# Projecting NFL Quarterback Readiness

**Amit Patankar** **Aseem Monga**  
Google Inc. Mountain View, CA NetApp, Sunnyvale, CA  
[amit.anil.patankar@gmail.com](mailto:amit.anil.patankar@gmail.com) [amonga@stanford.edu](mailto:amonga@stanford.edu)

## Abstract

The quarterback is the most important position on an NFL team. Teams often spend first round draft picks to potentially draft a future franchise quarterback. Every now and then some teams find themselves investing in a very promising prospect only to find out later that he is a bust. Our goal is to predict if a quarterback is a bust based on a player’s history, college stats, and the team that drafts him at a certain round and pick. Using a deep neural network we were able to predict with 73% test accuracy whether or not a quarterback drafted would be NFL-ready or a bust.

## Introduction

The NFL is the largest sports organization in the United States with 32 teams and nearly 200M viewers worldwide. Millions tune in each year to watch the NFL draft where teams who perform poorly in the last season have the opportunity to draft the most promising prospects from college football. Poor-performing teams can reel fans back in and boost merchandise sales by drafting new exciting players. Usually the quickest way to improve a team is to draft a franchise quarterback in the first round. A franchise quarterback is a starting quarterback who is usually the best player and face of the team. Notable examples include Tom Brady of the New England Patriots or Peyton Manning of the Indianapolis Colts (both who were selected by their respective teams in the NFL draft and turned their teams into perennial championship contenders). On the other side of the coin, every now and then an extremely talented player will be drafted early, fail to find their bearings in the NFL, and disappoint a franchise and millions of fans. Two of the biggest draft busts of all time were Ryan Leaf (picked 2nd in 1998 by the San Diego Chargers) and JaMarcus Russell (picked 1st in 2007 by the Oakland Raiders). At the time, both seemed like the right pick, but they both had red flags that some analysts were able to pick up. We wanted to see if a neural network could take these objective metrics and predict if a successful college quarterback is likely to be a bust or not.

## Related Work

It was very difficult to find machine learning approaches to the exact problem that we were trying to solve. Similar work in CS299 included “Machine Learning for Daily Fantasy Football Quarterback Selection” where the authors P. Dolan, H. Karaouni, A. Powell attempt to rank the best quarterback for daily fantasy sports. The only real useful metric we can derive from this approach was feature selection where they included similar passing and rushing metrics. Seonghyun Paik wrote a promising paper titled “Building an NFL performance metric”, but once again the analogous features in college data were next to impossible to find and collect in a short amount of time. “Using Machine Learning to Predict NFL Games” by E. Jones, C. Randall was also instrumental for data sources.

## Dataset and Features

We started by looking at a quarterback’s college statistics, their college, and the conference they played in. We also decided that some quarterback’s were more likely to do well on certain teams rather than others. For example the Carolina Panthers offense relies on a mobile quarterback such as Cam Newton whereas a pocket passing quarterback would find more success with a team like New England or Denver. Based on this observation we added the team that drafted a quarterback as a feature in our model as well. One of our more controversial decisions was whether to include the round and selection of a player as a feature. Many would argue a quarterback’s value should be irrespective of those features, but our logic is that the earlier you select a quarterback, the more likely you are to invest playing time and resources into them. This would potentially elevate a mediocre quarterback over a talented one.

Originally we wanted to predict a player’s actual rookie year performance in the NFL. After running experiments with data, we found that we had a high variance problem and our model was over-fitting to noise patterns that were not correlated to our input features. Rookie performance is also not necessarily an accurate indicator of future success. Jared Goff was the 2016 #1 pick and had a mediocre winless season with the Rams in his rookie campaign, but has turned it around with an impressive 9-3 (as of 12/08/2017) record this season. We then switched our criteria for NFL-ready vs. bust as a player who recorded 10 wins in their entire career as a starter. This criterion filtered out poor rookie performances and injuries and gave more weight to overall success. We also found that this criterion accurately classified many notorious NFL draft busts.

### Feature Set

An entry in our training dataset is about a player’s history, college stats, and the team that drafts him at a certain round and pick. It contains the following information

* **Player:** Name of the player
* **College:** Most recent college attended
* **Conference:** Athletic conference of the most recent college attended
* **Team:** NFLteam which drafted the player
* **Heisman:** 1 if player was awarded the Heisman trophy, 0 otherwise
* **Completions:** Pass completions
* **Attempts:** Pass Attempts
* **Yards:** Passing Yards
* **Touchdowns:** Passing Touchdowns
* **Interceptions:** Passing interceptions
* **Rush Attempts:** Rushing Attempts
* **Rush Yards:** Rushing Yards
* **Rush Touchdowns:** Rushing Touchdowns

Here is an example of what our dataset looks like:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Player** | **College** | **Conference** | **Team** | **Heisman** | **Classification (Bust, NFL-Ready)** |
| **Jameis Winston** | Florida St. | Atlantic Coast | TAM | 1 | 1 |
| **Marcus Mariota** | Oregon | Pac-12 | TEN | 1 | 1 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Completions** | **Attempts** | **Yards** | **Touchdowns** | **Interceptions** | **Rush Attempts** | **Rush Yards** | **Rush Touchdowns** |
| 562 | 851 | 7964 | 65 | 18 | 145 | 284 | 7 |
| 779 | 1167 | 10796 | 105 | 14 | 337 | 2237 | 29 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Draft Year** | **Round** | **Pick** | **Age** | **Games Played** |
| 2015 | 1 | 1 | 21 | 27 |
| 2015 | 1 | 2 | 21 | 41 |

\* above denotes college data

### Training and Test Set

Our training dataset has information of 150 plus quarterback’s that got drafted between year 1998 and 2015. Test Set consists of quarterback’s that got drafted in the year 2016.

### Preprocessing

Before feeding data to our machine learning algorithms, we went through a series of preprocessing steps.

* Text to numerical: One encoding to convert College and Team names which resulted in embedded vectors.
* Dropping Features: Date, time and the venue of the NFL draft are highly unlikely to have an impact on the readiness of the player hence we dropped these features.

Following these preprocessing steps, we ran some out-of-the box machine learning algorithms as a part of our initial exploratory steps. Our new feature set consisted of 7 features, all of which were now numeric in nature.

### Feature Addition

As we plunged deep into the problem, we felt that our dataset wasn’t complete enough to predict the readiness of a quarterback. To improve our feature set, we added Conference and Heisman features to our dataset. We felt that the addition of these features could improve our performance at measuring the readiness of a player.

Kaggle, UCIMLR, and the NFL don’t have this data in clean datasets, although plenty of individual data points are out there. Since our population size is roughly small, we decided the best way to do data collection was to manually look up features for each quarterback drafted. Using a simple filter feature selection algorithm, we noticed the college and draft age played almost no role in our performance and thus we removed them from our final model.

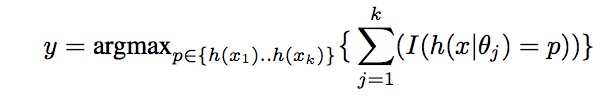
## Methods

After preprocessing our data and nailing down on our feature set, we processed to tackle our problem with an assortment of classification algorithms. The following sections explain the model we used in details

### Random Forest

Random Forests is a ensemble learning method, which builds a list of classifiers on the training data and combine the predictions of several base estimators built with a given learning algorithm in order to improve generalizability/robustness over a single estimator. Hence, the Random Forest algorithm is a variance-minimizing algorithm that utilizes randomness when making split decision to help avoid over-fitting on the training data.

It aggregates a family of classifiers ,,....,. Each classifier is a classification tree and m is the number of trees chosen. Each is a randomly chosen parameter vector. If T(x, y) denotes the training dataset, each tree in the ensemble is built using a different subset T(x, y) ⊂ D(x, y) of the training set. Each tree partitions the data based on the value of a particular feature (which is selected randomly from the subset), until the data is fully partitioned, or the maximum allowed depth is reached. The output y is obtained by building the results thus:



### Support Vector Machine

SVM is a traditional supervised learning model, which tries to find the maximal separating hyper-plane between two sets of points. For classes where y = {0, 1}, we will use parameters w, b, and write our classifier as

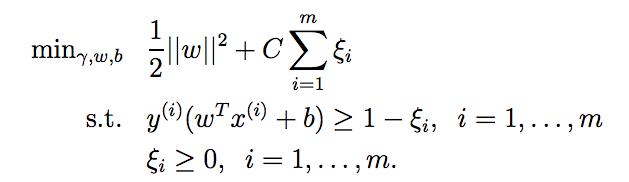
Macintosh HD:Users:ajm:Desktop:Screen Shot 2017-12-12 at 18.46.02.png

Here, g(z) = 1 if z ≥ 0, and g(z) = −1 otherwise.

We experimented with the following set of kernels

* Linear Kernel
* Polynomial Kernels (2nd and 3rd degree polynomials)
* RBF Kernels

The optimal margin classifier with L1 regularization is given as

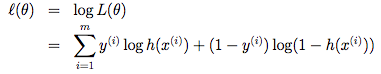


### Logistic Regression

Logistic regression predicts probabilities; rather than just class labels; hence we can fit the model using likelihood. For each training data-point, we have a vector of features, xi, and an observed class, yi. Where sigmoid function is used as hypothesis

Macintosh HD:Users:ajm:Desktop:Screen Shot 2017-12-12 at 19.06.12.png

The probability of a class is either , if yi = 1, or 1 − , if yi = 0. The log likelihood is given by following equation is maximized using gradient descent method



Stochastic ascent rule is given by

Macintosh HD:Users:ajm:Desktop:Screen Shot 2017-12-12 at 19.23.46.png

### Neural Network

We have input features x1, x2, x3…Xm, which are collectively, called the input layer, 50/100/50 hidden units which are collectively called the hidden layer one/two/three and one output neuron called the output layer. The term-hidden layer is called “hidden” because we do not have the ground truth/training value for the hidden units.

ReLU activation function was used in the hidden layers

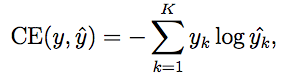
Macintosh HD:Users:ajm:Desktop:Screen Shot 2017-12-12 at 20.10.46.png

&

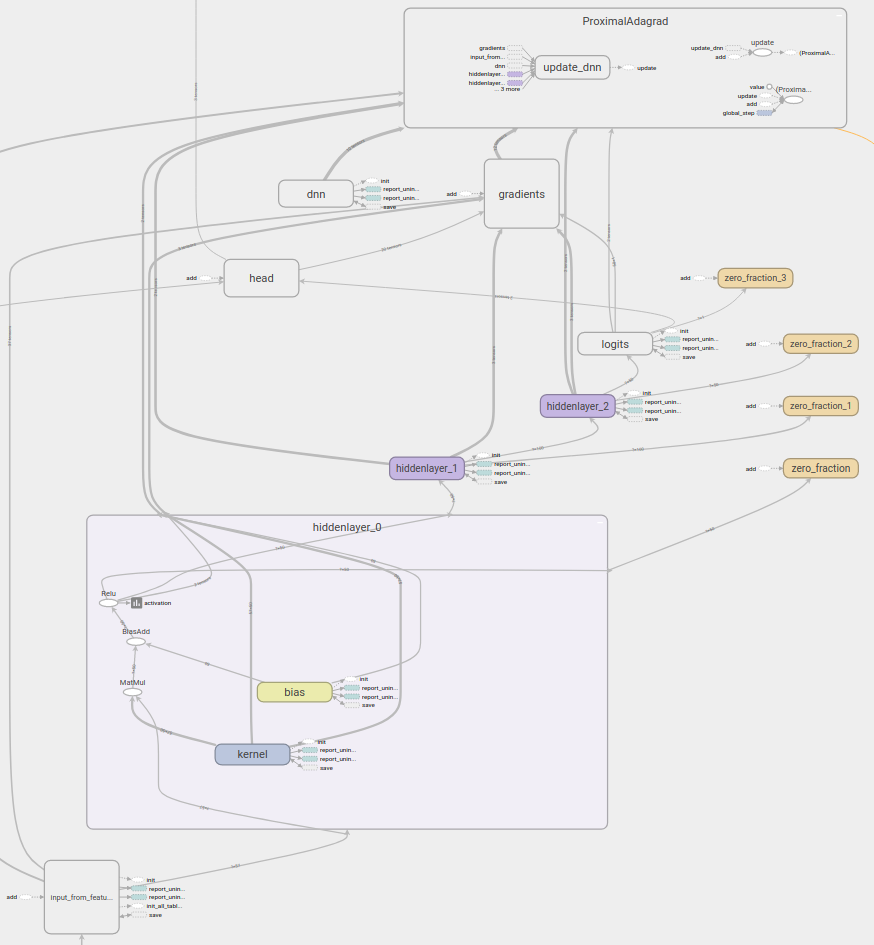
Softmax activation function was used for the output layer, where conditional distribution is given by

Macintosh HD:Users:ajm:Desktop:Screen Shot 2017-12-12 at 20.21.40.png

We evaluate our model using the cross entropy loss (CE). For a single example (x, y), the cross entropy loss is:



We used AdaGrad an optimization method, it allows different step sizes for different features and it increases the influence of rare but informative features.



## Experiments & Results

In this section, we will results obtained by applying classifiers described in the previous section on our dataset.

#### Random Forest

For Random forest classifiers, we experimented with various combinations of number of trees and maximum depth of the tree on our dataset. In the end, we picked the set of parameters, which gave better accuracy, precision and recall values.

Table : Fixed Depth(10) v/s varying Number of Trees

|  |  |  |  |
| --- | --- | --- | --- |
| **#Trees** | **Precision** | **Accuracy** | **Recall** |
| 1 | 0.57 | 0.56 | 0.56 |
| 5 | 0.67 | 0.67 | 0.67 |
| 10 | 0.69 | 0.70 | 0.69 |
| 15 | 0.72 | 0.72 | 0.72 |
| 20 | 0.69 | 0.69 | 0.69 |

Now keeping 15 trees, let’s try to find a optimal depth

Table 2: Fixed Number of Trees(15) v/s varying Tree Depth

|  |  |  |  |
| --- | --- | --- | --- |
| **Depth** | **Precision** | **Accuracy** | **Recall** |
| 1 | 0.71 | 0.71 | 0.71 |
| 5 | 0.69 | 0.69 | 0.69 |
| 10 | 0.72 | 0.72 | 0.72 |
| 15 | 0.71 | 0.71 | 0.71 |
| 20 | 0.69 | 0.69 | 0.69 |

Here, 15 trees in the forest with tree depth of 10 gave the best results.

#### Support Vector Machines

#### Logistic Regression

#### Neural Networks

## Conclusion AND Future work