Predicting NFL Game Outcomes with Machine Learning

Jericho Jacala, Alden George (Dropped)
jjacala@bu.edu

Abstract

In this paper, we examine how different machine learning methods can be used to classify NFL football games, and therefore predict game outcomes. We use 3 features, rush yards, pass yards, and turnovers for both the home and away team to train SVM, logistic regression, and multi-layer perceptron models. Then, for real world application, we try to predict the 2023 NFL Playoffs using expected metrics from the regular season with nearly 70% accuracy.

1 Introduction

Repository location Project Repository

American football is the most popular sport in America and touts the largest viewership in the nation. With sports betting on the rise, teams constantly innovating for new ways to win, and fans looking to make informed predictions about game outcomes, examination of what metrics are most important when considering which team will win the game is an important issue. Others in the past have built prediction models with only five statistics (Streitmatter). The goal of this study is to try to replicate these results despite having limited knowledge of the machine learning methods used.

We demonstrate SVM, logistic regression, and multi-layer perceptron models on historic NFL data. Then, using the SVM model trained on this data, we simulate predicting the 2023 NFL playoffs using projected stats for each game. The model was able to predict the outcomes for the playoffs with nearly 70% accuracy.

The historic data used is a collection of rush yards, pass yards, fumbles, and interceptions (fumbles and interceptions are combined into a single feature called turnovers for a majority of the study) for every team in every game from 2002-2023. This dataset can be found in the repository. NFL Playoff testing data was created by hand using playoff outcomes found on Wikipedia and official NFL team stats. Links can be found in the Supplementary Links section.

2 Related Work

Austin Streitmatter's Models This study is based on previous work by Austin Streitmatter Streitmatter (2023). While the actual machine learning methods are not disclosed, expected stats for each team are calculated using the average of the team stat per game and the opposing team stat allowed per game. For example, if the home team rushed for 100 yards per game and the away team allowed 200 rushing yards per game, the expected home team rushing yards for the game is 150. Streitmatter's best model was able to predict games with 70.7% accuracy. For comparison, 538's Elo predictor had an accuracy of 65.43%.

Committees of Artificial Neural Networks to Predict Game Outcomes Studies in the past have also tried to predict game outcomes using neural networks John A. David and Janning (2011). While studies like this one try to predict the scores of the games themselves, this study will focus primarily on determining the winner. Rather than using a single neural network, the model uses a committee of

machines (CoM). The model uses 500 neural networks, each of which has 4 neurons in the first layer and 6 neurons in the second layer and used a hyperbolic tangent sigmoid transfer activation function. Each network handled a different partition of the data, with the general idea being to take the most accurate networks and generate a consensus using various measures of central tendency. As the model predicted scores, error was measured using mean squared error. The study also trains on expected team stats that are calculated the same way as in Streitmatter's model rather than using actual historic data. In our study, we will use historic data as this models how the game actually was determined. When predicting new games, however, the model will use expected data as no one has access to the actual features until the game is played out. Our model will also construct a single multi-layer perceptron as opposed to a committee.

Mathematical Foundations This rest of this section will outline the different machine learning methods that will be used (namely, logistic regression, SVM, and neural networks). The following information was taken from "Undestanding Machine Learning," abbreviated as Shwartz and Ben-David (2014) on the CS 365 course website (also located in references):

Logistic Regression Determining the outcome of NFL games will be treated as a classification problem- either the home team wins, loses, or ties. Therefore, one of the prediction models developed will be based on logistic regression, a discriminative classification model that is widely used. In this model, denoted $p(y|x;\theta)$, where x is a fixed-dimensional input vector, y is the class label, and θ is the parameters. When the cardinality of the class label is 2, the model is called binary logistic regression.

Binary logistic regression can be represented as the following Bernoulli distribution taken from [SD] Book 1:

$$p(y|x;\theta) = Ber(y|\sigma(w^T x + b))$$

Essentially, the probability that an element is of the class label y is equivalent to the sigmoid function of the product of the weights and the input parameters plus the biases of the model. In this way, it is very similar to linear regression, but instead of producing a model through a least squares line, we calibrate a sigmoid function and assign a class label to each element based on whatever label the element is most likely to be (maximum likelihood). The sigmoid function can be represented as follows:

$$\sigma(a) = \frac{1}{(1+e)^{-a}}$$

where $a = w^T x + b$. We will refer to "a" as the logit for concision. By setting up a model for whether a game is a win or not, a tie or not, or a loss or not for the home team, we are able to generalize binary logistic regression to our situation with multiple class labels. Classifying training data will then come down to what outcome has the highest probability.

Support Vector Machines (SVM) The Support Vector Machine Paradigm (SVM) is used to classify data points by determining halfspaces in which the points can be partitioned. The objective of SVM is to not only partition the data correctly, but to ensure that the distance between the dividing hyperplane and the points is maximized.

In order for the data to be linearly separable (and by extension compatible with SVM), the data must be linearly separable. We can consider $S = (x_1, y_1)...(x_m, y_m)$ where each x is a vector of real numbers and each y is either -1 or 1. The data is linearly separable if there is a half space (w,b) such that yi has the sign of $< w, x_i > +b$.

However, linearly separating halfspaces are usually not unique, and we can formalize the aforementioned intuition about maximizing the distance between points using the concept of margin. Rather than maximizing the total distance between each point and the halfspace, we maximize the minimum margin, or distance between the closest point and the halfspace. This is illustrated in the Hard-SVM rule:

$$argmax_{(w.b):||w||=1} min_{i\in[m]} |< w, x_i > +b| \quad s.t. \quad y_i(< w, x_i > +b) > 0$$

The distance between a point x and the hyperplane is $< w, x_i > +b$, therefore we are constrained in that this distance must be greater than 0. If it is negative, the data is not linearly separable and there exists a point which is not on the correct side of the halfspace. The solution to this rule can be re-expressed by forcing a margin of 1. In doing this, the margin is inversely proportional to the norm of w. This means we must minimize the norm of w while meeting all constraints:

$$argmin_{(w,b)} ||w||^2 \quad s.t. \quad \forall y_i(< w, x_i > +b) \geq 1$$

We normalize (w,b) after finding the solution to this rule. If the data is not linearly separable, we can consider Soft-SVM, where we allow violation of each constraint by a certain constant epsilon:

$$y_i(< w, x_i > +b) \geq 1 - \epsilon_i$$

Therefore, we look to minimize both the norm of w and the average epsilon:

$$min_{w,b,\epsilon}(\lambda||w||^2 + \frac{1}{m}\Sigma_{i=1}^{m}\epsilon_i)$$
s.t. $\forall y_i(< w, x_i > +b) \geq 1 - \epsilon_i$ and $\epsilon_i \geq 0$ where λ is a positive parameter.

Neural Networks Neural networks model prediction models similarly to the brain. We focus on feedforward classification models for our purposes. Features are passed through an input layer and subsequently passed through multiple layers of neurons, each of which is represented by a scalar function which modifies the input until the output layer is reached, where the output is expressed. The cost function quantifies the error and, using a certain optimization algorithm, adjusts the biases to minimize cost. This study focuses on using stochastic gradient descent, which calculates the gradient of the loss function through backpropagation and minimizes this loss by adjusting the biases of the neurons.

3 Resources

Code execution was done through Google Colab with 12.67 GB of RAM and 107.7 GB of Disk space. While it should not be a factor, the code ran on a laptop with an AMD Ryzen 9 6900 HS. ChatGPT was consulted for some basic guidance on how to use the libraries. For example, we asked ChatGPT how to split the data into testing and training data, an example of general use of each model in sklearn, and guidance on functions for measuring model performance metrics.

4 Method

For splitting the dataset into training and testing data and implementing machine learning models, SKLearn was used. To graph data, Matplotlib was used. For reading the CSV files, Pandas was used. All code was implemented in Python. As a wrong choice constitutes equal loss regardless of whether something was a false negative, true positive, etc in our case, accuracy was the primary metric of model performance. Of the datasets, 80% of the data was used for training while the other 20% was used for testing.

The main machine learning methods used in the study were logistic regression, SVM, and multi-layer perceptrons (the mathematical foundations for these methods can be found in related work).

4.1 Experiment Overview

First, a logistic regression model was created using only home rush and pass yards from 2010-2016 as a preliminary test that everything worked correctly. Using the same time interval, new models were created for SVM, logistic regression and neural networks. How different numbers of principal components in preprocessing affected SVM and logistic regression was also studied. After preliminary tests, we simulate predicting the 2023 NFL playoffs using an SVM model trained on past games.

5 Experimental Results

As a proof of concept, the data was filtered into a cleaned dataset containing only the games from 2010 to 2016. For testing the implementation of models in code, it was logical to use a smaller dataset where season-by-season changes in how the game is played and computations could be done quickly. The goal was to make sure that the model worked and was implementable and sample complexity was not an issue at that time.

Initially, only the home team's pass yards and rush yards were used in preliminary experiments. The first model used was logistic regression, which maintained an accuracy of about 0.63. Knowing that the model appeared to work, we included rush yards, pass yards, and turnovers for both the home and away team:

Table 1: Accuracies across machine learning Methods with 6 preliminary features.

Logistic Regression	SVM	Neural Networks
0.6657	0.6350	0.6685

We were initially concerned that increasing the dimensionality of the data without increasing the sample size may adversely affect the data. To test this we used principal component analysis (PCA). By reducing the dimensionality to principal components along which most of the variation consists, we hoped to determine if the optimal number of features was lower than 6, or the number we currently had. However, model accuracy only grew with more principal components, suggesting that more information was better. We continued to use the six features for the rest of the analysis.

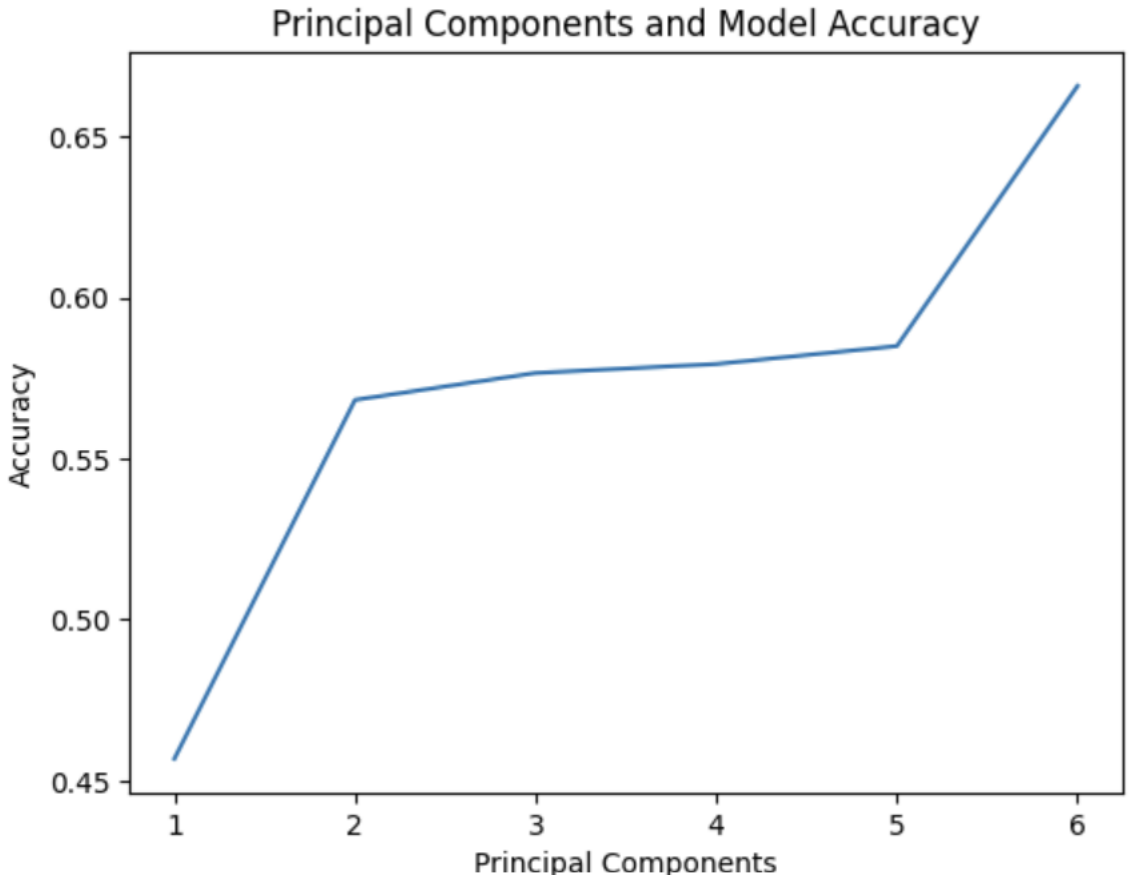

Figure 1: PCA on Preliminary Logistic Model

Table 2: PCA on Preliminary Logistic Regression Model.

1	2	3	4	5	6
0.4568	0.5682	0.5766	0.5794	0.5850	0.6657

Most notable is how model accuracy not only increases with the addition of more principal components, but that there is significant increases in accuracy from 1 to 2 components and from 5 to 6 components. It appears that 6 features has not created excessive sample complexity.

Next, we used support vector machines (SVM) to model predictions. SVM performed slightly better, with an accuracy of about 0.68 using the default radial basis function (RBF) kernel. Since it is difficult to visualize the data, it was difficult to hypothesize what kernel would work best. We tested linear, polynomial, and sigmoid kernels in addition to RBF, but all performed quite similarly. Lastly, we tested a Multi-Layer Classification Perceptron using the Stochastic Gradient Descent (SGD) optimization algorithm as outlined in the Research section. This had an accuracy of about 0.67.

After preliminary tests, we decided to test all the data points as opposed to just 2010-2016. Continuing with an SVM model with a polynomial kernel, the model showed improvements with increased data, with an accuracy of 0.82. Any attempts at dimensionality reduction through PCA once again, only diminished accuracy.

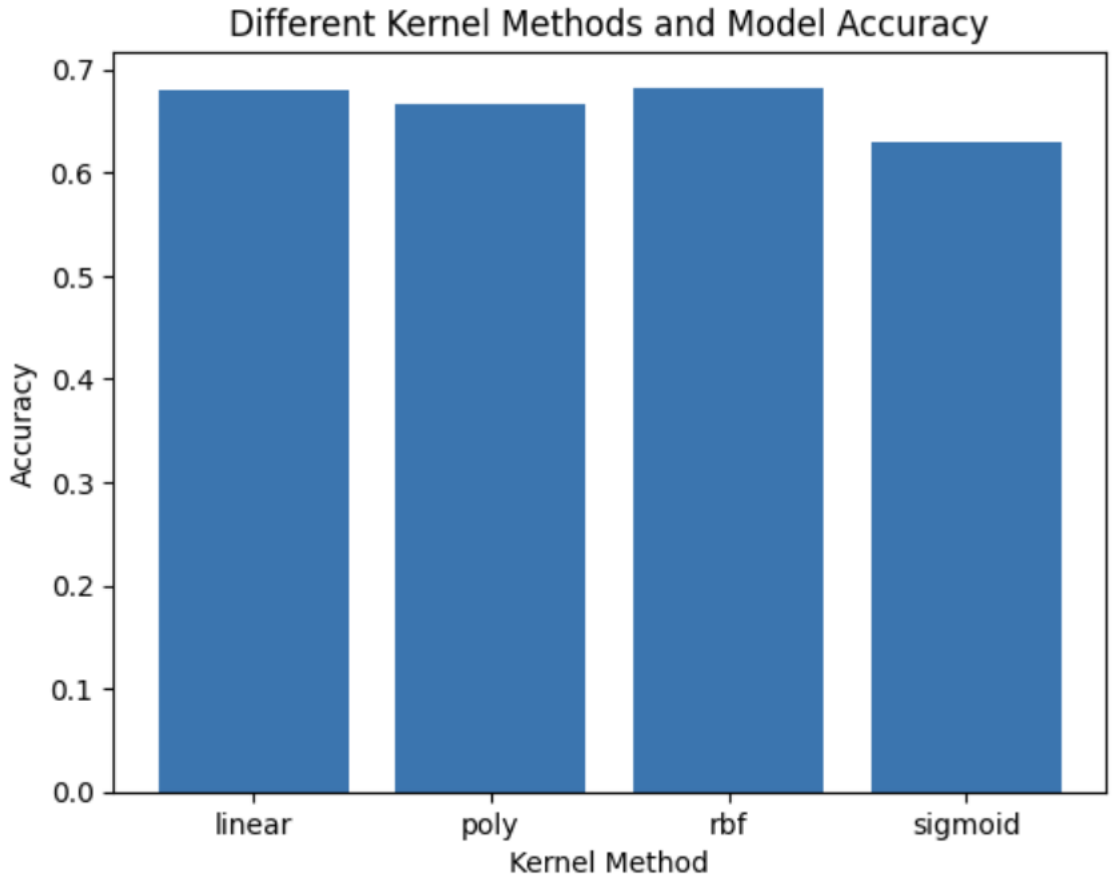

Figure 2: Accuracies of different Kernel methods

All kernels appear to perform relatively the same. The sigmoid function appears to lag behind slightly.

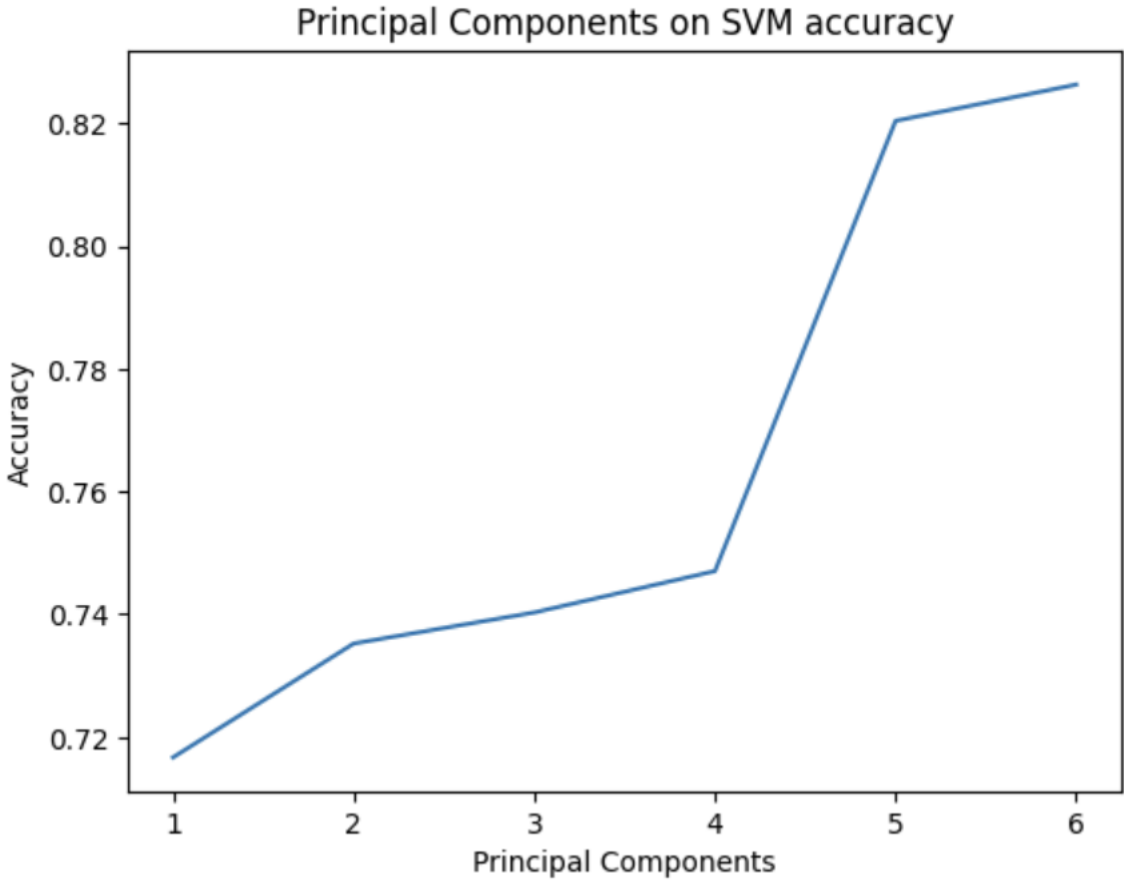

Figure 3: PCA on Preliminary Logistic Model

Again, increased components leads to increased model accuracy generally. Most interesting is how there is a great increase in accuracy from 4 to 5 components. It would be logical to hypothesize greater increases in accuracy when there are lower components, but perhaps this suggests the importance of every feature.

We wanted to apply the model to the real world, so using the same polynomial kernel model, we predicted the outcomes for the games of the 2023 playoffs. We decided to opt with keeping turnovers as one feature as opposed to both fumbles and interceptions to make creating the testing data easier. By taking the average of each team's offensive stats per game and the other team's allowed stats per game, we were able to generate an expected stat on which predictions were based. For example if the Home team averaged 100 rush yards per game and the Away team allowed on average 200 rush yards per game, then the Home team is expected to rush for 150 yards on the Away team. Using regular season data, we simulated making predictions for the playoffs, where the model had an accuracy of nearly 0.7. Precision was the same number, and the recall was 1.0.

Table 3: Performance Metrics for Final SVM Prediction Model.

Accuracy	Precision	Recall
0.6923	0.6923	1.0

With a recall of 1.0, it appears that the model was able to correctly predict the winner when the home team won every time. Games where the home team won constituted a majority of the wins in the testing set because the NFL playoffs give home field advantage to the team that performed better over the regular season. Accuracy and precision are both 0.6923, indicating the model performed worse at identifying when the away team won. In practical use, it appears the model does not perform well at

identifying possible upsets.

6 Conclusion

Experiments conducted suggest that Support Vector Machines, Multilayer Perceptrons, and Logistic Regression Models perform similarly to each other without hyperparameter tuning. The SVM and Logistic Regression models appear to not be sample deficient or overfitting as attempts at dimensionality reduction through PCA do not yield any improvements in model prediction. The lack of improvements in model performance using PCA suggest that rush yards, pass yards, and turnovers are all important factors in game prediction.

6.1 Simulation of Real World Application

In predicting the 2023 NFL Playoffs, the SVM model trained correctly predicted the outcome 69.23% of the time. Streitmatter's best models in the project upon which this one is based, was able to produce comparable results, claiming a 70.7% accuracy rate. However, more testing may be required to confirm the true accuracy of the model. The playoff data is very small, with a sample size of only 13 games. The reason for such a small test is that this test is a preliminary test meant to examine how the model holds up under modern conditions. Making the test data was extremely time consuming because the data had to be hand-entered (no spreadsheets recorded expected team stats based on regular season data). Despite this, model performance appears to be quite promising.

6.2 Potential Extension

Future projects expanding on this one may include tuning hyperparameters. Besides specifying a polynomial kernel, no hyperparameters were specified and we elected to use the default values. Future projects may attempt to predict actual game scores as done in John A. David and Janning (2011) or reattempting PCA while including more features. Perhaps extra information on margin of victory or other features like sacks could help increase model accuracy for our purposes.

Acknowledgements

This project was created for Boston University CAS CS 365 in Spring 2024. Alden George was initially a partner and played a role early on, completing preliminary tasks and helping write the milestone deliverables. However, George dropped the class midway through the semester.

References

M. Saif Ahmad John A. David, R. Drew Pasteur and Michael C. Janning. Nfl prediction using committees of artificial neural networks. *Journal of Quantitative Analysis in Sports*, 7(2), 2011.

Shai Shalev Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

Austin Streitmatter. How i built a competitive nfl prediction model with only five statistics. 2023.

Supplementary Links

The Feature Matrix for the 2023 Playoffs was found here
Playoff Outcomes found here