Predicting the Over/Under Result of NFL Games

Vincent Goyette, Blaise von Ohlen, Tyler Horwitz, Marcelo Castellanos
University of Notre Dame

ABSTRACT

This paper proposes a methodology for predicting the results of over/under (O/U) bets, specifically for NFL football games.

ACM Reference Format:

Vincent Goyette, Blaise von Ohlen, Tyler Horwitz, Marcelo Castellanos. 2021. Predicting the Over/Under Result of NFL games. In *Proceedings of the ACM Conference (Conference '21)*. ACM, New York, NY, USA, 1 page. https://doi.org/10.1145/1234567890

1 INTRODUCTION

The National Football League (NFL) is one of the most popular sports leagues in the United States. As a result of the recent wave of legalization of sports betting in the U.S., it has also become one of the most popular sports to gamble on as well. One of the easiest-to-understand bets is the over/under (O/U), which is a number set by bookmakers on what the combined score of a game will be. For example, if an O/U is listed at 50 by a sportsbook and the final score of the actual game is 42-21, then the "over" bet wins since the combined final was $63 > 50$. This particular bet is considered an "even" bet, meaning that a random guess should have a 50% chance of winning.

The goal of this project is to create a machine learning model that is able to use the statistics of a team in each matchup to successfully predict if the over or under bet will win for that game. We will be using teams' statistics such as yards per game, first downs per game, and turnovers, along with weather data, as features in the models. Our belief is that these features are indicative of team strength, and will allow our models to outperform random guesses.

The historical betting data needed to create our model came from a dataset available on Kaggle. It contains game and weather information for each game since 1966; it contains betting information for every game since 1981. Statistics for each game in the betting dataset were found by scraping team pages on *pro-football-reference.com*. We used several preprocessing methods to prepare our data, and then trained several different machine learning models.

2 RELATED WORK

The use of machine learning and other advanced statistical/analytical methods in sports betting is not a new endeavor, though generally not in the area of football over/under prediction. In 1996, Purucker used a neural network to achieve a 66% accuracy in predicting the winners of NFL games (1). Khan extended this work, achieving 75% accuracy across weeks 14 and 15 of the 2003 NFL season (2).

Machine learning has also been used for prediction in other sports as well. In 2008, McCabe and Trevathan used a neural network to predict results across four different leagues: three rugby leagues, along with the English Premier League (soccer). The average performance of their algorithm across leagues was 67.5% (3). Interestingly, Tax and Joustra (2015) used betting odds as features, and their model increased in accuracy from their ostatistics (4).

When looking at the other work done related to this field, we noticed that much of it was related to predicting *outcomes* in terms of win-loss, rather than *total score*. Our project manages to separate itself by distancing itself from the outcome of the game, and instead focuses more on how high-scoring the game will be.

3 PROBLEM DEFINITION

Given a particular matchup between two teams in the National Football League and an over/under for that game, can we create a machine learning model that successfully predicts the over/under result? How do our predictions compare to the real-life results?

4 DATA

4.1 Data Collection

To begin the process of solving this problem, we first had to collect the raw data. We began with a dataset that came from *kaggle.com*, called Spreadspoke Scores, consisting of

approximately 12,000 data objects. Each data object in this dataset can be considered a game. The features included general game information (team names, date, stadium, etc.), weather data, and betting data. Some were extraneousEach game had to have its combined score calculated so that a label could be assigned to each data object: 'over' or 'under'.

Team statistics also had to be obtained for every game and added to the dataset. To do so, we scraped *pro-football-reference.com* to get the team stats for every game in the original Spreadspoke Scores dataset. Once this was done, we had a dataset of approximately 12,000 data objects containing betting information and statistics for each team.

To further make this data relevant, we calculated the rolling 16 game average for each statistic. For example, say the game in question was between the Detroit Lions and the Chicago Bears, in Week 8 of the 2018 season. The team statistics associated with this game would be the *average* of the previous 16 games played by the Lions, and the previous 16 games played by the Bears. This was done so that the statistics for each team in a game better reflected each team's recent performance.

4.2 Data Objects

Each data object represents a game played in the National Football League. The dataset we used contains general information that is generally extraneous to the models (actual features are extracted later), but is useful for viewing and understanding the data. This information includes data like the date of the game, the names of the teams playing, the stadium name, and spread (different type of bet) information.

The information relevant to our actual models mostly includes the team statistics scraped from *pro-football-reference.com*, and then transformed using the rolling average described above. These features include the following statistics for both the home and away teams:

- First Yards per game
- Total Yards per game
- Rushing Yards per game
- Passing Yards per game
- Turnovers per game

Also relevant to our models was weather information, including the temperature (in Fahrenheit) at game time and the wind speed in miles per hour. The idea behind including weather data is that the temperature and wind can significantly affect the way a particular game is played. For example, a game that is played in cold and windy conditions will likely reduce the passing in a game.

5 PREPROCESSING

5.1 General Cleaning and Integration

The next step in the process was to ensure our data was properly cleaned and well integrated. We started by finding games that had no valid over/under number. We removed these games, along with games that had invalid weather data. Also, because we took a 16 game rolling average for each team, the first 16 games for each team in the dataset had to be dropped. After this preprocessing we were left with approximately 10,000 NFL games to base our models on.

5.2 Feature Extraction and PCA

During the initial development of our models, we used the original statistics we gathered from pro-football-reference.com as features. As we iterated on our models and attempted to make them better, we did analysis to figure out which features were highly correlated to one another. The correlation heatmap between features can be seen in Figure 1.

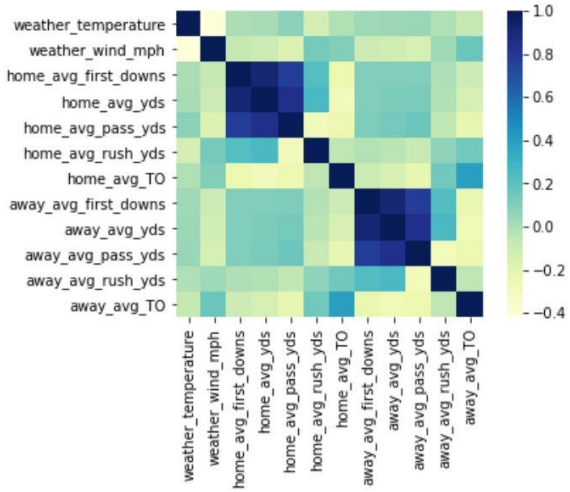

Figure 1: Correlation Heatmap of Features

As Figure 1 demonstrates, all of the yardage statistics (total yards, passing yards, rushing yards) were highly correlated; this makes sense, considering total yards is simply rushing and passing yards summed. After running PCA on our features, we were able to reduce our dataset's dimension from 12 to 8.

5.3 Feature Scaling and Label Encoding

The final steps we took to optimize our dataset for machine learning was to scale the data and encode the labels. We used a Z-Score scaler to scale our dataset; this type of scaling involves finding the mean of each feature. The rest of the values are then scaled based on this mean. This scaling allows for easier comparison of different features.

Another important preprocessing technique used was label encoding. Our original labels were strings, either "over" or "under". However, some machine learning models won't accept strings as arguments, and as a result we had to numerically encode our labels. To do so, we

encoded each "over" label as a -1, and each "under" label as a 1.

6 MODELING AND EVALUATION

With the data preprocessed, we were now ready to implement several different machine learning models. Each model used was a Scikit-Learn implementation. In total, we created 6 different models. We initially created three simpler classifiers using a decision tree, a Naive Bayes, and a logistic regression. We then moved on to creating three more complex models, including a support vector machine, a random forest, and a neural network. Every model underwent 10-fold cross validation and had pertinent metrics collected for evaluation.

6.1 Decision Tree

A Decision tree can be used both for classification and regression. It works by selecting features from the data objects that provide the highest information gain and creates a sequence of decisions based on that order.

We used entropy as our splitting criterion. For best results we tried increasing the max tree depth in increments of 5 within a range of 5 to 20. Our best results were achieved, however, when we did not restrict the depth and simply allowed the tree to continue splitting until it found no additional information gain.

	Mean	StDev
Accuracy	50.69%	±1.711
Precision	52.13%	±1.696
Recall	52.31%	±2.310
F1	52.19%	±1.680

It was a good model to use for our initial classification attempt. It gives slightly-better-than-average results in terms of all metrics we gathered. One interesting trend we noticed was that as max-depth increased, the metrics largely stayed the same; however, the standard deviation of each metric decreased.

6.2 Naïve Bayes

Naive Bayes is a simple yet powerful classification technique that relies heavily on the assumption that all features of the data object are independent of each other and that each is equally as impactful in the outcome. Some of its advantages are its low spatial and time complexities.

	Mean	StDev
Accuracy	52.05%	±1.279
Precision	53.04%	±1.072
Recall	62.97%	±15.623
F1	56.55%	±6.885

One surprise when using this model was that we had a high variability in recall which we attributed mostly to the error of giving all features the same importance, even with the PCA reducing the correlation between features.

6.3 Logistic Regression

A logistic regression is a type of classifier that uses a logistic function to model some binary variable. It is perfect for data such as this, in which it must model some outcome that is 1 of 2 choices.

For our model, we varied the hyperparameter of maximum iterations until convergence. We discovered that an ideal value for this hyperparameter in our model was 225.

	Mean	StDev
Accuracy	52.45%	±1.569
Precision	53.26%	±1.167
Recall	63.93%	±15.963
F1	57.09%	±7.162

This model performed in a very similar fashion to the Naive Bayes. Note that it had accuracy slightly better than a random guess; however, it once again had very high standard deviation in both recall and F1. This trend will be explored further in our analysis.

6.4 SVM

A support vector machine (SVM) works by constructing a hyperplane in a higher-dimensional space; this hyperplane can then be used to classify data objects.

There are several different kernels that can be used in SVMs; for our purposes we trained one model each on linear, polynomial, and radial basis function kernels. There is also a hyperparameter C which allows us to control error tolerance. For each model, we examined C values from 0.1 to 1, in increments of 0.1. The following were our best models for each kernel.

For the linear kernel, a C value of 0.7 produced the best results. A common theme among the different SVM models, it achieved very high recall but also had extreme variability within this metric.

C = 0.7	Mean	StDev
Accuracy	52.40%	±1.218
Precision	53.03%	±0.930
Recall	68.12%	±14.964
F1	58.81%	±6.186

For the polynomial kernel, we once again tested different C values to find which would give us the best results. This model was interesting in that it gave extremely high recall compared to other models, while its other metrics were relatively consistent to the others. This SVC model is the one we will examine in our analysis.

C = 0.4	Mean	StDev
Accuracy	52.30%	±1.123
Precision	52.34%	±0.933
Recall	85.36%	±7.747
F1	64.71%	±1.783

The final SVM we created used a radial basis function (RBF) kernel. This model had a very small C value of 0.1 and thus a large tolerance for misclassification. It performed similarly to the other SVM models created, with a high recall and F1 score.

C = 0.1	Mean	StDev
Accuracy	52.12%	±0.944
Precision	52.75%	±0.968
Recall	71.77%	±17.153
F1	59.74%	±6.638

6.5 Random Forest

Random forests are an ensemble learning method that constructs many decision trees and uses either the average prediction of the trees or the class which is selected by the most trees. This outcome then becomes the prediction of the model.

	Mean	StDev
Accuracy	51.79%	±1.663
Precision	53.04%	±1.694
Recall	56.19%	±5.485
F1	54.43%	±2.876

We were surprised by the very small increase in performance from the base decision tree to the random forest. The random forest performed very similarly, actually, to the original decision tree model.

6.6 Neural Network

Neural networks are machine learning models that take a vector of inputs and perform some computation with them, sending those outputs to the next layer of neurons, and repeat the process until a final output is delivered. As a classifier, the MLPClassifier uses a multi-layer perceptron algorithm that trains using backpropagation. It is especially useful for classification problems where inputs are assigned a label.

For our classification we used a network with 3 hidden layers of 10 neurons each and a maximum number of iterations of 1000.

	Mean	StDev
Accuracy	51.42%	±1.428
Precision	52.54%	±1.263
Recall	58.66%	±9.465
F1	55.09%	±4.237

The neural network produced results that were quite similar to our other models. Note that it also has high variability in its recall score, just like several other models we've examined so far.

7 ANALYSIS

Once the models had been run and results were collected, we were able to summarize our data by plotting each model's metrics.

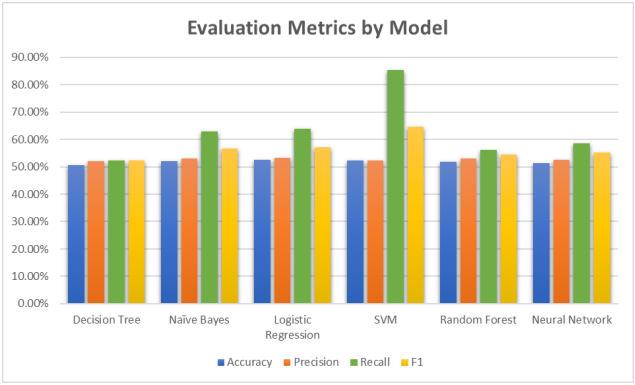

Figure 2: Metrics for Each Model

As can be seen in Figure 2, all of the models performed very similarly in terms of both accuracy and precision. The minimum average accuracy of any model was the decision tree, at 50.69%. The model with the highest average accuracy was the logistic regression model, with a value of 52.45%. This tiny gap between the best and the worst models indicates the fine margins that go with sports betting.

The most interesting trend we saw in our models was that tendency for them to have much higher recall, and sometimes F1, scores than any other metric. The most extreme case of this is the SVM model, which had over 80% recall. Other unusually high recall scores were also seen in the Naive Bayes model and the Logistic Regression. Upon further inspection, we noticed that these models tended to label nearly twice as many games "unders" as

they did "overs", despite the classes being nearly evenly distributed. These models are *very* good at predicting the under correctly; however, a game needs to be a very obvious candidate to be labelled an "over". In essence, these three models have a bias towards picking the under rather than the over.

To evaluate our best model - the logistic regression - we created an ROC curve that compares our results to a random, baseline guess. The following figure presents our results. Our model is clearly better than a 50% chance random guess, but only slightly.

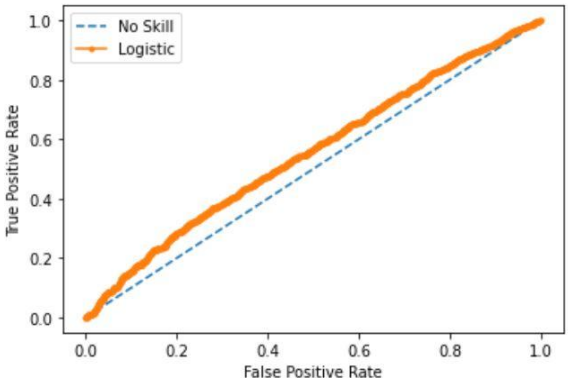

Figure 3: ROC for Logistic Regression

8 CONCLUSIONS & FUTURE WORK

To truly rise above the 50-55% range of prediction accuracy it would be essential for us to gather more in-depth data regarding the teams and their performance. A good set of features would correspond to team compositions, where there is information regarding the individual players on each team and how their performance affects the results of our predictions.

Another possible improvement that could be made on this project is the incorporation of more advanced statistics. As the NFL has gotten more popular, advanced statistical methods have been applied to give better indicators of team strength than the raw statistics we used. Incorporating these statistics into our dataset would hopefully give us better features. Unfortunately, these types of data weren't widely available until recently and thus there is a limited supply of it.

While the metrics collected for each model do not appear outstanding by themselves, we are still satisfied with our results. For context, professional bettors will rarely win more than 54% of their bets in a given time period[4]. Improvement in our model would potentially give it practical use in actual betting markets and give bettors an edge when looking at over/under bets.

9 REFERENCES

[1] Purucker, M C. “Neural Network Quarterbacking.” *IEEE Xplore*, IEEE, 1996, https://ieeexplore.ieee.org/abstract/document/535226.

[2] Silverio, Manuel. “My Findings on Using Machine Learning for Sports Betting: Do Bookmakers Always Win?” Medium, Towards Data Science, 26 July 2021, https://towardsdatascience.com/my-findings-on-using-machine-learning-for-sports-betting-do-bookmakers-always-win-6bc8684baa8c.

[3] McCabe, Alan and Trevathan, Jarrod. "Artificial Intelligence in Sports Prediction". *IEEE Xplore*, IEEE, 2008. https://ieeexplore.ieee.org/abstract/document/4492661

[4] Miller, J.V. and Miller, J.C. “Winning Percentages of Professional Sports Bettors”, *Professional Gambler*, ProfessionalGambler.org, 2021, https://professionalgambler.org/winning-percentages