CPTS 437 Final Project: Fantasy Football Classification

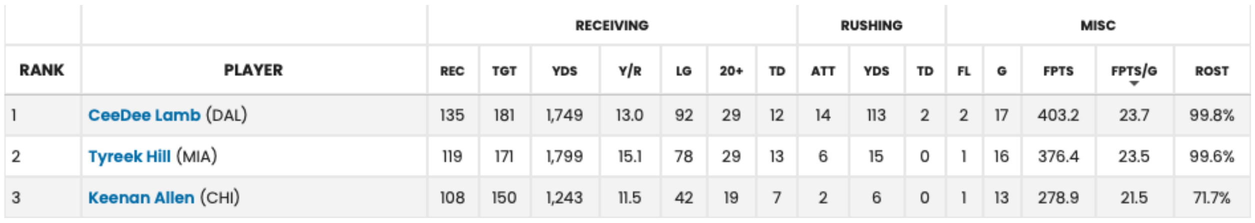
Caleb Anderson, Jayden Jinks, Joseph File, Darin Hardie

Background

In fantasy football, players earn points based on their real-life statistical performance each week, resulting in an abundance of statistics to analyze and many machine learning applications with these statistics. According to a 2023 USA Today article, Fantasy Football is an over $11 billion business in which over 29 million Americans compete yearly [1]. Thus, the ability to use machine learning to predict fantasy football outcomes is both a practical and valuable skill. In this project, our team has trained three classification models on fantasy football data from 2002-2023 to determine which is best suited for classifying the wide receiver (WR) and running back (RB) positions into tiers based on their statistics, as well as which is most capable of identifying potential trade targets based on data from the first 6 weeks of the 2024 season. To understand this project, it is crucial to first understand the format of fantasy football we are considering for this project. In a standard fantasy football league, a starting lineup consists of 1 quarterback, 2 running backs, 2 wide receivers, 1 tight end (TE), 1 FLEX, which can be filled by either a WR, RB, or TE, 1 defense/special team, and one kicker. Thus, RBs and WRs usually make up 5 of 8 starting spots, making them the most important positions in fantasy football, and the two positions we chose to examine in this project. On top of understanding the lineup format, it is important to understand the scoring format (how the players' real-life statistics translate to fantasy football scoring) to fully understand the tasks of this project. Our project considers a point-per-reception (PPR) scoring format and the scoring rules most relevant to our investigation of the WR and RB positions are that players receive 1 point per reception, 0.1 points per rushing or receiving yard, and 6 points for every rushing or receiving touchdown. As previously mentioned, our team has selected 3 machine learning classification models to compare their performance in classifying RBs and WRs into tiers based on their statistics. The three models we selected for this project are a decision tree and naïve bayes model, which we covered in class but provide a very different strategy from each other in how they classify, as well as a gradient boosting model which we did not cover in class and provides another different classification strategy. The final crucial background detail to understand our classification task is how our team is defining the tiers we are classifying players into. These tiers will be defined by where the players rank in fantasy points per game (PPG). In a standard fantasy football league, there are 12 teams, meaning if the talent is evenly distributed, each team will have one WR and one RB ranked between 1-12 in fantasy points per game at each position. These players belong to the tiers of WR1 and RB1 respectively. Similarly, players ranked 13-24 in PPG at each position will be placed into WR2 and RB2 tiers respectively, and those ranked 25-36 will belong to the FLEX tier. A standard fantasy football bench consists of 7 spots, at most 6 of which are usually filled by RBs and WRs, thus players ranked 37-60 at their position will be placed in the "Bench" tier, and all players ranked 60+ will be placed in the "Cut" tier. If we can accurately classify players into these tiers, we will be able to identify which players are currently underperforming their tier based on historical data and identify these players as trade targets, showing the application of classification tasks in fantasy football.

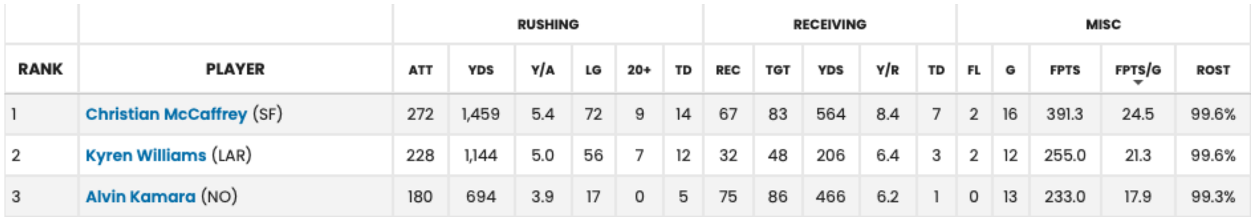
Dataset

For this project, our dataset comes from FantasyPros, a fantasy football site with data for each position dating back to the 2002 season [2]. Below, Figure 1 and Figure 2 provide data samples from the 2023 season for the WR and RB positions, respectively.



		RECEIVING							RUSHING			MISC				
RANK	PLAYER	REC	TGT	YDS	Y/R	LG	20+	TD	ATT	YDS	TD	FL	G	FPTS	FPTS/G	ROST
1	CeeDee Lamb (DAL)	135	181	1,749	13.0	92	29	12	14	113	2	2	17	403.2	23.7	99.8%
2	Tyreek Hill (MIA)	119	171	1,799	15.1	78	29	13	6	15	0	1	16	376.4	23.5	99.6%
3	Keenan Allen (CHI)	108	150	1,243	11.5	42	19	7	2	6	0	1	13	278.9	21.5	71.7%

Figure 1 - 2023 WR Data Sample



		RUSHING						RECEIVING					MISC				
RANK	PLAYER	ATT	YDS	Y/A	LG	20+	TD	REC	TGT	YDS	Y/R	TD	FL	G	FPTS	FPTS/G	ROST
1	Christian McCaffrey (SF)	272	1,459	5.4	72	9	14	67	83	564	8.4	7	2	16	391.3	24.5	99.6%
2	Kyren Williams (LAR)	228	1,144	5.0	56	7	12	32	48	206	6.4	3	2	12	255.0	21.3	99.6%
3	Alvin Kamara (NO)	180	694	3.9	17	0	5	75	86	466	6.2	1	0	13	233.0	17.9	99.3%

Figure 2 - 2023 RB Data Sample

As seen in the above figures, our data set provides statistics for receptions (REC), targets (TGT), receiving yards (YDS), rushing yards (YDS), rush attempts (ATT), rushing touchdowns (TD), receiving touchdowns (TD), yards per reception (Y/R), games played (G), and fantasy points per game (FPTS/G), for both the WR and RB datasets. The two datasets differ as in the WR dataset LG represents the longest reception of the season and 20+ represents the number of 20+ yard receptions for the player on the season. Meanwhile, for the RB dataset, LG represents the player's longest rush of the season, and 20+ represents the number of 20+ yard rushes for the player on the season. It is also important to note that each dataset is sorted by the FPTS/G column meaning the rank column, which is used to determine the players' tiers in the training data, is sorted in order of fantasy points per game.

Data Cleaning and Preprocessing

Before implementing the three classification models, our team first needs to perform data cleaning and preprocessing to load our data into our coding environment and alter the formatting of the data, so it is appropriate for our specific classification task. The first step of this process is loading all the datasets from 2002-2023 and combining these datasets into two data frames, one for WRs and the other for RBs. Because all our data is sorted by fantasy points per game, it is necessary to convert all the statistics which are cumulative across the entire season to per-game statistics using the games played statistic for each player, so that the statistics used for classification match the statistic used to determine rank, which determines the classification tier. These columns which needed to be converted include receptions, targets, receiving yards, rush attempts, rushing yards, receiving touchdowns, and rushing touchdowns. The next step in the preprocessing process is to create a tier column and convert each player's rank into their appropriate tier based on the previously defined tier guidelines. Finally, the last alteration that needs to be made to the data is to remove the statistics that hold little to no relevance to per-game fantasy score, and thus will not be used for classification, such as fumbles lost, games played, roster percentage, and total fantasy points. Once the data has been correctly cleaned and formatted, our team identified the target feature as the 'tier' column and split the data of each of the two data frames in 5 different random ways so that 80% of the data would be used for training and includes the 'tier' column, and 20% of the data would be used for testing and did not consider the 'tier' column.

Decision Tree

The first of three classification models that our team implemented was a decision tree classifier. A decision tree is a tree-shaped, flowchart-like structure which is made up of nodes containing decisions about attributes, branches which represent the outcomes of the decisions, and leaf nodes which represent the final classification prediction, in our case the tier the player belongs to [3]. In our implementation of the decision tree, we used an entropy calculation to determine which feature to split on, and where to split that feature by iterating through the features to determine which provides the most information gain. Our team also included a parameter in the decision tree implementation which allows us to select the maximum depth of the decision tree. This allows us to set different depths between the tree trained on RB data and the tree trained on WR data as well as test several depths to see which one provides the best testing accuracy. Our team performed this test plotting the training and testing accuracies of each model against the depth of the tree for depths 5-12, as seen in Figure 3 below.

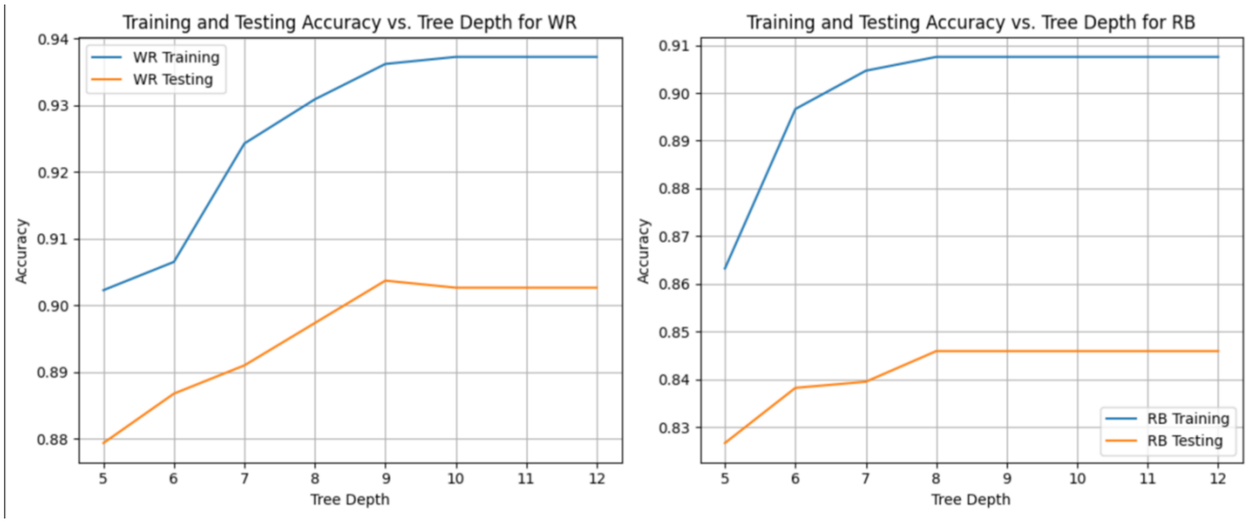


Figure 3 - Training and Testing Accuracies vs. Tree Depth for WRs and RBs

Based on the plots in Figure 3, our team decided to use a depth of 9 for the WR tree and a depth of 8 for the RB tree. Trees of these depths were then used to evaluate the performance of our models, as will be seen in the Results and Analysis section and determine possible trade targets in an application of our classification models, as will be explained in the Application section.

Naïve Bayes Classifier

The second classification model our team implemented is a naïve bayes classifier which is a type of classifier that predicts the probability that a data sample belongs to a class and selects the class with the highest probability. Naïve Bayes classifiers operate under the assumption that all features it considers are independent of each other, which may result in performance issues with our classification task as many of the features are strongly correlated [4]. However, this classification model provides a different type of classifier than the decision tree because it is an algorithm based and probabilistic classifier rather than the tree structure given by a decision tree, so our team still wanted to investigate how it would handle our classification task. Our naïve bayes model first calculates the prior probabilities of each class by determining the fraction of our data samples which belong to that class and calculates statistics for each feature including the mean (μ) and the variance (σ^2). Our team chose to implement a Gaussian naïve bayes classifier because it was what we were most familiar with, which means we use the following Gaussian probability density function to calculate the likelihood that a given feature x_i belongs to a class *c:*

$$P(x_i|C=c) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp(-\frac{(x_i-\mu)^2}{2\sigma^2})$$

Because a naïve bayes classifier assumes independence between features, these probabilities were then multiplied across all the features to determine the likelihood that a given sample belongs to a certain class. We then used the prior probabilities (P(C=c)) and the calculated likelihoods (P(x|C=c)) to calculate the posterior for a given sample x and class c using the following equation:

$$P(C = c|x) = P(C = c) * P(x|C = c)$$

Once performing this calculation for each possible tier as c, the model identifies the greatest posterior and predicts that as the tier value for that sample. The iterates through all the training data samples performing this process to predict the tier of each one.

Gradient Boosting Classifier

The third and final classification model that our team elected to implement is a gradient boosting model. A gradient boosting model combines several weaker learning models to make a stronger learning model by minimizing the loss function using gradient descent [5]. Our team chose to implement this type of classification model because it was something we were less familiar with and combined some of the qualities of the decision tree with the algorithm focus of the naive bayes model. To begin the classification process, each class is given an equal probability of 1/number of classes. We then compute the gradient using the below function with respect to the initialized predictions:

$$Gradient = y - P$$

where *y* is the class labels encoded numerically, and P is the probability distribution of the sample. Next, for each tier class, a decision tree is trained using the DecisionTreeRegressor imported from scikit learn and the gradient values updated in the previous step. Then these trees are used to update the prediction values by adding the product of the learning rate at the prediction value for the class based on the current tree for that class. This process then repeats itself improving the prediction and gradient with each iteration. At the end of the iterations, the prediction scores (F) are converted into probabilities using the following function, and the highest probability is the class predicted:

$$P_{i,c} = \frac{\exp\left(F_{i,c}\right)}{\sum_j \exp\left(F_{i,j}\right)}$$

Results and Analysis

After training and testing each model on over 20 seasons of data, from 2002-2023, which we split in 5 different ways using different splitting states, our team used the accuracy and F1 score functions provided by scikit learn to evaluate the performance of our models in classifying this historical data, allowing us to draw several conclusions about our models, the first of which being that the Decision Tree and Gradient boosting classification models consistently outperformed the Naïve Bayes model as seen below in Figure 4.

	RB	WR

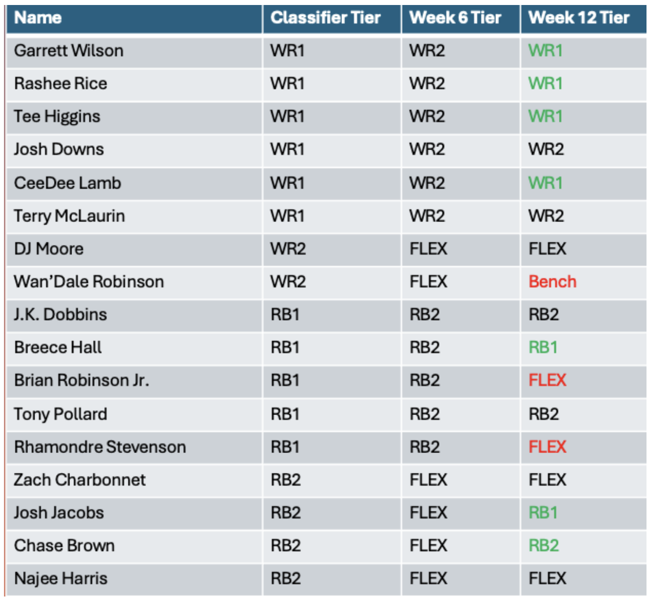
	Avg. Testing Accuracy	Accuracy Standard Dev.	Avg. F1 Score	F1 Score Standard Dev.	Avg. Testing Accuracy	Accuracy Standard Dev.	Avg. F1 Score	F1 Score Standard Dev.
Decision Tree	0.8513	0.0146	0.8494	0.0165	0.8942	0.0121	0.8933	0.0133
Naïve Bayes	0.7954	0.0155	0.8069	0.0128	0.8286	0.0130	0.8377	0.0136
Gradient Boosting	0.8714	0.0132	0.8678	0.0140	0.9031	0.0129	0.9004	0.0137

Figure 4 - Model Performance over 5 Random Splits

Our team concluded that this trend was a result of both our implementation of the naïve bayes model as well as the nature of a naïve bayes classifier as it relates to our specific task. We used a Gaussian Naïve Bayes classifier, which operates under the assumption that the data for the features would be normally distributed. Upon noticing this model was performing worse, our team attempted to fit a scaler to our data to normalize it, hoping to improve performance, but were unsuccessful, leaving the model to try to handle data that may be skewed, for example, a few long yardage plays may result in the yards per reception feature having a long tail. However, our team feels the naïve bayes classifier struggled not only because of our implementation but also because of its simplicity in nature causing it to struggle for our tasks. Decision trees and our gradient boosting model can handle interactions between features due to the several levels of trees. For example, a player with less receptions is likely going to need more yards per reception to meet reach the same tier as a player with more receptions, and a tree can represent both of these possible pathways to the same tier, while the naïve bayes classifier doesn't model these interactions between tiers, making it less well suited for the complex relationship between statistics in our fantasy football classification task. Another trend our team noticed was that the models consistently performed better at classifying WRs than RBs. Our team concluded that this trend was likely caused by the nature of the PPR scoring format. In this scoring format, the common WR statistics translate much more directly to points than RB statistics do, making it easier to classify them based on their statistics. For example, most of a WRs opportunities in a game will come in the form of receptions, which directly translate to 1 point in PPR scoring, meaning we know a WR who gets lots of targets and receptions is guaranteed to be receiving points for their opportunities, regardless of their yards, while a RB may receive lots of rush attempts but they do not receive a point for these opportunities alone, they are reliant on the yards they gain. Thus, the provided WR statistics translate much more directly to fantasy scoring than some of the RB statistics making it easier to classify WRs than RBs. While some may consider our testing accuracies to be low, our team was pleased with the accuracies of our Decision Tree and Gradient Boosting models because there are many factors to fantasy football that are not easily represented in statistics, which are going to result in more outliers and errors than in other strictly data-driven classification tasks. For example, our models cannot consider injuries either to players or to their teammates which could impact player performance either for the better or worse, potentially creating outliers. Additionally, our models cannot account for the change in the way football is played that has occurred across the 20+ seasons of data we have collected, whether that be differing rules or even just different tactics which could impact player performance. This may result in our models being unable to express all the trends that lead a player to a specific tier because there have been so many different playing styles resulting in differing player performance over these seasons. Thus, our team was pleased with the performance of our decision tree and gradient boosting models given the complexity of the classification task they were faced with.

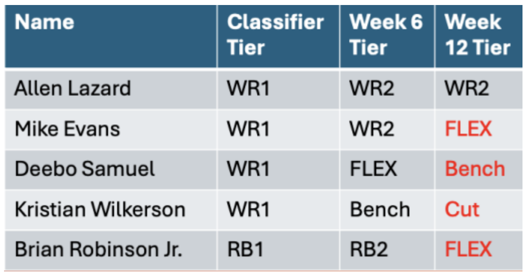
Application

While being able to classify fantasy football players into their appropriate tiers is interesting, the most crucial exploration of this project is how we can apply this classification ability in a manner that helps fantasy football players. In fantasy football, teams can trade players amongst each other, thus, if players can accurately classify players into tiers, they can identify trade targets by seeing which players are classified into tiers higher than their current performance indicates. To test our models on this application, our team gathered data from the first 6 weeks of the 2024 NFL season and examined all the players each model classified into the WR1, RB1, WR2, and RB2 tiers. We identified which of these players were classified into tiers higher than their rank would place them at that point of the season and deemed these players trade targets. Our team then examined the fantasy football data through 12 weeks of the 2024 season to determine which of these trade targets had moved up tiers, which had stayed the same, and which had moved down tiers. These trade targets can be seen below in Figures 5, 6, and 7, with green text in the "Week 12 Tier" columns representing an improvement in tier, black text representing no change, and red text representing that the player had moved down a tier since week 6.



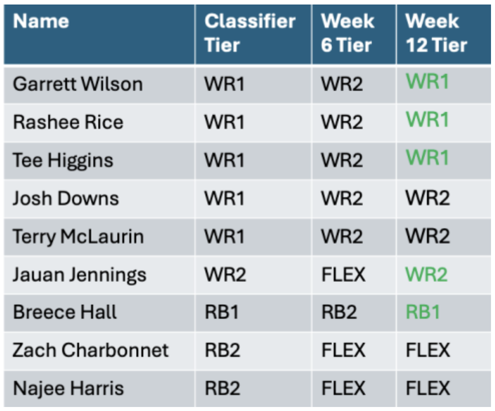
Name	Classifier Tier	Week 6 Tier	Week 12 Tier
Garrett Wilson	WR1	WR2	WR1
Rashee Rice	WR1	WR2	WR1
Tee Higgins	WR1	WR2	WR1
Josh Downs	WR1	WR2	WR2
CeeDee Lamb	WR1	WR2	WR1
Terry McLaurin	WR1	WR2	WR2
DJ Moore	WR2	FLEX	FLEX
Wan'Dale Robinson	WR2	FLEX	Bench
J.K. Dobbins	RB1	RB2	RB2
Breece Hall	RB1	RB2	RB1
Brian Robinson Jr.	RB1	RB2	FLEX
Tony Pollard	RB1	RB2	RB2
Rhamondre Stevenson	RB1	RB2	FLEX
Zach Charbonnet	RB2	FLEX	FLEX
Josh Jacobs	RB2	FLEX	RB1
Chase Brown	RB2	FLEX	RB2
Najee Harris	RB2	FLEX	FLEX

Figure 5 - Trade Targets Identified by Decision Tree Model



Name	Classifier Tier	Week 6 Tier	Week 12 Tier
Allen Lazard	WR1	WR2	WR2
Mike Evans	WR1	WR2	FLEX
Deebo Samuel	WR1	FLEX	Bench
Kristian Wilkerson	WR1	Bench	Cut
Brian Robinson Jr.	RB1	RB2	FLEX

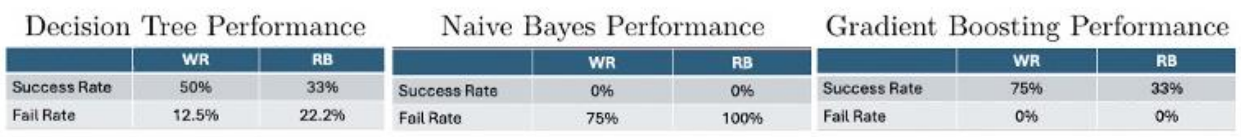
Figure 6 - Trade Targets Identified by Naive Bayes Model



Name	Classifier Tier	Week 6 Tier	Week 12 Tier
Garrett Wilson	WR1	WR2	WR1
Rashee Rice	WR1	WR2	WR1
Tee Higgins	WR1	WR2	WR1
Josh Downs	WR1	WR2	WR2
Terry McLaurin	WR1	WR2	WR2
Jauan Jennings	WR2	FLEX	WR2
Breece Hall	RB1	RB2	RB1
Zach Charbonnet	RB2	FLEX	FLEX
Najee Harris	RB2	FLEX	FLEX

Figure 7 - Trade Targets Identified by Gradient Boosting Model

From these trade targets, our team was able to make several observations, both about the number of trade targets, and the accuracy of their model. It is immediately apparent that the decision tree model identified far more trade targets than either of the other two models. Our team attributed this abundance of trade targets to the fact that the model has been trained on 20+ seasons of data, and across those seasons players have had many different statistical paths to reaching a specific tier. Thus, the decision tree is trying to represent all these possible paths, but not all these paths will result in a RB1 or WR1 finish in this season, resulting in too many trade targets being identified. Our team also noticed that the naïve bayes model identified the fewest trade targets. This is likely because of the model's difficulty handling interactions between features as discussed earlier which results in it failing to classify players outside of their current tiers. Beyond observations about the quantity of targets each model identified, our team also examined the performance of each model by determining the success rate (the percentage of targets that moved up in tiers) and the fail rate (the percentage of targets that moved down a tier) as seen in Figure 8 below.



Decision Tree Performance

	WR	RB
Success Rate	50%	33%
Fail Rate	12.5%	22.2%

Naive Bayes Performance

	WR	RB
Success Rate	0%	0%
Fail Rate	75%	100%

Gradient Boosting Performance

	WR	RB
Success Rate	75%	33%
Fail Rate	0%	0%

Figure 8 - Trade Target Identification Performance

In fantasy football, when making a trade, you of course hope that your player outperforms the value you got them at, but most important is that the player doesn’t underperform that value. For this reason, our team deemed the fail rate metric more valuable for judging model performance in this application. The naïve bayes model once again drastically underperformed the other two models with very high fail rates, and no success, however, these numbers are also inflated due to a very small sample size. Meanwhile, the decision tree and gradient boosting models both performed quite well, achieving relatively low fail rates, especially the gradient boosting model which had no targets decrease in tier. The decision tree did experience some failures, but this is to be expected considering the models cannot consider injuries to players or their teammates which could create outliers, which is why our team was incredibly pleased with the performance of the gradient boosting model for this application. On top of the zero failures, the gradient boosting model also had a 75% success rate on WR trade targets, which was very successful. While we didn’t achieve very high success rates for the RB position, the low failure rate was still encouraging, especially since many of the RB trade targets had their season impacted by injuries. It is also worth noting that one may be able to minimize the error rate of the decision tree model by restricting the maximum depths further, this will result in less trade targets, improving performance in this model, even if it hinders performance in testing. Ultimately, our team concluded that the decision tree and gradient boosting models could both be used successfully to identify trade targets in fantasy football, especially alongside some human analysis of how injuries have impacted a player’s statistics, and how they will do so for the rest of the season.

Future Work and Improvements

Our team felt that it was important to reflect on how our project could be expanded and improved upon, as well as how machine learning can be applied in other ways to the field of fantasy football. First, regarding improvements to our project, the most obvious would be improving our naïve bayes model. While our team felt that this model was not well suited to this task, we do believe it could still be improved upon by preprocessing the data to normalize it. Additionally, our team felt the performance of the other models could be improved regarding the trade target application by being more selective of the seasons we chose to train the models on. The way football is played is different now than it was in 2002, which results in our models learning statistical trends that may have been relevant in older seasons but not so much now, and when applied to the current season of statistics, causes the models to identify trade targets that follow trends that are no longer accurate, especially the decision tree model. Thus, if our team could determine a range better suited to this application, we could certainly improve our models’ performances when it comes to identifying trade targets. As far as expanding upon our current project, there are many other classifications models our team could try to apply to this classification task, such as neural networks, other boosting algorithms, or k-nearest neighbor. By testing more algorithms our team could determine which is best for identifying trade targets through classification. As far as other machine learning applications in fantasy football goes, our team identified a few other possibilities. First, we could investigate ranking algorithms to provide a more precise ranking of the players, rather than just using tiers. Our team could also explore much more in-depth data set which contain statistics regarding player matchups, as well as more in-depth statistics like routes run or target share. Finally, machine learning could be applied to fantasy football in the context of projecting a player's points on a weekly basis by developing an algorithm which considers the players average statistics as well as the average statistics of their opponent to try to predict the points that player will score in that coming week.

Conclusion

In conclusion, our team set out to see if classification models could be effectively used to classify fantasy football players into various tiers based on their statistics, and which of a decision tree, naïve bayes, and gradient boosting classifier would be most effective at this task. We found that, while imperfect, our decision tree and gradient boosting models can both be used to effectively classify players in the WR and RB positions and that the naïve bayes model was not well suited to the task because it was too simplified for the interactions between features. We also found that the decision tree and gradient boosting models were effective in the application of identifying trade targets based on the comparison between their classification and their tier to that point in the season, especially the gradient boosting model which had zero errors in this application. Overall, we determined that classification models can be a very valuable tool for fantasy football, especially when used alongside human analysis of factors like injuries, as long as you train a model complex enough to handle the connections between the real-life statistics.

References

[1] S. Gardner, “Money. Power. Women. The driving forces behind fantasy football’s skyrocketing popularity.,” *USA TODAY*, Dec. 15, 2023. https://www.usatoday.com/story/sports/nfl/fantasy/2023/12/15/fantasy-football-sports-economy/71870731007/

[2] “2023 NFL WR Statistics | Fantasy Football PPR | FantasyPros,” *www.fantasypros.com*. https://www.fantasypros.com/nfl/stats/wr.php?scoring=PPR

[3] GeeksForGeeks, “Decision tree - geeksforgeeks,” *GeeksforGeeks*, May 17, 2024. https://www.geeksforgeeks.org/decision-tree/

[4] GeeksForGeeks, “Naive Bayes Classifiers,” *GeeksforGeeks*, Mar. 03, 2017. https://www.geeksforgeeks.org/naive-bayes-classifiers/#

[5] GeeksForGeeks, “ML - Gradient Boosting,” *GeeksforGeeks*, Aug. 25, 2020. https://www.geeksforgeeks.org/ml-gradient-boosting/

Deliverables Links

Presentation Link:
https://drive.google.com/file/d/1Cy3PXIzTpbqAqc81NfZxf08p0b2ilo5W/view?usp=share_link

Demo Notebook Link:

https://colab.research.google.com/drive/1NjRn_6YRNZozrcdRdE4mDso0luGEWckT?usp=share_link

Demo Video Link:

- **Youtube: https://youtu.be/MuFgC27BalA**
- **Google Drive: https://drive.google.com/file/d/1mg1defVoWR5eoaxDtLya8yJ4qPUtpxml/view?usp=share_link**

Github Link:

https://github.com/31caleb/CPTS437-Fantasy-Football-Classification-Group12

Google Drive Folder Link:

https://drive.google.com/drive/folders/1sEZIc0JHK3eOsTobknxE5MvaMWnPqqAE?usp=share_link