Using Statistical Learning Techniques to Predict Winners of NFL Games from Half Time Metrics

John Knight

The University of Texas Rio Grande Valley

MATH-6333 Statistical Learning

Prof. Hansapani Rodrigo

5 May, 2024

Introduction

The NFL (National Football League) is the predominant professional league for the sport of American football (known in the U.S.A. as simply *football*). Much time and discussion is devoted to analyzing and predicting the winners of NFL games, and it is of particular interest to predict which team will win a given game having watched the first half. Other than the score of the game, and the prior assessment of the difference in quality between the two teams (taken from the Vegas line), how much value can be found by including statistics taken from the first half of the game? Is the first half of action largely just statistical noise, or does it contain a signal of how the two teams are performing on this particular day?

In this project, various statistical learning methods were used to predict the winner of each game in the period 2012 to 2023 using features based on statistics from the first half of the games, as well as the pre-game Vegas odds (for both the handicap line and the total points). There were two main objectives: firstly, to compare different modeling techniques and see which performed best at this task. And secondly, to see which features were the strongest predictors.

All the statistical learning techniques in this project were performed using Python. Python is the most popular language for data science, and boasts a large number of freely available packages and libraries for data science and many other tasks. Python was also used in the initial collection and cleaning of the NFL data. Game data was scraped from the website *Pro Football Reference* using the `beautifulsoup` and `sqlite3` libraries, and half time statistics were aggregated from play-by-play data.

Literature Review

May et al. (2010) examined statistics from NFL games to compare the importance of rushing and passing statistics, and concluded that passing efficiency is a better indicator of team strength than rushing efficiency. [1] Warner (2010) used a Gaussian process to predict the margin of victory in NFL games but found that their resulting predictions were not as accurate as the Vegas line, and were not sufficiently accurate to place profitable bets. [2]

Blaikie et al. (2011) created artificial neural network models to predict the outcomes of football games, claiming that their NFL model performed well compared to various expert predictions, while their college football version did not perform so well. [3] Ozuma & Nwachukwu (2015) used linear regression and K-nearest neighbors to create a hybrid model predicting the results of NFL games using various features, claiming a significantly high level of accuracy from their predictions. [4]

Finally, Pelechrinis (2018) presents a simple yet robust and well-calibrated model using logistic regression to predict the winning probability for the home team during NFL matches. [5]

Pelechrinis also finds that more complex, nonlinear models fail to improve on the performance of a simple linear model.

Methods

The initial dataset, `nfl_ht_data.csv` contained 3,260 observations (each representing an NFL game) and 40 variables. Some of these variables, such as nominal variables, were not relevant to the task and were removed. Categorical values with two levels were converted to a binary 1 or 0 value, and similarly, a `home_win` variable was created with a value of 1 or 0 based on whether `home_final_score` was greater than `away_final_score`. Tied games are fairly uncommon in the NFL, and for the purposes of this project the 12 games that ended as a tie were removed.

The only missing values in the data were some blank values in the columns relating to first downs. Upon investigation, it was found that these had occurred where a team had no first downs before half time. Therefore, these blank values were imputed with a value of zero. After this pre-processing, the dataset had 3,248 observations and 28 variables. These variables were comprised of the response variable, `home_win`, plus the 27 features described in Table 1.

Table 1. List and description of the 27 features in final dataset.

Variable	Description
vegas_total	The Las Vegas betting line for total points in the game.
ht_home_lead	The number of points the home team is leading by (negative if the home team is trailing).
home_vegas_line	The Las Vegas handicap line with respect to the home team (for example if the home team is a 5.5 point favorite, this would be -5.5).
away_offensive_plays	Number of offensive plays run by the away team.
home_offensive_plays	Number of offensive plays run by the home team.
away_yards_per_play	Mean yards gained by away team on offensive plays.
away_yards_per_pass_play	Mean yards gained by away team on passing plays.
away_yards_per_rush_play	Mean yards gained by away team on rushing plays.
home_yards_per_play	Mean yards gained by home team on offensive plays.
home_yards_per_pass_play	Mean yards gained by home team on passing plays.
home_yards_per_rush_play	Mean yards gained by home team on rushing plays.
away_first_downs	Number of first downs gained by away team.
home_first_downs	Number of first downs gained by away team.
away_first_downs_per_play	Number of first downs gained by away team divided by number of offensive plays.
home_first_downs_per_play	Number of first downs gained by home team divided by number of offensive plays.

away_first_downs_1_2	Number of first downs gained by away team on first or second down.
home_first_downs_1_2	Number of first downs gained by home team on first or second down.
away_turnovers	Number of turnovers by away team.
home_turnovers	Number of turnovers by home team.
away_sacked	Number of times away team quarterback sacked.
home_sacked	Number of times home team quarterback sacked.
away_penalties	Number of penalties committed by away team.
home_penalties	Number of penalties committed by home team.
home_second_half_choice	Does the home team have the choice to receive the kickoff in the second half?
is_playoff	Is this a playoff game?
neutral_venue	Is this a neutral venue (i.e. not the 'home' team's usual home stadium)?
net_home_plays_leading	Number of plays where home team is leading minus number of plays where away team is leading.

The data were then randomly split into an 80% training set and a 20% testing set. Although this data could be arranged chronologically, a random split was chosen rather than using the most recent seasons for testing. The main reason for this is because the 2020 season was affected by the COVID-19 outbreak, and it was undesirable to have all these matches go into either the training set or test set.

Some descriptive analysis was then performed on the data. A correlation matrix was created to identify any strong positive or negative correlations between variables, which might be considered redundant and removed. Next, histograms and box plots were created to inspect the distribution of each variable. Those with an apparent Gaussian distribution were transformed using the `StandardScaler` function, while those with a more skewed distribution underwent a log transformation, first adding 1 to ensure zero values could be handled without raising an error.

The seven statistical learning methods applied to the dataset were: logistic regression (using both backward stepwise selection and the Lasso method), linear discriminant analysis, random forest, boosting, neural network, and principal component regression. Here is a brief explanation of each method:

Logistic Regression (backward stepwise selection)

This model was built using the `LogisticRegression` function from Python's `scikit-learn` library. The `SFS` function was used along with 5-fold cross validation to determine the optimal number of features. The final model was then fit on the full training set using the selected

features. Additionally, interaction terms were added between the features showing strongest statistical significance, to see if this would further improve the model.

Logistic Regression (Lasso)

The lasso method uses a penalty parameter C to penalize the size of the regression coefficients. As a result, many of the coefficients can shrink to zero (unlike in ridge regression, where they can become small but cannot reach zero). This can help to combat overfitting, and aid the model in generalizing outside of the training set. In order to estimate the optimal value of C, cross validation was used to try 10 different values of C, and the value returning the lowest log loss was chosen.

Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) works by finding the linear combinations of features that best separate two or more classes or events. LDA assumes that different classes generate data based on Gaussian distributions with shared covariance matrices among the classes, and seeks to maximize the ratio of between-class variance to within-class variance in any particular dataset, thereby ensuring maximum class separability. The `LinearDiscriminantAnalysis` function from the `scikit-learn` library was utilized to perform LDA. The coefficients obtained from the model provide insights into the relative predictive importance of each feature.

Random Forest

In random forests, many decision trees are created using synthetic datasets created from the original training data using bootstrap aggregation ('bagging'). The crucial element of random forests is the parameter *m* which denotes the number of features randomly chosen to be considered for each split. This avoids the same features being used on every tree and de-correlates the trees, helping to prevent overfitting.

The `RandomForestClassifier` function was used for this task. To find the optimal value for *m*, 5-fold cross validation was used to try all values from 2 upward. The final model was then trained using this value of *m*, and the `feature_importances_` attribute was used to see which features had contributed most, based on each feature's contribution to the decrease in overall node impurity, as measured by the Gini index.

Boosting

Boosting is another tree-based method. However, unlike random forests, with boosting the trees are grown sequentially, with each tree fit using the residuals from the current model. Compared to random forests, boosting models are slightly more vulnerable to overfitting, and so three parameters were tuned using cross-validation: the number of trees, the learning rate, and the number of splits in each tree. Again, `feature_importances_` was used to output the most important features.

Neural Network

The neural network was fit using the `PyTorch` library. While neural networks can be powerful in their ability to uncover complex nonlinear relationships in the data, one of the challenges in creating them is the number and range of hyperparameters that need to be tuned. For this model, five parameters were tuned using 5-fold cross validation: the learning rate, the number of epochs, the size of the hidden layer, the batch size, and the dropout rate for regularization – important to help avoid overfitting. Unlike most of the other techniques, with neural networks there is no simple way to output feature importance, and in this sense it can be described as a 'black box' with strong predictive power that is not easily explained.

Principal Component Regression

With the possibility that many of the features in this dataset are redundant or irrelevant, a variable reduction method might improve the performance of the logistic regression. The PCA function from scikit-learn was used to apply principal component analysis to reduce the full set of features to a smaller number. 5-fold cross validation was used to try each number of variables, and the optimal value was then used to train a final logistic regression model using the resulting principal components.

After each model had been tuned and trained using the training data, it was used to make predictions on the untouched testing set. The scoring method chosen to compare models was the log likelihood. While it is common to use other measures such as accuracy or AUC (Area Under the Receiver Operating Curve), these measures rely on a 1 or 0 prediction for each game using some probability threshold (e.g. 0.5). However, in the context of football predictions, it is of limited interest to predict which team is more likely to win – often this is trivially obvious. Instead, it might be important (for example, for betting purposes) to predict whether a favored team has, say, a 75% or an 85% chance of winning the game.

Log likelihood is a measure of how well a statistical model explains the observed data, by comparing the probability of each prediction with the actual results. Higher log likelihood values (or, lower negative log likelihood values) indicate a model that better captures the underlying pattern of the data. It is calculated using the following formula:

$$\log L(\theta|x) = \sum_{i=1}^{n} \log f(x_i|\theta)$$

Results

The correlation matrix (Figure 1) showed very strong correlations between some variables. For example, `away_yards_per_play` correlated strongly with `away_yards_per_pass_play`, and `away_first_downs` correlated strongly with both `away_first_downs_per_play` and `away_first_downs_1_2`. Since this seemed like a redundancy of the same underlying

information, it was decided to remove four features: `away_yards_per_play`, `home_yards_per_play`, `away_first_downs`, and `home_first_downs`.

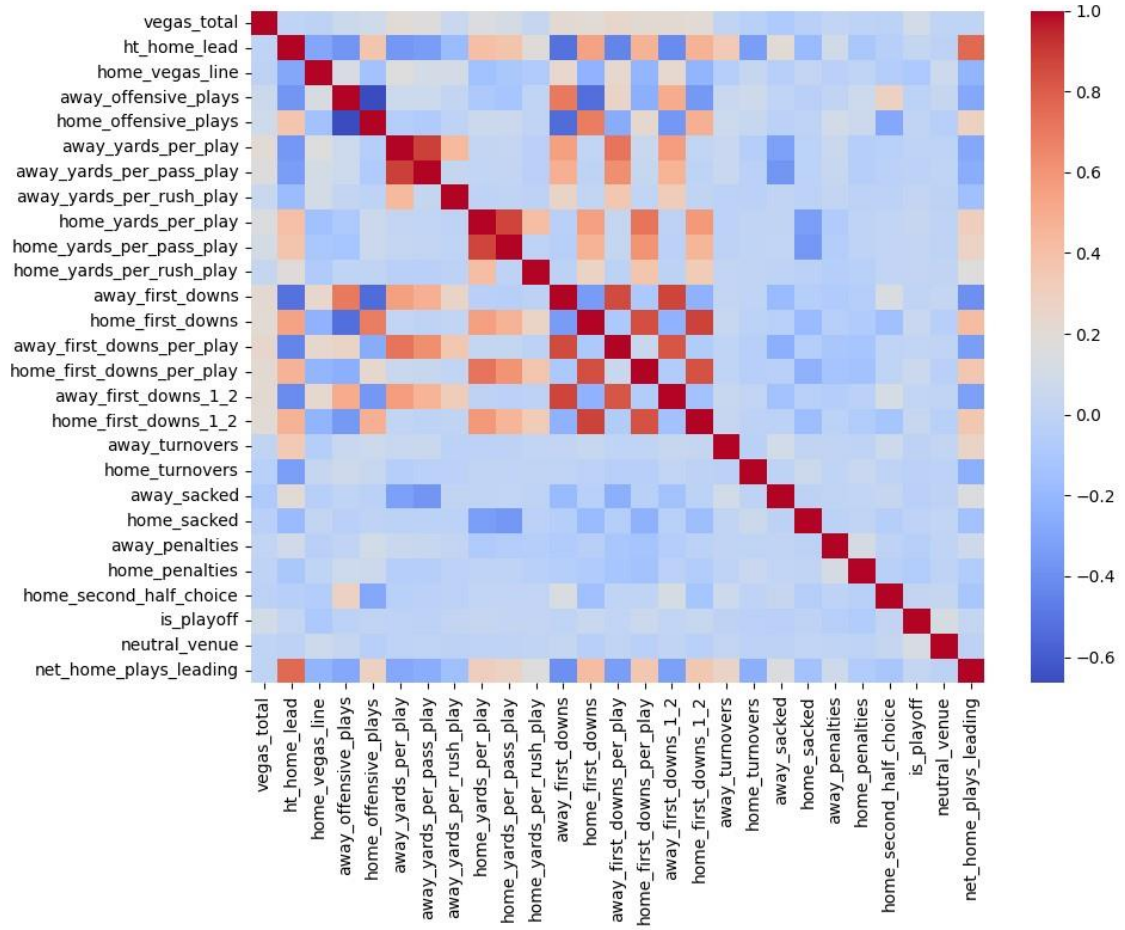

Figure 1. Correlation matrix for all features in the dataset.

Features were transformed and scaled based on their distribution, as described in the Methods section. An example of an approximately Gaussian distribution is `ht_home_lead` (Figure 2) while `away_turnovers` (Figure 3) is more skewed.

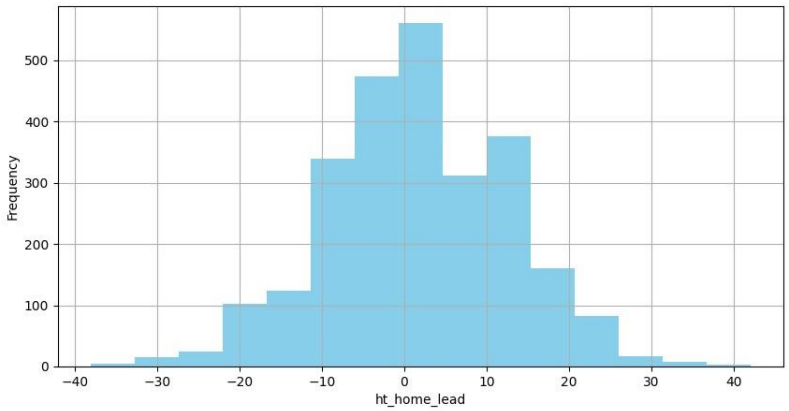

Figure 2. Histogram showing the distribution of the ht_home_lead variable.

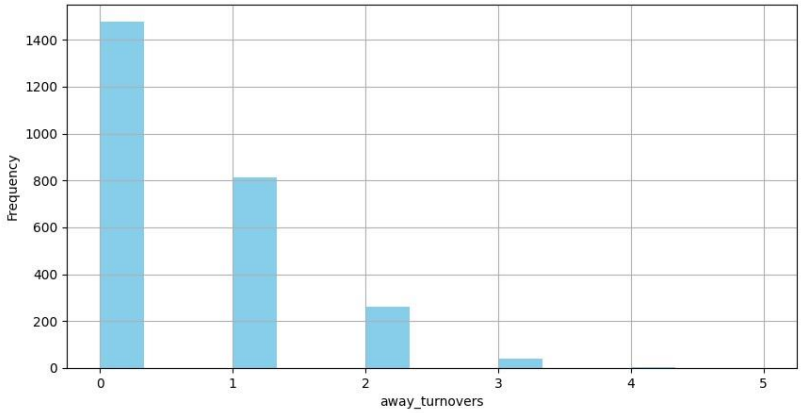

Figure 3. Histogram showing the distribution of the away_turnovers variable.

The logistic regression with backward stepwise selection method found that 5 variables were optimal for the model. These variables and their coefficients are shown in Table 2. Because the features were scaled, it is possible to compare the size of effects by the absolute value of each coefficient.

Table 2. Variables used in logistic regression with backward stepwise selection.

Variable	Coefficient
ht_home_lead	1.661
home_vegas_line	-0.754
net_home_plays_leading	-0.615
is_playoff	0.303
vegas_total	-0.130

The largest coefficient is for `ht_home_lead`, with a larger lead for the home team at half time implying an increase in the log odds of the team winning the game. There are negative coefficients for `home_vegas_line`, `net_home_plays_leading`, and `vegas_total`, meaning an increase in these variables would imply a decrease in the log odds of the team winning the game. Finally, the binary variable `is_playoff` increases the log odds by 0.303 when `is_playoff` = 1 (i.e. when it is a playoff game). Interactions were all found to be insignificant and did not improve the model.

For the lasso regression, using cross validation the optimal value for the cost parameter C was found to be 0.046. Note that in the Python function, C is the inverse of λ in the general lasso formula, and so the cost parameter was actually $\frac{1}{0.046}$ or 21.7.

After training the lasso regression, 13 of the feature coefficients shrank to zero. The coefficients of the remaining 10 variables can be seen in Table 3. Similarly to the backward stepwise selection method, `ht_home_lead` and `home_vegas_line` were the two most important predictors.

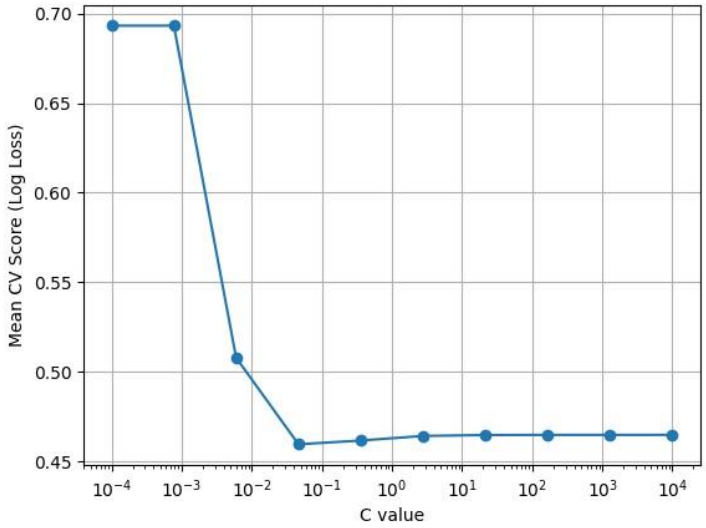

Figure 4. Plot of the log loss score of each value of C tried during cross validation for the Lasso regression.

Table 3. Variables with a nonzero coefficient in the Lasso logistic regression.

Variable	Coefficient
ht_home_lead	1.499
home_vegas_line	-0.678

is_playoff	0.172
home_second_half_choice	0.128
vegas_total	-0.081
home_yards_per_pass_play	0.034
home_yards_per_rush_play	0.030
home_penalties	0.030
home_first_downs_per_play	0.022
away_first_downs_per_play	-0.004

Applying the two final logistic regression models to the testing set, the log likelihood for the backward stepwise model was -305.36, and for the lasso it was -304.74, so the lasso performed slightly better on the holdout data.

The coefficients for the LDA, sorted by by their absolute value in descending order, can be seen in Table 4. Again, `ht_home_lead` and `home_vegas_line` are the two most important features. The log likelihood on the testing set was -305.09.

Table 4. Coefficients for the Linear Discriminant Analysis model.

Variable	Importance
ht_home_lead	1.425
home_vegas_line	-0.739
net_home_plays_leading	-0.514
is_playoff	0.341
home_first_downs_1_2	-0.151
home_first_downs_per_play	0.145
away_first_downs_per_play	-0.131
vegas_total	-0.129
home_turnovers	-0.125
away_turnovers	0.122
home_yards_per_pass_play	0.118
away_penalties	0.112
away_sacked	-0.103
home_sacked	-0.103
home_second_half_choice	0.093
home_yards_per_rush_play	0.086
away_yards_per_pass_play	0.056
home_penalties	-0.039
away_yards_per_rush_play	0.032
away_offensive_plays	-0.028
home_offensive_plays	0.025
neutral_venue	-0.019
away_first_downs_1_2	-0.005

For the random forest model, the optimal value for *m* found by cross validation was 9. However, as can be seen in Figure 5, the mean log loss was very similar for *m* = 6, and so it was decided in the context of this plot to use *m* = 6, since *m* = 9 is a bit more of an outlier relative to its close neighbors. 6 is also closer to the 'rule of thumb' for random forests which is to use m = $\sqrt{p}$ which in this case would be $\sqrt{23}$ = 4.8.

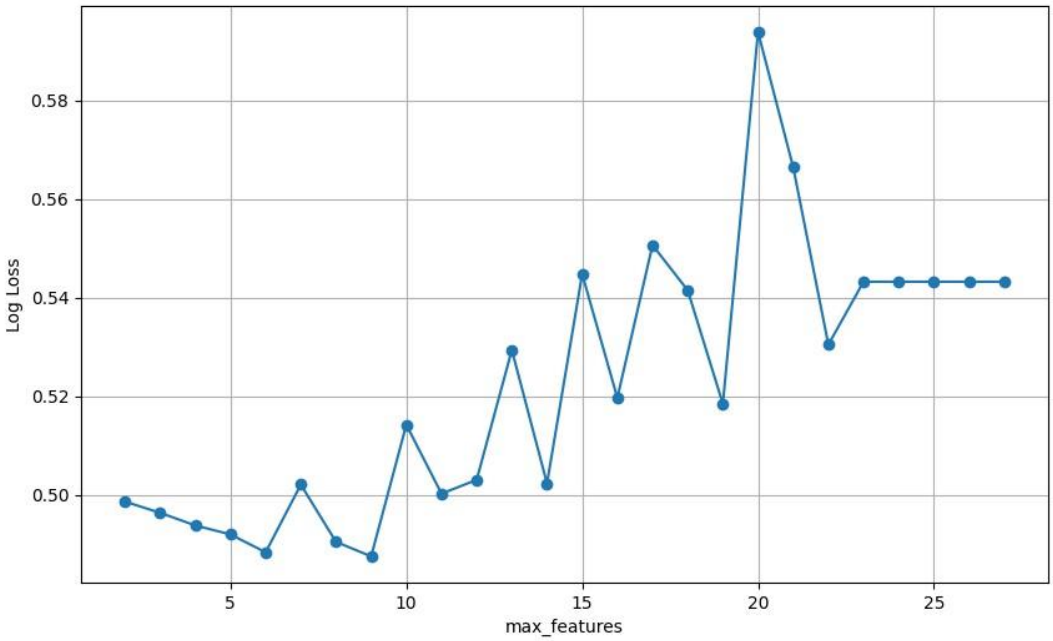

Figure 5. Plot of the log loss score of each value of m tried during cross validation for the random forest.

After training the final model with *m* = 6, it was used to make predictions on the test data and the log likelihood was -320.10. The variable importances can be seen in Table 5. Again, `ht_home_lead` and `home_vegas_line` are the two most important features.

Table 5. Variable importance for the random forest model, measured by decrease in overall node impurity.

Variable	Importance
ht_home_lead	0.213
home_vegas_line	0.109
away_turnovers	0.106
home_first_downs_per_play	0.056
away_first_downs_per_play	0.054
home_yards_per_pass_play	0.052

away_yards_per_pass_play	0.047
away_yards_per_rush_play	0.045
home_yards_per_rush_play	0.045
vegas_total	0.038
away_offensive_plays	0.033
home_offensive_plays	0.033
away_first_downs_1_2	0.030
home_first_downs_1_2	0.029
home_penalties	0.024
home_second_half_choice	0.024
away_penalties	0.015
home_sacked	0.015
away_sacked	0.012
home_turnovers	0.010
is_playoff	0.007
neutral_venue	0.004
net_home_plays_leading	0.001

For the boosting model, cross validation found the optimal parameters to be a learning rate of 0.1, max_depth 1, and n_estimators 100. The feature importances for the boosting model can be seen in Table 6. Once again, `ht_home_lead` and `home_vegas_line` are the two most important features, but it is notable how much less important the other variables are, relative to the random forest model. The log likelihood of the boosting model when applied to the test data was -310.47.

Table 6. Variable importance for the boosting model.

Variable	Importance
ht_home_lead	0.791
home_vegas_line	0.188
home_yards_per_pass_play	0.008
home_yards_per_rush_play	0.003
vegas_total	0.003
away_first_downs_per_play	0.003
is_playoff	0.002
away_yards_per_rush_play	0.001
away_first_downs_1_2	0.001
away_yards_per_pass_play	0.001
home_first_downs_per_play	0.001
away_turnovers	0.001
away_offensive_plays	0.000
home_offensive_plays	0.000
home_first_downs_1_2	0.000
home_turnovers	0.000

away_sacked	0.000
home_sacked	0.000
away_penalties	0.000
home_penalties	0.000
home_second_half_choice	0.000
neutral_venue	0.000
net_home_plays_leading	0.000

The optimal hyperparameters found for the neural network are shown in Table 7. When applied to the test set, the log likelihood for the neural network model was -308.54.

Table 7. Optimal hyperparameters for the neural network model, found by cross validation.

Parameter	Optimal Value
learning rate	0.001
epochs	10
batch size	32
hidden layer size	150
dropout rate	0.1

For the principal component regression, cross validation found the optimal number of components to be 21 (see Figure 6). The log likelihood on the test set was -307.06.

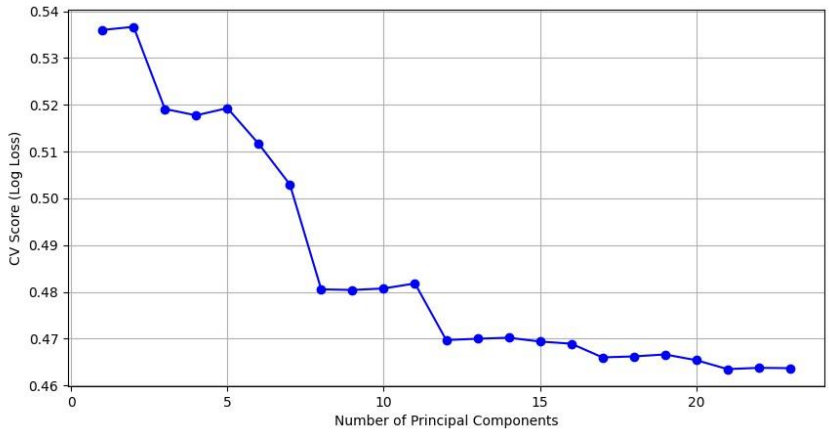

Figure 6. Plot of the log loss score of each number of principal components by cross validation.

A comparison of the log likelihood for each model on the test set can be seen in Table 8. The Lasso regression performed best with a log likelihood of -304.74, followed by the other linear methods. Random forest was the poorest performer, with a log likelihood of -320.10.

Table 8. Comparison of the performance of the 7 different models.

Model	Log Likelihood
Logistic Regression (Lasso)	-304.74
Linear Discriminant Analysis	-305.09
Logistic Regression (Stepwise)	-305.36
Principal Component Regression	-307.06
Neural Network	-308.54
Boosting	-310.47
Random Forest	-320.10

Discussion

It is interesting to see that the best performing models for this task were the more simple linear models such as logistic regression and LDA, whereas the techniques that allow for more complex interactions between variables performed less well. Why might this be the case? When there are a large number of variables with limited predictive power, or variables that are related to one another, overfitting can be a common problem. To overcome this limitation in a future repeat of this project, it may be possible to use feature engineering to create a smaller number of features that capture the same underlying information from the 23 features used in this project.

The most important variables were found by all models to be `ht_home_lead` and `home_vegas_line`. It comes as little surprise to learn that the two most important factors are the score and the relative quality of the two teams. In the best-performing models (logistic regression and LDA), the next most important features were `net_home_plays_leading` and `is_playoff`. The models found that `net_home_plays_leading` is negatively related to the log odds of `home_win`, while `is_playoff` is positively related.

It should be pointed out that these relationships exist in the context of all other variables being held equal. So, when the score and the quality of the teams is held equal, the more time a team has been leading, the less chance they have of winning. Some possible reasons for this are that it is mentally tiring to play with the lead for long periods, or that the opposing team may have momentum if they trailed for a long stretch and have recently taken the lead. The `is_playoff` result implies an extra second half advantage for home teams in the playoff, over and above the home advantage inbuilt in the pre-match Vegas line. This could possibly be due to the especially raucous home atmosphere in playoff games, and may be worthy of further investigation.

Overall, this project reminds us that despite the number of sophisticated nonlinear models available today, sometimes a simple linear approach still gives the best results when generalizing outside of the training data.

References

1. May, C., Dan, M., Carl, M., Ralph, A., & John, H. (2010). Rush versus pass: modeling the NFL. *Journal of Quantitative Analysis in Sports*.
2. Warner, J. (2010). Predicting margin of victory in NFL games: machine learning vs. the Las Vegas line. *Published on: Dec*, *17*.
3. Blaikie, A. D., Abud, G. J., David, J. A., & Pasteur, R. D. (2011). NFL & NCAA football prediction using artificial neural networks. In *Proceedings of the Midstates Conference for Undergraduate Research in Computer Science and Mathematics, Denison University, Granville, OH*.
4. Uzoma, A. O., & Nwachukwu, E. O. (2015). A hybrid prediction system for American NFL results. *International Journal of Computer Applications Technology and Research*, *4*(01), 42-47.
5. Pelechrinis, K. (2017). iwinrnfl: A simple, interpretable & well-calibrated in-game win probability model for NFL. *arXiv preprint arXiv:1704.00197*.