Predictive Machine Learning Models in Fantasy Football

Shaan Verma
Faculty of Science
University of Western Ontario
London, Canada
sverma43@uwo.ca

Alexander Hemming
Faculty of Science
University of Western Ontario
London, Canada
ahemmin@uwo.ca

Abdulaziz Chalya
Faculty of Engineering
University of Western Ontario
London, Canada
achalya@uwo.ca

Weichen Xu
Faculty of Science
University of Western Ontario
London, Canada
wxu283@uwo.ca

***Abstract*—The National Football League is without a doubt the largest, most televised football league world-wide racking in over $16 billion dollars in revenue. With this in mind it is imperative to make smart, educated decisions and in the era of big data this means data-based decision making. Now in order to use that data effectively we are going to need some high-powered tools and in this report we examine a few, looking at what insights we can gather from each. In this report we utilize PCA, clustering algorithms, tree-based algorithms and logistic regression.**

Keywords—PCA, Clustering Algorithms, Logistic Regression, Tree Algorithms.

I. Introduction

Fantasy Football is a game where participants take on the role of a coach or general manager of a virtual professional football team [1]. Participants build their own teams through a drafting process in which all relevant NFL players are available, and compete against the teams of other fantasy owners. Winners and losers are determined by the real-life statistics of the professional athletes that make up fantasy teams. As of 2017, over 59.3 million people have participated in Fantasy sports in the United states [1]. Given the popularity, there is often money involved and the stakes can get quite high. Prizes are awarded based on the prediction of player statistics, events, and team wins.

In recent years, analytics has made a big splash into sports, with many coaches and general managers using models to predict player performance to optimize their value given a set of constraints (e.g. salary cap) [12]. Machine learning provides a framework for this by utilizing data to train models and determine patterns to ultimately make an accurate prediction. This research project surveys and compares several different supervised and unsupervised learning models. Logistic regression, XGBoost, gaussian clustering, and agglomerative clustering are all explored to learn more about the data and predict player statistics.

II. Related Information

NFL modelling and prediction has been around for many years. Before machine learning and computers were prominent, mathematical formulas were used to manually calculate a player's performance (e.g. Passer Rating). As computers and statistics advanced, the field of sports analytics emerged as a topic of study with several papers implementing machine learning models such as Taylor [3] and Bunker [10]. Taylor used deep learning to predict the outcome of a play (e.g. pass or run) for in-game predictions. And Bunker used an artificial neural network to predict the final quarterback statistics of a game. Recently, Amazon's AWS has partnered with the NFL to provide a framework for analytics.

Deciding which model to use is often a difficult task because there are so many different metrics that can be predicted. Also, many of the models that are used are often proprietary and do not provide instructions to replicate and learn from it [8]. For example, QBR (Quarterback rating) is a proprietary metric used to measure a quarterback's overall performance [8]. Our project aims to explore different models to learn more about the data and to predict the outcome of a game (win/loss).

III. Data

The dataset used in our models was taken from Kaggle, which used Python to scrape the data from the official NFL website. The dataset contains basic statistics, career statistics and game logs for all players past and present (over 50 years) [6]. All of which are in a comma-separated values (CSV) format. The basic statistics section is composed of personal information such as name and sex. The career statistics is composed of metrics for a player's respective NFL position (e.g. passing yards for a quarterback). The game logs contain in-game information for each player that participated in the game (e.g. number of snaps played).

IV. Exploratory Data Analysis

The NFL data set contains a large number of highly correlated variables and needs to be reduced to a dimension that is easier to understand. This is achieved by transforming to a new set of variables the 'principle components' (PCs), which are uncorrelated, and which are ordered so that the first few retain most of the variation present in the original variables [7]. Three dimensions were chosen as the components since it retains most of the variation present (0.491) in the data set, and because it can be visually presented in a 3D graph to explore the data and visualize the explored clusters [13]. Moreover, the PCs were then fed as input into the Gaussian mixture and Agglomerate clustering models for further analysis.

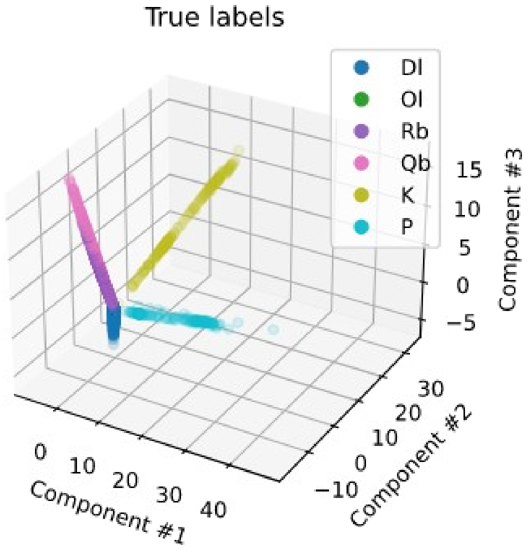

Fig. 1. Three component PCA graph with true data labels

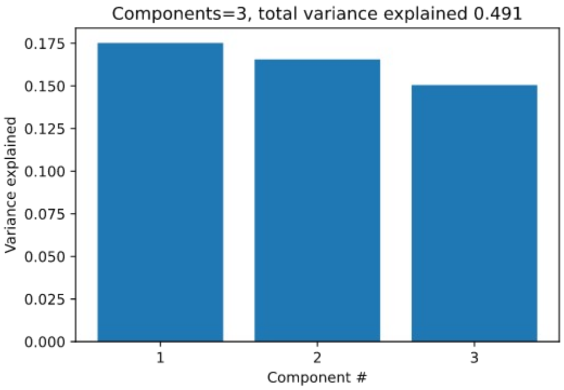

Fig. 2. Three component PCA barchart showing variance

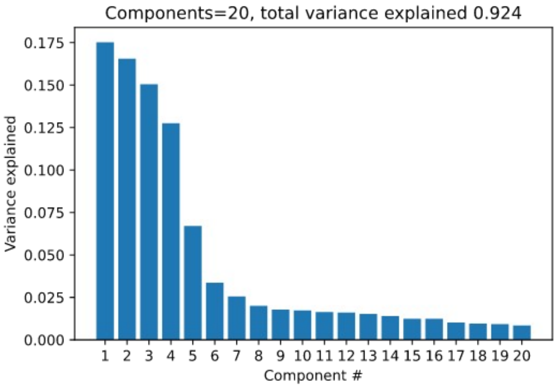

Fig. 3. 20 component PCA barchart showing bend in variance

V. Methods

This project aims to use both supervised and unsupervised machine learning methods to determine patterns and build predictive models within the NFL data set to ultimately be used to make informed choices in Fantasy Football. To determine patterns and correlations in the data set, PCA is first used to pre-process the data, followed by Gaussian mixture and Agglomerate clustering. The models created are used to predict the probability of wins, and are accomplished using both logistic regression and XGBoost.

A. Gaussian Mixture

The Gaussian mixture model (GMM) attempts to find a mixture of multi-dimensional Gaussian probability distributions that best model the output of the PCA clusters. This is used to probabilistically classify the NFL player position from a given data set [4]. In general, mixture models do not require knowing which subpopulation a data point belongs to, allowing the model to learn the sub-populations automatically (unsupervised) [6]. The multi-dimensional model was implemented in Python using the sklearn library, and is shown in the following formula:

$$p(\vec{x}) = \sum_{i=1}^{K} \phi_i \mathcal{N}(\vec{x} \mid \vec{\mu}_i, \Sigma_i) \tag{1}$$

$$\mathcal{N}(\vec{x} \mid \vec{\mu}_i, \Sigma_i) = \frac{1}{\sqrt{(2\pi)^K |\Sigma_i|}} \exp\left(-\frac{1}{2}(\vec{x}-\vec{\mu}_i)^{\mathrm{T}} \Sigma_i^{-1} (\vec{x}-\vec{\mu}_i)\right)$$

$$\sum_{i=1}^{K} \phi_i = 1 \tag{3}$$

Where the model is parameterized by the mixture component weights and the component means and variances/covariances. The kth component has a mean of μ_k and a covariance matrix of Σk. The mixture component weights are defined as ϕk, for component Ck, with the constraint that $\sum^{k}_{i=1}\phi_i=1$ so that the total probability distribution is normalized to 1 [4].

B. Agglomerative Clustering

Another model that was used to classify the position of NFL players from the output of PCA is Agglomerate clustering. It works in a “bottom-up” manner. That is, each object in the output of the PCA is initially considered as a single-element cluster (leaf) [5]. At each step of the algorithm, the two clusters that are the most similar are combined into a new bigger cluster (nodes). This procedure is iterated until all points are members of just one single big cluster (root) (see figure below) [5]. The Agglomerate clustering algorithm was implemented in Python using the sklearn library, and is described in following steps:

- Preparing the data
- Computing (dis)similarity information between every pair of objects in the data set.
- Using linkage function to group objects into hierarchical cluster trees, based on the distance information generated at step 1. Objects/clusters that are in close proximity are linked together using the linkage function.
- Determining where to cut the hierarchical tree into clusters. This creates a partition of the data.

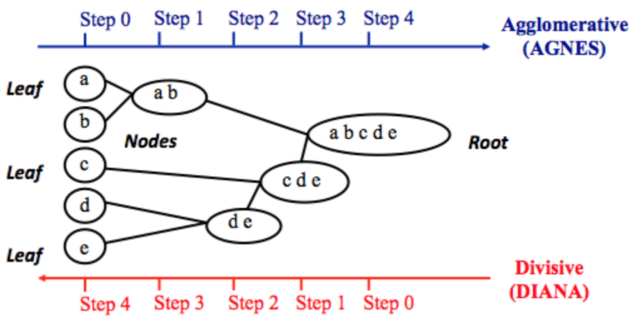

Fig. 4. Visual representation of the Agglomerative clustering algorithm. [5]

C. *Tree Algorithms (Random Forest and XG Boosting)*

A number of supervised tree-based methods were implemented, including both Random Forest and XG Boosting. A decision tree works by finding and storing input based upon constant functions at each node similar to how a binary tree will assign location based upon being larger or smaller than the parent node [9]. Trees provide us a non-parametric way in which to approximate functions, oftentimes much better than simply using polynomial expansion. More abstractly, what these algorithms are actually doing in higher-dimensional space is dividing our feature space into hyper-rectangles (each node deciding upon a feature) in which we fit constant functions making splits in our functions (branching off points in our trees) where we reduce our loss function the most [9]. The out of bounds error (OOB) decreases throughout training and eventually stabilizes/converges as some point signifying that training is complete. One of the limitations for trees is variance. Trees are extremely unstable with massive variance and can derive very different results even in the same dataset, however this is not entirely a bad thing, in fact as we will see this is a feature we exploit both in Random Forest and XGBoosting. The key is finding a way to reduce and manage this variance. This is where our algorithms come in, however, another popular method of variance reduction called bagging which is utilized by both our random forests and boosting algorithms must be discussed.

Bagging is a term short for bootstrap aggregation and as the name would imply works by building deep trees (high variance) on bootstrapped datasets (bootstrapping being a method of artificially increasing your data size and thus reducing variance through sampling with replacement from a sample population) and then combining these trees and averaging their predictions [11]. By averaging across these different datasets we are able to reduce our variance but because bagging alone is not very effective (all sorts of correlation problems between the trees it creates), random and boosting algorithms were created. Random forests essentially reduces the correlation between our trees but only showing random subsets of the data to the tree, changing the subset every time the tree has to make a split thus ensuring that our collection of trees (or forest) are unique from one another. Boosting on the other hand, fits an additive model using basis functions in succession where the parameters used in previous basis functions remain the same (The “XG” in XG Boosting comes from the fact we are using the gradient to optimize) [9]. Random forest can be thought of as essentially using sophisticated learners and forcing them to agree on a prediction, whereas in boosting we are using very weak learners (our basis functions) in succession where each learns from the last to reach a prediction. In the context of a classification model such as ours, what is actually happening is that essentially our nodes are voting on the class of the input and then deciding on the class based upon the proportions of these votes.

D. *Logistic Regression*

A logistic model was implemented on the NFL dataset to predict binary metrics (win/loss). It uses the following equation:

$$y = e\^(b0 + b1*x) / (1 + e\^(b0 + b1*x)) \tag{4}$$

Where y is the predicted output, b0 is the bias or intercept term and b1 is the coefficient for the single input value (x). Logistic regression models the probability of the default class, in our case win/loss [2]. In other words, we are modelling the probability that an input (X) belongs to the default class (Y=1). This can be written formally as:

$$P(X) = P(Y=1|X) \tag{5}$$

Pytorch was used to implement logistic regression on the NFL dataset, and the bias terms were estimated using maximum-likelihood estimation. The best coefficients results in a model that predicts a value very close to 1 (e.g. Win) for a default class and a value very close to 0 (e.g. Loss) for the other class. The intuition for maximum-likelihood for logistic regression is that a search procedure seeks values for the coefficients (Beta values) that minimize the error in the probabilities predicted by the model to those in the data (e.g. probability of 1 if the data is the primary class).

IV. RESULTS

To evaluate the four models, a test set consisting of a mixture of all the player positions with over 10000 data points are used for prediction.

A. *Agglomerative Clustering*

Figure 5 shows that the Agglomerative Clustering model (ACM) performed well on capturing Defensive Linemen, but had trouble with other positions. For example, in Figure 6, Kickers are made up of two clusters of points, which adds some complexity to the data. It seems the model has identified them as different positions which reduced the score. In Figure 7, we can see the model labelled most players as Defensive Linemen, which makes sense because they make up the vast majority of players and are all tightly packed.

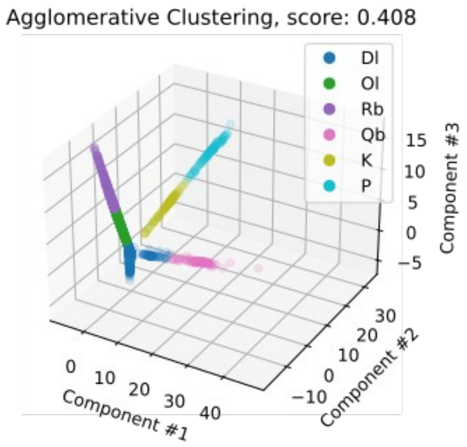

Fig. 5. 3D Agglomerative clustering model graph.

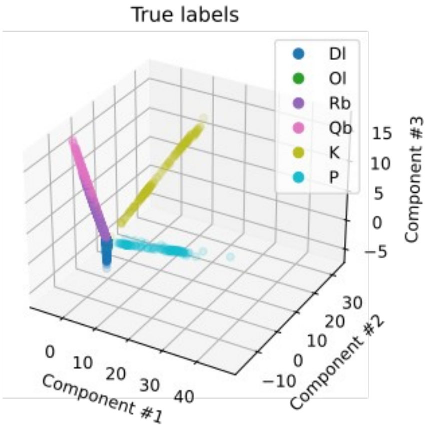

Fig. 6. 3D Agglomerative clustering model with with true data labels.

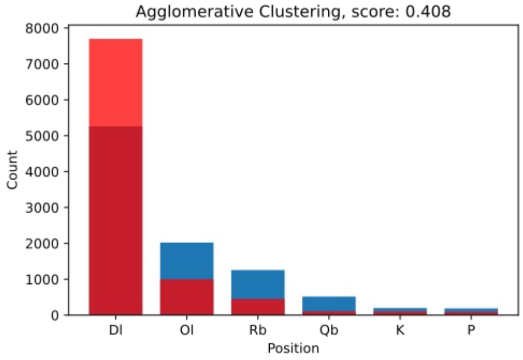

Fig. 7. Agglomerative clustering bar chart.

B. Gaussian Mixture Model (GMM)

The GMM is comparably similar to the previous model. It ran into similar issues as the ACM because some positions are very tightly packed as shown in Figure 8. Although it performed worse according to the score, it did correctly identify a half of the Punters, as well as the other half of Kickers that the ACM mislabelled. Furthermore, the GMM had trouble separating the line of points containing Offensive Linemen, Runningbacks, and Quarterbacks, into 3 clusters, instead it labelled the majority of them as Offensive Lineman. Figure 9 better visualizes these distinctions, compared to the ACM, the GMM was much worse at identifying Runningbacks and Quarterbacks, but did perform better in identifying Punters.

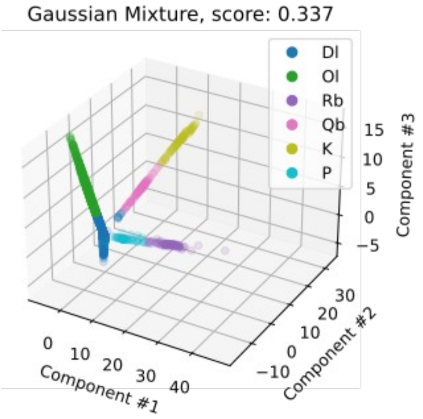

Fig. 8. 3D Gaussian clustering model graph.

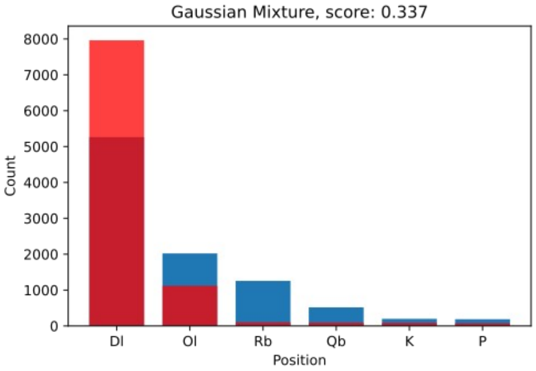

Fig. 9. Gaussian mixture bar chart.

C. Logistic Regression

For logistic regression, there are three main methods for evaluation. First, use the model to predict the test database to calculate the correct rate. After multiple rounds of testing, the accuracy stabilized at approximately 62%. Secondly, in order to evaluate the performance of the model more intuitively, a confusion matrix was created as shown in Figure 10. We call the prediction category 1 as Positive (win), and the prediction category 0 as Negative (lose). If the prediction is correct, it is True, and if the prediction is wrong, it is False.

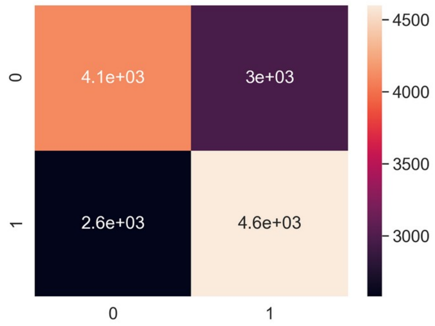

Fig. 10. Confusion matrix for logistic regression

Finally, in order to consider the classifier's ability to classify positive and negative examples at the same time, in the case of unbalanced samples, it is still possible to make a reasonable evaluation of the classifier. We introduced the AUC diagram (Figure 11). This leads to the two concepts of True Positive Rate and False Positive Rate:

$$\text{TP Rate} = \text{TP} / \text{TP} + \text{FN} \tag{6}$$

$$\text{FP Rate} = \text{FP} / \text{FP} + \text{TN} \tag{7}$$

The meaning of TP Rate is the proportion of all samples with a true category of 1, and a predicted category of 1. The meaning of FP Rate is the proportion of all samples with a true category of 0 and a predicted category of 1. We can get two values through the confusion matrix to get the AUC curve. The final ACU value is approximately 0.6566.

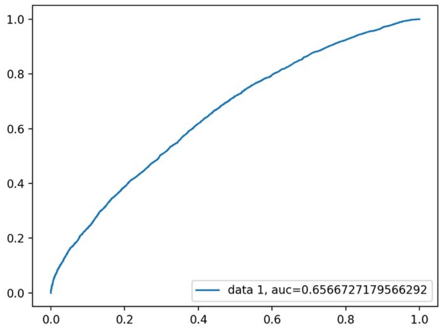

Fig. 11. AUC curve for logistic regression

D. *Tree Algorithms (Random Forest and XG Boosting)*

In our use of the Random Forest algorithm, the first thing we had to do was to determine the optimal number of trees we need to generate in the training process. There are a few ways in which to do this but we have opted to use the out-of-bounds error (OOB Error) and as explained in our overview of tree-based methods we will be able to determine this by graphing the OOB error and seeing where it stabilizes (this point signifying the number of trees required for no more significant training to occur). For our data the following graphs were generated as shown in Figure 12.

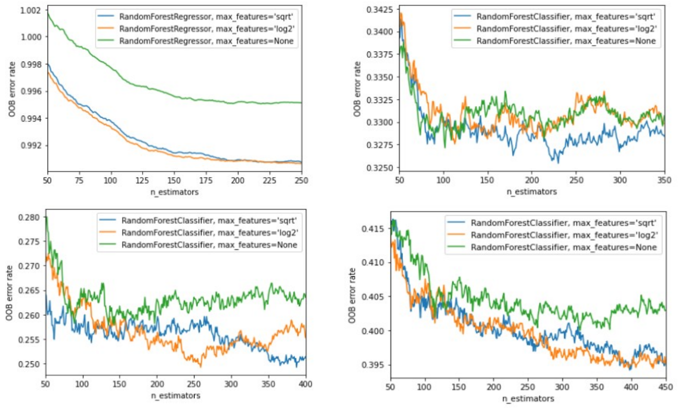

Fig. 12. Top-left: Defensive Linemen, Top-right: Quarterbacks, Bottom-left: Kickers, Bottom-right: Runningbacks.

So from the graphs in Figure 12 we were able to determine the number of estimators or trees in our case to train over. For instance our kickers dataset (bottom-left of Figure 12) appears to stabilize 380 estimators which was carried out throughout the rest of our analysis on that dataset. On it's own this only serves to aid in optimizing our algorithm, so what sort of information are we actually able to obtain through the use of our models? Well one of the very useful attributes of tree-based models is that they can be very easily retrofitted to give us the relative importances of the different features we have in our dataset, a very useful characteristic in the feature selection process. The variable importances are rated using the Gini Importance (also known as Mean Decrease in Impurity) which works by summing over all trees containing the feature and then obtaining the relative proportion over all trees generated by the model. This can be visualized in a barplot as seen in Figure 13 where these proportions are ranked from most important at the top to least important at the bottom. This allows us to see for instance that the most influential feature in whether running backs have an influence on the outcome of the game is to get as many attempts at rushing as possible (as can be seen in bottom-left of Figure 13). Now this intuitively makes sense and isn't exactly the most interesting discovery, obviously the more attempts these players have, the more likely they are to impact the game but this does give us an idea of how well our model has performed if it appears to recreate things we see in reality. This also gives us more confidence when our model generates results which may not be so intuitive at first. For example looking at the data we have on the quarterbacks (bottom-right of Figure 13) it's not so obvious that completion percentage would be more important than passing yards or that in spite of this that passing yards per attempt is ranked so high. The other information we can get from our importance barplot is that we can get an idea of the value of the various positions. For instance you will notice a huge discrepancy between the defensive linemen and the quarterbacks (top-left and bottom-right of Figure 13). You will notice that in the defensive linemen dataset that the vast majority of the most important features are not actually player data. A lot of these are actually team metrics meaning that the influence of these players clearly is much smaller than for instance the impact of what team they are playing (the opponent features). This would suggest that these players overall have less impact and therefore value in determining the outcome of the game. This is in strong contrast to the data on the quarterbacks which is the complete opposite. Clearly the performance of the quarterback has a huge effect on the outcome on the game whereas defensive linemen evidently do not. This could be seen as further evidence in favour of why an average quarterback is paid so much more than the average defensive lineman.

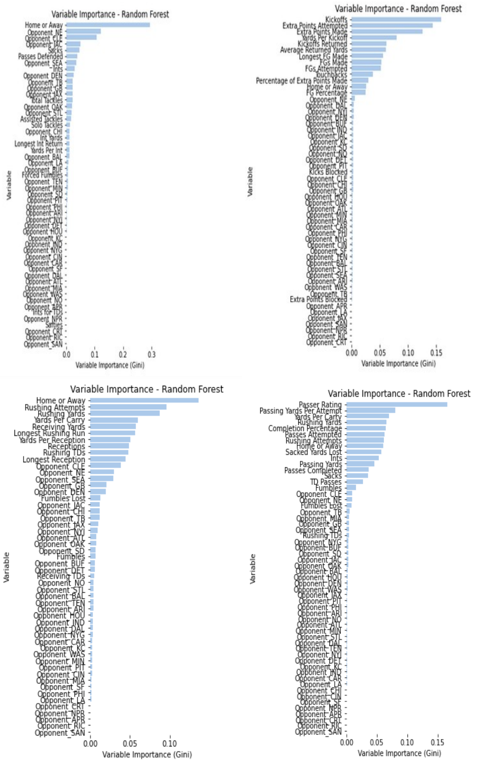

Figure. 13. Top-left: Defensive Linemen, Top-right: Kickers, Bottom-left: Runningbacks, Bottom-right: Quarterbacks

The last part of our random forest analysis both relate to the predictive performance of our models. The AUROC is the area under our ROC curve and tells us how effective our algorithm is at determining that for any positive input it should be ranked to the left of any negative input. This is essentially a less sophisticated way of showing our predictive performance over our confusion matrix which gives us the relative proportions. Clearly we can see that our model performed best on the kickers and worst on the defensive linemen with AUROCs of 0.847 and 0.647 respectively (top-left and top-right of Figure 14). This is again seen in our confusion matrices (Figure 15) where our model performed best on the kickers dataset although it performed worst on the running backs in this case.

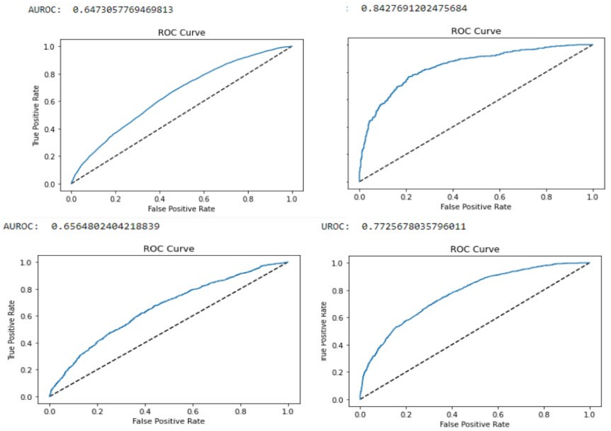

Figure. 14. Top-left: Defensive Linemen, Top-right: Kickers, Bottom-left: Runningbacks, Bottom-right: Quarterbacks.

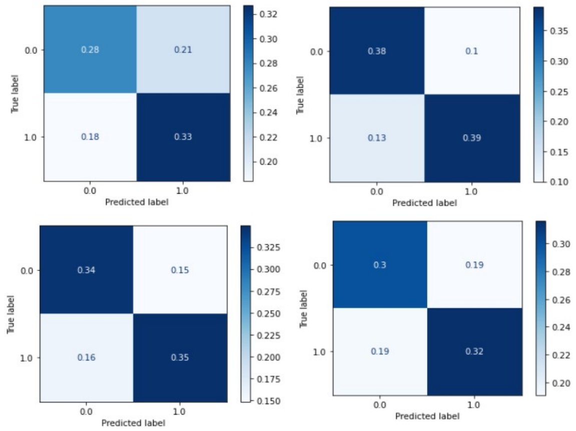

Figure. 15. Top-left: Defensive Linemen, Top-right: Kickers, Bottom-left: Quarterbacks, Bottom-right: Runningbacks

Now for our XGBoosting models we see very similar results but there are some interesting differences. As expected, due to our small sample size the boosting algorithms on the whole perform better (they are optimized for smaller datasets after all). What's much more interesting is the impact that the algorithm has on the rest of the data. For instance, in the importance barplots (Figure 16) we see that the importance of all the features have decreased over all the datasets but the relative importances have been smoothed out. It's almost as if the data has been normalized under our XGBoost algorithm as it compares to Random Forest. This is carried out over all of our plots and Figures 16-18.

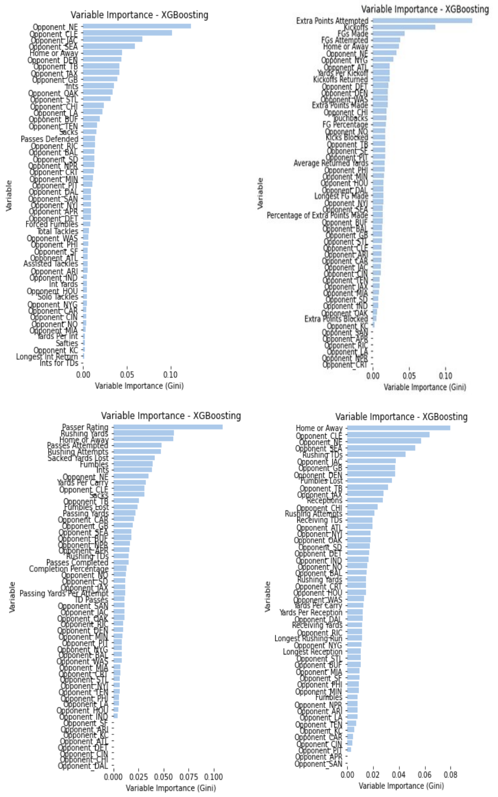

Figure. 16. Top-left: Defensive Linemen, Top-right: Kickers, Bottom-left: Quarterbacks, Bottom-right: Runningbacks

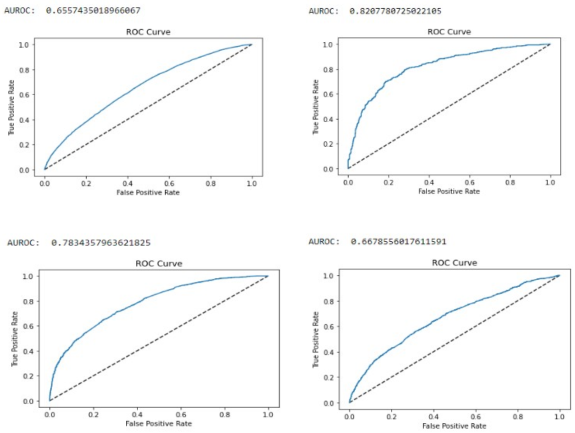

Figure. 17. Top-left: Defensive Linemen, Top-right: Kickers, Bottom-left: Quarterbacks, Bottom-right: Runningbacks

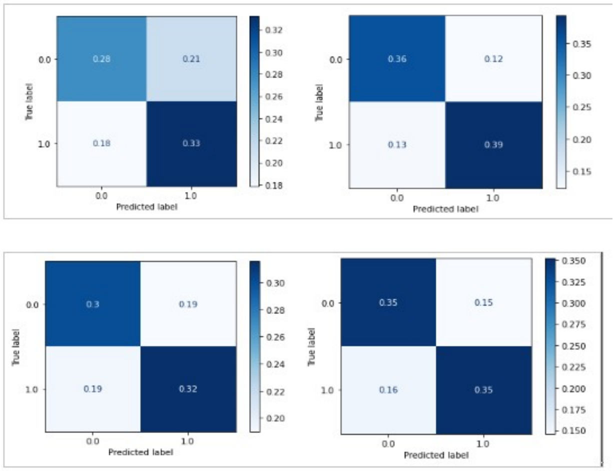

Figure. 18. Top-left: Defensive Linemen, Top-right: Kickers, Bottom-left: Quarterbacks, Bottom-right: Runningbacks

VI. Conclusion

In conclusion, we have utilized several modern AI algorithms and techniques to generate models and figures capable of providing us insights into the characteristics of our sample population. We used these models in the context of analyzing over 45 years of player data to show off the usefulness of these algorithms in the real-world data analysis process.

VII. References

[1] B. Dwyer and Y. Kim, “For Love or Money: Developing and Validating a Motivational Scale for Fantasy Football Participation,” *Journal of Sport Management*, vol. 25, no. 1, pp. 70–83, 2011.

[2] C. Molnar, “Interpretable Machine Learning,” *4.2 Logistic Regression*, 08-Apr-2021. [Online]. Available: https://christophm.github.io/interpretable-ml-book/logistic.html. [Accessed: 12-Apr-2021].

[3] C. Taylor, “Deep Learning for In-Game NFL Predictions,” *stanford.edu*, 2020.[Online]. Available: http://cs230.stanford.edu/projects_winter_2020/reports/32263160.pdf. [Accessed: 2021].

[4] “Gaussian Mixture Model,” *Brilliant Math & Science Wiki*. [Online]. Available: https://brilliant.org/wiki/gaussian-mixture-model/. [Accessed: 12-Apr-2021].

[5] Jovana, Kassambara, Kassambara, Serkan Korkmaz, S. Korkmaz, and Jason, “Agglomerative Hierarchical Clustering,” *Datanovia*, 20-Oct-2018.[Online].Available:https://www.datanovia.com/en/lessons/agglomerative-hierarchical-clustering/. [Accessed: 12-Apr-2021].

[6] KendallGillies, “NFL Statistics,” *Kaggle*, 09-Jun-2017. [Online]. Available:https://www.kaggle.com/kendallgillies/nflstatistics?fbclid=IwAR2duOY3nl2jZwNeMyfjSRUbFtDds5_k5iL74VBn9UtrTKCgpOlnpv251-g&select=Career_Stats_Field_Goal_Kickers.csv. [Accessed: 12-Apr-2021].

[7] M. Brems, “A One-Stop Shop for Principal Component Analysis,” *Medium*,10-Jun-2019.[Online].Available: https://towardsdatascience.com/a-one-stop-shop-for-principal-component-analysis-5582fb7e0a9c. [Accessed: 12-Apr-2021].

[8] *Medium*. [Online]. Available: https://medium.com/@gzil/how-to-calculate-nfl-passer-rating-using-a-formula-in-excel-or-google-sheets-54eb07246d1e. [Accessed: 12-Apr-2021].

[9] *Medium*.[Online].Available:https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d. [Accessed: 12-Apr-2021].

[10] R. P. Bunker and F. Thabtah, “A machine learning framework for sport result prediction,” *Applied Computing and Informatics*, 19-Sep-2017. [Online].Available:https://www.sciencedirect.com/science/article/pii/S2210832717301485. [Accessed: 12-Apr-2021].

[11] “Random Forests Leo Breiman and Adele Cutler,” *Random forests – classification description*.[Online].Available: https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm. [Accessed: 12-Apr-2021].

[12] T. O. Comeau, “Fantasy football participation and media usage,” 2019.

[13] Z. Jaadi, “A Step-by-Step Explanation of Principal Component Analysis (PCA),”*BuiltIn*. [Online]. Available: https://builtin.com/data-science/step-step-explanation-principal-component-analysis. [Accessed: 12-Apr-2021].