Predicting NFL Game Outcomes

Sunil Sabnis, Joseph Iovine, Abhinav Girish

December 12, 2022

Abstract

Whether it's for beating Vegas, gaining respect as an NFL analyst, or bragging rights with a friend, for decades people have been trying to accurately predict NFL game outcomes. With the fairly recent introduction of high quality and easily accessible data, it was only logical to utilize machine learning for this problem. Using a logistic regression model, a random forest, and a neural network for NFL game data over the past decade we were able to accurately predict game outcomes 67.03%, 64.43%, and 66.76% of the time, respectively.

1 Introduction

The incorporation of advanced statistical analysis in the NFL has been a long debated topic. The older generations of coaches, players and fans think you can't capture the essence of the game in numbers, nonetheless effectively use that data to aid a team's strategy or describe a player's performance in context of a specific game. In 2017 the NFL introduced Next Gen Stats. Utilizing 100s of RFID tags and ultra-wide band receivers, this tracking system provides the raw data that is necessary to calculate performance metrics and derive advanced statistics through machine learning. These metrics come with a fair share of criticism (mostly from people who don't understand the process) and the question remains: is this stuff useful? One way to convince a non-technical audience that machine learning has a place in professional sports is to present it in layman's terms. We will be discussing how to predict the outcome of NFL matchups using play-by-play data and advanced team/player metrics collected over the course of several seasons.

2 Context

We'll start with some background information in the advanced statistics that are most commonly used by NFL stats analysts: EPA and EPA/play. Respectively these stand for expected points added and expected points added per play and are the holy grail for evaluating NFL offenses and defenses with proper context. The theory behind EPA is simple, try and quantify how much a specific player or play is worth to a team. When looking at EPA we must break it up into two categories: team and individual. From a team's perspective, EPA looks at the body of work through a play-by-play lens and while the actual outcome of a play is important, context is just as important. For example, if the situation is third down with three yards to go for a first down, a play that gains four yards will add more EPA opposed to a situation where it is third down with six yards to go and now the offense must punt because they fell short of the first down marker. Similarly, for an individual EPA is affected by plays where they are involved and is most heavily used when evaluating a quarterback. EPA for a team and EPA/play for a quarterback are two of the most commonly used statistics when rating a team's overall performance.

A body of work that we initially referenced was *Intro to NFL game modeling in Python* by Ben Dominguez. In the article, a logistic regression model was used containing eight features, all based on different EPA/play metrics for both the home and away teams. We aimed to improve on two things from this model: utilizing a more complex classification algorithm, and incorporating more human elements. For the former, we opted for a random forest classification and neural network in

addition to the logistic regression model. For the latter, as great as EPA is, it does have its flaws: it does not separate players performances from others, it does not adjust for opponents, and it does not account for luck and other randomness. We attempted to fix this by substituting Elo ratings for EPA. Elo ratings are a simple system for ranking a team based on performance, this includes a mix of advanced statistics such as EPA, along with the more random factors that cannot be quantified as easily and require human analysis. Theoretically this should fix the issues with EPA that were listed above. The results for the random forest and neural network classification will be discussed later, but for the logistic regression model the article reported an accuracy of 63.5%, whereas our model had an average accuracy of 67.03%. This is a promising result that we plan on improving with a more complex model.

3 Method

We decided to use Logistic regression, Random Forest, and Neural Network methods since our project deals with a classification problem - we're essentially trying to decide whether the home team wins or loses in a game given a set of features characteristics of the game.

We utilized Logistic Regression since it seemed like a simple baseline model that could output probabilities for a binary classification like our problem. This baseline could then be used to further evaluate other types of models on the dataset. Before applying the Logistic Regression model (our baseline) we incorporated a few data pre-processing techniques such as normalization using Scikit-learn's MinMaxScaler preprocessing module. This would have made sure that no single feature has excessive influence over the predicted outcome outputted by our classifier. In a separate experiment, we created another type of Logistic Regression model using one-hot encodings of a particular 'playoffs' feature. This was to take advantage of the differing variety of discrete values and ensure that some values are not rated higher than others.

We used a Random Forest algorithm because it didn't require too much hassle in terms of data preparation and thought it would work well with structured data such as what we had. We did not do any additional pre-processing such as normalization for the Random Forest regression due to the way that the algorithm for Decision Trees works. In other words, A decision tree algorithm performs a split on the data due to a specific threshold value for a feature. Normalization of the input does not really affect how the splits in the data occur, and thus it is not necessary.

We used neural networks because we felt this type of model would be complex enough to be able to take into account all the nuances in the features of the data in order to be able to squeeze out greater accuracy.

4 Setup

We join three datasets for our experiment. Namely, we have a dataset on historical weather data for every NFL game since 2000, a dataset containing all NFL games with game_ids since 2009, and a dataset containing football team metrics prior to a game (ELO scores, QB ratings). Each of these datasets required substantial cleaning and preprocessing. I will refer to the all games dataset as games, football metrics as ELO, and weather as weather.

We needed a dataset with NFL game_ids because the game_id field is a key leveraged to join with the table containing weather metrics. The game_id field is a unique identifier for a NFL game. Multiple games are played on Sundays, and as such the game_id field helps distinguish between games. A dataset containing this information over an extended period of time did not exist. We were, however, able to obtain individual CSVs of game data for each season. With a simple script we constructed a file containing all NFL games with game_ids since 2009 (agg_game_history.py).

Within the games dataset, there was some missing data. We dropped any rows where the home or away score was undefined. Furthermore, we had created an additional column for a binary variable to indicate whether or not the home team had won. For each game, we compared the value in the 'home_score' column to the value in the 'away_score' column - if the home score was greater, we set the binary variable to 1, and 0 otherwise.

Another problem we ran into was that the ELO dataset did not have game_ids. Thus, to join the data, we needed to create our own unique key. A NFL game is unique because of the date and the teams

playing against each other; thus, we joined the datasets following this methodology. However, a problem we quickly encountered was that datasets had different team abbreviations. This is due to the fact that NFL teams switch locations and have name changes (e.g., San Diego Chargers → Los Angeles Chargers, Washington Redskins → Washington Commanders). The set difference (what is in ELO and not in games) between the ELO game dataset and games included: 'LAR', 'WSH'. The set difference between the games dataset and ELO included: 'RIC', 'NPR', 'APR', 'CRT', 'SD', 'LA', 'STL', 'WAS', 'SAN', 'JAC'. From the games dataset, we inspected APR, CRT, NPR, RIC, SAN and saw that these teams were 'nonsense' as there were only a couple rows containing these teams and they only played against each other. Rows containing these team abbreviations were dropped. As a next step we capitalized all team abbreviations, and then applied the proper mappings.

To verify that the mappings are 1:1, we computed the set differences again, which were empty as desired. The ELO dataset has a dates column, but the games dataset does not. However, the games dataset has a URL to each historical game (NFL site). We noticed that the URL contains dates within it. We wrote a simple parser to extract the date from the URL and then converted this string into a datetime object. With the date and team name columns defined for each dataset, we joined the two datasets together. After creating this intermediary dataset, we joined it with the weather dataset. This was a more straightforward join because the weather dataset has a game_id field which matches the game_id field from the games dataset. From the weather dataset we only keep the first weather measurement, which happens prior to a game because we are predicting NFL outcomes prior to any play. Our final dataset constructed contains 3207 rows and 20 features. The following table describes the features we chose to use.

Feature	Description
playoff	whether the game was a playoff game or not
elo1_pre	Home team's Elo rating before the game
elo2_pre	Away team's Elo rating before the game
elo_prob1	Home team's probability of winning according to Elo ratings
elo_prob2	Away team's probability of winning according to Elo ratings
qbelo1_pre	Home team's quarterback-adjusted base rating before the game
qbelo2_pre	Away team's quarterback-adjusted base rating before the game
qb1_value_pre	Home starting quarterbacks's raw Elo value before the game
qb2_value_pre	Away starting quarterbacks's raw Elo value before the game
qb1_adj	Home starting quarterbacks's Elo adjustment for the game
qb2_adj	Away starting quarterbacks's Elo adjustment for the game
qbelo_prob1	Home team's probability of winning according to quarterback-adjusted Elo
qbelo_prob2	Away team's probability of winning according to quarterback-adjusted Elo
quality	Rating of game's quality, based on the harmonic mean of the teams' pregame Elo ratings, scaled from 0-100
Temperature	Gameday temperature
DewPoint	Gameday dewpoint
Humidity	Gameday humidity
Precipitation	Gameday precipitation
WindSpeed	Gameday windspeed
Pressure	Gameday Pressure

Table 1: Feature Descriptions

We tested the accuracy of Logistic Regression, Random Forest model, and Neural network model on the dataset we had for the reasons that were described in the previous section. Furthermore, we used cross-validation to get a more realistic average score for each type of model. PCA was applied to the data to see which features had the greatest impact on the predictions of models.

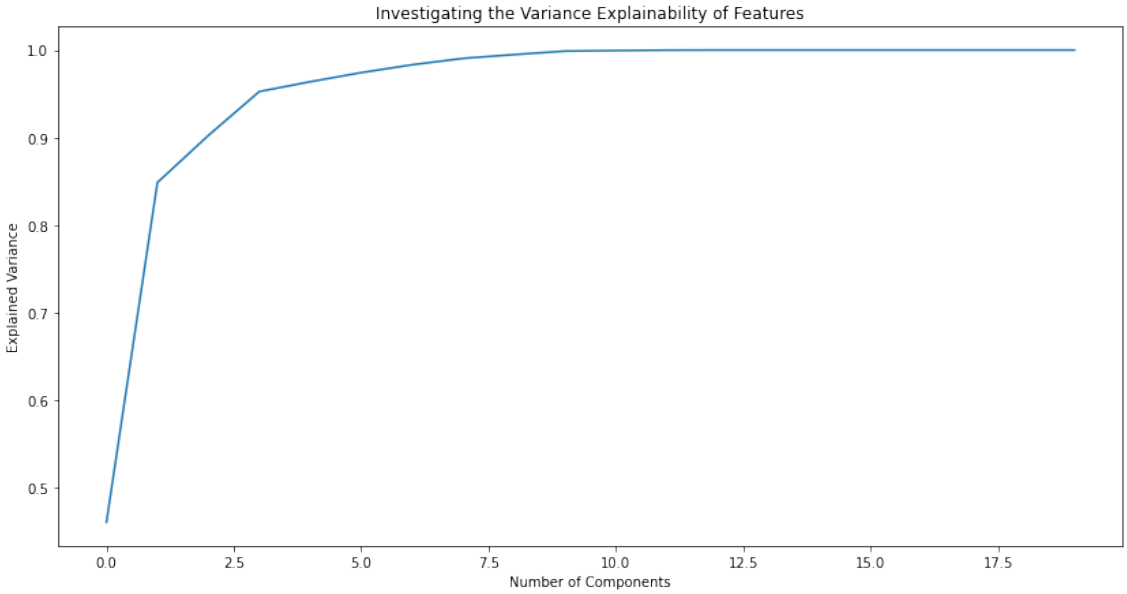
We performed PCA on the dataset to determine which features had been the most important in terms of deciding the outcomes of the football games due to maximum variability. We found that these

features were ‘elo1_pre’, ‘qbelo1_pre’, and ‘elo2_pre’ (please see the descriptions for these features in Table 1). Judging from the plot below of the number of components vs the explained variance, we see that the variance stops decreasing significantly after we have 2 components. This is corroborated further by the fact that the cumulative variance sum of the first and second component is close to 85%. The first principal component explains 46% of the explained variance and the second principal component explains 39% of the explained variance. The ‘elo1_pre’, ‘qbelo1_pre’, ‘elo2_pre’, and ‘qb1_value_pre’ are the most important features for these top components.

5 Outcomes

Now, we’ll be discussing the outcomes for each of our experiments, including PCA analysis, running different model types on our data, as well as cross-validation. Furthermore, we’ll analyze why we get the accuracies from each of the models and potential improvements and/or steps forward.

We performed PCA on the dataset to determine which features had been the most important in terms of deciding the outcomes of the football games due to maximum variability. We found that these features were ‘elo1_pre’, ‘qbelo1_pre’, and ‘elo2_pre’ (please see the descriptions for these features in Table 1). Judging from the plot below of the number of components vs the explained variance, we see that the variance stops decreasing significantly after we have 2 components. This is corroborated further by the fact that the cumulative variance sum of the first and second component is close to 85%. The first principal component explains 46% of the explained variance and the second principal component explains 39% of the explained variance.



Our project deals with creating models that predict whether or not the home team won each given game. The outcome was operationalized by a binary variable that was equal to 1 if the home team won and 0 if the home team lost. We also generated a probability for a home team winning or losing a game against another team. Furthermore, after applying the logistic regression to the aggregated data, we got an accuracy of 67%. This served as a baseline experiment for our other predictions. In addition to incorporating ELO statistics, we utilized other features related to the weather on the day of the game such as dew point, humidity, precipitation, wind speed, and air pressure.

To improve our Logistic Regression model, we decided to utilize one-hot encodings. We also realized that in the initial preprocessing (in clean.ipynb), we were converting the ‘playoff’ feature of the data into a binary value of 1 and 0 depending on whether the value at each row for that column was NaN or not. Upon closer observation, we noticed that there were different discrete values of the ‘playoff’ feature that were equal to ‘c’, ‘d’, and ‘s’. We thought these different values might play a significant role in determining which team would win the game. Therefore, we thought it might be a good idea

to one-hot encode this feature and expand it into different features, one for each possible discrete value, as is done in the clean_data_2.ipynb file. This would ultimately give us more features to work with to make our model more expressive. After incorporating the new one-hot encoded features into our model, we found that the maximum accuracy we could get was 66%.

To attempt to improve the performance of our model even further, we then decided to apply a Random Forest classification to the data since we thought this would be more robust to outliers in the data. Using a single decision tree would likely overfit the data, but a Random Forest would likely mitigate this effect due to the averaging of results of multiple trees. After applying the Random Forest classifier model to the same data, we were only able to achieve a maximum accuracy of around 64-65%.

We also tried using a neural network because we thought it would be a more expressive model with more complex features and would be able to approximate complex functions, which might be the case with forecasting football game outcomes. It turned out that the accuracy of this model for the test set was 66.76%.

5.1 Analysis:

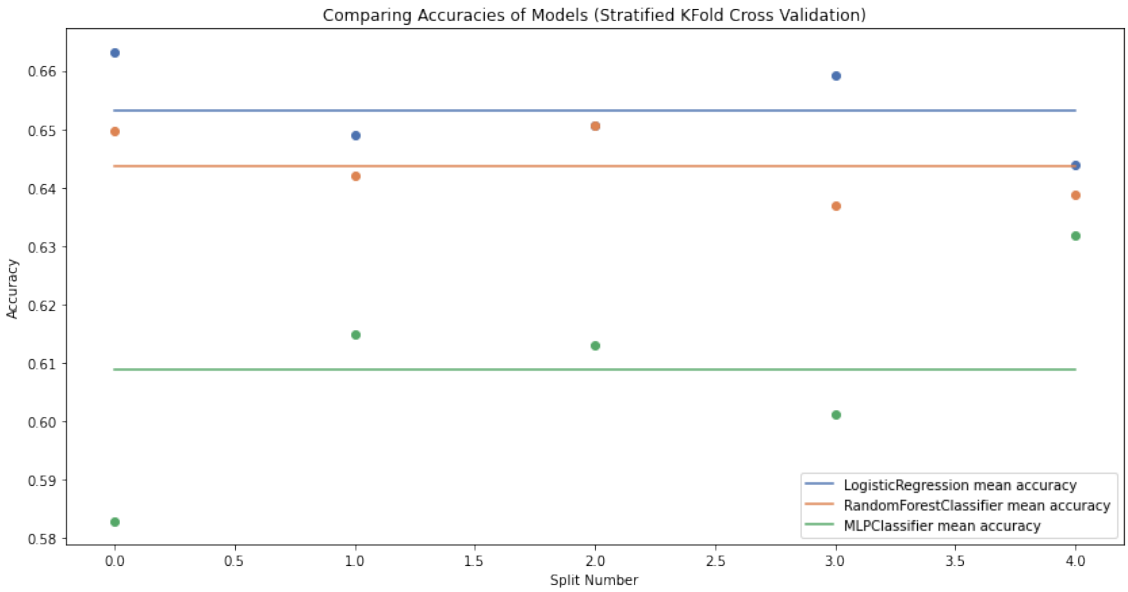
Our results for the Logistic Regression, Enhanced Logistic Regression (with one-hot encoded values), Random Forest Classification, and Neural Network classification for our data set is as follows:

Table 2: Features

Model	Train Accuracy	Test Accuracy
Logistic Regression	58.63 %	67.03 %
Random Forest Classifier	100.0 %	64.43 %
Logistic Regression with 1-hot encodings for playoffs	65.16%	66.62 %
Neural Network	66.03 %	66.76 %

Table 3: Model Evaluation with 80-20 train-test data split

We also performed cross validation with 5 folds to further evaluate the accuracy for each model. For each of the five folds in the data, each model's accuracy is determined, as depicted by the points in the graph below. The lines represent the average accuracy for each model.



Given all these results, the maximum test accuracy we could obtain was 67%. It's clear from the data that the Random Forest model overfit the data and both Logistic Regression models as well as the

Neural Network underfit the data. It's quite possible that we didn't have enough features to train our data with. In particular, it could have been useful to incorporate expert opinions prior to each game and use metrics based off of these opinions as additional features. None of this data unfortunately was available in a .csv format and would likely have had to be scraped individually for each game from each expert before getting aggregated and merged into the data we already had. In other words, all of this data pertaining to expert opinions for each of these games is widely dispersed and would involve strenuous effort to gather and consolidate into a convenient format to train our models. If we had the extra time and resources to do this, this would be a potential next step in our experimentation. Moreover, to improve the performance of the Neural Network and the Logistic Regression models, we could have used more data to train on. We utilized data from games between 2009 and 2019 and perhaps could have included more seasons.

6 Github Project Repository

https://github.com/sunil-2000/NFL_predictions

7 References

Dominguez (2021, Feb. 4). Open Source Football: Modeling NFL game outcomes using Python and scikit-learn. Retrieved from https://mrcaseb.github.io/open-source-football/posts/2021-01-21-nfl-game-prediction-using-logistic-regression/

https://github.com/fivethirtyeight/data/tree/master/nfl-elo

https://github.com/ryurko/nflscrapR-data/tree/master/games_data

https://www.datawithbliss.com/weather-data

https://www.activestate.com/blog/how-to-predict-nfl-winners-with-python/

https://github.com/kuleshov/cornell-cs5785-2022-applied-ml/blob/main/slides/lecture18-neural-networks.ipynb

https://github.com/kuleshov/cornell-cs5785-2022-applied-ml/blob/main/slides/lecture15-decision-trees.ipynb

https://github.com/kuleshov/cornell-cs5785-2022-applied-ml/blob/main/slides/lecture4-classification.ipynb