Final Project – Report

Ben Garski

School of Information Studies, University of Wisconsin Milwaukee

Computer Science 657: Machine Learning and Applications

Rohit Kate

May 17th, 2020

Contents

Project Description:

The prediction task for this project was to see if we can predict which NFL combine participants will be drafted in the NFL. This may be important for players and their agents to have a better gauge on if they have a chance to be drafted. Further, if a player knows their NFL combine numbers, they could then use that to estimate if they have a chance at being drafted. Time is money in the NFL, so I could see value with making these predictions on combine data, which is one of the most important factors of being drafted. Therefore, I made predictions if a player will be drafted or not based off their position, height, weight, and all of the NFL combine tests (40-yard dash, 3-cone, shuttle, broad jump, vertical jump, and bench press).

The Data:

The data came from https://www.pro-football-reference.com/. I used ten years' worth of NFL combine data, specifically, ten separate CSV files were downloaded from the site. They had all of the data mentioned above, like the player's name, school, position, height, weight, 40-yard dash time, shuttle time, 3-cone time, vertical jump height, bench press reps, broad jump distance, and what round/team/pick they were drafted.

Data Preprocessing/Data Exploration:

There was a lot that was done for preprocessing the data and getting it ready for analysis, all which can be viewed and explained here. I first had to join all the datasets from all the years. I then removed duplicate rows (there were duplicate players for some reason). Also, I converted height into inches, since it was in the feet-inches format. After exploring the data further, I found out that some of the positions were not represented well, like nose-tackle (a position not drafted often) and defensive back (which is basically the same as FS, SS, and CB).

So, rows with these positions were removed. Next, I found out that kickers and punters are missing the majority of their testing numbers, so these positions would need to be removed in order to get accurate predictions. Plus, for imputation to be beneficial, there needs to be enough data to gather the average on.

I used one-hot encoding/dummy variables for the position column, so that each position was its own column. Then, I normalized the numerical columns so that a method like k-nearest neighbors, was not affected by different units of measure. Lastly, before imputation I wanted to remove records if the player missed three or more tests at the combine. This to me shows that the player was injured. I don't want their values to be estimated off imputation, since these could be inaccurate. Finally, I used imputation using the position group medians. This is important because I did not want to simply use the column medians since results very so drastically by position. For example, giving an offensive-lineman a faster 40-yard dash time from the median would not make sense.

From exploring the data, I found out that the 40-yard dash is the least missed test at the combine. This makes sense to me since this test holds the greatest prestige in the combine. Also, I predicted beforehand that the 40-yard dash will be the most important factor in the machine learning model.

After analyzing the correlation matrix of all the variables, all of the correlations looked accurate. For instance, as player weight increases you would expect players to be slower, so this was true. Also, similar tests had correlations closer to one. For instance, shuttle and 3-cone times had a correlation of 0.88.

Machine Learning Tools/Techniques:

I used Python for the entire project, specifically, Spyder and Jupyter Notebook. Using Jupyter Notebook allowed me to create a nice report with markdown and have it viewable in GitHub. Python was used since it has the greatest machine learning capabilities for a data science language as of today. I used random forest, Naïve Bayes, k-nearest neighbors (with a k of three), and logistic regression for the classification methods. I chose these methods since they are popular methods and because they are different types of methods (information-based/ensemble, probability-based, similarity-based, and error-based). Thus, I should get an estimation of which type of method may be best suited for this application problem. Lastly, 10-fold cross-validation was used since the dataset was smaller and I wanted to get a good estimation of the model accuracies on unseen testing data.

Results:

All of the results can be viewed here in GitHub. Specifically, every fold's confusion matrix, accuracy, specificity, sensitivity, FPR, and FNR. Logistic regression had the highest mean accuracy over all the folds, but random forest was not far behind (70.8 % vs. 70.3 %). If we were to use false-positive rate instead, which would be the case if we want to assess which model reduces the number of false positives (predicting a player will be drafted and they actually won't be), then k-nearest neighbors would be the better model. The false-positives seem to be more important than false-negatives (predicting a player will not be drafted, and they actually do get drafted). Thus, if this model would be used in production, perhaps we would use FPR instead of accuracy to decide on the model. For my purposes, I will stick with accuracy as this is a good overall metric of the model.

```
Logistic Regression Cross-Validation Mean: 0.708
Random Forest Cross-Validation Mean: 0.703
Naive Bayes Cross-Validation Mean: 0.599
KNN Cross-Validation Mean: 0.663
```

Figure 1: Model Mean Accuracies

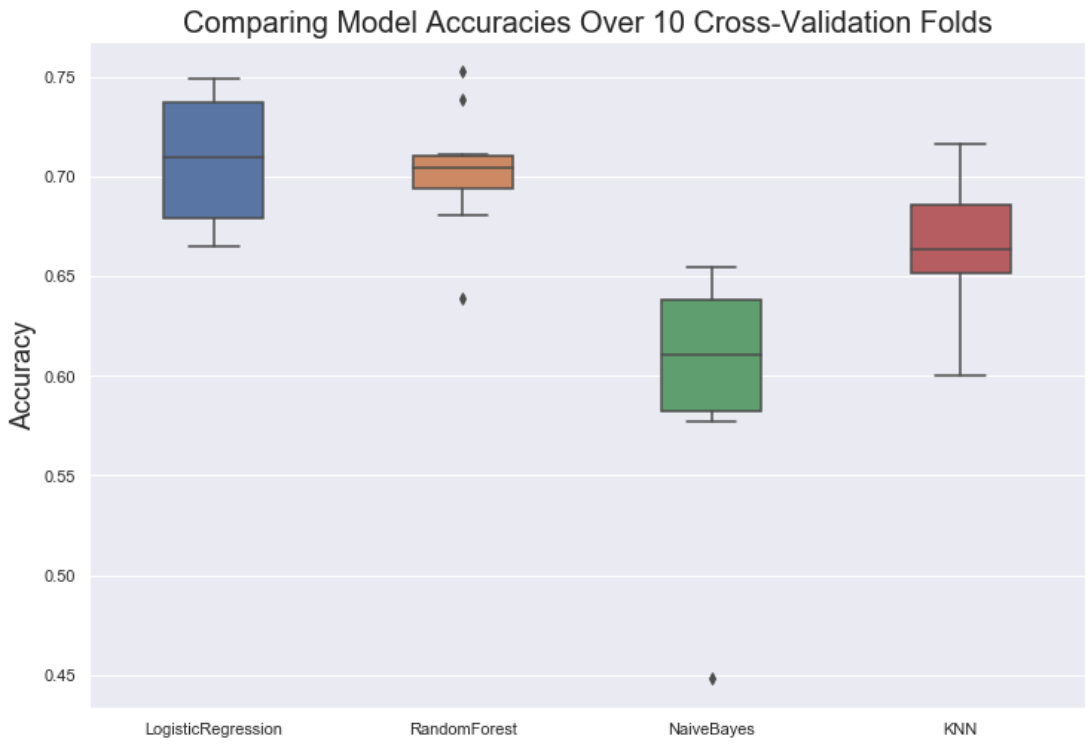

Figure 2: Boxplot Mean Accuracy

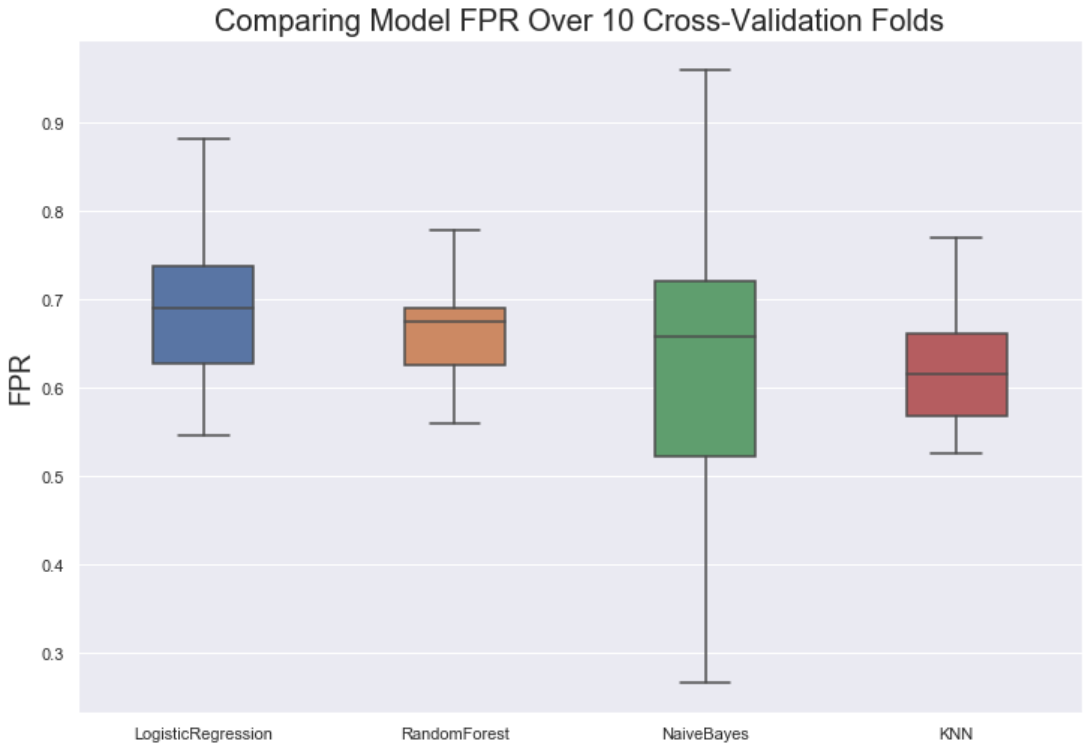

Figure 3: Boxplot Mean False-Positive Rate

After running the Levene's test to assess the equality of variances (an assumption for the ANOVA test), I used the fold accuracies to run significance testing to see if these models differ significantly from one another using the ANOVA test. Next, to find out which model(s) specifically differ, I used the Tukey's HSD test. The results show that KNN had a significantly lower accuracy compared to Naïve Bayes. Also, logistic regression had a significantly higher accuracy compared to Naïve Bayes. Finally, random forest had a significantly higher accuracy compared to Naïve Bayes.

Multiple Comparison of Means - Tukey HSD, FWER=0.05

group1	group2	meandiff	p-adj	lower	upper	reject
KNN	LogisticRegression	0.0447	0.0925	-0.0052	0.0947	False
KNN	NaiveBayes	-0.0639	0.0076	-0.1138	-0.014	True
KNN	RandomForest	0.0401	0.1533	-0.0098	0.09	False
LogisticRegression	NaiveBayes	-0.1086	0.001	-0.1585	-0.0587	True
LogisticRegression	RandomForest	-0.0046	0.9	-0.0546	0.0453	False
NaiveBayes	RandomForest	0.104	0.001	0.0541	0.1539	True

Figure 4: Tukey's Test Results

These significance testing results in combination with the mean accuracies allow me to confirm that either logistic regression or random forest would be the best model to choose for this classification problem given the decision to use accuracy as the metric to analyze. Looking at the feature importance score using the random forest algorithm, we can see that my hypothesis of the 40-yard dash being important was true. In fact, it looked like a player's weight was also pretty important.

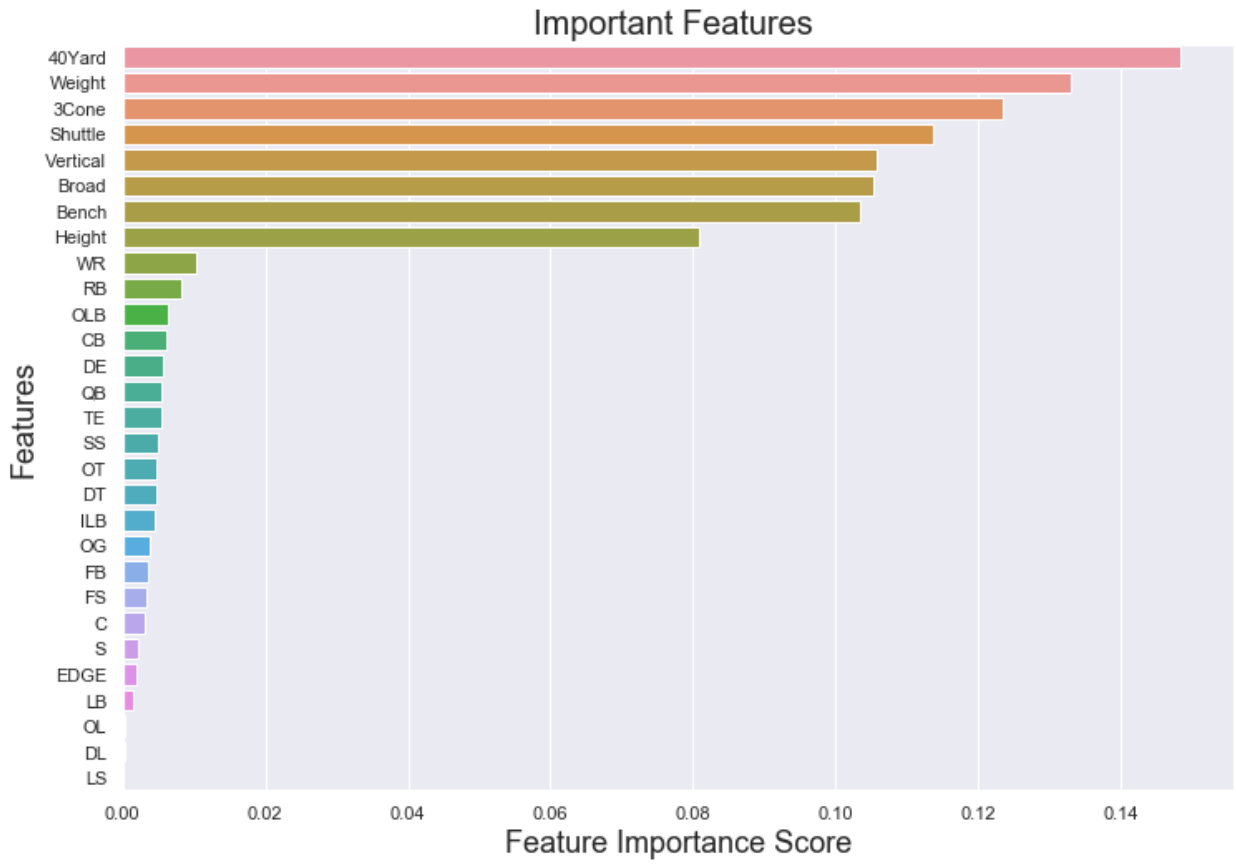

Figure 5: Random Forest Important Features

Conclusion:

In conclusion, it looks like logistic regression and random forest had the highest accuracy in classifying if an NFL combine participant would be drafted, while KNN had the ability to reduce false positives the most. Some of these results may be due to the fact that the data was relatively small (< 3000 rows). For instance, Naïve Bayes would probably improve if there was more data available. I contemplated using another ten years' worth of data, but the relevance of the data starts to decline, so the model relevance would as well I figured. Further, the NFL has changed a lot in ten years, so applying the model even further back would only exacerbate this effect. Future work would include analyzing data going back further, or simply

analyzing the years 2000-2010 to see if anything has changed. Lastly, trying out different feature selection methods could be beneficial to see if we can improve the accuracy at all.

Links:

- Data Source
- GitHub Page
- Jupyter Notebook