Projecting NFL Quarterback Readiness

Amit Patankar

Google Inc. Mountain View, CA amit.anil.patankar@gmail.com

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

Aseem J Monga

NetApp Inc. Sunnyvale, CA aseem.jolly@gmail.com

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 quarterbacks 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.

A. 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:** NFL team which drafted the player
- **Heisman:** 1 if player was awarded the Heisman trophy, 0 otherwise
- **Completions:** Pass completions

Attempts: Pass AttemptsYards: Passing Yards

Touchdowns: Passing TouchdownsInterceptions: Passing interceptions

• Rush Attempts: Rushing Attempts

• Rush Yards: Rushing Yards

• Rush Touchdowns: Rushing Touchdowns

• **Draft Year:** Year in which the player was drafted

• **Round:** Round of the NFL draft process

• **Pick:** Position with in a round of NFL draft

• **Age:** Age at the time of the NFL draft

Game Played

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

Completi	Attem pts	Yar ds	Touchdo wns	Intercepti ons	Rush Attem pts	Rus h Yar ds	Rush Touchdo wns
562	851	796 4	65	18	145	284	7
779	1167	107 96	105	14	337	223 7	29

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

^{*} above denotes college data

B. Training and Test Set

Our training dataset has information of 150 plus quarterback's that got drafted between year 1998 and 2013. Test Set consists of quarterback's that got drafted in the year 2014 and 2015.

C. Preprocessing

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

- Text to numerical: One hot 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.

D. 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 models we used in detail

A. Random Forest

Random Forests is an 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 generates a family of classifiers $h(x|\theta 1), h(x|\theta 2),, h(x|\theta m)$. Each classifier $h(x|\theta m)$ is a classification tree and m is the number of trees chosen. Each θm 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 $T\theta m(x, y) \subset T(x, y)$ of the training set. Each tree partitions the data based on the value of a particular feature, until the data is fully partitioned, or the maximum allowed depth is reached. The output y is obtained by building the results thus:

$$y = \operatorname{argmax}_{p \in \{h(x_1)..h(x_k)\}} \big\{ \sum_{j=1}^k (I(h(x| heta_j) = p)) ig\}$$

B. 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

$$h_{w,b}(x) = g(w^T x + b).$$

Here, g(z) = 1 if $z \ge 0$, 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

$$\min_{\gamma, w, b} \quad \frac{1}{2} ||w||^2 + C \sum_{i=1}^m \xi_i$$
s.t. $y^{(i)}(w^T x^{(i)} + b) \ge 1 - \xi_i, \quad i = 1, \dots, m$

$$\xi_i \ge 0, \quad i = 1, \dots, m.$$

C. 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, x_i , and an observed class, y_i . Where sigmoid function is used as hypothesis

$$h_{ heta}(x) = g(heta^T x) = rac{1}{1 + e^{- heta^T x}},$$

The probability of a class is either \emptyset , if $y_i = 1$, or $1 - \emptyset$, if $y_i = 0$. The log likelihood is given by following equation is maximized using gradient descent method

$$\begin{array}{lll} \ell(\theta) & = & \log L(\theta) \\ & = & \sum_{i=1}^m y^{(i)} \log h(x^{(i)}) + (1-y^{(i)}) \log (1-h(x^{(i)})) \end{array}$$

Stochastic ascent rule is given by

$$\theta_j := \theta_j + \alpha \left(y^{(i)} - h_{\theta}(x^{(i)}) \right) x_j^{(i)}$$

D. Neural Network

We have input features x_1 , x_2 , x_3 ... x_m , 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

$$g(z) = \max(z, 0)$$

&

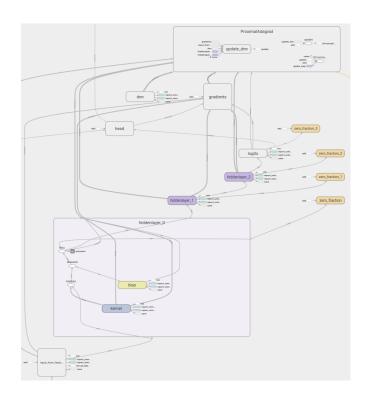
Softmax activation function was used for the output layer, where conditional distribution is given by

$$\operatorname{softmax}(z) = \exp(z) / \sum_k \exp(z_k)$$

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

$$ext{CE}(y, \hat{y}) = -\sum_{k=1}^K y_k \log \hat{y_k},$$

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

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

Model Selection Algorithm

We applied a slightly modified version of K fold cross-validation algorithm to do model selection. In our K* fold cross-validation algorithm, each fold consisted of players selected in a particular year, e.g. $1^{\rm st}$ fold consisted of players drafted in 2015 and so on. Now we will apply this algorithm on the models mentioned above and select the model, which gave the best result out of the lot.

A. 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 1: 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

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.

B. Support Vector Machines

We experimented with three different kernel functions, and various permutations of parameters C(penalty parameter C of the error term) and γ (kernel coefficient)

Linear Kernel

Table 3: Linear Kernel with Varying C

С	Precision	Accuracy	Recall
1	0.74	0.74	0.74
0.5	0.76	0.76	0.76
1	0.77	0.77	0.77
5	0.78	0.78	0.78

Polynomial Kernel

Table 4: Fixed γ (2) v/s varying C with second degree polynomial kernel

С	Precision	Accuracy	Recall
1	0.59	0.59	0.59
5	0.62	0.61	0.62
6	0.60	0.59	0.60
10	0.61	0.60	0.60

Table 5: Fixed C (5) v/s varying γ with second degree polynomial kernel

γ	Precision	Accuracy	Recall
2	0.62	0.61	0.62
5	0.61	0.60	0.60
10	0.61	0.60	0.60

Gaussian Kernel (Radial Basis Function)

Table 6: Fixed γ (2) v/s varying C

С	Precision	Accuracy	Recall
1	0.78	0.65	0.54
10	0.71	0.64	0.53

Table 7: Fixed C (1) v/s varying γ

γ	Precision	Accuracy	Recall
2	0.78	0.65	0.54
10	0.78	0.65	0.54

Linear kernel gave the best results among various SVM kernels.

C. Logistic Regression

There are two parameters to tune with logistic regression, one is the regularization (l1, l2, etc) and another is C(strength of regularization)

Table 8: Varying C with L1 regularization

С	Precision	Accuracy	Recall
0.01	0.38	0.61	0.47
0.1	0.77	0.76	0.77
0.5	0.74	0.74	0.74
10	0.76	0.76	0.76

Table 9: Varying C with L2 regularization

С	Precision	Accuracy	Recall
0.001	0.73	0.72	0.72
0.01	0.75	0.75	0.75
0.1	0.73	0.73	0.73
0.5	0.74	0.74	0.74
10 0.76		0.76	0.76

Logistic regression with L1 regularization performed better with scaling factor of $0.1\,$

D. Neural Networks

We experimented with varying degree of width and depth of the Neural Network. We used TensonFlow's DNN classifier library to run these experiments, and that surely was a big learning curve.

Table 10: fixed depth (1) v/s varying hidden units

L1 Precision Hidden Units		Accuracy	Recall
1	0.36	0.6	0.45
10	0.52	0.50	0.51
50	0.60	0.60	0.60
100	0.48	0.50	0.50

Table 11: fixed depth (2) v/s varying hidden units in L1 & L2

Depth	Precision	Accuracy	Recall
[10,20]	0.22	0.20	0.20
[25,25]	0.60	0.60	0.60
[50,50]	0.57	0.60	0.56
[50,100]	0.60	0.60	0.60

Table 12: fixed depth (3) v/s varying hidden units in L1, L2 & 13

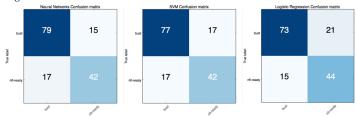
Depth	Precision	Accuracy	Recall
[10,20,10]	0.36	0.60	0.45
[20,40,20]	0.60	0.60	0.60
[50,50,50]	0.52	0.51	0.51
[50,100,10]	0.36	0.60	0.45
[50,100,40]	0.56	0.60	0.56
[50,100,50]	0.79	0.79	0.79
[50,100,100]	0.65	0.60	0.60
[100, 100, 100]	0.36	0.60	0.45

Neural network with three hidden layers, and with 50,100 and 50 hidden units respectively in each layer performed better than all other combination of depth and width of NN.

Results

Out of various models that we tried, the top three models were logistic regression, SVM with linear kernel, and neural networks. Random forests and SVM (Polynomial Kernel) would classify a majority of the players as busts and had lower recall, precision and f1-scores. We also considered the fact that predicting a player as a bust who was actually NFL-ready was a less critical mistake than drafting a bust.

Table 13: Neural Networks v/s SVM Linear Kernel v/s Logistic Regression



Training Accuracy v/s Test Accuracy comparison

Training Accuracy	Test Accuracy	
96.4%	73.7%	

A large gap between training and test accuracy suggests that model overfits the data and suffers from high variance, it's not possible to get more data to fix high variance as every year only a handful of quarterbacks make it to NFL. Reduction in feature space resulted in poor test accuracy.

2018 Draft Predictions

As we mentioned before, it is very difficult for our model to have statistically significant test data as there are only roughly 200 quarterbacks that have ever been drafted. As a fun experiment we assumed the mock draft from Chris Trepasso of CBS Sports was accurate. We applied our model to his draft and got some interesting predictions. This was a fun way of evaluating our model.

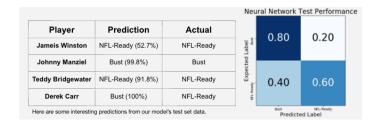
Table 14: 2018 Draft Predictions

Quarterback	Pick	Team	Prediction	Confidence
Lamar Jackson	1	CLE	NFL-ready	99.9%
Josh Rosen	2	NYG	NFL-ready	97.9%
Sam Darnold	9	CIN	NFL-ready	99.4%
Mason Rudolph	12	WAS	Bust	73.4%

It looks like we have a very successful quarterback class in 2018. Despite going to the Cleveland Browns (who have the largest QB turnover in the NFL) the model is very confident that Lamar Jackson will be NFL-ready. Washington should

beware that releasing current quarterback Kirk Cousins (who is definitely NFL-ready) in favor of incoming Oklahoma State phenomenon Mason Rudolph might be costly.

Conclusion & Future work



Our NN model classified NFL-ready player as bust with high degree of confidence that suggests that we need to come up with features that make a big difference in making a player a success at the pro-level such has NFL team's previous record, or optimization on coaches etc.

Also, the biggest improvements we can make are defining better labeling criteria that is more universally accepted and increasing our dataset size as more quarterbacks get drafted.

Acknowledgements

We would like to thank TA staff for their feedback at every stage of our project. We also owe a debt of gratitude to Tensor Flow open-source community, Derek Murray and Geo Hsu, And of course, our Professors Andrew Ng, and Dan Boneh for their invaluable guidance throughout the class.

References

- [1]https://www.sports-reference.com
- [2] https://www.pro-football-reference.com
- [3] http://www.ncaa.com
- [4] http://scikit-learn.org/stable/
- [5] https://www.tensorflow.org/
- [6] J. Duchi, Y. Singer. Efficient Learning using Forward-Backward Splitting, 2009.
- [7] E. Jones, C. Randall. Using Machine Learning to Predict NFL Game
- [8] Random Forests: Leo Breiman and Adele Cutler

Contributions

Team Members:

Amit Patankar (05739492)

 Responsibilities: Data collection, feature and model selection, model generation (NN, and Logistic), Training and Test Error Analysis, Poster, Project Report

Aseem J Monga (06247049)

 Responsibilities: Data collection, feature and model selection, model generation (SVM and Random Forest), Training and Test Error Analysis, Poster, Project Report