to much of the analytics explosion within the NFL.

## Research Question

In this study, I will determine if it is possible to build a predictive model using a wide receiver’s collegiate and NFL combine statistics to accurately predict 80% of the wide receivers that will average 10 fantasy points per game.

## Data and Variables

As mentioned previously, there are very few places where you can get extensive historical collegiate data for players who now or did play in the NFL. There are of course popular websites that host basic statistics for each player’s collegiate stats, but these statistics are the basic metrics such as yards, touchdowns, etc. Those who are familiar with fantasy football know that each of these metrics can be broken down into many, many more metrics, and combinations of these metrics can lead to hundreds more.

The data source I will use for this project, is perhaps the only one of its kind. It contains hundreds of variables for each player and is maintained by Dave Wright who is a well-known author at DynastyLeagueFootball.com (Wright, n.d.). He provides this dataset free of charge in an effort to evangelize analytics in fantasy football. It contains player statistics from the 2018-2022 seasons. The dataset also is unique in that it is broken down exclusively into players who are wide receivers. There are 836 rows (players) in the dataset and 602 columns. The dataset contains both numeric and categorical data.

For each player in the dataset, there are several descriptive columns that have critical information for each player and don’t contain missing data. These columns are:

* Player Name
* School - college they attended
* Conference - the athletic conference the school is in
* Date of Birth (DOB)
* Draft Round - the round the player was selected in the NFL draft (there are 7 rounds)
* Draft Pick - what number the player was selected overall in the draft
* Draft Year
* Draft Age - how old the player was when they were drafted
* NCAA Seasons - how many seasons they played in the NCAA

Following these columns, there are three types of numeric columns: yearly statistics from the 2018-2022 seasons, aggregated metrics from these seasons such as average receptions per game or total career touch downs, and metrics from the NFL combine such as height, weight, 40 time, hand size, etc. In total, there are 100 different metrics. Some of these metrics are straight forward and well known, others are rather specific and are derived from calculations of other columns. A full listing of metrics within the dataset are presented below as well as their definition, the type of metric it is (if it is from a season, a count, from the combine, etc.) and how they are calculated:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Term | Abbreviations | Type | Definition | Formula |
| **20 Yard Shuttle** | **Shuttle, Short Shuttle** | Combine | measures an player’s lateral quickness by having the player move laterally 5 yards to touch a line, then 10 yards to the left to touch a line, and finally 5 yards to the right to finish. | - |
| **3 Cone Drill** | **3 Cone** | Combine | measures a player’s ability to change directions as they run around 3 cones placed in an L-shape. | - |
| **40 Time** | **-** | Combine | a sprint measuring 40 yards. Times not recorded at the NFL combine are adjusted by adding .05 seconds to the player time. | - |
| **Adjusted Yards Per Attempt** | **AY/A, AYA** | Metric | created by Bob Carroll, Pete Palmer, and Jim Thorn in The Hidden Game of Football\* to weight the significance of Pass TDs, Pass Yds, and INT when considering Y/PA. \*source Pro Football Reference. | (Pass YDs + 20\*(Pass TD) - 45\*(INT))/(Pass ATTs) |
| **Agility Score** | **-** | Combine | a metric intended to measure a player’s lateral agility and quickness | Shuttle + 3 Cone |
| **Arm Length** | **-** | Combine | the distance from the end of the bicep or shoulder blade to the tip of the middle finger with the arm extended. | - |
| **Bench Press** | **Bench** | Combine | measures the amount of reps a player can lift 225 lbs in the bench press lifting exercise. | - |
| **Body Mass Index** | **BMI** | Combine | an index of an individual’s weight compared to their height to describe someone’s level of thickness. Derived by Belgian Adolphe Quetelet. | “=((weight in pounds/height in inches)\*703)” |
| **Breakout Age** | **BOA** | Metric | a metric created by Frank Dupont and Shawn Siegele where a player’s production eclipses a certain threshold to represent they have become a significant contributor for their college team | “stat/metric>threshold” |
| **Broad Jump** | **Broad** | Combine | measures the player’s ability to jump horizontally from a balanced stance. | - |
| **Career Average** | **-** | - | the college career average of a stat or metric for a player. | “=AVERAGE(cell for age 18 szn:cell for age 23 szn)” |
| **Career Best** | **-** | - | a stat or metric’s “best” value in a particular season. For TDs it would be the highest value and for INTs it would be the lowest value. | “=MAX(cell for age 18 szn:cell for age 23 szn)” |
| **Career Last** | **-** | - | the value of a stat or metric from the last season of a player’s college career. | “=LOOKUP(9999999999,(cell for age 18 szn:cell for age 23 szn)” |
| **Career Total** | **-** | Counting | the total counting stat for a player’s college career | - |
| **College Dominator** | **CD, DOM** | Metric | metric originated by Frank DuPont in his book Game Plan, it represents the player’s percentage of their team’s offense in respect to the team’s yardage and touchdowns. | “=(Career Last + Career Best)/2” |
| **Combine Scrimmage Market Share** | **COMB SCRIM MS** | Metric | a metric designed to represent a player’s total offensive production within their offense. | ((SCRIM YDs/TM SCRIM Yds)+(SCRIM TDs/TM SCRIM TDs)/2) |
| **Combined QB Market Share** | **COMB QB MS** | Metric | a metric designed to represent a QBs production in their team’s passing offense. | ((Pass YDs/TM Pass Yds)/2)+((Pass TDs/TM Pass TDs)/2)) |
| **Combined QB Team %** | **QB COMB TM %** | Metric | the percentage of the team yards that were Pass YDs by a player divided by two added to the percentage of the team TDs that were Pass TDs by a player divided by two. | ((Pass YDs/TM Yds)/2+(Pass TDs/TM TDs)2) |
| **Combined Receiving Market Share** | **Comb REC MS** | Metric | a metric designed to represent a player’s receiving production in their college offense | ((REC YDs/TM REC Yds)/2)+((REC TDs/TM REC TDs)/2)) |
| **Combined Rush Market Share** | **COMB Rush MS** | Metric | a metric designed to represent a QBs production in their team’s rushing offense. | ((Rush YDs/TM Rush Yds)/2)+((Rush TDs/TM Rush TDs)/2)) |
| **Conference** | **Conf** | Bio | the college football conference of the team a player played for in their last year of college football. (Non-CFB means non Division 1 schools) | - |
| **Fantasy Points** | **PPR Points** | Metric |  | (REC)+(scrimmage yards\*.1)+(Total TDs\*6) |
| **Freak Score** | **-** | Combine | an athleticism metric that accounts for the height, weight, and speed of a prospect | - |
| **Games Played** | **GP** | Counting | player makes appearance on the field regardless of position. | - |
| **Hand Size** | **-** | Combine | the distance from the tip of the pinky to the tip of the thumb with the fingers spread out | - |
| **Height** | **-** | Combine | the amount of distance from the bottom of a player’s heals to the top of their head In inches. | - |
| **Interception %** | **INT %** | Metric | Interceptions divided by Pass ATTs | INT/Pass ATTs |
| **Kick Return** | **Kick Returns** | Counting | when a player receives a kickoff from the opposing team and attempts to advance the ball down the field. | - |
| **Kick Return Touchdown** | **KR TDs** | Counting | when a kick return results in a touchdown for the team that receives it. | - |
| **Kick Return Yard** | **KR YDs** | Counting | the yards a player gains after receiving a kickoff from the opposing team | - |
| **Over Average** | **OVR AVG** | Metric | a stat or metric over the average of certain cohort of players at that player’s position. | - |
| **Over Expected** | **OVR EXP** | Metric | a stat or metric over the average of that stat or metric for the player’s in that conference. | - |
| **Over Team Average** | **Over TM AVG** | Metric | a stat or metric compared to the player’s college team performance for the same stat or metric. | stat/metrcic minus the college team’s performance in that stat/metric for the season |
| **Pass Attempt** | **Pass ATT, PA** | Counting | offensive player throws the ball forward when behind line of scrimmage. | - |
| **Pass Attempt % of Team Plays** | **Pass ATT % of TM Plays** | Metric | the percentage of the team plays that were Pass ATTs by a player. | Pass ATT/TM Plays |
| **Pass College Dominator** | **Pass CD** | Metric | a metric that equally weighs the career last and career best seasons in terms of market share of team yards and touchdowns. | “=((Career Last QB COMB TM %+Career Best QB COMB TM %)/2)” |
| **Pass Completion Percentage** | **Comp %** | Metric | Pass Completions divided by pass attempts | Pass Comp/Pass ATTs |
| **Pass Interceptions** | **INT** | Counting | opposing defense catches a pass attempt by the offense. | - |
| **Pass TD %** | **Pass TD %** | Metric | Pass TDs divided by Pass ATTs | Pass TDs/Pass ATTs |
| **Pass TDs % of Team TDs** | **Pass TDs % of Team TDs** | Metric | the percentage of the team TDs that were Pass TDs by a player. | Pass TDs/TM TDs |
| **Pass Touchdowns** | **Pass TDs** | Counting | pass thrown behind line of scrimmage that results in a touchdown for the offense. | - |
| **Pass Yards** | **Pass Yds, Pass YDs** | Counting | yards gained by offense on a completed pass. | - |
| **Pass Yards % of Team Yards** | **Pass Yds % of TM YDs** | Metric | the percentage of the team yards that were Pass YDs by a player. | Pass YDs/TM YDs |
| **NCAA Passing Efficiency** | **-** | Metric | A passing metric designed to measure QB performance. | ((8.4\*Pass Yds)+(330\*Pass TDs)+(100\*Pass Comp)-(200\*INTs))/Pass ATT |
| **NFL Passer Rating** | **-** | Metric | created by Don Smith, Seymour Siwoff, and Don Weiss to provide a standard for comparing QB passing based on completion % per attempt, yards gained per attempt, % of TD passes per attempt, and % of INTs per attempt. | (((Pass Comp/Pass ATTs -.3)\*5+(Pass Yds/Pass ATTs -.3)\*.25+(Pass TDs/Pass ATTs)\*20+2.375-(INT/Pass ATTs\*25))/6)\*100) |
| **Per Carry (Per Rush Attempt)** | **/Carry or /Rush ATT** | Metric | a stat divided by the number of carries/rush attempts in season/career | - |
| **Per Game** | **/GP** | Metric | stat divided by number of games in season/career. | - |
| **Per Team Pass Attempt** | **/TM Pass ATT** | Metric | stat divided by the player’s team pass attempts in a season | - |
| **Per Team Pass Touchdown** | **/TM Pass TD** | Metric | stat divided by the player’s team pass touchdowns in a season | - |
| **Per Play** | **/Play** | Metric | stat divided by the player’s team plays in a season | - |
| **Per Reception** | **/REC** | Metric | stat divided by number of receptions in a season/career | - |
| **Per Team Rush Attempt** | **/TM Rush ATT** | Metric | stat divided by player’s team rush attempts in a season. | - |
| **Per Team Rush Touchdown** | **/TM Rush TD** | Metric | stat divided by player’s team rush touchdowns in a season. | - |
| **Per Touch** | **/Touch** | Metric | stat divided by number of touches in season/career. | - |
| **Punt Return** | **Punt Returns** | Counting | when a player receives a punt from the opposing team and attempts to advance the ball down the field. | - |
| **Punt Return Touchdown** | **PR TDs** | Counting | when a punt return results in a touchdown for the team that receives it. | - |
| **Punt Return Yard** | **PR YDs** | Counting | the yards a player gains after receiving a punt from the opposing team. | - |
| **QB Pass Attempt Market Share** | **QB Pass ATT MS** | Metric | the percentage of total team Pass ATTs thrown by a QB | Pass ATT/TM Pass ATT |
| **QB Pass TD Market Share** | **QB Pass TD MS** | Metric | the percentage of total team Pass TDs thrown by a QB | Pass TDs/TM Pass TDs |
| **QB Pass Yards Market Share** | **QB Pass YDs MS** | Metric | the percentage of total team Pass YDs thrown by a QB | Pass YDs/TM Pass YDs |
| **Quarterback Breakout Age** | **QB BOA** | Metric | Quarterback BOA traditionally is measured as the age a QB first posted a QBR of 50 or higher in college while average 20 action plays per team game. My data for QBR does not go back to 2003 so I came up with a different criteria for QB BOA. QB BOA is when a QB has greater than 50% of their team’s passing attempts in a season. | “QB Pass ATT MS> or =50% and >100 Pass ATTs” |
| **Raw QBR** | **-** | Metric | created by ESPN to measure QB efficiency based on the down, distance, yard line at the snap, and the QBs contribution to the result of the play. Then this is divided by the total QB action plays in the game including penalties. | g(Adjusted EPA/Action Plays) |
| **Receiving College Dominator** | **REC CD** | Metric | a metric that equally weighs the career last and career best seasons in terms of market share of team receiving yards and receiving touchdowns. | "=((Career Last REC Yards MS+Career Last REC TDs MS+Career Best REC Yards MS+Career Best REC Yards MS)/4) |
| **Receiving TD Market Share** | **REC TD MS** | Metric | the percentage of total team receiving TDs by a player. | REC TDs/TM Pass TDs |
| **Receiving Touchdowns** | **REC TDs** | Counting | a reception that results in a touchdown. | - |
| **Receiving Yards** | **REC YDs** | Counting | the number of yards gained from the line of scrimmage by a player after catching a forward pass. | - |
| **Receiving Yards Market Share** | **REC YDs MS** | Metric | the percentage of total team receiving yards by a player. | REC YDs/TM Pass YDs |
| **Receptions** | **REC** | Counting | when a player catches and secures possession of a forward pass in bounds. | - |
| **Receptions Market Share** | **REC MS** | Metric | the percentage of total team receptions by a player. | REC/TM Completions |
| **Receptions Per Team Pass ATT** | **REC/TM Pass ATT** | Metric | a receiving efficiency metric that measures the number of receptions per team pass attempt | REC/TM Pass ATT |
| **Relative Athletic Score** | **RAS** | Metric | a metric created by Kent Lee Platte that can easily and intuitively gauge a player’s athletic abilities relative to the position they play | **-** |
| **Return** | **-** | Counting | when a player receives a kickoff or punt from the opposing team and attempts to advance the ball down the field. | KR+PR |
| **Return Touchdown** | **Return TD** | Counting | when a kick return or punt return results in a touchdown for the team that receives it. | KR TDs+PR TDs |
| **Return Yard** | **Return YDs** | Counting | the yards a player gains after receiving a kickoff or punt from the opposing team | KR YDs+PR YDs |
| **Rush Attempt** | **Rush ATT** | Counting | when a player tries to advance the ball up field getting the ball behind the line of scrimmage. | - |
| **Rush Attempt Market Share** | **Rush ATT MS** | Metric | the percentage of total team Rush ATTs by a player. | Rush ATT/TM Rush ATT |
| **Rush TD Market Share** | **Rush TDs MS** | Metric | the percentage of total team Rush TDs by a player. | Rush TDs/TM Rush TDs |
| **Rush Touchdowns** | **Rush TDs** | Counting | a Rush ATT that results in a touchdown | - |
| **Rush Yards** | **Rush YDs** | Counting | number of yards gained when a player carries the football. | - |
| **Rush Yards Market Share** | **Rush YDs MS** | Metric | the percentage of total team Rush YDs by a player. | Rush YDs/TM Rush YDs |
| **Rushing College Dominator** | **Rush CD** | Metric | a metric that equally weighs the career last and career best seasons in terms of market share of team rushing yards and rushing touchdowns. | "=((Career Last RUSH Yards MS+Career Last RUSH TDs MS+Career Best RUSH Yards MS+Career Best RUSH Yards MS)/4) |
| **Scrimmage College Dominator** | **SCRIM CD** | Metric | a metric that equally weighs the career last and career best seasons in terms of market share of team scrimmage yards and scrimmage touchdowns. | "=((Career Last SCRIM Yards MS+Career Last SCRIM TDs MS+Career Best SCRIM Yards MS+Career Best SCRIM Yards MS)/4) |
| **Scrimmage TD Market Share** | **SCRIM TD MS** | Metric | the percentage of total team scrimmage touchdowns by a player. | SCRIM TDs/TM SCRIM TDs |
| **Scrimmage Touchdowns** | **SCRIM TDs** | Counting | the total number of rushing touchdowns and receiving touchdowns by a player | REC TDs+Rush TDs |
| **Scrimmage Yards** | **SCRIM YDs** | Counting | the total number of rushing yards and receiving yards by a player | REC YDs+Rush YDs |
| **Scrimmage Yards Market Share** | **SCRIM YDs MS** | Metric | the percentage of total team scrimmage yards by a player. | SCRIM YDs/TM SCRIM YDs |
| **TE Height Adjusted Speed Score** | **HaSS** | Combine | an athleticism metric developed by Shawn Siegele that creates a metric for tight ends that adjusts a player’s 40 time for height. | (weight\*200)/(40 time\*40 time\*40 time\*40 time)\*(height/76.4)^2 |
| **Tight End Breakout Age** | **TE BOA** | Metric | the age in a season a tight end first achieves a greater than 15% receiving dominator rating. | “TE CD> or =15%” |
| **Total Adjusted Net Yards Per Attempt** | **TANY/A** | Metric | a schedule-adjusted, rushing-inclusive version of ANY/A or adjusted net yet yards per passing attempt created by Sean Slavin. My version does not adjust for sacks because my college data doesn’t have sacks and sack yards. | (Total Yards+20\* Total TDs-45\*INTs)/Total Attempts |
| **Total QB Market Share** | **Total QB MS** | Metric | a metric designed to represent a QBs production based on the percentage the QB had of team Rush YDs, Rush TDs, Pass YDs, and Pass TDs weighed equally. | ((Pass YDs/TM Pass Yds)/2)+((Pass TDs/TM Pass TDs)+(Rush YDs/TM Rush Yds)/2)+((Rush TDs/TM Rush TDs)/4) |
| **Total QBR** | **-** | Metric | created by ESPN to measure QB efficiency based on the down, distance, yard line at the snap, and the QBs contribution to the result of the play. Then this is divided by the total QB action plays in the game including penalties. This is then adjusted for play difficulty and the quality of opposing defenses. | ??? |
| **Total Touchdowns** | **Total TDs** | Counting | the total amount of rush touchdowns, receiving touchdowns, kick return touchdowns, and punt return touchdowns for a player. | SCRIM TDs+Return TDs |
| **Total Touches** | **-** | Counting | the total amount of rush attempts, receptions, kick returns, and punt returns for a player. | Touches+Returns |
| **Total Yards** | **Total YDs** | Counting | the total amount of rush yards, receiving yards, kick return yards, and punt return yards for a player. | SCRIM YDs+Return YDs |
| **Touch Market Share** | **Touch MS** | Metric | the percentage of total team touches by a player. | Touches/TM Plays |
| **Touchdown:Interception Ratio** | **TD:INT Ratio** | Metric | Pass TDs divided by Interceptions | Pass TDs/INT |
| **Touches** | **-** | Counting | the total number of receptions and rush attempts by a player | REC+Rush ATT |
| **Vertical Jump** | **Vert** | Combine | measures a player’s ability to jump off the ground from a flat-footed stance. | - |
| **Weight** | **-** | Combine | the body’s relative mass or the quantity of matter contained by it | - |
| **Weight Adjusted Speed Score** | **WaSS** | Combine | an athleticism metric developed by Bill Barnwell in Pro Football Prospectus that adjusts a player’s 40 yard dash time for their weight. | “=(weight\*200)/(40 time\*40 time\*40 time\*40 time)” |
| **Wide Receiver Breakout Age** | **WR BOA** | Metric | the age in a season a wide receiver first achieves a greater than 20% (30% Shawn Siegele) receiving dominator rating. | “REC DOM>20% (30%)” |
| **WR Height Adjusted Speed Score** | **HaSS** | Combine | an athleticism metric developed by Shawn Siegele that creates a metric for wide receivers that adjusts a player’s 40 time for height. | (weight\*200)/(40 time\*40 time\*40 time\*40 time)\*(height/73.5)^2 |

## Statistical Methods

Our variable of interest will be a variable called “Break Out” that indicates whether a player averages 10 fantasy football points a game or not. The field will be a binary Yes/No field.

When generating the predictions for this field, it will be useful to also be able to see not only the classification, but the associated probability of that classification so that we can raise or lower the threshold as needed when we tune our model.

Because this is a classification problem, I will test most of the well known algorithms to see which model produces the most accurate results. I will explore:

* Random Forest
* K-Nearest Neighbors
* Support Vector Machines (SVM)
* XGBOOST

To test each of these models, our dataset will be broken into a training (75%) and test set (25%). All of the model tuning and training will take place on the training dataset, and our test set will be the holdout set that we run our final models against to check the model’s accuracy.

Ultimately, the goal of this project is to develop a model that can predict which receivers will average 10 fantasy football points per game with 80% accuracy. That being said, accuracy will be the metric I use for my evaluation criteria. However, in addition to accuracy, I’ll also show common classification metrics such as Precision, Sensitivity, and Specificity.

The most difficult part of working with this dataset, is determining what to do with the missing data in each scenario. Almost every row of data has at least one missing value. The reasons for this vary, but common reasons include, the player was injured and missed a season, the player retired and stopped playing during the period of time this dataset contains, the player was drafted and played a season and then didn’t play after that or didn’t play for several seasons because more seasoned players were ahead of them on the team. For the majority of these cases, the missing values can be filled with 0’s because the player truly didn’t have a metric.

One area where I decided not to fill missing values with zero was the columns related to the NFL combine tests. Combine results are heavily utilized when players are being drafted and so I am making an assumption that they will be important to the predictive power of our model. It is fairly common for players to miss certain tests within the combine (due to injury, for example), so they may have values for some metrics but not others. A smaller population opt to not take part in the combine, while others may be injured and can’t participate.

Because most of the players have values for several of the combine metrics, I assume that players with recorded values in certain metrics may also have similar values in other metrics. For this reason, I will be imputing these missing values using k-means clustering, under the assumptions that a players “nearest neighbor” would have similar statistics.

The other difficulty with this dataset is the sheer number of columns. As described in the data section above, working through 600+ columns can make modeling extremely difficult, therefore, efforts to reduce the dimensionality of the dataset will be attempted while still maintaining prediction accuracy. I will explore the use of principal component analysis (PCA) to reduce the size of the dataset to make it more manageable during modeling.

## Discussion of Results

The summarized results from modeling are shown in the table below:

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Accuracy | Specificity | Sensitivity |
| Random Forest | 84.67% | 23.53% | 94.39% |
| XGBoost | 75.81% | 58.82% | 78.50% |
| K-Nearest Neigbors | 57.26% | 64.71% | 56.07% |
| Support Vector Machine | 73.39% | 58.82% | 75.70% |

While the overall accuracy of the Random Forest model is greater than 80%, the precision (or how accurate the model is at classifying breakout wide receivers) of the model is only ~24%%, which is less than the 80% threshold I had set for this research. The XGBoost model has the most balanced results with an accuracy of ~76% and a specificity of ~59%. While these results are disappointing from the perspective that creating an extremely accurate model may not be possible, these are most likely the expected results. As mentioned previously, every year there are analysts who create predictions for rookie wide receivers and each year there are unexpected breakouts as well as unexpected flops.

Part of the reason why this problem is so difficult to model is that there are important metrics that are not always captured in the data. For example, some players may have amazing metrics because they were a star player at a smaller college with less competition and were part of a less competitive conference. The opposite is also true. There may be a very good player who was a second string player at an amazing school who may not have been able to play as much and put up the statistics they were capable of. Additionally, the NFL is composed of players competing at the highest level. Some athletes will adjust to this better than others. These are just examples and by no means is it meant to be an exhaustive list.

While the results are not what I had hoped, I do think they are promising and open the door for further advancement and research. As can be see in the results for each model, it’s clear that the models are picking up on different trends and patterns. For example the Random Forest model has the overall highest accuracy because it makes the least amount of mistakes when it comes to false negatives by bucketing most players as ‘not a breakout’:

|  |  |  |
| --- | --- | --- |
|  | No | Yes |
| No | 101 | 13 |
| Yes | 6 | 4 |

Because of the class imbalance, this automatically makes the model more accurate just by guessing ‘No’ a majority of the time. Out of 17 breakouts in the dataset, it correctly identified 4, predicted that 6 non-breakouts would be breakouts, and missed 13 of the 17. These results tell us that the random forest model is being more conservative in who it predicts as a breakout and is overall favoring the accuracy of the overall model as opposed to the class of interest.

On the other end of the spectrum, the KNN model has the lowest accuracy because it is far less conservative with its allocation to “breakout” and inaccurately classifies 47 non-breakout receivers as “breakouts”. In allocating more receivers to “breakout” it does correctly classify 11 of the 17 true breakouts making its sensitivity 56%.

|  |  |  |
| --- | --- | --- |
|  | No | Yes |
| No | 60 | 6 |
| Yes | 47 | 11 |

In the middle of the spectrum is the XGBoost model. This model had an overall accuracy of ~76%. The model correctly identified 10 of the 17 breakout wide receivers which is a specificity of ~59%, which is the second highest behind the KNN model. Unlike the KNN model, it only had 23 false positives.

|  |  |  |
| --- | --- | --- |
|  | No | Yes |
| No | 84 | 7 |
| Yes | 23 | 10 |

Overall, the XGBoost model seems to be the most balanced, and if needed, this would be the model chosen to productionalize. The SVM model had very similar results to the XGBoost model, however, they were slightly less accurate. In the above discussion, as mentioned, the results show that most of the models seem to be picking up on different patterns within the data. This seems to suggest that an ensemble model may be able to provide a more accurate model above that of an individual algorithm. An attempt at building an ensemble model was made, however, because of compute power restraints and the length of time the model was taking to run, the attempt was abandoned. As a next step in this research, building an ensemble model using GPU’s or cloud computing power would be an area of focus.

## Conclusion

The research performed in this study is quite possibly the first of it’s kind. Being a pioneer in a research field can often be difficult as there are no examples or sources to look to or learn from, however, the research performed provides a good starting point for further research in the area and will open the doors for future researchers to investigate similar questions. While I was unable to build a model to a model that could predict a “breakout” wide receiver with 80% accuracy, I was able to build some directional models as well as outline a path forward for potential improvements using ensembling and additional computing resources. This research is just the tip of the iceberg in this new field of analytics within NFL Dynasty League Football.

## References

Hampton, B. (2020, May 24). *Blake’s Predictive 2020 Wide Receiver Model*. Retrieved from The Undroppables: <https://www.theundroppables.com/blakes-predictive-2020-wide-receiver-model/>

Lindsey, J. (2016, January 13). *The NFL Is Finally Tapping Into the Power of Data*. Retrieved from Wired: <https://www.wired.com/2016/01/the-nfls-impending-data-revolution/>

Subramanian, P. (2014, August 29). *The 3 biggest prizes in fantasy football*. Retrieved from Yahoo Finance: <https://www.yahoo.com/entertainment/news/the-3-biggest-prizes-in-fantasy-football-174712081.html>

Wright, D. (n.d.). *DAVID WRIGHT*. Retrieved from <https://dynastyleaguefootball.com/author/spacemnspiff207/:> <https://dynastyleaguefootball.com/author/spacemnspiff207/>

## Appendix

## --------------------------DATA PREP AND CLEANING ---------------------------------------------------  
  
wr <- readr::read\_csv("https://raw.githubusercontent.com/christianthieme/masters\_research\_project/master/wide\_receiver\_databaseV1.csv") %>%   
 mutate\_if(is.character, as.factor)  
  
  
opportunity <- readr::read\_csv("https://raw.githubusercontent.com/christianthieme/masters\_research\_project/main/WReceiverOpp.csv") %>%   
 mutate\_if(is.character, as.factor) %>%  
 select(Team\_Next\_Year, Opportunity)  
  
wr2 <- wr %>%   
 select('Player','Team', 'School', 'Conf','DOB', 'DR', 'DP','Draft\_Year', 'Draft\_Age', 'BreakoutAge\_WRBOA\_twentypercent','BreakoutAge\_WRBOA\_thirtypercent','TotalCareer\_REC', 'GamesPlayed\_18','GamesPlayed\_19','GamesPlayed\_20', 'GamesPlayed\_21', 'GamesPlayed\_22', "RushATTs\_18" ,   
 "RushATTs\_19", "RushATTs\_20" , "RushATTs\_21",   
 "RushATTs\_22" , "RushATTs\_23" , "RushYards\_18" ,   
 "RushYards\_19" , "RushYards\_20" , "RushYards\_21",   
 "RushYards\_22" , "RushYards\_23" , "RushTDs\_18" ,   
 "RushTDs\_19" , "RushTDs\_20" , "RushTDs\_21" ,   
 "RushTDs\_22" , "RushTDs\_23" , "MSRushATTs\_18" ,   
 "MSRushATTs\_19" , "MSRushATTs\_20" , "MSRushATTs\_21" ,   
 "MSRushATTs\_22" , "MSRushATTs\_23" , "MSRushYards\_18" ,   
 "MSRushYards\_19" , "MSRushYards\_20" , "MSRushYards\_21" ,   
 "MSRushYards\_22" , "MSRushYards\_23" , "MSRushTDs\_18" ,   
 "MSRushTDs\_19" , "MSRushTDs\_20" , "MSRushTDs\_21" ,   
 "MSRushTDs\_22" , "MSRushTDs\_23" , "CombinedRUSHMS\_18" ,   
 "CombinedRUSHMS\_19" , "CombinedRUSHMS\_20" , "CombinedRUSHMS\_21" ,   
 "CombinedRUSHMS\_22" , "CombinedRUSHMS\_23" , "Yards\_Carry\_18" ,   
 "Yards\_Carry\_19" , "Yards\_Carry\_20" , "Yards\_Carry\_21" ,   
 "Yards\_Carry\_22" , "Yards\_Carry\_23" , "Yards\_TMRushAttempt\_18" ,   
 "Yards\_TMRushAttempt\_19" , "Yards\_TMRushAttempt\_20" , "Yards\_TMRushAttempt\_21" ,   
 "Yards\_TMRushAttempt\_22" , "Yards\_TMRushAttempt\_23" , "TDs\_TMRushAttempt\_18" ,   
 "TDs\_TMRushAttempt\_19" , "TDs\_TMRushAttempt\_20" , "TDs\_TMRushAttempt\_21" ,   
 "TDs\_TMRushAttempt\_22" , "TDs\_TMRushAttempt\_23" , "Receptions\_18" ,   
 "Receptions\_19" , "Receptions\_20" , "Receptions\_21" ,   
 "Receptions\_22" , "Receptions\_23" , "ReceivingYards\_18" ,   
 "ReceivingYards\_19", "ReceivingYards\_20" , "ReceivingYards\_21" ,   
 "ReceivingYards\_22" , "ReceivingYards\_23" , "ReceivingTDs\_18" ,   
 "ReceivingTDs\_19" , "ReceivingTDs\_20" , "ReceivingTDs\_21" ,   
 "ReceivingTDs\_22" , "ReceivingTDs\_23" , "MSReceptions\_18" ,   
 "MSReceptions\_19" , "MSReceptions\_20" , "MSReceptions\_21" ,   
 "MSReceptions\_22" , "MSReceptions\_23" , "MSReceivingYards\_18" ,   
 "MSReceivingYards\_19" , "MSReceivingYards\_20" , "MSReceivingYards\_21" ,   
 "MSReceivingYards\_22" , "MSReceivingYards\_23" , "MSReceivingTDs\_18" ,   
 "MSReceivingTDs\_19" , "MSReceivingTDs\_20" , "MSReceivingTDs\_21" ,   
 "MSReceivingTDs\_22" , "MSReceivingTDs\_23" , "CombinedRECMS\_18" ,   
 "CombinedRECMS\_19" , "CombinedRECMS\_20" , "CombinedRECMS\_21" ,   
 "CombinedRECMS\_22" , "CombinedRECMS\_23" , "TotalCombinedMS\_18" ,   
 "TotalCombinedMS\_19" , "TotalCombinedMS\_20" , "TotalCombinedMS\_21" ,   
 "TotalCombinedMS\_22" , "TotalCombinedMS\_23" , "Yards\_Reception\_18" ,   
 "Yards\_Reception\_19" , "Yards\_Reception\_20" , "Yards\_Reception\_21" ,   
"Yards\_Reception\_22" , "Yards\_Reception\_23" , "Yards\_ReceptionOverTMAverage\_18" ,   
 "Yards\_ReceptionOverTMAverage\_19" , "Yards\_ReceptionOverTMAverage\_20" , "Yards\_ReceptionOverTMAverage\_21" ,  
 "Touches\_18", "Touches\_19" , "Touches\_20" , "Touches\_21" ,   
 "Touches\_22" , "Touches\_23", "ScrimmageYards\_18" ,   
 "ScrimmageYards\_19" , "ScrimmageYards\_20" , "ScrimmageYards\_21" ,   
 "ScrimmageYards\_22" , "ScrimmageYards\_23" , "ScrimmageTDs\_18" ,   
 "ScrimmageTDs\_19" , "ScrimmageTDs\_20" , "ScrimmageTDs\_21" ,   
 "ScrimmageTDs\_22" , "ScrimmageTDs\_23" , "TouchMS\_18" ,   
 "TouchMS\_19" , "TouchMS\_20" , "TouchMS\_21" ,   
 "TouchMS\_22" , "TouchMS\_23" , "ScrimmageYardsMS\_18" ,   
 "ScrimmageYardsMS\_19" , "ScrimmageYardsMS\_20" , "ScrimmageYardsMS\_21" ,   
 "ScrimmageYardsMS\_22" , "ScrimmageYardsMS\_23" , "ScrimmageTDsMS\_18" ,   
 "ScrimmageTDsMS\_19" , "ScrimmageTDsMS\_20" , "ScrimmageTDsMS\_21" ,   
 "ScrimmageTDsMS\_22" , "ScrimmageTDsMS\_23" , "CombineSCRIMMS\_18" ,   
 "CombineSCRIMMS\_19" , "CombineSCRIMMS\_20" , "CombineSCRIMMS\_21" ,   
 "CombineSCRIMMS\_22" , "CombineSCRIMMS\_23" , "ScrimmageYards\_Touch\_18" ,   
 "ScrimmageYards\_Touch\_19" , "ScrimmageYards\_Touch\_20" , "ScrimmageYards\_Touch\_21" ,   
 "ScrimmageYards\_Touch\_22" , "ScrimmageYards\_Touch\_23" , "ScrimYards\_TouchOverTMAVG\_18" ,   
 "ScrimYards\_TouchOverTMAVG\_19" , "ScrimYards\_TouchOverTMAVG\_20" , "ScrimYards\_TouchOverTMAVG\_21" ,   
 "ScrimYards\_TouchOverTMAVG\_22" , "ScrimYards\_TouchOverTMAVG\_23" , "Touch\_Play\_18" ,   
 "Touch\_Play\_19" , "Touch\_Play\_20" , "Touch\_Play\_21" ,   
 "Touch\_Play\_22" , "Touch\_Play\_23" , "ScrimmageYards\_Play\_18" ,   
"ScrimmageYards\_Play\_19", "ScrimmageYards\_Play\_20" , "ScrimmageYards\_Play\_21" ,  
 "ScrimmageYards\_Play\_22" , "ScrimmageYards\_Play\_23" , "ScrimmageTDs\_Play\_18" ,  
 "ScrimmageTDs\_Play\_19" , "ScrimmageTDs\_Play\_20" , "ScrimmageTDs\_Play\_21" ,  
 "ScrimmageTDs\_Play\_22" , "ScrimmageTDs\_Play\_23" , "PPRFantasyPoints\_18" ,  
 "PPRFantasyPoints\_19" , "PPRFantasyPoints\_20" , "PPRFantasyPoints\_21" ,  
 "PPRFantasyPoints\_22" , "PPRFantasyPoints\_23",  
 "TotalTouches\_19" , "TotalTouches\_20" , "TotalTouches\_21" ,   
 "TotalTouches\_22" , "TotalTouches\_23" , "TotalYards\_18" ,   
 "TotalYards\_19" , "TotalYards\_20" , "TotalYards\_21" ,   
 "TotalYards\_22" , "TotalYards\_23" , "TotalTDs\_18" ,   
 "TotalTDs\_19" , "TotalTDs\_20" , "TotalTDs\_21" ,   
 "TotalTDs\_22" , "TotalTDs\_23" ,  
"Combine\_Height" , "Combine\_Weight" , "Combine\_HandSize" ,   
 "Combine\_ArmLength" , "Combine\_40time" , "Combine\_Bench" ,   
 "Combine\_Vertical", "Combine\_Broad" , "Combine\_Shuttle" ,   
 "Combine\_3Cone",  
 "Yards\_ReceptionOverTMAverage\_22" , "Yards\_ReceptionOverTMAverage\_23" ,'TotalCareer\_REC\_GP', 'TotalCareer\_RECYards', 'TotalCareer\_RECYards\_GP', 'TotalCareer\_RECTDs', 'TotalCareer\_RECTDs\_GP', 'TotalCareer\_Yards\_REC', 'TotalCareer\_Touches', 'TotalCareer\_Touches\_GP', 'TotalCareer\_PPRPoints', 'TotalCareer\_PPR\_GP', 'TotalCareer\_PPR\_Touch', 'TotalCareer\_TotalTouches\_GP', 'TotalCareer\_TotalYards', 'TotalCareer\_TotalYards\_GP', 'TotalCareer\_TotalTDs', 'TotalCareer\_TotalTDs\_GP', 'CareerBest\_REC', 'CareerBest\_RECs\_GP', 'CareerBest\_RECYards', 'CareerBest\_RECYards\_GP', 'CareerBest\_RECTDs', 'CareerBest\_RECTDs\_GP', 'CareerBest\_RECMS', 'CareerBest\_RECYardsMS', 'CareerBest\_RECTDsMS', 'CareerBest\_CombinedRECMS', 'CareerBest\_TotalMS', 'CareerBest\_Yards\_REC', 'CareerBest\_YDs\_RECOverTMAVG', 'CareerBest\_REC\_TMPA', 'CareerBest\_RECYards\_TMPA', 'CareerBest\_RECTDS\_TMPA', 'CareerBest\_Touches', 'CareerBest\_Touches\_GP', 'CareerBest\_SCRIMYards', 'CareerBest\_SCRIMYards\_GP', 'CareerBest\_SCRIMTDs', 'CareerBest\_SCRIMTDs\_GP', 'CareerBest\_TouchMS', 'CareerBest\_SCRIMYardsMS', 'CareerBest\_SCRIMTDsMS', 'CareerBest\_COMBSCRIMMS', 'CareerBest\_SCRIMYards\_Touch', 'CareerBest\_SCRIMYDs\_TouchOverTMAVG', 'CareerBest\_Touch\_Play', 'CareerBest\_SCRIMYards\_Play', 'CareerBest\_SCRIMTDs\_Play', 'CareerBest\_PPRPoints', 'CareerBest\_PPR\_GP', 'CareerBest\_PPR\_Touch', 'CareerBest\_TotalTouches', 'CareerBest\_TotalTouches\_GP', 'CareerBest\_TotalYards', 'CareerBest\_TotalYards\_GP', 'CareerBest\_TotalTDs', 'CareerBest\_TotalTDs\_GP', 'CareerLast\_RECs\_GP', 'CareerLast\_RECYards', 'CareerLast\_RECYards\_GP', 'CareerLast\_RECTDs', 'CareerLast\_RECTDs\_GP', 'CareerLast\_RECMS', 'CareerLast\_RECYardsMS', 'CareerLast\_RECTDsMS', 'CareerLast\_CombinedRECMS', 'CareerLast\_TotalMS', 'CareerLast\_Yards\_REC', 'CareerLast\_YDs\_RECOverTMAVG', 'CareerLast\_REC\_TMPA', 'CareerLast\_RECYards\_TMPA', 'CareerLast\_RECTDS\_TMPA', 'CareerLast\_Touches', 'CareerLast\_Touches\_GP', 'CareerLast\_SCRIMYards', 'CareerLast\_SCRIMYards\_GP', 'CareerLast\_SCRIMTDs', 'CareerLast\_TotalTouches\_GP', 'CareerLast\_TotalYards', 'CareerLast\_TotalYards\_GP', 'CareerLast\_TotalTDs', 'CareerAverage\_RECYards', 'CareerAverage\_RECYards\_GP', 'CareerAverage\_RECTDs', 'CareerAverage\_RECTDs\_GP', 'CareerAverage\_RECMS', 'CareerAverage\_RECYardsMS', 'CareerAverage\_RECTDsMS', 'CareerAverage\_CombinedRECMS', 'CareerAverage\_TotalMS', 'CareerAverage\_Yards\_REC', 'CareerAverage\_YDs\_RECOverTMAVG', 'CareerAverage\_REC\_TMPA',   
'CareerAverage\_RECYards\_TMPA', 'CareerAverage\_RECTDS\_TMPA', 'CareerAverage\_Touches', 'CareerAverage\_Touches\_GP', 'CareerAverage\_SCRIMYards', 'CareerAverage\_SCRIMYards\_GP','CareerAverage\_SCRIMTDs', 'CareerAverage\_SCRIMTDs\_GP', 'CareerAverage\_TouchMS', 'CareerAverage\_SCRIMYardsMS', 'CareerAverage\_SCRIMTDsMS', 'CareerAverage\_COMBSCRIMMS', 'CareerAverage\_SCRIMYards\_Touch', 'CareerAverage\_SCRIMYDs\_TouchOverTMAVG', 'CareerAverage\_Touch\_Play', 'CareerAverage\_SCRIMYards\_Play',  
'CareerAverage\_SCRIMTDs\_Play', 'CareerAverage\_PPRPoints', 'CareerAverage\_PPR\_GP', 'CareerAverage\_PPR\_Touch', 'CareerAverage\_TotalTouches', 'CareerAverage\_TotalTouches\_GP', 'CareerAverage\_TotalYards', 'CareerAverage\_TotalYards\_GP', 'CareerAverage\_TotalTDs', 'CareerAverage\_TotalTDs\_GP', 'CollegeDominator\_RECCD', 'CollegeDominator\_SCRIMCD', 'Combine\_BMI', 'Combine\_Height', 'Combine\_Weight', 'Combine\_HaSS', 'RAS','Agility\_Score', 'Burst\_Score', 'Freak\_Score', 'fantasy\_ppg' ) %>%  
 filter(Draft\_Year < 2019 ) %>%  
 select(-contains("\_23")) %>%   
 mutate(Drafted = as.factor(ifelse(DR == 'UDFA', 'No', 'Yes'))) %>%   
 mutate(DR = ifelse(DR == 'UDFA', 10, as.numeric(as.character(DR)))) %>%  
 mutate(DP = ifelse(DP == 'UDFA',999, as.numeric(as.character(DP)))) %>%   
 mutate(breakout = as.factor(ifelse(fantasy\_ppg >= 10, 'Yes', 'No'))) %>%   
 filter(Team != 'bum') %>%   
## mutate(Draft\_year = as.factor(Draft\_Year)) %>%   
 #mutate(CareerAverage\_KRTDs = as.numeric(CareerAverage\_KRTDs)) %>%  
 # mutate(CareerAverage\_PuntReturns = as.numeric(CareerAverage\_PuntReturns)) %>%  
 # mutate(CareerAverage\_PRYards = as.numeric(CareerAverage\_PRYards)) %>%  
 # mutate(CareerAverage\_PRTDs = as.numeric(CareerAverage\_PRTDs)) %>%  
 select(-fantasy\_ppg) %>%   
 mutate(Team\_Next\_Year = paste(Team, Draft\_Year, sep = '-'))   
  
  
wr2 <- wr2 %>%   
 left\_join(opportunity, by = 'Team\_Next\_Year')  
  
wr\_fill\_0 <- wr2 %>%   
 select(-Combine\_BMI, -Combine\_Height, -Combine\_Weight, -Combine\_HandSize, -Combine\_ArmLength, -Combine\_40time, -Combine\_Bench, -Combine\_Vertical, -Combine\_Broad, -Combine\_Shuttle, -Combine\_3Cone, -Combine\_HaSS, -RAS, -Agility\_Score, -Burst\_Score, -Freak\_Score) %>%  
 mutate(key = paste0(Player, School, Conf, DOB))  
  
wr\_fill\_0[is.na(wr\_fill\_0)] <- 0   
  
  
wr\_fill\_impute\_select <- wr2 %>%   
 select(Player, School, Conf, DOB, Combine\_BMI, Combine\_Height, Combine\_Weight, Combine\_HandSize, Combine\_ArmLength, Combine\_40time, Combine\_Bench, Combine\_Vertical, Combine\_Broad, Combine\_Shuttle, Combine\_3Cone, Combine\_HaSS, RAS, Agility\_Score, Burst\_Score, Freak\_Score) %>%  
 mutate(key = paste0(Player, School, Conf, DOB)) %>%   
 select(-Player, -School, -Conf, -DOB)  
  
wr\_no\_nulls <- wr\_fill\_0 %>%   
 left\_join(wr\_fill\_impute\_select, by = "key") %>%  
 select(-Draft\_Year, -key)  
  
null\_cols<- colSums(is.na(wr\_no\_nulls))  
null\_cols[null\_cols>0]  
  
wr\_no\_nulls <- wr\_no\_nulls %>%  
 janitor::clean\_names()  
  
## ----------------------------------- TRAINING/TEST DATA ------------------------------------  
  
  
set.seed(222)  
train\_test\_split <- initial\_split(wr\_no\_nulls, strata = breakout)  
  
wr\_train <- training(train\_test\_split)  
wr\_test <- testing(train\_test\_split)  
  
set.seed(234)  
  
val\_set <- validation\_split(wr\_train,   
 strata = breakout,   
 prop = 0.80)  
  
## ------------------------------------- RF Model -------------------------------------------------  
  
  
cores <- parallel::detectCores()  
  
rf\_mod <-   
 rand\_forest(mtry = tune(),   
 min\_n = tune(),   
 trees = 1500) %>%  
 set\_engine("ranger", num.threads = cores) %>%  
 set\_mode("classification")  
  
  
rf\_recipe <-   
 recipe(breakout ~ ., data = wr\_train) %>%  
 update\_role(player , new\_role = "ID") %>%  
 step\_novel(all\_nominal(), -all\_outcomes()) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes()) %>%  
 step\_zv(all\_predictors()) %>% #remove all variables with only a single value  
 step\_normalize(all\_predictors()) %>%  
 step\_impute\_knn(all\_predictors()) %>%  
 # step\_pca(all\_numeric(), -all\_outcomes()) %>%  
 themis::step\_upsample(breakout, over\_ratio = 1.0)  
  
  
rf\_workflow <-   
 workflow() %>%  
 add\_model(rf\_mod) %>%  
 add\_recipe(rf\_recipe)  
  
rf\_res <-   
 rf\_workflow %>%  
 tune\_grid(val\_set,   
 grid = 15,   
 # split\_rule = "gini",  
 control = control\_grid(save\_pred =TRUE),   
 metrics = metric\_set(roc\_auc)  
 )  
  
  
rf\_best <-   
 rf\_res %>%  
 select\_best(metric = "roc\_auc")  
  
last\_rf\_mod <-   
 rand\_forest(mtry = 23, min\_n = 25, trees = 1500) %>% #  
 set\_engine("ranger", num.threads = cores, importance = "impurity") %>%  
 set\_mode("classification")  
  
last\_rf\_workflow <-   
 rf\_workflow %>%  
 update\_model(last\_rf\_mod)   
  
set.seed(345)  
last\_rf\_fit <-   
 last\_rf\_workflow %>%  
   
 last\_fit(train\_test\_split)  
  
fitted <- last\_rf\_fit %>%   
 collect\_predictions()   
  
confusionMatrix(data =fitted$.pred\_class , reference = fitted$breakout)  
  
predictions <- last\_rf\_fit %>%   
 collect\_predictions() %>%   
 mutate(row\_number = row\_number())  
  
final <- wr\_test %>%   
 mutate(row\_number = row\_number()) %>%  
 select(row\_number, player, school, dr, dp, draft\_age ) %>%   
 left\_join(predictions, by = "row\_number")  
  
## ----------------------------- XGBOOST MODEL -----------------------------------------  
  
xgb\_wr <- boost\_tree(  
 trees = 1000,   
 tree\_depth = tune(),   
 min\_n = tune(),   
 loss\_reduction = tune(), ## first three: model complexity  
 sample\_size = tune(),  
 mtry = tune(), ## randomness  
 learn\_rate = tune(), ## step size  
) %>%   
 set\_engine("xgboost") %>%   
 set\_mode("classification")  
  
xgb\_grid <- grid\_latin\_hypercube(  
 tree\_depth(),  
 min\_n(),  
 loss\_reduction(),  
 sample\_size = sample\_prop(),  
 finalize(mtry(), wr\_train),  
 learn\_rate(),  
 size = 20  
)  
  
xgb\_wr <- workflow() %>%  
 add\_recipe(rf\_recipe) %>%  
 add\_model(xgb\_wr)  
  
xgb\_res <- xgb\_wr %>% tune\_grid(  
 val\_set,   
 grid = xgb\_grid,  
 control = control\_grid(save\_pred = TRUE),  
 metrics = metric\_set(roc\_auc)  
)  
  
best\_auc <- select\_best(xgb\_res, "roc\_auc")  
  
final\_xgb <- finalize\_workflow(  
 xgb\_wr,  
 best\_auc  
)  
  
final\_res <- last\_fit(final\_xgb, train\_test\_split)  
  
collect\_metrics(final\_res)  
  
xg\_pred <- final\_res %>%   
 collect\_predictions()  
  
confusionMatrix(data = xg\_pred$.pred\_class, reference = xg\_pred$breakout)  
  
predictions <- last\_rf\_fit %>%   
 collect\_predictions() %>%   
 mutate(row\_number = row\_number())  
  
final <- wr\_test %>%   
 mutate(row\_number = row\_number()) %>%  
 select(row\_number, player, school, dr, dp, draft\_age ) %>%   
 left\_join(predictions, by = "row\_number")  
  
## ------------------------------------------------ KNN MOdel ----------------------------------------  
  
  
knn\_model <-  
 nearest\_neighbor(  
 mode = "classification",  
 neighbors = tune("K"),  
 weight\_func = tune(),  
 dist\_power = tune()  
 ) %>%  
 set\_engine("kknn")  
  
knn\_wflow <-  
 workflow() %>%  
 add\_recipe(rf\_recipe) %>%  
 add\_model(knn\_model)  
  
set.seed(123)  
  
knn\_set <- parameters(knn\_wflow) %>%  
 update(K = neighbors(c(1, 10)))  
  
knn\_grid <- knn\_set %>%  
 grid\_max\_entropy(size = 10)  
  
knn\_grid\_search <- knn\_wflow %>%   
 tune\_grid(  
 val\_set,  
 grid = knn\_grid,  
 control = control\_grid(save\_pred = TRUE),  
 metrics = metric\_set(roc\_auc)  
 )  
  
best\_auc\_knn <- select\_best(knn\_grid\_search, "roc\_auc")  
  
final\_knn <- finalize\_workflow(  
 knn\_wflow,  
 best\_auc\_knn  
)  
  
final\_knn <- last\_fit(final\_knn, train\_test\_split)  
  
knn\_pred <- final\_knn %>%   
 collect\_predictions()  
  
confusionMatrix(data = knn\_pred$.pred\_class, reference = knn\_pred$breakout)  
  
## ---------------------------------------------- SVM Model-------------------------------------  
  
svm\_model <-  
 svm\_rbf(cost = tune(), rbf\_sigma = tune()) %>%  
 set\_mode("classification") %>%  
 set\_engine("kernlab")  
  
svm\_wflow <-  
 workflow() %>%  
 add\_recipe(rf\_recipe) %>%  
 add\_model(svm\_model)  
  
set.seed(123)  
  
svm\_param <- parameters(svm\_wflow) %>%  
 update(rbf\_sigma = rbf\_sigma(c(-7, -1)))  
  
svm\_grid <- svm\_param %>%   
 update(  
 cost = cost(c(-6, 1)),  
 rbf\_sigma = rbf\_sigma(c(-6, -4))) %>%   
 grid\_regular(levels = 2)  
  
svm\_grid\_search <- svm\_wflow %>%   
 tune\_grid(  
 val\_set,  
 grid = svm\_grid,  
 control = control\_grid(save\_pred = TRUE),  
 metrics = metric\_set(roc\_auc)  
 )  
  
best\_auc\_svm <- select\_best(svm\_grid\_search, "roc\_auc")  
  
final\_svm <- finalize\_workflow(  
 svm\_wflow,  
 best\_auc\_svm  
)  
  
final\_svm <- last\_fit(final\_svm, train\_test\_split)  
  
svm\_pred <- final\_svm %>%   
 collect\_predictions()  
  
confusionMatrix(data = svm\_pred$.pred\_class, reference = svm\_pred$breakout)  
  
## ---------------------------------------------------- ENSEMBLE -------------------------------------------------   
  
  
# ctrl\_grid <- control\_stack\_grid()  
  
  
# set.seed(234)  
  
# rf\_res <-  
# rf\_workflow %>%  
# tune\_grid(val\_set,  
# grid = 15,  
# # split\_rule = "gini",  
# control = ctrl\_grid,  
# metrics = metric\_set(roc\_auc)  
# )  
#  
# xgb\_res <- xgb\_wr %>% tune\_grid(  
# val\_set,  
# grid = xgb\_grid,  
# control = ctrl\_grid,  
# metrics = metric\_set(roc\_auc)  
# )  
#  
# knn\_grid\_search <- knn\_wflow %>%  
# tune\_grid(  
# val\_set,  
# grid = knn\_grid,  
# control = ctrl\_grid,  
# metrics = metric\_set(roc\_auc)  
# )  
#  
#  
# svm\_grid\_search <- svm\_wflow %>%  
# tune\_grid(  
# val\_set,  
# grid = svm\_grid,  
# control = ctrl\_grid,,  
# metrics = metric\_set(roc\_auc)  
# )  
  
  
# ensemble\_data\_st <-  
# stacks::stacks() %>%  
# stacks::add\_candidates(rf\_res) %>%  
# stacks::add\_candidates(xgb\_res) %>%  
# stacks::add\_candidates(knn\_grid\_search) %>%  
# stacks::add\_candidates(svm\_grid\_search)  
  
# ensemble\_model\_st <-  
# ensemble\_data\_st %>%  
# blend\_predictions()  
  
# theme\_set(theme\_bw())  
# autoplot(ensemble\_model\_st)  
  
# autoplot(ensemble\_model\_st, type = "members")  
  
# autoplot(ensemble\_model\_st, type = "weights")  
  
# ensemble\_model\_st <- ensemble\_model\_st %>%  
# fit\_members()  
  
# ensemble\_model\_st %>%  
# collect\_parameters()  
  
# ensemble\_model\_test <- wr\_test %>%  
# bind\_cols(predict(ensemble\_model\_st,.))  
  
# table(ensemble\_model\_test$breakout,ensemble\_model\_test$.pred\_class)  
  
##-------------------------------- Scraper Code------------------------------------------------  
  
  
# scraper\_func <- function(base\_url, team, year, ending) {  
#  
# build\_url <- paste0(base\_url, team, "/", year, ending)  
#  
# url <- xml2::read\_html(build\_url)  
#  
# data <- url %>%  
# html\_nodes('\*[id="all\_rushing\_and\_receiving"]') %>%  
# html\_nodes(xpath = 'comment()') %>%  
# html\_text() %>%  
# read\_html() %>%  
# html\_node('table') %>%  
# html\_table()  
#  
# names(data) <- c("No.", "Player", "Age", "Pos", "Games\_G", "Games\_GS", "Rushing\_Att", "Rushing\_Yds", "Rushing\_TD", "Rushing\_Lng", "Rushing\_YA", "Rushing\_YG", "Rushing\_AG", "Receiving\_Tgt", "Receiving\_Rec", "Receiving\_Yds", "Receiving\_YR","Receiving\_TD", "Receiving\_Lng", "Receiving\_RG", "Receiving\_YG", "Receiving\_Ctch", "Receiving\_Y\_Tgt", "Total\_Yds\_Touch", "Total\_Yds\_Y\_Tch", "Total\_Yds\_Y\_Scm", "RRTD", "Fmb")  
#  
# data <- data[c(2:length(data)),]  
#  
# wide\_receiver <- data %>%  
# dplyr::filter(Player != 'Team Total') %>%  
# dplyr::filter(Player != 'Opp Total') %>%  
# #dplyr::filter(Pos %in% c('WR', 'wr', 'TE')) %>%  
# dplyr::select(Player, Pos, Receiving\_Rec, Receiving\_Yds) %>%  
# dplyr::arrange(desc(as.numeric(Receiving\_Yds))) %>%  
# dplyr::top\_n(1, wt = as.numeric(Receiving\_Yds)) %>%  
# mutate(  
# Receiving\_Yds = as.numeric(Receiving\_Yds),  
# Opportunity = ifelse(Receiving\_Yds >= 1465, 1,  
# ifelse(Receiving\_Yds >= 1240, 2,  
# ifelse(Receiving\_Yds >= 929, 3,  
# ifelse(Receiving\_Yds >= 657, 4,  
# 5))))) %>%  
# mutate(  
# Team = team,  
# Year = year,  
# Next\_Year = as.numeric(year) + 1,  
# Team\_Next\_Year = paste(team, Next\_Year, sep = "-")  
# )  
#  
# return(wide\_receiver)  
#  
# }  
  
# wr\_df <- data.frame()  
#  
# teams <- c('crd', 'atl','rav', 'buf','car', 'chi', 'cin', 'cle', 'dal', 'den', 'det', 'gnb', 'htx', 'clt', 'jax', 'kan', 'sdg', 'ram', 'mia', 'min', 'nwe', 'nor', 'nyg', 'nyj', 'rai', 'phi', 'pit', 'sfo', 'sea', 'tam', 'oti', 'was')  
# years <- seq(2002,2021)  
#  
#  
# for (team in teams){  
# for (year in years){  
# receiver <- scraper\_func("https://www.pro-football-reference.com/teams/", team, year, ".htm#rushing\_and\_receiving")  
# wr\_df <- rbind(wr\_df, receiver)  
# }  
# }  
#  
# readr::write\_csv(wr\_df, 'C:/Users/chris/OneDrive/Master Of Data Science - CUNY/last\_semester/masters\_research\_project/WReceiverOpp.csv')