Method or Madness: Predicting College Football Playoff Rankings using Machine Learning

Will Fossett

Aerospace Engineering and Mechanics, The University of Alabama, Tuscaloosa, USA

wefossett@crimson.ua.edu

December 5, 2022

Abstract

This paper discusses the use of different machine learning methods utilized in an attempt to predict the final CFP rankings of college football teams using their season's statistics. The data set consists of 14 numeric features from each Division 1 college football team starting in 2015, when all such data was readily available. The algorithms used are: k-nearest neighbors, artificial neural network, and decision tree with feature extraction. Data was processed and analyzed in Python and MATLAB. Each team's features were fed into the regression algorithms to produce an estimated final ranking, and then compared to the true end-of-year ranking to calculate error. Each minor subcategory of ranking - top 4, top 10, top 25, and outside of the top 25 - were also used as classes in a secondary classification to determine algorithmic accuracy with predicting approximate rankings. Accuracy metrics are presented at the end of the paper for review. As a final test, the 2022 CFP rankings were estimated using team data through December 3rd, 2022.

Keywords: College Football; machine learning; neural networks; k-nearest neighbors; decision tree; regression

Contents

List of Figures

1 Context and motivation

As a lifelong fan of college football and an engineer who is interested in science's ability to predict how complex systems behave, I have always wondered if seemingly non-deterministic systems like college sports can be accurately described using deterministic analyses. College sports are notoriously difficult to predict, with only a few teams able to muster repeated consistency year-over-year. Machine learning allows us to attempt to find discrete patterns and important features in an otherwise noisy and chaotic sea of data, like college football statistics. This paper discusses multiple methods to predict a team's final CFP ranking using their year's statistics for training and testing.

Data analytics for sports is a huge market. Millions of dollars are bet on the outcomes of sports, and entire companies have been formed to attempt to bring order and determinism to the chaotic field of amateur athletics. Multiple entities have gathered their own data, processed it, and used the data to create their own, unique predictions for a college sport. These methods are not often publicized but are virtually guaranteed to use their own combination of data features, picked due to previous research and from each feature's importance. I will attempt the same, sourcing my own data from multiple sources based on proven importance as well as anecdotal belief in a statistic's relevance. Because I will be using my own feature set as well as multiple machine learning methods, this analysis will be completely different and unique from previous attempts, and any results resembling results of other methods only lends credence to the validity of the analysis. This paper will serve as the cumulative term paper for the course ME 591: Machine Learning, taught by Dr. Hwan-Sik Yoon at the University of Alabama.

2 Data Set description

The data is sourced from multiple online repositories, such as cfbstats.com, sports-reference.com, espn.com, and others. All data sources are linked in the data sheets and in the references section. No changes were made to the data, except for team names to standardize abbreviations across data sets. For each year, starting in the 2015-2016 season when all data features were available, an Excel workbook was made to store the data. Each workbook has a data sheet, where logic pulls in the desired values from the remaining sheets and sources. Each year's workbook starts with a column of team names of all of the Division 1 football teams from that year. The next column is the year-end CFP rank, which is the desired regression value to learn. The features input to the algorithms are presented in the following order in the data workbooks: team win/loss percentage, average team scoring per game, average team points allowed per game, strength of schedule metric (with 0 being average and positive values being more difficult schedules), prior year's final Associated Press (AP) ranking, offensive efficiency metric (with 0 being average and positive values being more efficient at scoring per play), defensive efficiency metric (with the same characteristics as offensive efficiency), net yards per play (the difference between average offensive yards per play gained and defensive yards per play allowed), explosiveness metric (with 0 being least explosive and larger values meaning a team is more likely to offensively generate plays of 10+ yards), average number of times per play that a defense gives up 10+ yards, turnover margin, percent of offensive team returning, percent of defensive team returning, and time of possession percent. These features were chosen based on research from other analyses and anecdotal belief in a feature's importance, like the human-centric aspect of returning player percentage. Using dimensionality reduction and feature extraction, features with little impact on the overall learning are mitigated.

3 Methods

Multiple machine learning methods were utilized in this study: k-Nearest Neighbors, shallow Neural Network, and decision trees. These methods are non-parametric learning algorithms, which do not assume any underlying probability densities that may have generated the data. The operating assumption for non-parametric methods is that similar inputs produce similar outputs, so data near the test data can be assumed to correspond to the label of the test data. This "lazy" learning method is useful when the data is assumed to be a function of many variables that are not easily quantified. By not assuming the data is a function of any set of independent variables, the algorithm is free to learn the data in any form. This tends to lead to lower prediction accuracy compared to parametric data when there is an underlying probability density function but higher accuracy otherwise. In this case, it was assumed that a team's ranking is not a simple, linear function of finite probability densities: it would take numerous, perhaps un-quantifiable features to accurately describe the model. As such, non-parametric methods were chosen.

3.1 Eigenvectors and Data Projection

As discussed Section 2, the data features were chosen based on established importance to a team's success, such as win/loss percent, as well as human-centric, anecdotal belief in their importance, such as returning player percentage. As expected, not all of these features capture as much data variance as the others. To simplify calculations, the data should be projected onto a smaller input space that still captures as much variance as possible, such as 99.9% of the variance. To do this, a method called Principal Component Analysis (PCA) is used. PCA projects the input space onto the k largest eigenvectors of the data.

First, the covariance matrix of the data matrix is calculated, which is a matrix containing the variance of each feature and the covariance between each set of two features. Both Python (using the NumPy package) and MATLAB have built-in methods for calculating this matrix. Then, the eigenvalues and eigenvectors of the matrix are calculated with similar numeric packages. Then, using the eigenvectors corresponding to the eigenvalues that sum to 99.9% of the total sum of eigenvalues, the eigenvector matrix W can be composed as a matrix whose columns are the eigenvectors of the corresponding eigenvalues. The mean vector of each data feature is then subtracted from the data; this serves to "center" the data at the center of the input space. Finally, the dot product of the eigenvector matrix and this centered data projects the data onto the n eigenvector space. This reduces the data input space from d dimensions to n (where n is the number of eigenvalues required to reach the proportion of variance of 99.9%, for example), allowing for simpler computations in the reduced space while still capturing the most data variance. The projection also creates new features which are all of the same scale and with zero covariance, allowing the quicker Euclidean distance to be used in the k-NN algorithm instead of the computationally-slow Mahalanobis distance, which is the more accurate distance when data points are strongly correlated and are of different scales and units.

3.2 k-Nearest Neighbors (k-NN)

After projecting the training data into the reduced dimensional input space, each test instance must also be projected into the same space using the same W and mean vector. This is done for each test instance, after which the Euclidean distance between the test instance and every training data instance is calculated. The distances are then sorted to produce a list of the smallest to largest distances from the test instance to all training instances. This is how the k-NN gets its name: it utilizes only the 'k' smallest of these distances in its regression and/or classification. For the regression, the k distances are averaged to produce an average regression for that test instance, which is stored before iterating over the next test instance.

In college football, where there are approximately 130 Division 1 teams each year, only 25 are ranked in the top 25. This leads approximately 100 non-top 25 teams each year. In this study, if there were no prior classification, the presence of these 100 additional instances per year would greatly skew the regressions towards lower rankings and lead to poor accuracy for estimating high-ranked teams: the algorithm would estimate that a team truly ranked 4, for example, should be regressed as an 11 due to a high number of low ranking teams. To overcome this, the augmented k-NN algorithm first attempts a classification, where it checks if the majority of the k nearest neighbors are classified as '26', denoting outside of the top 25. If so, it assumes this test instance should be outside of the top 25 and returns a regression of 26. If the majority are not 26, the algorithm averages the k-nearest neighbors that are not 26 and returns that values as the regression for the test instance. This reduces the impact of the additional non top 25 teams and improves the accuracy for high-ranked teams. This method is essentially classification then regression: the algorithm classifies the test instance as top 25 or not, then regresses the true ranking in the top 25. In this study, k values from 1 to 25 were tested, with 8 being close to optimal for the training data.

3.2.1 Error Calculation and Classification

For the k-NN, error was calculated as absolute value of the difference between the regression ranking and the true ranking for the training and testing data. The average error was also calculated for all top 25 regressions and non-top 25 regressions, to see the impact of each class. Finally, each regression value for the training and testing was classified into top 4, top 10, top 25, and not top 25 and compared to the true class and tabulated. This is to show the accuracy of the algorithm at predicting those four main classes of interest in college football. The results are shown in the Results section.

3.3 Artificial Neural Network (ANN)

ANNs are very popular methods for regression and classification, but are computationally expensive. They rely on multiple layers of nodes, which each interact with the nodes in the layers before and after them with unique weights. Each node uses a weighted average of the inputs it receives, then passes that value into the next layer. Using a method called backpropagation, the error between the final output and the true label can be calculated, differentiated with respect to each layer, and used to change the weights of the connections throughout the network. Backpropagation allows the previous experience of the network to influence its future weights so that the network can learn from its past mistakes. This method is iterated thousands of times to reduce the error. Each iteration is called an epoch, and this study used 4250 to produce the ANN.

The structure of a shallow ANN is as follows: a layer of input nodes, corresponding to the number of input features; one layer of hidden nodes, which each connect to the previous input nodes and output nodes; and the output nodes, which produce the values or classes of interest. In this study, there are 14 input nodes corresponding to the 14 features of interest, and one output node which produces the regression value. Then number of hidden nodes is a changeable parameter, where a higher number of hidden nodes may improve accuracy and a smaller number may reduce computation time. Because computation time is a function of data set complexity, reducing the hidden nodes may not improve computation time considerably. In this study, 14 hidden nodes was found to be an optimal value for accuracy and computation time. The final four parameters of the ANN, denoted by Greek letters κ, ϕ, θ, and μ are parameters that change how much the ANN weights previous iterations in the backpropagation. In this study, values of 0.1, 0.5, 0.7, and 0.7, respectively, were used.

3.3.1 Error Calculation

For each epoch of the ANN, the error is calculated by the difference between the true values and the estimated values with the current weights. This value is stored before backpropagation begins for another epoch. The same errors as discussed in Section 3.2.1 are calculated and shown in the Results section. The ANN has much better accuracy, especially for high ranking teams, since it does not rely on the neighbors of the data point which are not guaranteed to be the same value. The ANN is able to learn the relationship between the features and the label, although not in a decipherable way.

3.4 Decision Trees

Univariate decision trees break the classification or regression into a finite number of decision nodes, where each feature is quantifiably compared to a threshold value to determine the direction after the decision. Features more important to the result are placed earlier in the tree and impact more of the data. Univariate decision trees allow users to see the relationship and importance each feature has, often at the cost of some accuracy. The structure of a decision tree is as follows: the 'root' node is the first decision node the data comes across, where the feature in the node is compared to the threshold value. If the value is in a certain range, the data progresses down one 'branch' or the other, until it reaches another decision node with another feature comparison. This feature is then compared, and so on and so forth until the instance reaches a 'leaf' node where there is no decision to be made. This node represents the estimated classification of the data instance. Following the branches back up to the root, one can see the features that were important to the classification as well as the threshold values. Using MATLAB's classification learner, two decision trees were generated and are presented in the Results section.

3.4.1 Error Calculation

Decision trees are most easily utilized for classification rather than regression, so this study utilized a decision tree for classification instead of regression as was done in the k-NN and ANN. The error is calculated simply as if the test instance was classified by the decision tree as its true rank or not.

4 Results and discussion

4.1 k-NN

First, the data is projected into a reduced input space using the proportion of variance, which is a ratio of the sum of the n eigenvalues to the total sum of eigenvalues, shown in Figure 1 below.

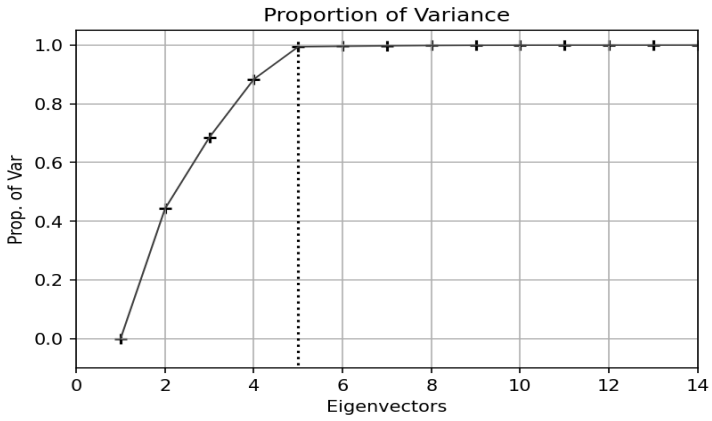
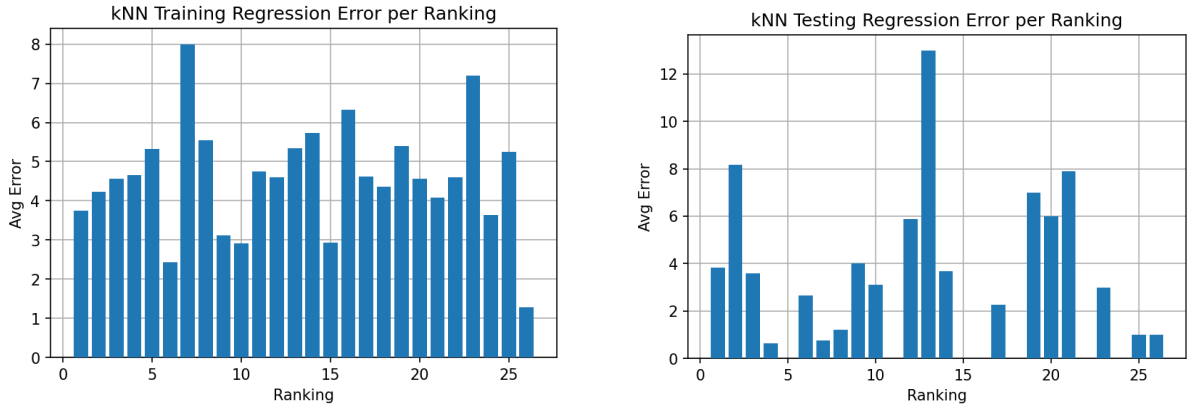


Figure 1: CFB Proportion of Variance Plot

The figure shows that a proportion of variance of nearly 1 is achieved after only 5 eigenvectors; this implies that only 5 dimensional space is needed for this data set. The data is then projected onto the eigenvectors corresponding to these 5 largest eigenvalues before further computations.

The k-NN algorithm is then run, and the following figures show the resultant error. The first two figures show the average regression error for each rank, which is calculated as the average distance between what should have been labeled as a rank and the regression value that was calculated for that instance.



(a) Training (b) Testing

Figure 2: k-NN Ranking Error

The average training error is 4.58 ranks, and the average testing error is 3.02 ranks. This shows that the k-NN algorithm is good at regressing the ranks of around 10 to 15, but the lack of example instances in the top 5 and bottom 5 ranks cause the error to increase. This is to be expected, as data was gathered from the 2015 to 2021 seasons; this leads to only 24 instances of top 4 teams in the whole data set. More instances would allow the k-NN algorithm to have more nearest neighbors to approximate from. k-NN algorithms greatly benefit from additional samples so more neighbors can be utilized.

Prediction Estimates of Common CFB Categories after kNN Regression

Category	True Training	Training Est	True Testing	Testing Est
Top 4	20	7	4	2
Top 10	27	35	7	9
Top 25	68	118	12	23
Not Top 25	601	556	157	146

Figure 3: k-NN Classification Error Table

Figure 3 shows the classes of interest in the data and the number of estimated samples in that class. Again, the table shows that the algorithm is very strong at predicting whether or not a team will be in the top 25 or out of the top 25, but more specific rankings are more difficult due to fewer neighbors. These classes are specific, so an instance in the top 10 only belongs in the top 10 and not also in the top 25, as a team's highest class is the most important. With more samples from more years, the algorithm would be more accurate at predicting top 4 teams.

Overall, the k-NN accuracy is about 4 ranks off, which is very accurate. With the features provided,

the algorithm can predict the final rank of that team to within 4 ranks, much more accurate than a fan would be able to predict for many teams at the end of the season. A team that finishes in spot 1 could be ranked as low as 5 with the algorithm, which is still close enough to be in the discussion of the top 4 and potentially top 1 teams in the country. The k-NN algorithm optimizes computational time with easy implementation and produces great results.

4.2 ANN

The neural network's backpropagation algorithm allows the neural network to take in the full input space without dimensionality reduction. The input space must, however, be normalized according to each feature so that each value is between -1 and 1, the range that the logistic function utilizes in the node value calculations. As such, the network input is the full 14-dimensional input space with one output being the regression value for the instance. Multiple network architecture variables were used before settling on the approximately optimal ones mentioned previously. The network stores the total regression error for each epoch. That error vs epoch is shown below in Figure 4 to show the network learning the relationships between the features.

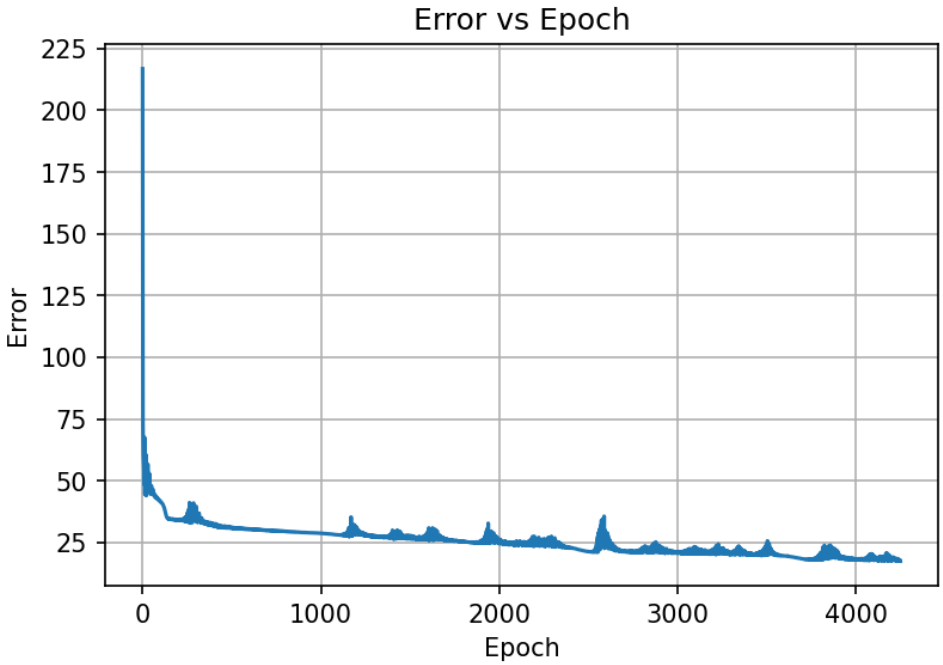
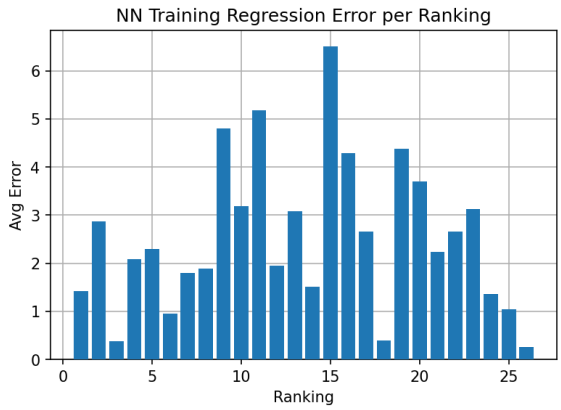
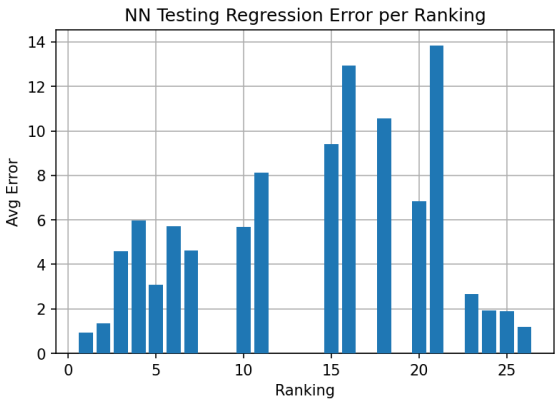


Figure 4: NN Error vs Epoch

As with the k-NN method, the neural network was used to calculate the regression error per rank for the same training data and testing data. These are shown in Figure 5 below; when compared to Figure 2 for the k-NN, the ANN method shows a great improvement over the k-NN. This comes at a cost, as running 4250 epochs of the neural network takes around 10 minutes on this computer.



(a) Training



(b) Testing

Figure 5: NN Ranking Error

As seen in Figure 5 for both training and testing, the NN does a better job than the k-NN at regressing the values for higher ranking teams. This is important, as the ranking of these teams is more important for the playoff structure than lower ranked teams. At an average error for rank 1 teams of only 1 on the training data, the ANN is able to, on average, correctly predict that a team will be rank 1. Due to potential overfitting of the the network to the noisy input space, the network has large errors for specific testing ranks. In comparison to the k-NN method, the ANN has higher accuracy for highly ranked teams but lower accuracy for lower ranked reams, possibly due to overfitting.
Figure 6 below shows how the network classified each of the four major categories listed previously.

Prediction Estimates of Common CFB Categories after NN Regression

Category	True Training	Training Est	True Testing	Testing Est
Top 4	16	17	8	16
Top 10	29	26	5	8
Top 25	66	186	14	38
Not Top 25	605	487	153	118

Figure 6: NN Classification

4.3 Decision Trees

Using MATLAB's Classification Learner, a decision tree with various numbers of branches and leaves can be constructed. This was done by uploading the training data in .csv format into the MATLAB workspace, importing it into the Classification Learner, and classifying the data. MATLAB also has a regression learner, but the results were clearer and more decipherable when using the Classification Learner with the 26 classes/regression values of this data set. The first tree shown in Figure 7 is MATLAB's "fine tree," which includes the most branches and nodes out of any of the trees.

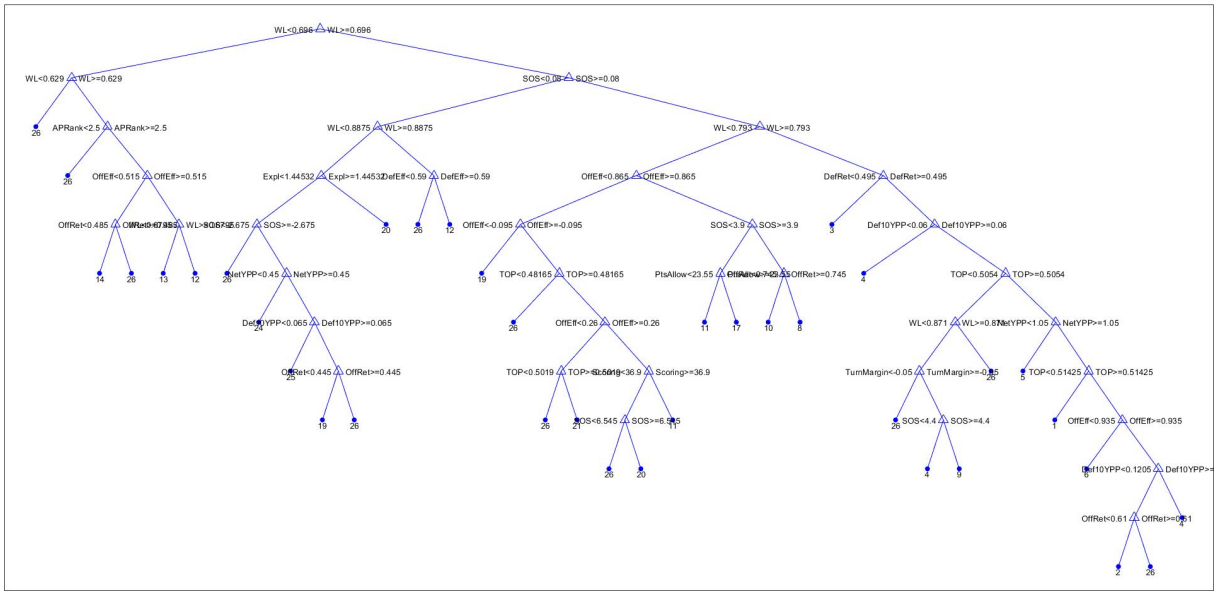


Figure 7: Fine Decision Tree

At this scale and with the feature labels, many of the decision node labels at the bottom are difficult to read; however, as is the case with decision trees, the most important features that divide the input space are nearer the root, which are legible. This fine tree can be compared to Figure **8**, which is a medium tree. This tree uses fewer branches and features and is less accurate for this data set.

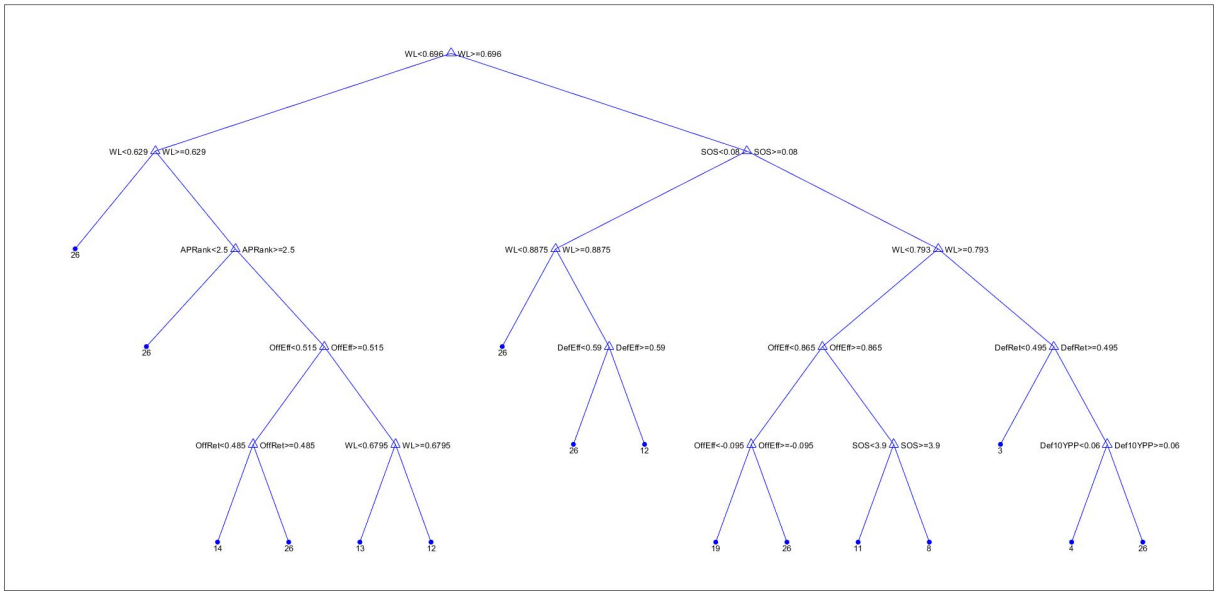


Figure 8: Medium Decision Tree

For both trees, the most important feature, the root feature, is a team's win/loss ratio. This intuitively makes sense: a team that wins more is more likely to finish in a higher rank. This decision boundary is at a win/loss ratio of 0.696, or just over 8 wins out of a typical 12 game season. The next major node on the left side of the fine tree further splits teams with smaller win/loss ratios: if a team has a win/loss ratio of less than 0.629, or less than about 7.5 wins, the classifier is confident that that team will not be ranked in the top 25 and gives it a class of 26. This too makes sense, as teams who finish worse than 8-4 are rarely seen in the top 25. The highest rank teams of 1, 2, 3, etc. are found down the ride side

of the fine tree, with win/loss ratios of greater than 0.793 and strengths of schedule of greater than 0.08. Further features divide the top ranked teams, but there is still a way for a team to have these statistics and have a class of 26. This is likely due to conference strength, where a team may win many games in a weak conference with few other strong teams. This team would not be considered one of the top 25 teams, as they had not competed against other strong teams. According to this classifier, which has an accuracy of 81.2%, the major factors for a team to be ranked highly are as follows: have a win/loss ratio of greater than 0.793 or 9.5 wins per season and a strength of schedule of greater than 0.08. The further delineations are between ranks of 3 and 4, or between 5 and 1, showing that if a team meets the above qualifications, they are on track for a high ranking.

5 Implications/Applications

The major implications of the study are as follows:

- Methods like k-NN and ANN are very strong at predicting a team's final CFP ranking when given the features listed, despite some of the features being extraneous as shown by PCA. Despite the arguably "arbitrary" selection of the top teams, this analysis shows that the teams must still have quantifiable, demonstrable success in order to achieve those rankings.
- The Decision Tree analysis has shown, as expected, that some features matter more than others. The features that matter most in a team's final CFP ranking are its win/loss ratio, strength of schedule, returning production, and explosiveness.

As mentioned previously, data analytics for sports is a large market, with millions of dollars being bet on the results of these games. Data analysts are responsible for generating the models that determine the predicted team to win and the score from any given match up. Machine learning techniques like dimensionality reduction and feature extraction allow data analysts to determine the most important features to feed into their models, so that the models are accurate while still computationally optimized. Using those techniques, models can be made using non-parametric methods like k-NN or ANNs that are accurate and as fast as possible. With constantly changing data sets and large monetary values, the models need to be both.

Outside of data analytics and sports, machine learning techniques like these are applicable particularly in engineering. These techniques allow scientists to ascertain the specific parameters that define physical phenomena in an otherwise chaotic and noisy world.

5.1 Using Model to Predict 2022 Rankings

Since, as the time of writing this, the 2022 CFB season is complete, this study will attempt to apply the models to teams' current statistics, assuming they are available. This will allow us to predict what rank each team would achieve. The final CFP rankings were announced on December 4, and the top 4 are as follows: Georgia, Michigan, Texas Christian, and Ohio State, followed by Alabama, Tennessee, and Utah. Using the cumulative statistics including conference championship weekend, the k-NN and ANN algorithms were run on the 2022 data.

2022 CFP Predictions

Estimated CFP Rank	k-NN	NN
1	Georgia	Georgia
2	Florida State	Kansas State
3	Ohio State	Ohio State
4	Michigan	Alabama

Figure 9: Final 2022 CFP Predictions

Between the two algorithms, five of the top seven true CFP teams are represented, showing that the algorithms are accurately predicting the approximate rankings of the teams. The algorithms weight metrics differently than the CFP committee, as expected. As discussed in the decision tree section, the algorithms weigh a teams win/loss record heavily but only up to around a 9-3 record; for teams with better records, the algorithms weigh the performance of the teams more so than their record. This explains the presence of 3 loss teams like Kansas State and Florida State in the high ranks.

The largest discrepancy between the estimated rankings and the true rankings is the absence of Texas Christian (TCU). This is likely due to the difference between TCU's points scored and points allowed, effectively the margin of victory. TCU won many games this season by less than 7 points. As seen in the decision tree, both the medium and fine trees differentiate between teams with good win/loss ratios based on offensive and defensive efficiency, which would hurt TCU in the estimated rankings. TCU also played few top ranked teams, unike Georgia, Ohio State, Michigan, and Alabama, who all played teams in the top 5 estimated rankings. The decision trees show that Strength of Schedule (SOS) is an important metric, so these teams are benefitted by having a strong schedule.

6 Conclusion

Both the k-Nearest Neighbors and Artificial Neural Network algorithms provide great approximations for a team's final CFP ranking, based on the available data from earlier seasons. Since approximate rankings are able to adequately predicted, the algorithms show that, to some degree, the CFP selection committee considers quantifiable statistics for the teams instead of solely relying the often maligned "eye test." Since methods like k-NN and ANN can adequately capture the variance in college football statistics, these methods are shown to be useful in other chaotic, seemingly non-deterministic fields.

7 References

Below are the metrics used and the databases they were found in at the time of writing.
Final CFP Rankings:

- College Football Playoff. [Online]. Available: https://collegefootballplayoff.com/rankings.aspx

Team win/loss, scoring, points allowed, strength of schedule, and prior AP ranking:

- "College Football National Champions and Seasons," Sports Reference. [Online]. Available: https://www.sports-reference.com/cfb/years/

Offensive and defensive efficiency:

- "FEI Ratings," BCF Toys. [Online]. Available: https://www.bcftoys.com/2022-fei/

Net yards per play and defensive 10+ yards per play:

- "Yards Per Play," BCF Toys. [Online]. Available: https://www.bcftoys.com/2022-ypp/

Explosiveness:

- "Advanced team metrics by season," CollegeFootballData.com. [Online]. Available: https://collegefootballdata.com/exporter/stats/season/advanced?year=2022

Turnover margin:

- "College FB team turnover margin per game," College Football Stats - College FB Team Turnover Margin per Game — TeamRankings.com. [Online]. Available: https://www.teamrankings.com/college-football/stat/turnover-margin-per-game

Offensive and defensive returning production:

- B. Connelly, "Here's a better way to measure 2019 college football returning production," SBNation.com, 31-Jan-2019. [Online]. Available: https://www.sbnation.com/college-football/2019/1/31/18204093/2019-ncaa-football-returning-starters-experience
- T. Nettuno, "Florida football has a lot of work ahead in replacing production for 2021 season," USA Today, 01-Feb-2021. [Online]. Available: https://gatorswire.usatoday.com/2021/02/01/florida-gators-returning-talent-ranking-2021/
- B. Connelly, "College Football Teams' returning production for 2022 season," ESPN, 08-Feb-2022. [Online]. Available: https://www.espn.com/college-football/insider/story//id/33237908/college-football-teams-returning-production-2022-season

Time of Possession:

- "College FB team 1st half time of Possession Share %," College Football Stats - College FB Team 1st Half Time of Possession Share % — TeamRankings.com. [Online]. Available: https://www.teamrankings.com/college-football/stat/1st-half-time-of-possession-share-pct.

All data can also be found on my github repository: https://github.com/willfossett/CFB