Leveraging Classification Machine Learning Models to Predict The Outcome of Games in The National Football Association

Ethan Dinh
University of Oregon
edinh@uoregon.edu

Annika Huston
University of Oregon
ahuston@uoregon.edu

Abstract

Recently the increase in volume of data related to sporting events has led to the development and increased use of machine learning (ML) as a tool to extract information. Taking sports into consideration, the utilization of ML has appealed to the field of sports betters who desire to exploit the betting markets. In this work, we implemented a probabilistic betting algorithm which utilizes model ensembles that combine the predictions from multiple tuned binary classification models to accurately predict the winner of a National Football League (NFL) game. We evaluated four common betting classifiers – Decision Tree, Logistic Regression, XGBoost, and Random Forest – and analyzed the accuracy of predicting NFL games for each model. Our results demonstrated that of the four models, the Decision Tree model performed the worse with a cross-validation accuracy of 69.5%. Using our three best models, we utilized a calibrated classifier model with a voting classifier as the base to build a probabilistic betting algorithm. Overall, on unseen data from the 2019 NFL season, our betting algorithm placed 207 bets out of the possible 251 games and won 170 of those bets. This resulted in a win percentage of 82.13%.

1 Introduction

Across the United States of America, the sports scene has grown drastically over the last decades and has developed into a formidable financial market. Concurrently, the sports betting industry has followed the trend and currently stands as a multi-billion-dollar industry. Estimates will vary, but the Nevada Gaming Commission reported over $3.2 billion dollars in legal gambling in 2011, with 41% of that money begin wagered on football alone (Spear, 2013). In fact, for the upcoming Super Bowls – the national championship game – 31.4 million Americans are expected to place $7.6 billion in legal bets according to the American Gaming Association (Chen, 2022). With a large market for sports betting, predicting outcomes has become appealing not only to sports workers but also to the wider audience, particularly those who want to leverage their odds.

Sports betting itself is a phrase to mean placing a wager on a subset of outcomes of sports events, each of which is associated with a corresponding profit predefined by a bookmaker (Hubáček et al., 2019). If the outcome of the event is predicted correctly, the better wins back the wager plus the profit. If the prediction is incorrect, the better will lose the wager to the bookmaker. Historically, this type of betting was legally operated through in-person shops, however, the expansion of the internet allowed for betting to occur remotely on websites such as DraftKings and FanDuel. Despite the large number various betting opportunities, our work will specifically focus on moneyline bets – a bet in which the better will need to predict the winner of the game.

One solution to this problem of predicting the winner of a game involves the implementation of binary classification, or the combination of several classification models. In specific, these models learn from a series of features related to the relative performance of each team. In sport prediction, large numbers of features can be collected including the historical performance of the teams, results of matches, and data on players, to help betters analyze the odds of winning or losing forthcoming matches. For the purpose of our project, these metrics include: Vegas win probabilities, quarterback performance, weather information, betting odds, and game schedule.

Our project leverages a probabilistic betting algorithm which utilizes model ensembles that combine the predictions from multiple tuned binary classification models to accurately predict the winner of a football game. Using a dataset comprised of all National Football Association games from 1980 to 2019, we trained and compared the accuracy of four different learners (Decision Trees, XGBoost, Logistic Regression, and Random Forest).

The rest of the paper is organized as follows: Section II details the problem we are attempting to solve in two parts: validation and extrapolation; Section III provides an overview of our methodology; Section IV covers our experiments and analyzes results; and Section V highlights our conclusions.

2 Background

The problem we are attempting to solve can be divided into two subsections: validation and extrapolation.

2.1 Validation

One of the most common machine learning (ML) tasks involves predicting a target variable in previously unseen data. This task is referred to as **classification**. The aim of classification is to predict a target variable by building a classification model on a training dataset, and then utilizing that model to predict the value of the class in test data (noa, 2019). In the case of this project, we are attempting to predict the winner of an NFL game. Although ML has been implemented in previous works to predict sports games, we must validate that with our dataset, ML models are able to predict the outcome of game with an accuracy greater than 50%. Since the goal of the project is to develop a betting model that will profit the better, if none of the tested models perform better than 50%, there is no point in building a betting model. In fact, it would be better to just guess the outcome. Once we determine if our data possess the necessary features and a model is able the outcome of games, we will attempt to extrapolate using model ensembles.

2.2 Extrapolation

The overall goal of this project is to produce a betting algorithm that will profit the better. Consequently, we must construct a model that will correctly classify the winner of games with an accuracy greater than 50%. With the assumption that all bets are of equal value, if our model performs better than 50%, the better will therefore profit. Although this assumption does not perfectly reflect the current betting scene, this baseline allows for a realistic evaluation of our model. Given that this is a classification project, we want to build a model that produces a probabilistic value for each outcome. This probability will allow our model to decide whether or not to place a bet. In doing so, the model increases the better's profit as they will lose less when the model is unsure of the winner.

3 Methodology

In this section, we provide a detailed overview of our methodology for this work. We will analyze the dataset we used and how we filtered our data, the features we selected for our models, the binary classifiers we compared, and how we built our probabilistic betting model.

3.1 NFL Dataset

A total of three datasets was used in this project: NFL teams, NFL games, and NFL betting odds. The NFL teams dataset comprised of the team name and the shorthand abbreviation for each respective team. The NFL games dataset contained data relating to all of the NFL games from 1966 to 2019. Each game had data relating to: scheduled playing date, season, score, competing teams, stadium neutrality, win probabilities, and quarterback performance metrics. The NFL betting odds dataset was similar to the games in that they both contained scheduled playing date and competing teams. However, this dataset also contained the Vegas betting odds and favorites, and weather information.

To effectively parse the data into features that could be associated with determining the winner of each game, the game and betting odds dataset had to be uniform. Starting with the names of each team, we used the team dataset to convert all long names to their corresponding abbreviations. Since each dataset possessed the scheduled playing date, we had to convert the date to a datetime type so that the datasets could be effectively merged. We then merged the datasets and created the following features:

1. If the home team was the favorite (0 or 1)
2. If the score was above the over-under line (0 or 1)
3. If the game was a playoff game (0 or 1)
4. If the stadium was neutral (0 or 1)
5. If the home team was won (0 or 1) – target variable

We then removed all games scheduled prior to the 1980 season. We did this because the NFL has evolved drastically since the 1980s. As a result, if we had left those games, the data would become noisy and thus result in our model performing worse. We then dropped redundant columns and filled the empty values with the mean of the column.

3.2 Features

With our dataset cleaned we were left with 16 distinct feature. In order to identify the most useful features for the classification task, we utilized the recursive feature elimination algorithm (RFE). In general, RFE works by searching for a subset of features by starting with all features in the training dataset and removing features until the desired number of features remain. This process is done by fitting the given machine learning model, ranking features by importance, removing the least important features, and refitting the model (Brownlee, 2018). RFE is a wrapper-type feature selection algorithm which implies that a different ML model will be used to help assist in the feature selection process. In our case, we used the linear discriminant analysis algorithm (LDA) to assist the RFE. LDA is a powerful statistical method to find a combination of features that separates classes ("Linear Discriminant Analysis — What Is Linear Discriminant Analysis"). However, an important hyperparameter in the RFE model is the number of features to select. To determine this number, we implemented an algorithm that increments the number of features and evaluates a decision tree model fit to the selected features. We evaluated the model on accuracy via cross validation with 10 splits and 3 repeats per split.

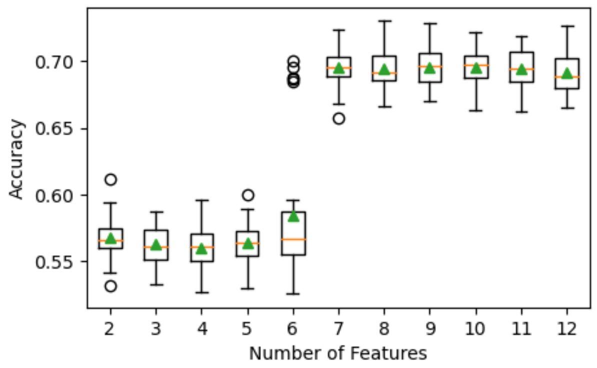

Figure 1: How the number of features influence the accuracy of a decision tree model

After doing so, we determined the best number of features to be 7 with a bias on minimizing the total number of features to aid in the speed of training each model (figure 1). With the number of features in mind, we ran the RFE model, and it discovered that the following features were best:

1. Whether or not it was a playoff game
2. Stadium neutrality

3. Vegas probability that the home team would win
4. Vegas probability that the home quarterback would outperform the away quarterback
5. Whether or not the home team was favored to win
6. The over-under betting line
7. The relative performance of the quarterbacks

3.3 Classifiers

We decided to compare four classification models to determine which model will be best for building our betting model. The four models we compared are:

1. *Logistic Regression* – Logistic regression is a common machine learning model used to predict a binary outcome. We decided upon this model to determine whether our model was linearly separable. If the accuracy of this model performed well, we could conclude that our data was in fact linearly separable.
2. *Decision Tree* – Decision trees are very simple and can model non-linearly separable data. Due to its simplicity, it does not take long to train and test.
3. *Random Forest* – Random Forest is an ensemble learning method that constructs a multitude of decision trees. As a result, it would be a good baseline for the application of more complicated models.
4. *XGBoost Classifier* – XGBoost Classifier is among the leading ML models for classification and is another ensemble learning method that constructs a series of decision trees combined to correct errors made by existing models.

4 Experiments

In the following sections, we detail the experiments we conducted and analyze the results we found. These experiments include the comparison between classifiers, the hyper parameter tuning of the best three classifiers, and the evaluation of our betting model.

4.1 Classifier Comparison

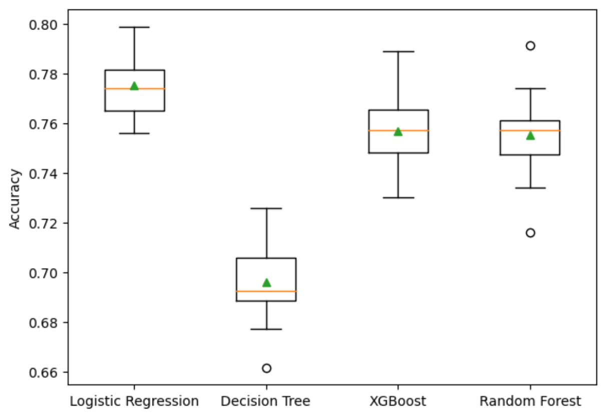

Figure 2: Classifier Average Test Accuracy

In order to determine if ML models can accurately predict the winner of NFL games, we took our whole dataset and utilized stratified k–fold cross validation in order to evaluate each model. We found that of the four classifiers, decision trees performed the worst with an average test accuracy of 69.5%. This is no surprise as compared to the other models, the decision tree model was the most simple. Although there are cases in which simplicity is preferred, due to the complexity of NFL games, its simplicity may reduce its overall performance. On the other hand, the logistic regression model performed best with an average test accuracy of 77.5%. This is surprising as one would not expect the NFL dataset to be linearly separable. Both the other models (XGBoost Classifier & Random Forest) had an average test accuracy of approximately 75%. Overall, since all of the models were able to predict more than 50% of the outcomes, we concluded that it was indeed possible to model a betting algorithm to increase the odds of betting on NFL games.

4.2 Tuning Hyper Parameters

In order to begin developing our betting model, we first had to tune the hyper parameters of our current three best models. However, when comparing the models, we used the entire dataset. Since we have demonstrated that it is possible to predict the outcome of games, we constricted our training dataset to games prior to the 2019 season. Using the training dataset, we tuned the hyper parameters of all three classifiers by utilizing Bayesian optimization. After tuning, we discovered the following trends:

Random Forest: When tuning the random forest classifier, there were three parameters to consider: number of estimators, maximum depth, and criterion. From the Bayesian optimization, we found that best number of estimators was 30. As seen in figure 3, as loss was minimized when the parameter was set to 25 and as it increased, loss increased. This increase is loss can be attributed to the fact that as the number of trees grows, the model most likely over-fits the data and thus loses its ability to generalize. As for the maximum depth, the Bayesian algorithm found the best option to be 4. However, there were little notable differences in loss between the various maximum depth values. Lastly, regarding the criterion, the Bayesian algorithm selected the gini criterion as it minimized the loss. However, between entropy and gini, the loss was nearly identical. Overall, it seems that the only hyper parameter that significantly influences the model's performance is the number of estimators as varying this parameter resulted in the largest changes in loss.

Logistic Regression: When tuning the logistic regression classifier, there was one parameters to consider: C. A high C value informs the model to give more weight to the training data, and a lower weight to the complexity penalty. On the other hand, a low C value does the opposite; it raises the weight to the complexity penalty. When attempting to find the ideal C value via the Bayesian algorithm, we noticed that loss did not significantly change when the algorithm varied the model's C value. As a result, we attempted once more using a grid search approach. The grid search indicated the best C value was 0.1 since it correlated with the highest accuracy. Interestingly, however, as the C value incremented, the accuracy of the model seemed to have not dropped significantly. On the other hand, when the C values were low, the accuracy decreased significantly

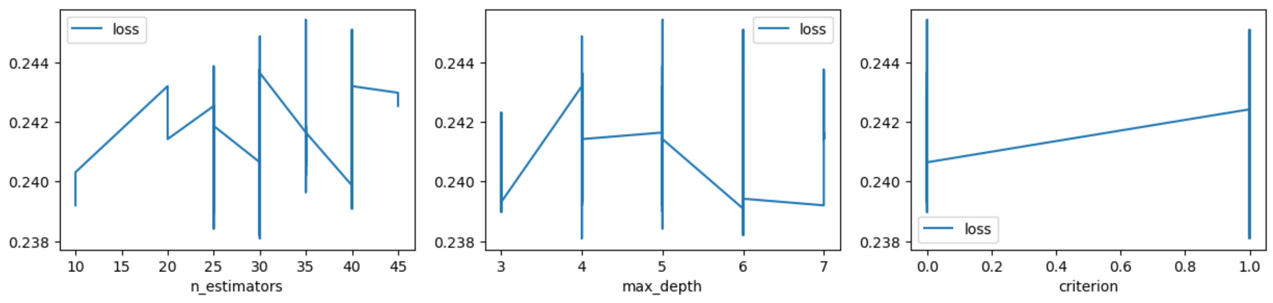

Figure 3: Random Forest Hyper Parameter Tuning Results

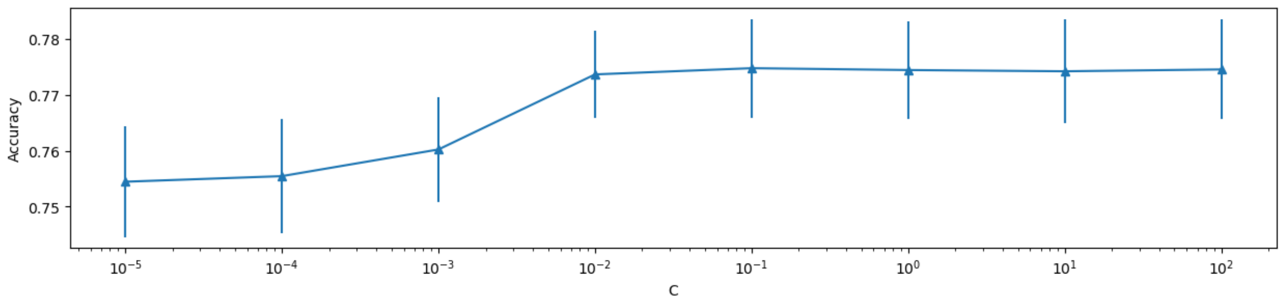

Figure 4: Logistic Regression Hyper Parameter Tuning Results (Notice that the x-axis is on a logarithmic scale)

more, indicating that placing more weight on the complexity penalty worsened the model's performance. Overall, we selected the C value to be 0.1 and deemed the logistic regression modeled tuned.

XGBoost Classifier: When tuning the XGBoost Classifier, there were three parameters to consider: learning rate, maximum tree depth, and number of estimators. The Bayesian algorithm determined that the best learning rate was 0.12. In figure 5, as the learning rate increased from 0.12, the loss drastically increased. The larger learning rate may have not allowed the model to converge properly and as a result, decrease its overall performance. As for the number of estimators, the algorithm determined it to be 343. As this value increased, however, it is evident that the loss also increased. This trend is likely due to the fact that as the number of estimators increased, the model began over-fitting to the training data and thus failed to generalize. Lastly, the maximum depth was determined to be 0. This value, however, does not make sense in the scheme of the model. Since the XGBoost classier is dependent upon the trees that it creates, it does not seem likely that it would create trees with 0 depth. As a result, we kept the maximum depth at its default.

4.3 Evaluating Our Betting Algorithm

With our tuned classifiers, we implemented a betting algorithm using a calibrated classifier with a voting classifier as the base model. Calibrated classifiers are crucial in decision making tasks as they provide a reliable probabilistic estimate of a prediction (Cohen and Goldszmidt, 2004). This model has been used in weather forecasting, game theory, and more recently, in machine learning. We are implementing this model to allow our three classifiers to produce a probability value. Since we are betting, we wanted to ensure that our model does not place a bet if it is unsure of the result. In specific, our betting model generates a probability of the home team winning. If the probability is greater than 60%, the model will place a bet on the home team. Similarly, if the probability is below 40%, the model will place a bet on the away team. Anywhere in between 60% and 40%, the model will not place a bet. We trained the betting model on all games prior to the 2019 season and tested the model's performance on the 2019 season. In table 1, notice that our model was able to win 81.82% of the bets it made. In total, the model made 198 bets out of the 251 possible games in the season. Overall, with an accuracy of 81.82%, we concluded that our model would successfully profit the better. However, as Vegas betting is not always about the wins and losses, we cannot say for certain that our model will perform well in the real world.

Metric	Value
Model Win Percentage	**81.82%**
Total Number of Bets Won	162
Total Number of Bets Made	198
Number of Possible Games	251

Table 1: Our Betting Model's Statistics.

5 Conclusion

In this work, analyzed whether or not NFL games can be predicted via conventional classifying machine learning models and built a moneyline betting model to profit those who bet on NFL games. Further, we detailed the feature selection process, the model comparisons, and the hyper parameter tuning process. We demonstrated that the best models were logistic regression, random forest, and XGBoost.

Ultimately, we conclude that despite our betting model achieving an accuracy of 81.82%, further testing on future seasons must be done to truly understand whether or not we can profit the better. Also, since our algorithm does not

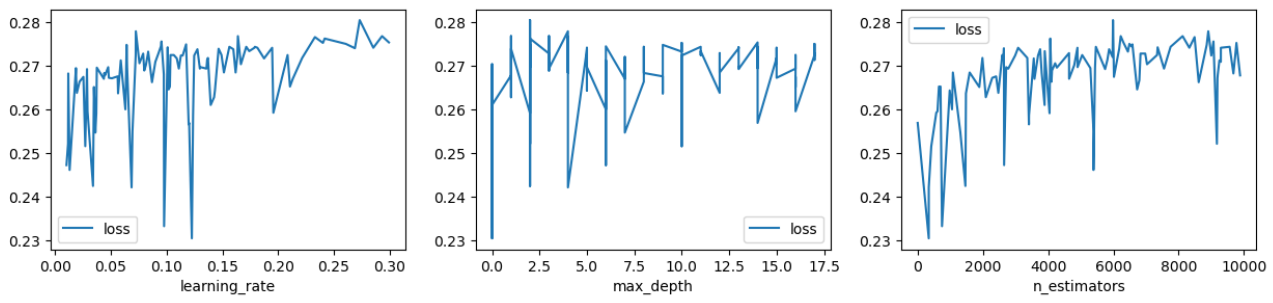

Figure 5: XGBoost Classifier Hyper Parameter Tuning Results

account for the predetermined profit margin, future work could implement that metric to determine how much a better will make given our predictions. For now, it is satisfactory to state that our model was able to win 81.82% of the bets it made on unseen data.

References

Jason Brownlee. 2018. How and When to Use a Calibrated Classification Model with scikit-learn, September.

David W. Chen. 2022. The N.F.L.'s About-Face on Sports Gambling. *The New York Times*, February.

Ira Cohen and Moises Goldszmidt. 2004. Properties and Benefits of Calibrated Classifiers. In Jean-François Boulicaut, Floriana Esposito, Fosca Giannotti, and Dino Pedreschi, editors, *Knowledge Discovery in Databases: PKDD 2004*, Lecture Notes in Computer Science, pages 125–136, Berlin, Heidelberg. Springer.

Ondřej Hubáček, Gustav Šourek, and Filip Železný. 2019. Exploiting sports-betting market using machine learning. *International Journal of Forecasting*, 35(2):783–796, April.

2019. ML | Voting Classifier using Sklearn, November. Section: Machine Learning.

Gillian Spear. 2013. Think sports gambling isn't big money? Wanna bet?, July.