# Data Analytics Project Report

## Topic: Predicting offensive plays in the National Football League

Name: Shounak Rangwala

Netid: snr85

## Introduction:

Ever since Brad Pitt wowed us by his performance in the film "Moneyball", an important aspect of sports has been brought into public focus: the merging of data analytics and sports. Data analytics had been called upon in so many different businesses to help them in their problems, the sports industry never figured out how to harness the full power of this tool until very recently. Major sports leagues such as the NFL, NBA, MLB and many international soccer leagues are planning to integrate such analytics on a deeper level.

There are basically two ways data analytics can be used in the sports industry: off-field data analytics and on-field data analytics. Off-field data analytics is concerned with the factors and parameters surrounding the game/sport. This includes a wide variety of areas ranging from fan approval rating from Twitter sentiment analysis, to finding out the best business deals that each sport franchise can enter such that it's a win-win situation for both the parties, to helping out with crowd control and predicting what will be the best way to manage traffic after sporting events and lastly to help analysis the sports data to help people make wagers on the outcome of the games for sport. While the opportunities of research and projects on these fields are limitless, the more interesting use of analytics is the on-field analytics.

On-field analytics is analytics that is concerned in the actual game/sport and the ways to play it so that the maximum performance/ best result are guaranteed. The data is directly collected by historical game details and the same is used in innovative models to predict different aspects of the game. One prominent example of this is about the addition of SportVU1/ cameras in all NBA arenas for the 2013-2014 season, which record the x, y, and z coordinates of the players and the ball 25 times per second, has added a huge amount of data that teams can utilize. This is instrumental in predicting the efficiency of basketball players in different positions on the basketball court and thus helping players make the shots that will ensure them a higher percentage of making the shot. This report, however, focusses on applying the principles of data-analytics in the National Football league. We are trying to predict if the next play called on the field will be a "pass" play or a "run" play based on previously accumulated data and trying to identify existing tendencies in the league, all the time looking for a predictive model that provides the best precision in the results.

The National Football League has become an American tradition over its 100-year history and has been the most widely watched sport in America. This year amid the COVID-19 pandemic, Americans have sorely needed the sport more than ever, a great escape from the problems of everyday life. An advertisement run during the matches stresses on this very point even going so

far as to call football a “microcosm” of America. The NFL has 16 teams and makes approximately $1.12 billion every year. Each team is limited to spending $198.3 million every year on employees. Judging sorely by these numbers, we know that all the teams are going to try as hard as possible to win as many games in the regular season as they can. This objective can be achieved by harnessing the power of data analytics and predicting offensive plays of different teams. By doing this teams can come up with offensive plays that will guarantee them a higher chance of “success” and defensive teams can come up with ways of countering the said strategies. Tonnes of research has been done in these aspects and many predictive models have been made to predict various aspects of the game so as to come up with a model that has the highest accuracy in predicting plays. Football coaches believe that while there will always be some intangibles that wary person-to-person and situation-to-situation that will need experience and intuition, such a prediction system will greatly help the staff in preparation for the game and also sometimes in execution of the game. This report will detail the data used to perform the analysis upon, the detailed cleaning and preprocessing steps taken that will prepare the data for our use, the various attributes the data has and the importance of the key attributes to the predictions and also the additional data features engineered to help in the analysis. I will summarize the research papers I read to prepare the base for carrying out this project to fruition. We will also talk about the methods that will be used to make the model, the challenges and the expectations from each of these methods. We will talk about the parameters that we will potentially use to grade our model for its efficiency. We will discuss the results and the plausibility of the practical application of these models. Lastly, we will propose the future work that can be done to add onto the project to that we the efficiency of the models can be improved upon.

## Background:

The game of football is played between 2 teams. At any point of time, there’s one offensive team and one defensive team. The goal of the game is to drive the football down the length of the field from your (offensive teams) half to the defensive teams endzone for the touchdown and score points. The team with the greatest number of points will win the game. Each touchdown will award the offensive team 6 points. After a touchdown the offensive team has the option of either going for an extra point play or a 2-point attempt. In the extra point play, the kicker of the offensive team will try to kick the ball through the goal posts and get an additional point for his team. For a two point attempt the offensive team will try to score another touchdown from within the 5-yard line to get 2 points and they have one attempt at both these plays.

The offensive team is given a set of 4 “down” (aka chances) within which they have to drive the ball forward by 10 yards. If the teams successfully complete this, then they are awarded 4 more downs and so on till they manage to score. Usually, most teams try to make plays (either passor run plays) to get the ball through, but if the defensive teams stop them on the first 4 downs, the offensive team has 3 options: they can punt the ball to the defense team, they can try to go for the first down at the risk of failing and giving the opposite team a good starting position or they can kick a field goal and score 3 points if they make it.

The game is divided into 2 halves and each half is divided into 2 quarters of 15 minutes each. At the end of this time, the game will be paused (in case the quarter has ended), or the ball will be kicked again (in case the half has ended) or the game will end and no more plays will be allowed. Thus, time management plays an important role when it comes to deciding which play to run and how to get more points quickly.

The offensive team consists of a quarterback who gets the ball at the start of the play and then he gets to make a decision: he can either pass the ball (which means he can throw it to another teammate of his who will then run the ball as far as he can before the defense catches upto him and tackles him) or he can run the ball (which means he will hand the ball over to another player who will then run with the ball and try to get as far as possible before being tackled. The are other plays the quarterback does is taking a knee or spiking the ball. We won't be discussing these plays because we will not consider them in our analysis.

At the end of each down, the offensive team gets 40 seconds when they can huddle and decide which play to run next and then take positions in any formation they choose before the countdown has ended. It is within these 40 seconds that the defense should ideally predict the play the offense will run and make necessary changes to their formation too. Thus, any predictive model, that can be used in this setting needs to work in a way that delivers the output within 40 seconds so that it can be used by the defense.

An additional situation that can arise are the case when the quarterback, throws the ball and then instead of his teammate, the opposite team player catches the ball (aka interception). This is a turnover, and these situations will not be considered in our analysis because we do not know if that play would have worked or not. All penalties (both from the offensive team and the defensive team) result in addition or subtraction of yards and these will not be considered in our analysis too because these render the entire play dead. Also if during the play the defense reaches the quarterback and then sacks him, the down will be counted, but we will to consider these situations in our analysis because we do not know if the pay was going to be a run or a pass.

## Literature review:

*Paper 1: Football Play Type Prediction and Tendency Analysis [9]*

The premise of this paper was to underscore how important it is to recognize the tendencies of teams to make a certain play in a certain situation and then come up with plans to counter them. The author made explained certain technical terms related to football like "down", "distance", "score differential" and "field position". These will be the key data attributes he used in making his predictive model. The paper discusses how important it is to make a "feature" vectors of values that are normalized between 0-1 to avoid scaling bias. Using this format of the data, they partitioned the dataset into training and test and carried out 2 different forms of predictions: aggregate and situational. The aggregate analysis tried to predict tendencies in the NFL as a whole. For example, which kind of plays are more favoured by "downs" number in the NFL in general. They used a neural network to predict the results of this analysis and also made a baseline model for comparison. Their model was better than the baseline by at least 10% score. The researcher also reported an interview with the MIT football team coach about this model and

its possible uses. The coach's perspective was that while an aggregate analysis can greatly help reduce the load on the manpower, it would be more useful to the coach if the model could predict on a more situational level. Finetuning the data used to specific teams and their plays in specific situation is comes closest to a coach seeing video for preparation against that team. A play would be influenced by the yardage remaining, or by the time remaining or by the field position. Using these attributes for a situational analysis, the researcher was able to distinguish between the tendency (which is the general assumption) and predictability (which is most favoured). They used average accuracy as the metric to judge their models with the baseline/naïve model. They came up with 3 logistic regression models: based on downs, based on field position and based on downs and distance and in all three models the average accuracy surpasses the naïve model by 4%. The paper recognizes that while the increase in accuracy is significant, there is room for further improvement by incorporating more data and also by perfecting the encoding process for the data. By which they mean that the normalization of the data done at the very beginning could be done in ways that would make the predictions better. Also modelling based on player capability and role would provide better insight than the situational analysis because certain players are better in some situations than others.

*Paper 2: Predicting plays in the National Football League [10]*

This paper builds off on the earlier research done in predicting NFL plays including Paper-1. The researchers here highlight that previous papers while focusing on coming up with models with good accuracy do not pay attention to the false negative parameter of their results. The paper reasons that the false negative would be the case where the model predicts a run instead of a pass and the defense that would be set up for a pass can also be used to stop a run but not vice-versa. The researchers try to come up with a model that would reflect their belief in false negative playing a bigger role in predictions. The unique feature of this paper is that it tries to incorporate player ratings from Madden in its prediction model. This, they reason, would eliminate the bias that would be present in the model which does not account for how good or bad the current roster is. A good team can pull off a wrong prediction and a bad team cannot pull off the correct prediction. The first part of the paper deals with using various complex models like neural networks, k-nearest neighbours, random forests and classification trees to get the model that will give them the highest prediction accuracy and the lowest false negative rate. The reason for doing this is to verify that the data they are using can provide the results that the previous research provides and also to setup a baseline for a simpler model they wish to achieve which they feel would be practical and effective enough to be used in actual games. After running multiple analysis, they settle on a neural network model that provides 75% accuracy and false negative rate of about 10%. The second part of their paper discusses the importance to have a simpler, faster and practical model that will be able to predict plays on the fly during the game. Preferably, within the first 25 seconds of the 40 second time intervals between plays. It should also be an interpretable and simple model whose workings could be understood by the coach and would not rely on technology that is prohibited by NFL to have on the field. In the tradeoff between simplicity and accuracy, they finalized upon a decision tree model with a 65% accuracy which corresponds to an 85% accuracy on their baseline neural network. This tree uses only 3 variables: "down", "yards to go" and "point difference". A possible extrapolation of this tree to specific teams is also discussed in the paper. When facing a specific team, we can use a new attribute of "formation" and predict accurately with an average accuracy of 73%. Indeed, this would give the coaching staff a veritable arsenal of analytics they could run during the game to

predict plays. However, the researchers say that nothing can replace the coach's experience and intuition and that this tool can should be used to inform the decisions. They say that in the future, ways of incorporating the weather, injury stats etc. can make this model more robust.

*Paper 3: NFL Play Prediction [8]*

The researcher follows the same objective in the previous two papers. They too want to identify the optimum way of selecting the attributes which when used in the prediction would give the best accuracy. The paper revolves around the possibility of finding this holy grail of a model. The focus of this paper is not to produce a practical model that will be used in the actual game, it focusses more on comparing and contrasting different models. One of the most interesting things this paper discusses is the encoding issue the researchers talked about in Paper-1. In order to convert/encode the categorical data attribute like name of the team, the paper proposes to introduce a 32-bit binary number which would have 31 0's and 1 one. Based on the position of the 1, the name of the team will be determined. They also believe that recall, precision and accuracy all the 3 have to compared to judge the best model. Judging solely by accuracy will not help us address the false positives that will be predicted by the model. The paper suggests using the F1 score which takes into consideration both recall and precision. They also have a unique take on the data output which happens through a function where the number of the current down, the number of yards gained in the play and the number of yards needed for a first down/touchdown are used. They then go on to perform analysis through several complex models such as classification trees, regression trees, Nearest centroid, linear discriminant analysis, support vector machine and neural networks. The performance of these models was evaluated based on their accuracy, precision and recall.

## Approach:

In this section, I have detailed the entire process from start to end of performing analytics on fro predicting NFL plays. It has 3 sections:

A) ***Data Source***: Initially, I had selected the data set from nflsavant.com which included plays from the seasons 2015 till 2019. It had almost over 40000 records in it and was available in .csv format. However, after studying the data, I decided not to use this because it did not have certain key attributes which I felt would be very important in the predictive analysis. Since the nature of the plays depends on the time remaining in the game, half or quarter, it is important for us to have these attributes too in our dataset. These were absent in this dataset. The new dataset that I used is a popular NFL dataset created by Max Horowitz in 2016 and hosted on Kaggle. This includes almost 255 attributes and has the plays from the seasons 2009 to 2018. This dataset is massive and has the following info

```
play_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 449371 entries, 0 to 449370
Columns: 255 entries, play_id to defensive_extra_point_conv
dtypes: float64(135), int64(18), object(102)
memory usage: 874.2+ MB
```

B) ***Data cleaning:*** Since the dataset has so many attributes, I had to select a handful of attributes that had very little correlation with each other so that they can independently affect the outcome of the model. They are as follows:

```
play_df.columns
```

```
Index(['game_id', 'posteam', 'defteam', 'yardline_100', 'game_date',
       'quarter_seconds_remaining', 'half_seconds_remaining',
       'game_seconds_remaining', 'qtr', 'down', 'ydstogo', 'play_type',
       'yards_gained', 'shotgun', 'no_huddle', 'score_differential',
       'rush_attempt', 'pass_attempt'],
      dtype='object')
```

The next step of data cleaning required some amount of domain knowledge. I knew that in the NFL rosters change significantly over an average period of 5 years and so it would not be plausible to keep records from early years for our analysis. Thus, I selected the data only from the years 2014 to 2018 to conduct the analysis on.

Also, in the last couple of seasons, several franchises have shifted to new locations so their names/codes in the data may be different, but the team and the staff remain the same. Also, some teams were coded in two different variations of their names. These were the St. Louis Rams which shifted to Los Angeles and the San Diego Chargers that shifted to Los Angeles. Bothe their names were renames from 'STR' to 'LAR' and 'SDC' to LAC' respectively. Jacksonville jaguars were labelled as both 'JAC' and 'JAG' so we picked one and changed all the other's instances to it.

After this step, I addressed the "`play_type`" attribute in the data. Since we are only concerned about the plays of the types: "run" or "pass", I removed all the other kinds of `play_types` in the data. This may include plays like 'kickoff', 'punt', 'extra point', 'field goal' etc. For sake of validating the data cleaning process so far, I compared the '`play_type`' column values to the '`rush_attempt`' and '`pass_attempt`' columns. If the play was a run play, then the corresponding value in the '`rush_attempt`' column would be 0 and the value in the '`pass_attempt`' column will be 1. The reverse is true for when the play is a pass play.

Now, I checked for missing values in the data and in which column they existed. If the missing values would be in the colmn containing important categorical data such as "`down`" or "`posteam`", then the record cannot be kept for analysis. But if the null values were in columns like "`pass_attempt`", or "`rush_attempt`" (columns that are used for the validation purposes, then we can keep those records since those columns will not be used in the data analysis anyway. Counting the null values in the dataset, we got this result:

```
play_df.isna().sum()
```

```
game_id                       0
posteam                       0
defteam                       0
yardline_100                  0
game_date                     0
quarter_seconds_remaining     0
half_seconds_remaining       17
game_seconds_remaining       14
qtr                           0
down                        430
ydstogo                       0
play_type                     0
yards_gained                  0
shotgun                       0
no_huddle                     0
score_differential            0
rush_attempt                  0
pass_attempt                  0
dtype: int64
```

At his point, we have around 159,000 records and of these records, 430 have null values for "`down`" and of those 430 values, 17 and 14 have null values for `half_seconds_remaining` and `game_seconds_remaining`. Since "`downs`" is a categorical attribute that we cannot replace/substitute by performing any statistical operations on the rest of the values in the column, we have to discard these records as well. Lucky for us, the number of null value rows (430) are very small than the total records we have (159,000). Once we drop the records, we have a dataset of the shape (158158,18).

C) ***Data engineering:*** At this step of preprocessing phase, we have a dataset of relevant attributes without any null values. However, there are some attributes that have to be used to obtain better attributes.

In the game of football, sometimes the offensive team just wishes to get within a certain distance from where they can kick a field goal and get easy 3 points. They also may accordingly, so to account for this scenario, I engineered a new attribute called "`field_goal_range`". They attribute "`yardline_100`" was used to generate this attribute.
If the `yardline_100` value was above 45 yards the corresponding value in the "`field_goal_range`" is 4. If the yards were between 45 and 35, the corresponding value is 3. If the yards were between 25 and 35, then the corresponding value is 2. For anything lesser than 25, the value will be 1.

Focusing on the kind of plays called between consecutive first downs, we see a pattern in the kind of plays that are called at certain distances from the first down. One way to rationalize this is that the play called at a distance of 1 yard to go and at a distance of 2 yards will be similar. Also, the play called when the first down line is 20 yards away is similar to the play called at a distance of 18 yards. Thus, we can group this yardage into

different ranges, where similar plays will be called for yards belonging to the same range (high probability of them being similar). A new attribute called '`first_down_dis`' is a categorical attribute that has the values corresponding to the ranges of distance. The ranges are as follows. If the yards to first down are less than 3; the corresponding value in the new column is 1. If the yards are between 4 and 10 yards, then the corresponding value is 2. If the yards are between 15 and 10 yards, then the corresponding value is 3. Anything larger than 15 comes under category 4.

Last step of pre-processing is to encode the categorical data attributes of "`posteam`", "`defteam`" and `play_type`. For this step we used the LabelEncoder() method of the sklearn.preprocessing library. The run plays were encoded 0 and the pass plays were encoded as 1. All the 32 teams were encoded by numbers between 1 to 32. This integer value in needed especially when the matrix multiplications take place in predictive models like neural networks. At this final step we have a data frame of (158158,23) dimensions. The data is stored into another .csv file so that it can be loaded from there directly for future applications/models etc. The correlation matrix for the data after all pre-processing steps looks like this.

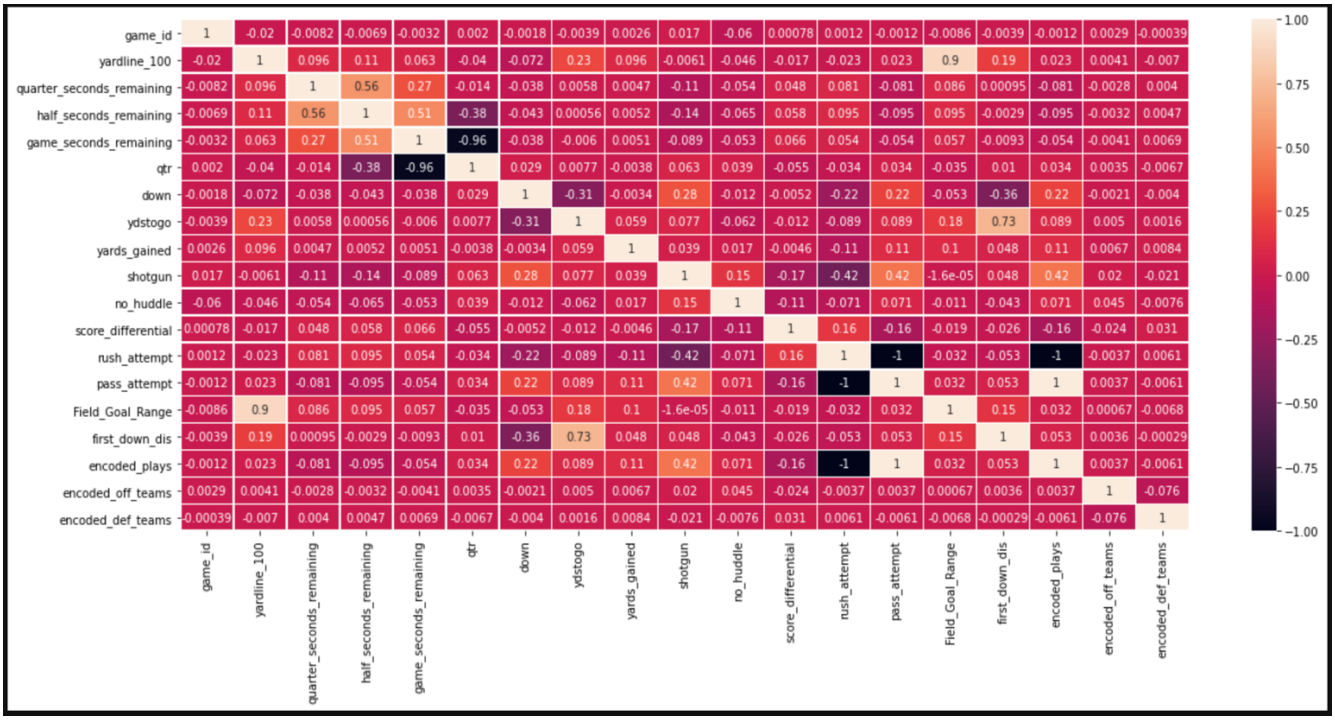


*Figure 1 Correlation matrix*

D) ***Analysis:*** Using this data, we can perform 2 different types of analysis. The first one is purely statistical analysis which is done by studying the occurrences of specific situations in our data. This can also be used as a baseline for our second type of analysis: predictive analysis. In this analysis we try to harness different predictive models like: Logistic Regression, Logistic Regression with Ridge classifier, neural networks, KNN classifiers,

random forest classifier and gradient boosting classifier. The performance of these is compared amongst each other and also with a baseline so that we can see improvements in our models.

## Models (Description and results)

There will be two parts to this section.

### *A) Tendency analysis:*

Since the NFL has been around for almost 100 years, there are going to be tendencies that are apparent across the league. Some plays in some situations are going to be played and everyone is aware of them being played. This is because people have seen those plays being called for so long, there is nothing predictable about them.
This logic can be applied to certain teams and their tendencies to play in certain kinds of situations. Bear in mind that different editions of the same team will have different play-calling tendencies, and this is why we have collected the latest data ranging over 5 years so that this difference is minimal. For a good defensive coordinator, this information will be a confirmation of his own knowledge but for someone who wishes to understand how the league defensive strategies are being called, this provides valuable insight.

The first result that I collected was how many total run plays and pass plays have been called between 2014 and 2018 seasons. The graph is as shown below.

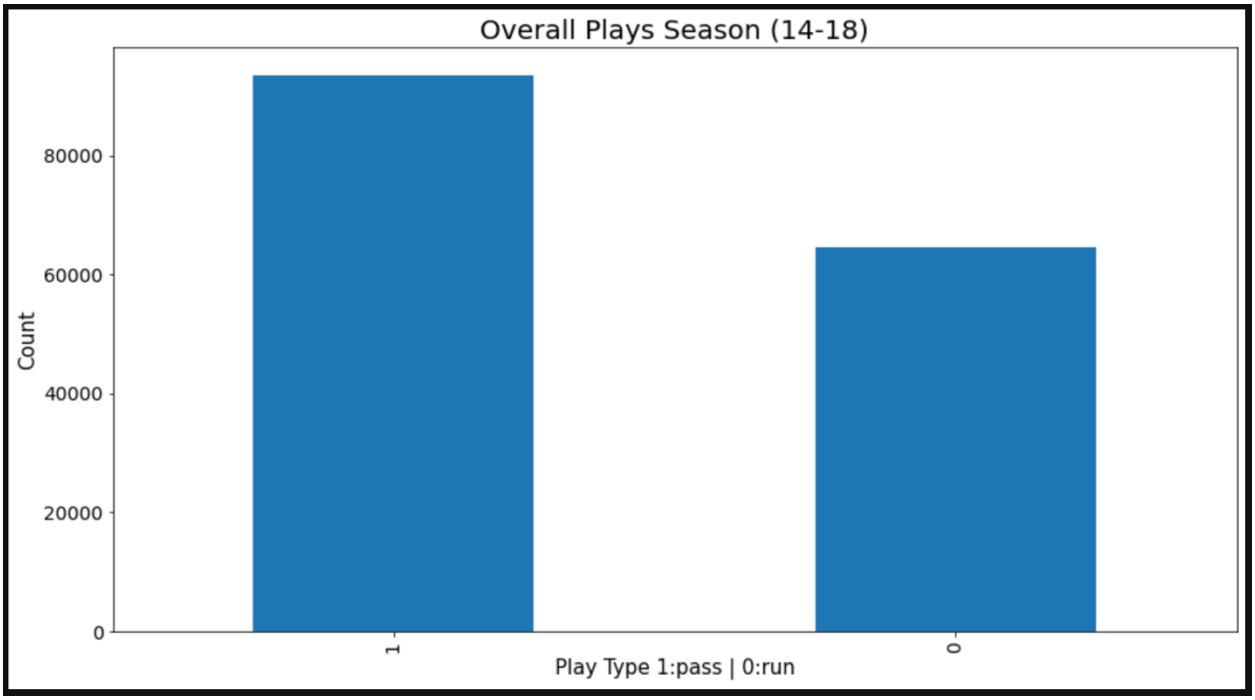


*Figure 2 Pass vs. Run (League stats)*

Statistically, the pass plays are called 59.1% of the time and run plays are called 40.9% of the time. This is considered a tendency of the league in general to favor more passing plays than run plays.

The second result that I considered was performing a similar comparison between the run plays and the pass plays but separated by the "`down`" they were called in. By this we go a little deeper in this statistical analysis by getting the probability of a run or pass play by down so this can be considered as a summary of how to play the game when just taking the down into comparison. The result obtained was as follows.

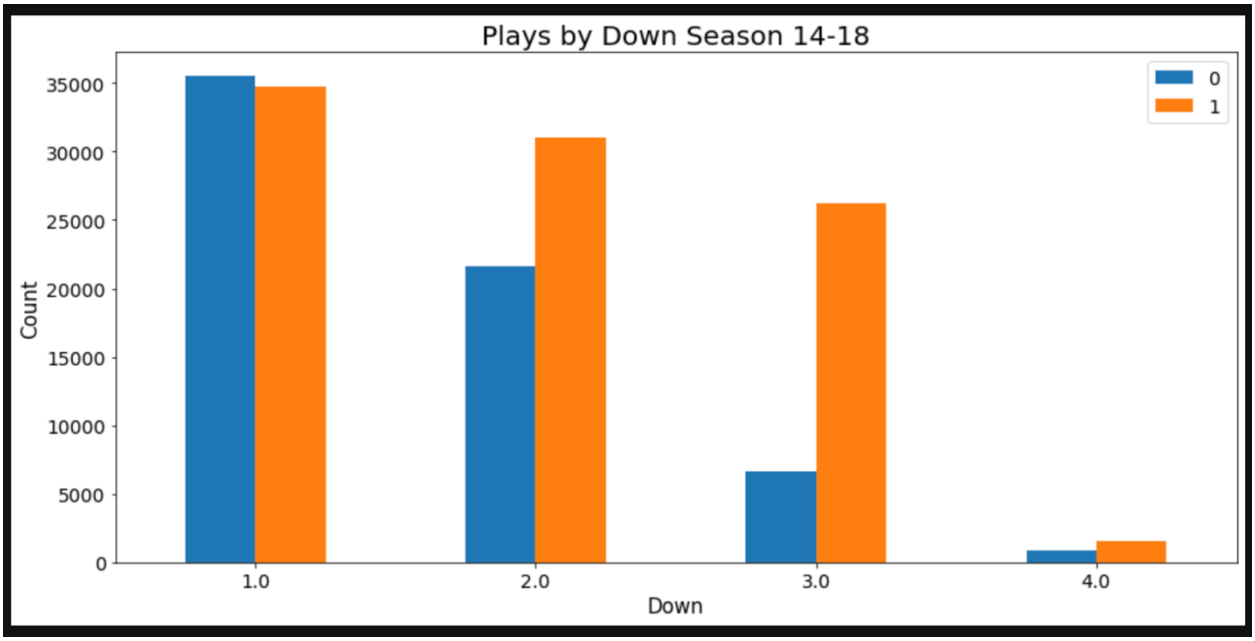


*Figure 3 Plays by the "down"*

As you can see, the number of records that correspond to first downs are more than other downs because that's the starting position every time. One important observation they can be made is that teams are slightly more inclined to run the ball than pass because this makes sure that you can trick the defense who may be more spaced out (expecting a pass) and gain some yards from where a pass play may be more successful in converting to a first down. Also, the number of plays in fourth down are so less because usually teams, after third down, either kick a field goal or punt the ball instead of taking a risk of giving the opposition team a better field starting position.

The third result was a deeper dive into my above hypothesis about first downs. I added the "`first_down_distance`" range to the analysis and broke up plays based on downs and this range. The result was as shown in the image below.

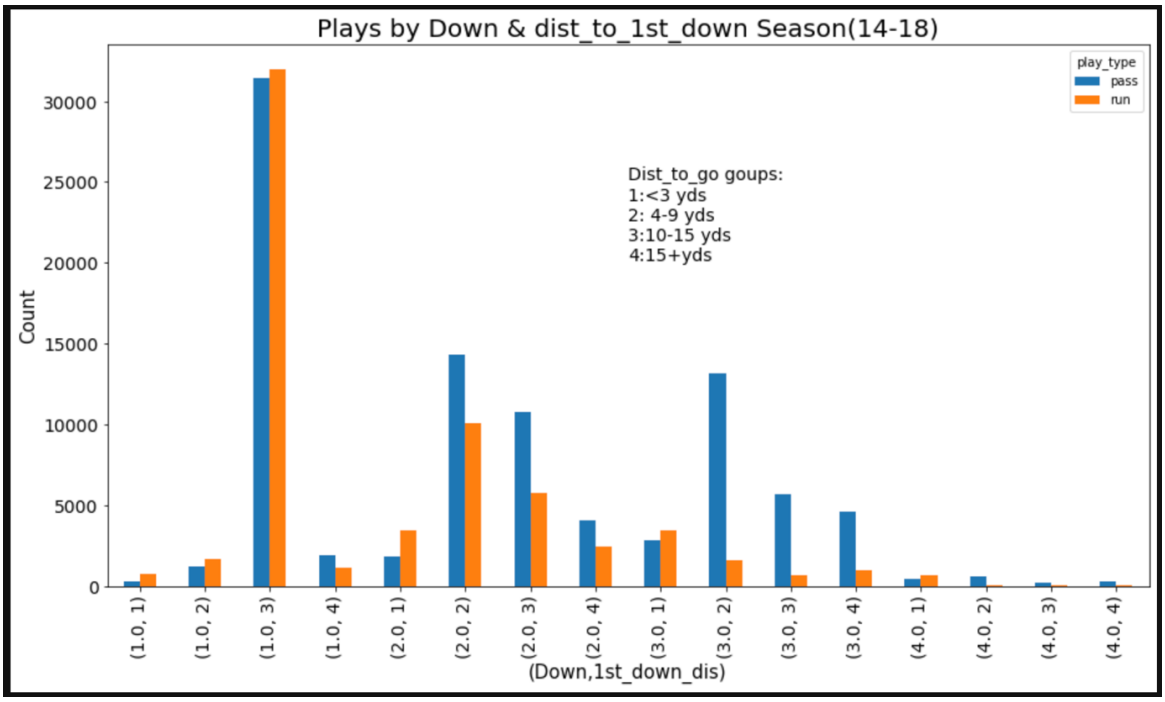


*Figure 4 Plays by down and dist. to first down*

Several observations can be made from this graph. Firstly, since the starting positions are usually first down and 10 yards to next first down, the (1.0,3) value has so many values. Also, it shows that the run plays are indeed preferred to pass plays at the starting position. Rare cases include a first down and less yardage. In that case (which is usually in the endzone), to avoid the risk of getting intercepted and get the touchdown for sure, a run play was preferred. Similarly, on the other end of the graph, any time a play is being called in the fourth down we know that the offensive team is really desperate to get a first down/ touch down. Across all the downs, we observe that if the yards to first down are very less, a run play is called because this will get them the first down for sure without running the risk of getting intercepted or sacked. As seen if its third down and the distance to the first down is 10 or more yards, almost certainly a pass play will be called because the team would rather throw the ball over the yards required, then run a ball with a player and risk him getting tackled before reaching first down thus sending the team to fourth down.

A tabular representation of this result showing the percentage probability of a play type according to down and distance to down is as shown below. This is the tendency of play-calling of the entire league in general.

| down | yds_range | pass_call_percent | run_call_percent |
|---|---|---|---|
| 1.0 | <3 | 28.57% | 71.43% |
| | 4-9 | 40.91% | 59.09% |
| | 10-15 | 49.63% | 50.37% |
| | 20+ | 62.30% | 37.70% |
| 2.0 | <3 | 34.57% | 65.43% |
| | 4-9 | 58.78% | 41.22% |
| | 10-15 | 65.31% | 34.69% |
| | 20+ | 62.56% | 37.44% |
| 3.0 | <3 | 44.84% | 55.16% |
| | 4-9 | 89.14% | 10.86% |
| | 10-15 | 89.75% | 10.25% |
| | 20+ | 82.50% | 17.50% |
| 4.0 | <3 | 39.46% | 60.54% |
| | 4-9 | 88.96% | 11.04% |
| | 10-15 | 91.05% | 8.95% |
| | 20+ | 89.94% | 10.06% |

Similar analysis can be done team-wise. This is useful when you are facing a particular team and want to know what this teams' tendencies in certain situation is. Below are the tables for my 2 favorite teams: Green Bay on the left and Kansas City on the right.

GB

| down | yds_range | pass_call_percent | run_call_percent |
|---|---|---|---|
| 1.0 | <3 | 0.470588 | 0.529412 |
| | 4-9 | 0.531250 | 0.468750 |
| | 10-15 | 0.532762 | 0.467238 |
| | 20+ | 0.768293 | 0.231707 |
| 2.0 | <3 | 0.437838 | 0.562162 |
| | 4-9 | 0.590674 | 0.409326 |
| | 10-15 | 0.681188 | 0.318812 |
| | 20+ | 0.659794 | 0.340206 |
| 3.0 | <3 | 0.516279 | 0.483721 |
| | 4-9 | 0.880734 | 0.119266 |
| | 10-15 | 0.904762 | 0.095238 |
| | 20+ | 0.828025 | 0.171975 |
| 4.0 | <3 | 0.341463 | 0.658537 |
| | 4-9 | 0.935484 | 0.064516 |
| | 10-15 | NaN | NaN |
| | 20+ | 0.750000 | 0.250000 |

KC

| down | yds_range | pass_call_percent | run_call_percent |
|---|---|---|---|
| 1.0 | <3 | 0.291667 | 0.708333 |
| | 4-9 | 0.394231 | 0.605769 |
| | 10-15 | 0.497679 | 0.502321 |
| | 20+ | 0.640449 | 0.359551 |
| 2.0 | <3 | 0.410959 | 0.589041 |
| | 4-9 | 0.602165 | 0.397835 |
| | 10-15 | 0.652977 | 0.347023 |
| | 20+ | 0.596939 | 0.403061 |
| 3.0 | <3 | 0.395604 | 0.604396 |
| | 4-9 | 0.843891 | 0.156109 |
| | 10-15 | 0.850829 | 0.149171 |
| | 20+ | 0.828402 | 0.171598 |
| 4.0 | <3 | 0.300000 | 0.700000 |
| | 4-9 | 0.692308 | 0.307692 |
| | 10-15 | 0.666667 | 0.333333 |
| | 20+ | NaN | NaN |

## *B) Predictive analysis*

In this section we make predictive models and try to finetune them so that they can give a better performance than a decided baseline model. We also compare these models with one another to decide which will be the ideal model to be used in practice.

Baseline model: The baseline model for our project would be one which would predict a pass play every time. Since on an average the pass play is 59.1% probable while the run play is just 40.9% probable. This baseline model has more than 50% accuracy and so we can consider this a good baseline.

Before the predictions, I used the train_test_split module to create the train set and the test set to use in these models. The test set was 20% of the total data. I also dropped several attributes such as the "`posteam`", "`defteam`", "`play_type`" etc. which were encoded into different attributes or were converted to other attributes. Lastly both the x_train and the x_test datasets were standardized by the StandardScalar module of the preprocessing library since we are feeding the data in the form of vectors into the models and since all the data is integers, we can standardize it to values that can be easily computed on by the models.

I considered 6 different predictive models. For each of the models listed below, we first created a normal classifier with all default parameters to check if the data could be used in the model. After this verification, we used the RandomSearchCV() library to take in a dictionary of parameters that can be fed into the model and train the model so that it can decide which combination of the parameters will give the best precision and recall. This type of cross validation is performed because its runtime is faster than traditional GridSearchCV although it may have more noise. The best estimator obtained is then used to predict on the test and train data and the precision and recall values are compared among all the models.

They are as follows:

1) **Logistic Regression**:
   Logistic regression is the first choice for this type of problem because we have essentially a binary classification problem and the performance for a logistic regression is best for such a problem. Using the LogisticRegression() module in sklearn we were able to create the model and predict the output.
   The final best model estimated, and its accuracy is as follows. The max_iter parameter sets the number of iteration the solver takes to converge. The solver in this case is the optimization algorithm that is used for the regression.

```
y_pred = clf.predict(x_test)
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.67      0.61      0.64     12840
           1       0.75      0.79      0.77     18792

    accuracy                           0.72     31632
   macro avg       0.71      0.70      0.71     31632
weighted avg       0.72      0.72      0.72     31632
```

```
clf.best_estimator_
```

```
LogisticRegression(max_iter=10000, solver='newton-cg')
```

2) **Logistic Regression with Ridge classifier**:
The ridge classifier was used here because the regularization brought by this classifier might help avoid overfitting in the case of using just Logistic Regression. The additional parameter we have to assign to the model is "alpha" which determines how aggressively regularization is being done. We included a number of possible alpha values and carried out RandomSearchCV. The best performing model obtained was this.

```
y_pred = clf.predict(x_test)
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.67      0.62      0.65     12840
           1       0.75      0.79      0.77     18792

    accuracy                           0.72     31632
   macro avg       0.71      0.71      0.71     31632
weighted avg       0.72      0.72      0.72     31632
```

```
clf.best_estimator_
```

```
RidgeClassifier(alpha=5.0, max_iter=20000, solver='sag')
```

3) **Neural Network**:
The most exciting model would definitely be the Neural network, not because of its performance but because of the way it is architected and the flexibility it provides us in its structure. Neural networks are very useful in classification problems. They have a series of hidden layers between the input layer and the final output layer. For finding the best model, I selected 3 architectures: (20,50,30), (50,100,50) and (28,50,30,14). The activation function used is ReLU and I also decided to give 4 different values of alphas and 3 different batch sizes. A batch size is the set of data that is sent into the model at each epoch. For the model, the learning rate of the model was set to adaptive, which means that the model would decide the best learning rate by itself. Given so many hyperparameters, the training time is significantly more (about 11.2 mins). The results are as follows.

```
y_pred = clf.predict(x_test)
print(classification_report(y_test,y_pred))
clf.best_estimator_
```

```
              precision    recall  f1-score   support

           0       0.66      0.65      0.66     12879
           1       0.76      0.78      0.77     18753

    accuracy                           0.72     31632
   macro avg       0.71      0.71      0.71     31632
weighted avg       0.72      0.72      0.72     31632

MLPClassifier(batch_size=1000, hidden_layer_sizes=(50, 100, 50), solver='sgd')
```

4) **KNN classifier:**

This classifier compares the distance of the datapoint with k nearest points. The hyperparameter that can be controlled are the value of k (I have set the values as 5, 10 and 15). The time taken for the training the classifier is second to neural networks (5.2 mins). The results and final model are as follows.

```
y_pred = clf.predict(x_test)
print(classification_report(y_test,y_pred))
clf.best_estimator_
```

```
              precision    recall  f1-score   support

           0       0.62      0.59      0.61     12879
           1       0.73      0.76      0.74     18753

    accuracy                           0.69     31632
   macro avg       0.68      0.67      0.67     31632
weighted avg       0.69      0.69      0.69     31632

KNeighborsClassifier(algorithm='ball_tree', weights='distance')
```

5) **Random Forest classifier:**

I preferred to use random forest classifier over the decision tree classifier because of the bagging nature of random forests, the variance is low. Also, the training of the model can be done in parallel and the predictions are faster than traditional trees. Since this is an average of all different kinds of trees, this model has a low bias and a low variance. In our model the number of estimators (trees in the forest) were considered as parameters to optimize our model, so we tried 50, 100 and 150 trees. Another parameter we could optimize on was the max_depth of each of these trees, so we tried using 10, 15 and 20 as the depths of these trees. The trees were using entropy as the criterion to split further. The result that we got from the classifier is as shown in the image below. We can also get a comparision about which feature held the most importance in splitting the tree such that the information gain was maximum. This gives us an insight of which attribute the model considered important (not to be confused with the attribute historical/statistical data deem important). The graph of feature importances is as shown below. As you can see, the “shotgun” feature is considered very important. SO if the offensive team is in a shotgun formation or not

in a certain situation will most definitely tell you what play will be called in that situation.

```
y_pred = clf.predict(x_test)
print(classification_report(y_test,y_pred))
clf.best_estimator_
```

```
              precision    recall  f1-score   support

           0       0.67      0.64      0.65     12879
           1       0.76      0.79      0.77     18753

    accuracy                           0.73     31632
   macro avg       0.72      0.71      0.71     31632
weighted avg       0.72      0.73      0.72     31632

RandomForestClassifier(criterion='entropy', max_depth=10, n_estimators=150)
```

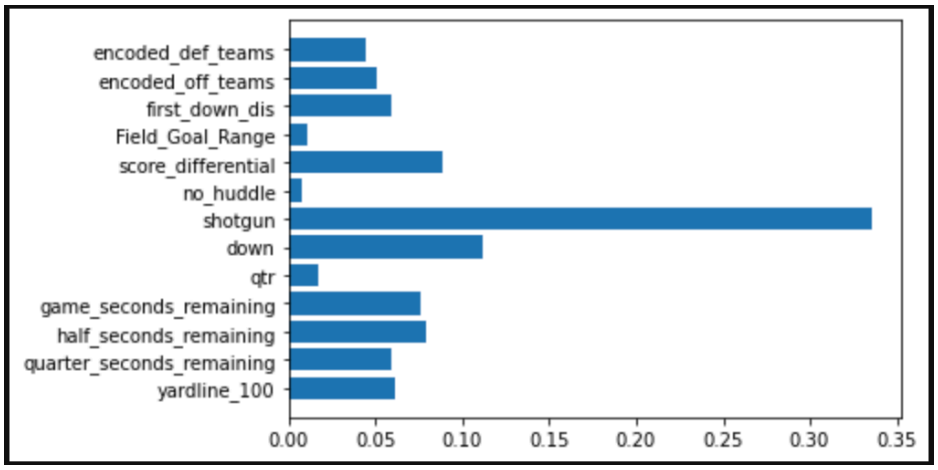


*Figure 5 feature importance graph for random forest*

6) **Gradient Boosting Classifier:**

This is the classifier that is also one of the most powerful classification models because of the fact that it consists of a loss function that is improved on regularly. Unlike random forest where the trees are separated randomly and finally added together, in this the trees are added simultaneously as we make the model, and a gradient descent metric is used to measure the loss when the trees are getting added together. The parameters that I included to make the best model from include number of estimators (number of trees that are added), the loss function to be used, the learning rate of the model and the maximum depth of the trees generated. The results of this model are as follows:

```
y_pred = clf.predict(x_test)
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.66      0.64      0.65     12879
           1       0.76      0.78      0.77     18753

    accuracy                           0.72     31632
   macro avg       0.71      0.71      0.71     31632
weighted avg       0.72      0.72      0.72     31632
```

```
clf.best_estimator_
```

```
GradientBoostingClassifier(learning_rate=1.0, loss='exponential', max_depth=5,
                           n_estimators=50, random_state=5)
```

## Conclusion and discussion

Since all these predictive models were supposed to perform better than the baseline (59.1% accuracy, all these models were successful. However, it is important to consider exactly which metrics are most important and why.

Precision is important because we want to know how many correct predictions are being made by the model. This is generally the most important criteria to judge the model by. Speaking strictly on the basis of precision, the best model we have would be Random Forest classifier which has an overall accuracy of 73%. The worst model in this would be the KNN classifier. This is surprising because the KNN model took longer than most models to train but still has lesser accuracy. One possible reason for this to exist is that we have equally weighted all the attributes of the data. Perhaps if the weights given to each attribute would have been different, then we could have seen the classifier converge better and have a higher accuracy score.

In the case of football, I think that recall plays a more important role than accuracy because of the implications it has defense wise. For example, if the defensive coordinator is confident in his team's defense to stop all running plays but cannot stop pass plays unless they are prepared for it, he will need the model to predict the pass plays better than the run plays. Similarly, on the other hand if the defense can adapt well to a pass play but cannot stop runs, they will need the model to predict runs more accurately than pass plays. This is where the false negative metric of the prediction comes into play. We want the recall to be as high as possible so that the model can be used in special need conditions too. Thus, during the RandomizedSearchCV, the scoring of the different model was done on the basis if recall. As I found out that there would be a trade-off in the accuracy and the desired recall if we want the model to be weighing more on preventing false negatives. In our project, the neural network provides a marginally better recall score than the random forest classifier and even though its accuracy is lesser than the random forest, I would recommend the neural network to be used.

## Future Work

In the future, many modification and improvements can be made to these models. We can finetune the hyperparameters more depending on the computing resources you have at hand. In one of the literature reviews, they had mentioned including team player data into the analysis because who plays in the team makes a major difference especially if the player is a MVP level player like Patrick Mahomes etc. To do this they planned to either scrap the NFL official website to get player data and then decide which features of it are most significant. Another option they mentioned was to use the Madden video game player rankings since the player rankings are constructed by all this data mined and are a ready-made summary/aggregate of the data from the official data source.

## References:


1) https://www.thespax.com/nfl/predicting-nfl-offensive-play-calling-with-python/

2) https://www.thinkful.com/blog/dont-bet-against-this-data-science-student/

3) https://sites.northwestern.edu/msia/2020/01/31/nfl-tendency-analysis-and-basic-play-type-prediction/

4) https://rahuljain28.github.io/NFLPredictions/

5) https://rstudio-pubs-static.s3.amazonaws.com/234204_5df1d9fac39247b3bf884a5480d1d1c9.html

6) http://nflsavant.com/about.php

7) https://www.nfl.com/news/nfl-salary-cap-will-increase-to-198-2m-in-2020-0ap3000001106260#:~:text=In%20the%20aftermath%20of%20NFL,sources%20informed%20of%20the%20situation.

8) Teich, Brendan, Roman Lutz, and Valentin Kassarnig. "NFL Play Prediction." arXiv preprint arXiv:1601.00574 (2016).

9) Ota, Karson L. *Football play type prediction and tendency analysis*. Diss. Massachusetts Institute of Technology, 2017.

10) Joash Fernandes, Craig, et al. "Predicting plays in the National Football League." *Journal of Sports Analytics* Preprint (2019): 1-9.