Machine Learning the NFL Playoffs

KEVIN SU and LEO LUNG, Carnegie Mellon University

Sports predictions are a common form of entertainment in our society. Prediction tournaments are held around us all the time such as online fantasy sports teams, March Madness brackets among friends, and sports betting. In most cases, experts make predictions based on past statistics. Most of these methods are algorithmic in nature. This led us to believe that we can apply machine learning and artificial intelligence to sports prediction. American football is the most watched sport in the US, so we based our project around the National Football League (NFL). Our main goal was to predict the postseason game outcomes of the NFL using the team statistics from the regular season.

Categories and Subject Descriptors: I.2.6 [**Learning**]: Machine Learning—*Decision Trees and Neural Networks*

General Terms: Machine Learning

Additional Key Words and Phrases: Machine learning, NFL, decision trees, neural networks

ACM Reference Format:

Kevin Su and Leo Lung. 2013. Machine Learning the NFL Playoffs. *ACM Trans. Appl. Percept.* 1, 1, Article 1 (May 2013), 11 pages.
DOI:http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

Our main goal was to build a model to predict NFL playoff outcomes based on regular season statistics to each team. We hoped to build a classifier that could accurately predict the results of the post season playoffs. However, given the nature of sporting events, we understood that there is an upper bound on accuracy. It is unlikely that we would be able to predict player performance and achieve a very high level of accuracy. We used several benchmarks in checking our results which we discuss later.

1.1 Motivation

The question we were most interested in answering was "Can we predict the National Football League playoffs?" The NFL playoff games are the most widely watched sporting events in the US. Additionally, in the US, sports betting generally centers around the NFL, especially the post season. Given the popularity of football here, it seemed natural to begin with the NFL.

This work is done as a term project for 15-780 Graduate Artificial Intelligence at Carnegie Mellon University, taught by Tuomas Sandholm (sandholm@cs.cmu.edu) and Manuela Veloso (mmv@cs.cmu.edu).

DOI:http://dx.doi.org/10.1145/0000000.0000000

Additionally, if the NFL can accurately be predicted with machine learning techniques, those techniques might also translate directly to other sports such as the NHL or MLB. Since sports betting is extremely prevalent in the culture of professional athletics, there would be great interest in an accurate classifier. Heuristics that work for American football might have analogs in other sports, and lastly, if the results are not as successful as we want them to be, we hope to be able to offer insight into further work.

1.2 Goals

As stated, we wanted to accurately predict the NFL playoffs. This would require us to collect data, build models, and improve our results. While the main goal was to predict the results, we also had the following goals:

1). Collect data over past seasons
2). Build a program to process the data
3). Train a model over the data
4). Improve the accuracy of our classifier

1.3 Organization of the Paper

In Section 2, we first discuss some background topics to familiarize the reader with the structure of the NFL playoffs as well as basic machine learning techniques. In Section 3, we cover the methodology we used in building classifiers. The results are covered in more detail in Section 4. We give our conclusions in Section 5, and lastly in Section 6 we offer some comments on future work that might be done.

2. BACKGROUND

2.1 Weka

We used the Weka (Waikato Environment for Knowledge Analysis) machine learning suite for our analysis. Weka is written in Java, and is developed at the University of Waikato, New Zealand. Since Weka is free software under the GNU General Public License, we thought it was an appropriate choice for our project given the scope and timeframe. The Weka suite contains a huge collection of algorithms and tools for visualizing data analysis and predictive modeling, backed by a user-friendly graphic user interface that is easy to use and understand.

2.2 ARFF

Weka uses a file format called the Attribute Relationship File Format (ARFF) to store data in a database. The file structure is quite simple. For example, if we were to describe a ”car” in an ARFF file, it would look like:

```
. @ relation car
. @ attribute make {Honda, Toyota, Chevy}
. @ attribute length real
. @ attribute width real
. @ attribute frontwheeldrive {TRUE, FALSE}
```

The ARFF file is made up of a header section and a data section. The first line of the header is where the relation name is. Following that is a list of the attributes, where each attribute consists of a name and a type. The name has to be unique for obvious reasons. The type describes what kind of a variable it is and also the potential values it can be. There are four types of variables: numeric, nominal, string, and date.

2.3 Football Overview

From Wikipedia: American football, known in the United States as football, is a team sport. It is played by two teams, eleven players to a side, who advance an oval ball over a rectangular field that is 120 yards long by 53.3 yards wide and has goalposts at both ends. The team in possession of the ball (the offense) attempts to advance down the field by running with or passing the ball. To continue their drive, the offense must advance the ball at least 10 yards down the field in a series of four downs. If they succeed, they receive a new set of four downs to continue their drive. Otherwise, they lose control of the ball to the opposing team. The offense can score points by advancing the ball into the end zone (a touchdown) or by kicking the ball through the opponent's goalpost (a field goal), while the defense can score points by forcing an offensive turnover and advancing the ball into the offense's end zone or by tackling the ballcarrier in the offense's end zone (a safety). The team that has scored the most points by the end of the game wins.

2.4 Football Team Statistics

Pts. Points Scored by Team

Yds. Total Yards Gained

Ply. Offensive Plays: Pass Attempts + Rush Attempts + Times Sacked

Y/P. Yards per Offensive Play

1stD. Number of First Downs

Passing Cmp. Passes Completed

Passing Att. Passes Attempted

Passing Yds. Yards Gained by Passing

Passing TD. Passing Touchdowns

Passing Int. Interceptions Thrown

Passing NY/A. Net Yards Gained per Pass Attempt

Passing 1stD. First Downs by Passing

Rushing Att. Rushing Attempts (sacks not included)

Rushing Yds. Rushing Yards (sacked yardage not included)

Rushing TD. Rushing Touchdowns

Rushing Y/A. Rushing Yards per Attempt

Rushing 1stD. First Downs by Rushing

TO. Team Turnovers

FL. Fumbles Lost by Player or Team

1stPy. First Down Penalty

3. METHODOLOGY

3.1 Gathering Data

We successfully found a website that hosts all relevant football statistics for over 50 seasons. Our initial plan was to get regular season data from each team that made the playoffs in that season, so we could try to predict the postseason results based on the regular season statistics. However, there was no good way to compile all of the data, as each team's data was hosted on a different web page. Thus, we resorted to manually compiling the results into an Excel spreadsheet, then exporting a CSV file which is used as the input to our program.

At first, we gathered the regular season statistics for each team that made the playoffs from 2009 to 2012, which is 4 seasons worth of statistics. All of these team statistics are pretty standard; each team has separate offensive and defensive stats. The offensive and defensive statistics are those that are described above. Most of these statistics are easy to understand for anyone that knows the basic rules of football.

In addition to team statistics, we also recorded all postseason game results and scores, including which team had the home field advantage. We had roughly 40 games worth of data. This data is compiled into a single spreadsheet, which is then parsed by the program.

After we finished data collection, we hypothesized that we might actually need more data. As we will mention later on, we will be relying on decision trees and neural networks to machine learn the data. Given that we only have 40 games and 4 seasons of team data, there is a very real possibility that the neural network may find unreliable patterns (such as "if the total yards scored is greater than 10 times the rushing attempts against the same teams defense, then this team has an advantage"). Since the sample size was relatively small, there was a good chance that standard machine learning algorithms would overfit the data. Such unreliable patterns would have much less of an effect if the sample data set was larger in size. To compensate, we tripled the size of our data set, collecting data for the past 12 seasons. We now had data for 132 games, which we thought would improve our accuracy.

To conclude the data collection phase, we wrote a Java program that parsed the recorded CSV data into an ARFF file for Weka.

3.2 Machine Learning Models

Fundamentally, there were two things we wished to be able to predict: the winner of the game and the point differential. We defined the point differential to be the home team's score minus the away team's score. Additionally, we wanted a way to benchmark our results.

For the binary classifier, we were able to compare the results to always choosing the home team, which is favored in all playoff games except the Super Bowl. This is equivalent to always choosing the higher seeded team. For the point spread, it was more difficult to find a benchmark. However, we decided to calculate the average error as well as the standard deviation. There is a lot of literature on point spreads and their accuracy, and we decided to compare our results to the predictions of Las Vegas sports betting.

Note that the two classifiers are both useful in their own right. For predicting the winner, even though it might be relatively trivial if the favored team wins a large percentage of the time, one could apply these algorithms to a bracket-style competition such as NCAA's March Madness tournament for collegiate basketball. For March Madness, the ubiquitous competition is to predict the winners in the form of a bracket. In that case, point differentials would be irrelevant, and we would concern ourselves only with predicting the winner of each game. However, for sports betting on NFL games, it is almost universally done by a point spread, where the goal is to predict how much a team will win or lose by. In this case, the point differential trumps winner prediction.

3.3 Heuristics

One of the first things we noticed was that while we had over 80 signals, our initial dataset consisted of just 42 games. We knew that this would lead to excessive overfitting, even with automated pruning. The first step was to gather more data, which we did. By tripling our data size, we hoped to reduce irrelevant patterns in the data which would hurt our classifiers' accuracies. However, even with 132 games, we reasoned that there might still be a significant amount of overfitting. To alleviate this, we came up with some heuristics.

The key insight was that we expected many of the signals to be uncorrelated with each other. For example, a teams average rush yards allowed per play is unlikely to be related with the number of passing attempts they had on offense. We would actually prefer our classifiers to not compare these signals in anyway. More importantly, that statistic is most likely correlated to the other teams average rush yards achieved per play. These two statistics, one teams rushing defense and the others rushing offense, are very likely correlated to each other, in terms of the outcome of the game. We wanted a way to inherently compare the two and take that into account. We combined several of these signals with basic operations such as adding the two together or multiplying them to create new signals. Most of the offensive statistics have inverses as defense statics, which allowed us to reduce our signal set down to around 40 signals.

In short, we combined relevant signals together to create a smaller set of what we believed to be stronger signals. An important thing to note is that in doing so, we used our knowledge of football to manually combine signals. This should improve the accuracy of our classifiers by modeling the data in a more domain-specific manner.

3.4 Training the Models

We used the Java machine learning library Weka to train the models. We wrote a program in Java to parse the data we collected. After representing the data in our program, we processed it using our heuristics to produce an ARFF file.

3.4.1 *Binary Classifier.* We decided to use decision trees to try to classify the winner of the game. For the binary classifier, each instance has a "winner" attribute which can take one of two values, *HOME* or *AWAY*. While decision trees make cuts perpendicular to the feature set, we thought that given that our heuristics combined relevant signals, we might achieve a reasonable amount of accuracy here. As for a specific algorithm, we used C4.5, which is the successor to ID3. C4.5 made several improvements on ID3, notably the ability to handle continuous attributes. It does so by creating a threshold for the attribute and splitting the training set over the threshold. Additionally, C4.5 post-prunes the resulting tree, replacing unhelpful nodes with leaf nodes.

First, we examined the benchmark that we would be comparing our results to. We counted the number of games in which the favored (home) team won. In our testing set, the favored team won 62.22% of the games. We made sure to compute this over the same data we were going to test on.

We first built a classifier without our heuristic. Using a standard $\frac{1}{3}$ withhold of the data set for testing, we found our classifier to correctly classify 44.44% of the testing set. Note that this is extraordinarily poor. We could take the inverse of each classification and achieve an accuracy of almost 56%. Even if we flipped the classifier, we still could not do better than choosing the favored team.

The next step was to experiment with the heuristic. Using the same method but with the heuristic, we found that the resulting classifier had an accuracy of 71.11%. This was a significant improvement over the previous result. We saw a nearly 27% increase in accuracy. Notably, we were now able to do slightly better than a predictor that always chose the favored team.

3.4.2 *Point Differential.* For predicting the point differential, we needed to choose a different type of classifier. We needed the classifier to take the continuous valued feature set as input and produce a single continuous value as output. We thought the features could likely be linearly compared to each other. For example, a possible approach would be to check if one team's offensive passing statistic was greater than another team's offensive passing statistic. Given this, we decided to use a multilayer perceptron network. Neural networks excel at modeling linear functions of continuous data, and additionally, with a sigmoidal function, we should be able to output a single continuous prediction.

Without the heuristic, we found our neural network model to have a mean absolute error of 19.5 points. However, with the heuristic, we were able to reduce the mean absolute error to 17.6, almost 2 whole points better. The standard deviation of the error in our results was 20.2, which we thought to be rather high.

4. EXPERIMENTAL RESULTS

Binary Classifier

—The benchmark of choosing the favored team had an accuracy of 62.22%

—The decision tree trained without heuristics had an accuracy of 44.44%

—With our heuristics, we were able to improve the accuracy to 71.11%

Point Differential

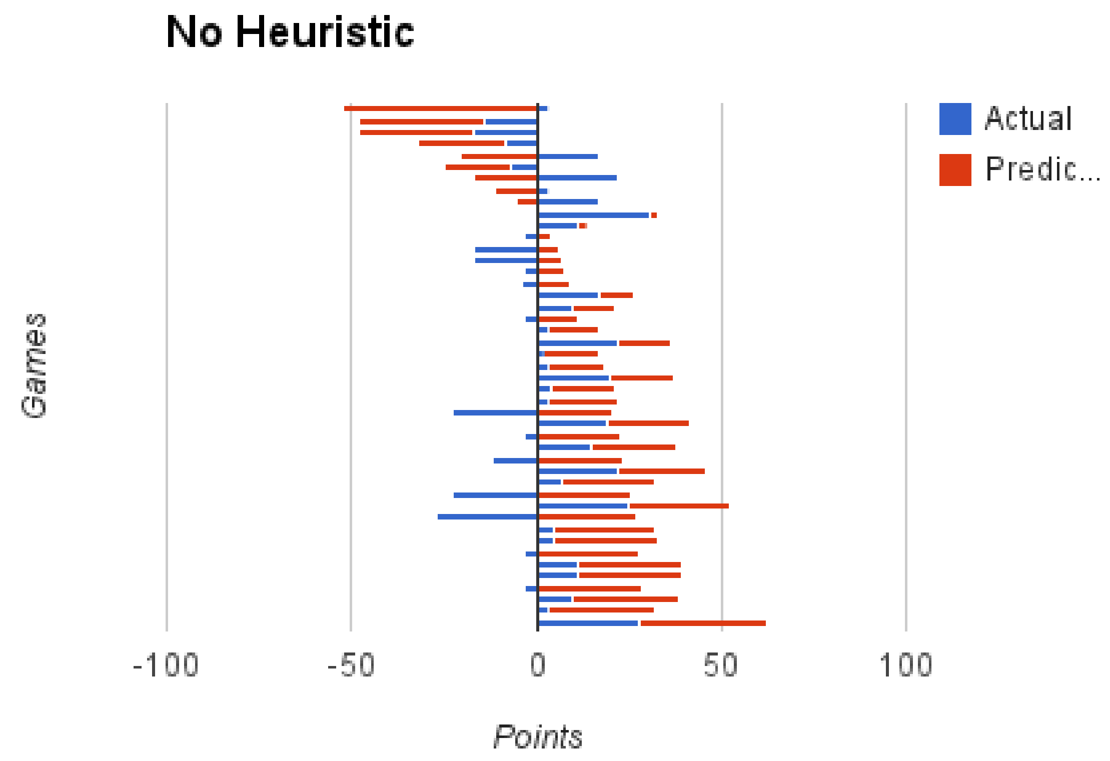
—Las Vegas sports betting can generally achieve a mean error of -1.6 with a standard deviation of 13.3 points (over a sample of 240 games)

—Our neural network without the heuristics had a mean absolute error of 19.5 points. The mean error was 6.9 with a standard deviation of 22.8

—With our heuristics, we were able to improve the mean absolute error to 17.6 points. We found a mean error of -4.48 with a standard deviation of 20.2

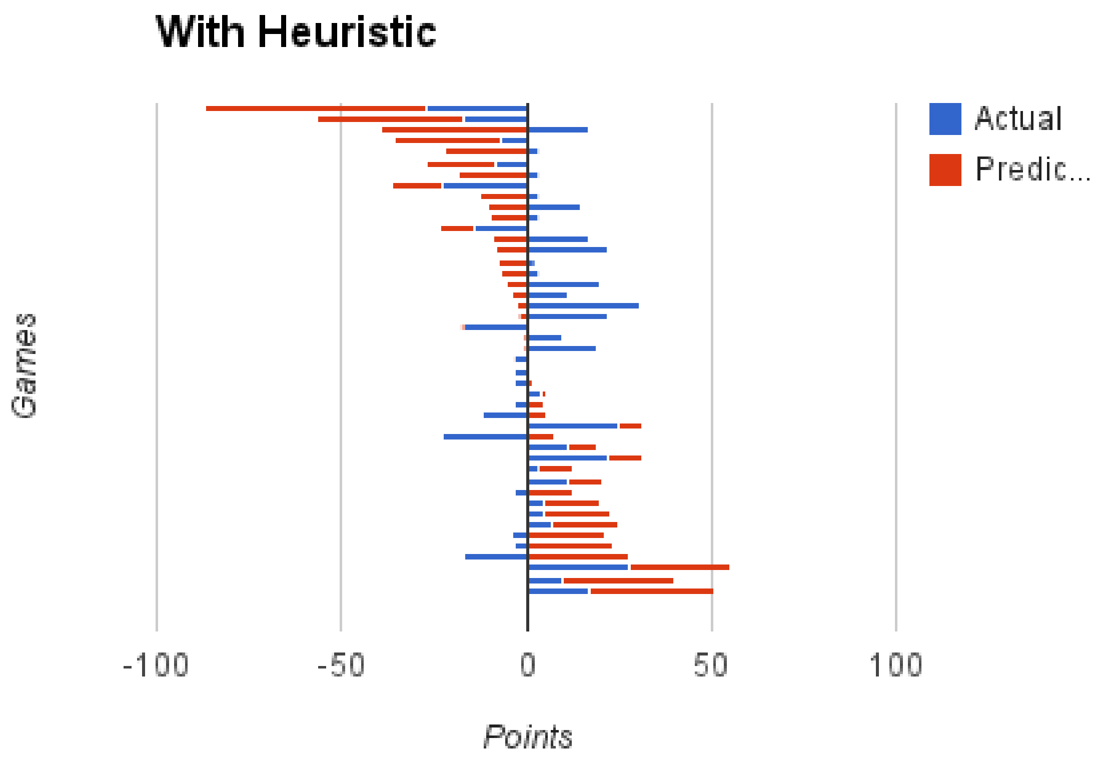
A text version of the decision tree is given in the Appendix. Numbers wise, we see that for predicting the winner, we did quite well. We were able to do slightly better than the benchmark, and the heuristics improved our results greatly. On the other hand, for predicting the point differential, Las Vegas sports bettors do significantly better than our model, and the heuristics only slightly improved the results.

We give a visualization of the errors for the point differential below, generated without our heuristic:



The games are sorted by predicted score. The point differential is measured in points, and is on the x-axis. The predicted score is depicted in red, and is stacked on top of the actual score. An interesting thing to note is that most of the large errors occur near the middle, where our model predicts a win by 10-15 points for the home team but the away team wins by 15-20 points. Additionally, our model seems to predict exaggerated point totals. Note that in general the red bars are larger than the blue bars. In one case it predicted the away team would win by more than 50 points, whereas any NFL aficionado would ridicule that prediction.

With the heuristic, we obtained the following graph:



An interesting thing to note is that there are more errors in the top half. This means the model predicts an upset, whereas the actual result is the home team wins by a small to medium margin. Also, there are more win/loss errors with the heuristic (roughly 20 compared to about 10 without the heuristic). Additionally, the predictions are even more exaggerated, with one prediction being near a -80 point differential. However, the mean errors are smaller as is the standard deviation. We would not necessarily call this model better, since it is still fairly inaccurate.

5. CONCLUSIONS

At the beginning of the project, we originally thought the NFL playoffs to be unpredictable. With our heuristics, we were able to somewhat accurately predict the winner. Unfortunately, we were unable to predict the point differential to any acceptable level of accuracy.

However, we were not able to achieve the accuracy we desired. We only did slightly better than choosing the favorite. In manually inspecting the games where our classifier did poorly (particularly with predicting the points), we found that in some games in which the error was greatest, the anomaly was caused by unpredictable events such as horrendous player performance. This led us to believe that a large percentage of the error is due to events not predictable from our statistics (such as poor player performance and injuries). In the end, we found that there was still some room for improvement in binary classification and even more so in point prediction.

We set out to answer several questions when we started the project. Through our work, we found that we were able to satisfactorily answer these questions. Our experiments offered much insight into the following:

—**Can NFL games be predicted with any reasonable amount of accuracy by machine learning?** In our experiments, we found that it might be possible to predict the winner of games accurately. We were able to do slightly better than choosing the favorite. With additional heuristics, more data, and more signals, we think that it should be possible to predict the winner with a high degree of accuracy. On the topic of predicting the points, we believe our current approach is ill-suited for predicting the point differential.

—**Given our model over all the seasons but one, how well did we do?** We wanted to see if we could predict the 2012 season. While we had the Baltimore Ravens (Super Bowl champions) correctly winning their first round, we had them losing to the Broncos in their second game, which they won.

—**What models work best for sports?** We tried to use decision trees for binary classification. One of the main issues with this approach is that decision trees split the dataset perpendicular to the feature set. That is, it does not compute any explicit relations between the signal set. We overcame this with domain specific heuristics, but if we wanted to extend this to other domains, we might want to look at feature selection and regression-based algorithms. We thought neural networks would be well suited to predicting the score (continuous input, continuous output), but we did not have very successful results with that.

—**What features are actually relevant? Are there statistics that don't matter?** Not surprisingly, we found that the root node (which is the node with the most gain in C4.5) was the touchdown statistic. This correlates with our intuition, since touchdowns are the primary method of scoring points in football. Other important statistics were passing yards and first downs, which football enthusiasts will tell you are traditionally very important statistics. Our findings are inline with conventional knowledge. Interestingly, the rushing statistics were pruned from the tree.

6. FURTHER COMMENTS

There are many areas for improvement and future work in this project. We present a couple that we wish to have implemented in this project, but were not able to due to time constraints.

The first logical step is to further increase the size of the data set. Even though we tripled our data set midway through the project, we still had a tiny dataset, consisting of only 132 games. However, we also have to keep in mind that older statistics may not necessarily improve our results. The game of football has changed notably over the past few decades, and statistics and patterns that worked for teams 30 years ago will probably not apply to modern day football.

Since increasing the data set by including more football seasons may not necessarily prove fruitful, we could also start by including player statistics as well. This wwould greatly increase our data sets, but wwould also increase the difficulty in data collection. Obtaining player statistics is easy, but to include them in the appropriate games is hard. We would need full roster and play time for each regular season game in order to have player statistics be meaningful.

A large part of football analysis deals not with pure statistics, but with non-numerical data as well. Modeling the matchup more accurately by including non-numerical data such as coaching proficiency or player match-ups might help. Additionally, there are other factors for the game such as weather. While New England is used to playing in Foxborough, MA, the Miami Dolphins would be ill-suited to play there. This factor could be amplified many times over if there was a significant amount of snow on game day. Additionally, some teams have trouble at Denver due to the mile-high altitude.

By increasing the size of the data set both by harvesting more games and including more features, we reason that the classifiers could be significantly improved.

APPENDIX

We give the text format of the decision tree for the binary classifier below. While the numbers are not sensible since they are developed from our heuristics, it may be of interest to see which signals have the most gain. Specifically, touchdowns are at the root of the tree, which is to be expected since they are the primary method of scoring points.

```
to_h <= 0.548387
|   pass_nypa_h <= 1.296875: AWAY (12.0)
|   pass_nypa_h > 1.296875
|   |   pass_nypa_a <= 1.029851: AWAY (4.0)
|   |   pass_nypa_a > 1.029851: HOME (6.0)
to_h > 0.548387
|   fdpy_h <= 1.428571
|   |   fdpy_a <= 0.966667
|   |   |   pass_cmp_a <= 1.006757: HOME (18.0)
|   |   |   pass_cmp_a > 1.006757
|   |   |   |   fl_h <= 0.714286: HOME (7.0)
|   |   |   |   fl_h > 0.714286
|   |   |   |   |   fl_h <= 0.923077: AWAY (4.0)
|   |   |   |   |   fl_h > 0.923077
|   |   |   |   |   |   rush_nypa_h <= 1.027027: HOME (8.0)
|   |   |   |   |   |   rush_nypa_h > 1.027027
|   |   |   |   |   |   |   pass_cmp_a <= 1.032448: HOME (2.0)
|   |   |   |   |   |   |   pass_cmp_a > 1.032448: AWAY (4.0)
|   |   fdpy_a > 0.966667
|   |   |   ypp_h <= 0.943396: HOME (6.0)
|   |   |   ypp_h > 0.943396
|   |   |   |   yds_h <= 1.043418: AWAY (8.0)
|   |   |   |   yds_h > 1.043418
|   |   |   |   |   pass_td_a <= 1.642857: AWAY (35.0/15.0)
|   |   |   |   |   pass_td_a > 1.642857: HOME (6.0)
|   fdpy_h > 1.428571: HOME (12.0)
```

Bibliography

—"Multilayer Perceptron." Wikipedia. 2013.

—Pro Football Reference. http://www.pro-football-reference.com 2013.

—Quinlan, J. R. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, 1993.
—Weka. The University of Waikato. http://www.cs.waikato.ac.nz/ml/weka/ 2013.