Furman Basketball Report
Matt Smith, Marissa Pennings, John Rado

Introduction:

The primary purpose of our project is to use data mining to analyze Furman basketball performance training data to identify key attributes that influence our class attribute of game outcome. We used historical performance data from each game since the 2010-2011 season to find patterns and trends that contribute to winning or losing. With the help of the machine learning algorithms in Weka, we look to turn raw data into actual insights that can support coaching staff when deciding what to focus on in a game or practice.

Our objective is to find the classification model with the highest accuracy and interpretability based on our performance-related attributes. We would want to take this project to Coach Richey with tangible results to give him something of interest to better understand what metrics are important for wins. Through this, we will attempt to answer the questions of which attributes provide us the most information gain about winning or losing, we can do this prediction with non-score related attributes, and which of these attributes are furthering our predictions versus which are just noise? We use various algorithms, including ZeroR, OneR, Naive Bayes, IBk, J48, and Random Forest, to see which handles this type of data best and how feature selection can improve our accuracy.

Data Set Description:

Our data set is from sports-reference.com, and we went to Furman's page for statistics. The set was created by our team, as we had to go extract the data through a CSV file from each game of each season, one at a time. The set is made up of 489 instances and 48 attributes. Each game from the fifteen years of Furman basketball is categorized in order, with instance 1 being the first game of the 2010-2011 season and instance 489 being Furman's most recent NIT game against North Texas. The attributes in this dataset provide the time, place, and basic statistics on both Furman and the opposition. Luckily, the data from sports-reference.com was robust. From Furman's 2010-2011 season to the present, all 489 instances had all data available from all 48 attributes. We used data from the 2010 season through the 2021 season as training data and the 2022 season to the present as test data. We will keep in mind potential redundant attributes, such as the percentage of two attributes divided.

Attribute	Description	Values
Location	Where did the game take place?	Home Away Neutral
Type	Basketball game based on its context	REG-(Non-Conf) REG-(Conf) CTOURN NIT

		ROUND-64 ROUND-32 CIT
Furman_Score	Amount of points Furman scored in a game	numeric
Opponent_Score	Amount of points Furman's opponent scored in a game	numeric
OT_happen	Did the game go into overtime?	No OT 2OT
FG_Furman	How many field goals Furman made	numeric
FGA_Furman	How many field goals Furman attempted	numeric
FG_Percent_Furman	Ratio of Furman's made field goals to attempted field goals	numeric
3P_Furman	How many three pointers Furman made	numeric
3PA_Furman	How many three pointers Furman attempted	numeric
3P_Percent_Furman	Ratio of Furman's made three pointers to attempted three pointers	numeric
2P_Furman	How many two pointers Furman made	numeric
2PA_Furman	How many two pointers Furman attempted	numeric
2P_Percent_Furman	Ratio of Furman's made two pointers to attempted two pointers	numeric
eFG_Percent_Furman	Ratio of Furman's made baskets, weighting a 3P more due to it scoring more points	numeric
FT_Furman	How many free throws	numeric

	Furman made	
FTA_Furman	How many free throws Furman attempted	numeric
FT_Percent_Furman	Ratio of Furman's made free throws to attempted free throws	numeric
ORB_Furman	How many offensive rebounds Furman tallied	numeric
DRB_Furman	How many defensive rebounds Furman tallied	numeric
TRB_Furman	How many total rebounds Furman tallied	numeric
AST_Furman	How many assists Furman tallied	numeric
STL_Furman	How many steals Furman tallied	numeric
BLK_Furman	How many blocks Furman tallied	numeric
TOV_Furman	How many turnovers Furman tallied	numeric
PF_Furman	How many fouls Furman tallied	numeric
FG_Opponent	How many field goals opponents made	numeric
FGA_Opponent	How many field goals opponents attempted	numeric
FG_Percent_Opponent	Ratio of opponent's made field goals to attempted field goals	numeric
3P_Opponent	How many three pointers opponents made	numeric
3PA_Opponent	How many three pointers	numeric

	opponents attempted	
3P_Percent_Opponent	Ratio of opponent's made three pointers to attempted three pointers	numeric
2P_Opponent	How many two pointers opponents made	numeric
2PA_Opponent	How many two pointers opponents attempted	numeric
2P_Percent_Opponent	Ratio of opponent's made two pointers to attempted two pointers	numeric
eFG_Percent_Opponent	Ratio of opponent's made baskets, weighting a 3P more due to it scoring more points	numeric
FT_Opponent	How many free throws opponents made	numeric
FTA_Opponent	How many free throws opponents attempted	numeric
FT_Percent_Opponent	Ratio of opponent's made free throws to attempted free throws	numeric
ORB_Opponent	How many offensive rebounds opponents tallied	numeric
DRB_Opponent	How many defensive rebounds opponents tallied	numeric
TRB_Opponent	How many total rebounds opponents tallied	numeric
AST_Opponent	How many assists opponents tallied	numeric
STL_Opponent	How many steals opponents tallied	numeric
BLK_Opponent	How many blocks opponents tallied	numeric

TOV_Opponent	How many turnovers opponents tallied	numeric
PF_Opponent	How many fouls opponents tallied	numeric
Result	The outcome of the game for Furman	Win Lose

Data Preparation:

Our data preparation process was fairly straightforward but tedious, due to the significant amount of manual appending required. Fortunately, no data cleaning was necessary. As mentioned earlier, Sports-Reference.com served as a robust and reliable data source, with no missing values or outliers.

The primary steps in preparing the dataset involved reduction and transformation. We began by copying the data for both Furman and their opponent into an Excel spreadsheet. From there, we removed irrelevant attributes, including player names and minutes played. These variables lacked predictive value, particularly player names, as our model spans a 15-year period and players typically remain on a team for a maximum of five years, making this attribute non-generalizable.

Next, we transformed the data from an individual-player level to a team-level format. The original dataset listed statistics by player, but our goal was to evaluate team performance. Therefore, we aggregated the data: cumulative stats such as points, rebounds, and fouls were summed, while percentage-based metrics (e.g., field goal percentage) were averaged. After compiling team-level statistics for both Furman and their opponent, we removed all individual-level data and retained only the final aggregated values for each game.

The next step was data extraction. We manually added the class attribute, whether Furman won or lost, which was available on the same page as the statistics. To capture additional context that might influence game outcomes, we also included fields for game location, game type, and whether the game went into overtime. This full process was repeated for every game instance, using data pulled from Sports-Reference.com.

The final steps were in Weka. We were a bit concerned after loading our dataset into Weka that there could be some potentially redundant attributes, as some attributes were just a ratio of two other attributes. For example, 3P_Furman is divided by 3PA_Furman to get 3P_Percent_Furman. We ran both CorrelationAttributeEval and InfoGainAttributeEval and concluded that none of these potentially redundant attributes and identical or even similar correlation or info gain to their counterparts, so we are confident keeping them in the dataset. We are also keeping attributes that have a correlation of 0, due to the nonlinear potential in Random Forest. Most of our attributes needed to be discretized as we were using a nominal class attribute, so that was the final step in Weka for pre-processing before testing our data.

Data Analysis:

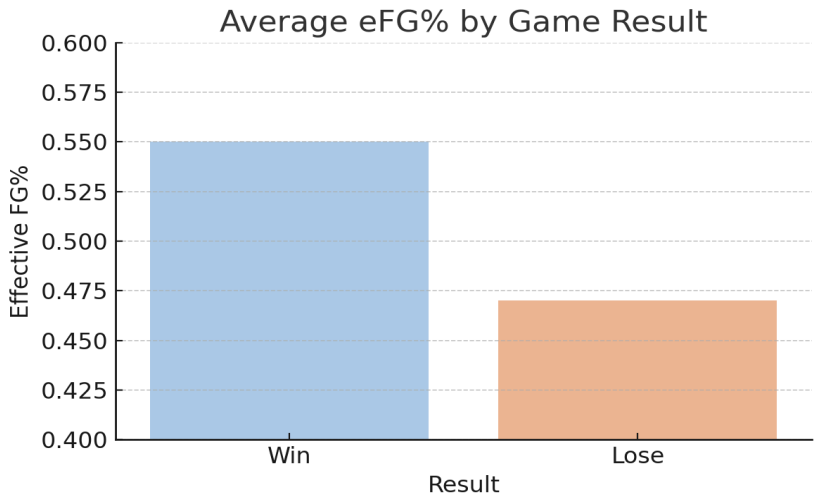
We began our analysis with **ZeroR**. ZeroR is a baseline classification algorithm that does not take into account any attributes of the data, rather, it predicts the majority class for every instance. This was a quick algorithm for us to use as a reference point as we moved throughout our analysis.

We found that with ZeroR, we were able to predict our class attribute of Win or Lose with 67.3077% accuracy. This algorithm classified all of our 104 testing instances as a Win, correctly predicting 70 instances while incorrectly predicting 34 instances.

After ZeroR to form another baseline, we looked at the slightly more complex algorithm **OneR**. OneR uses a single attribute to create one rule that classifies our instances. To pick the attribute, Weka runs through each to see which has the highest predictiveness or the lowest error rate. This benchmark is a useful tool, but does miss out on potential interactions between attributes. OneR was able to predict our class attribute with 73.0769% accuracy. eFG_Percent_Furman was chosen by Weka as our most predictive attribute. eFG_Percent_Furman is an aggregate of two-point and three-point percentage, weighing three-point percentage a bit more due to it resulting in more points. With this rule, OneR correctly predicted 56 wins and 20 losses, while misclassifying 14 wins and 14 losses. Due to the complexity of the rule used, we should worry greatly about overfitting with this algorithm. When thinking about this logically with basketball, it does make sense that Furman's effective field goal percentage should be fairly accurate at predicting their wins and losses, although it doesn't intuitively make sense that the class attributes values continue to change as eFG_Percent_Furman increases.

eFG_Percent_Furman:
- < 0.4915 -> Lose
- < 0.5265 -> Win
- < 0.5295000000000001 -> Lose
- < 0.5925 -> Win
- < 0.601 -> Lose
- >= 0.601 -> Win

(310/385 training instances correct)



As we see, eFG_Percentage_Furman is extremely predictive but provide no insight on how these wins are accomplished. We want to look past this into more hidden information.

After we had these baseline results, we began moving into more complex algorithms. **IBk Nearest Neighbor** was the first one we evaluated. This algorithm finds the “k” number of nearest neighbors to our instance and bases the class attribute prediction on those instance class attributes.
IBk was able to predict our class attribute on the test data with 80.7692% accuracy. It correctly predicted 56 wins and 28 losses while misclassifying 6 losses as wins as well as 14 wins as losses. Our accuracy of predicting wins has not changed compared to OneR, with still having 56 correctly classified instances; however, with this model, we see improvement on the accuracy of predicting losses, with 8 more correctly predicted losses.

Naive Bayes is a probabilistic machine learning algorithm that uses Bayes' Theorem to predict a class attribute based on the probability of the class attribute occurring with the other given attributes. This algorithm has “naive” in the title due to the assumption that all attributes are independent of one another. This is something we need to be cautious about because there is a high likelihood that basketball statistics are not independent of each other.
This algorithm gained accuracy when predicting both wins and losses as the class attribute. Our overall accuracy rose to 92.3077% with 64 true positives and 32 true negatives, along with 2 false positives and 6 false negatives.

We next explored the ensemble learning algorithm, **Random Forests**. Random Forest is an algorithm that combines the results of multiple decision trees to improve prediction accuracy. Each decision tree is trained on a random subset of the data, and a random subset of features is considered at each split in each tree. In Weka, the predictions from all the trees are analyzed, and the majority class is chosen as the final prediction for each instance. Using multiple trees helps reduce the risk of overfitting, which is very important with a dataset like ours.

When we ran a random forest algorithm on our dataset, our class attribute was predicted with 91.3462% accuracy. There were 65 true positives and 30 true negatives predicted. Along with this came 4 false positives as well as 5 false negatives.

Attribute Selection and Reduction:

When we reached the point in our study focused on attribute selection, we noticed something about our predictions that we weren't entirely satisfied with. We used Weka's "Select Attributes" tab and chose Information Gain as our attribute evaluator. This revealed that the three attributes with the highest information gain were the Furman score, the opponent's field goal percentage, and Furman's field goal percentage. While this outcome wasn't surprising to us, we wanted to explore a different perspective of the dataset. Specifically, we aimed to focus only on attributes that were (more or less) controllable by Furman. In other words, we wanted to isolate the variables a basketball player could directly influence and then determine which of those had the highest information gain. In other words, we wanted to follow our goal of finding tangible information for Coach Richey and his staff. If we walked into their office and told them the most important parts of the game were Furman's score, the opponent's field goal percentage, and Furman's field goal percentage, we would get laughed out of the room. We wanted to explore more hidden knowledge.

We removed the attributes that had a direct influence on the score first. These attributes were Furman_Score, Opponent_Score, FG_Furman, FG_Percent_Furman, 3P_Furman, 3P_Percent_Furman, 2P_Furman, 2P_Percent_Furman, eFG_Percent_Furman, FT_Furman, FT_Percent_Furman, FG_Opponent, FG_Percent_Opponent, 3P_Opponent, 3P_Percent_Opponent, 2P_Opponent, 2P_Percent_Opponent, eFG_Percent_Opponent, FT_Opponent, and FT_Percent_Opponent. We saved this dataset and had our shadow dataset.

After this reduction in attributes, we found that the three attributes with the highest information gain were assists by Furman, defensive rebounds by Furman, and assists by the opponent. We wanted a reset of some of our baselines and found that OneR had a 70.1932% accuracy with the rule Weka found as following

AST_Furman:
< 10.5 -> Lose
< 12.5 -> Win
< 13.5 -> Lose
>= 13.5 -> Win
(282/385 instances correct)

In the next section, we will refer to the attributes selected by CfsSubsetEval. **CfsSubsetEval** is a feature selection evaluator in Weka that determines which of our attributes are the most predictive as a group. The subsets are ranked based on how well they are correlated with our class attribute and how little the attributes correlate with each other. CfsSubsetEval found that Location, FTA_Furman, DRB_Furman, TRB_Furman, AST_Furman,

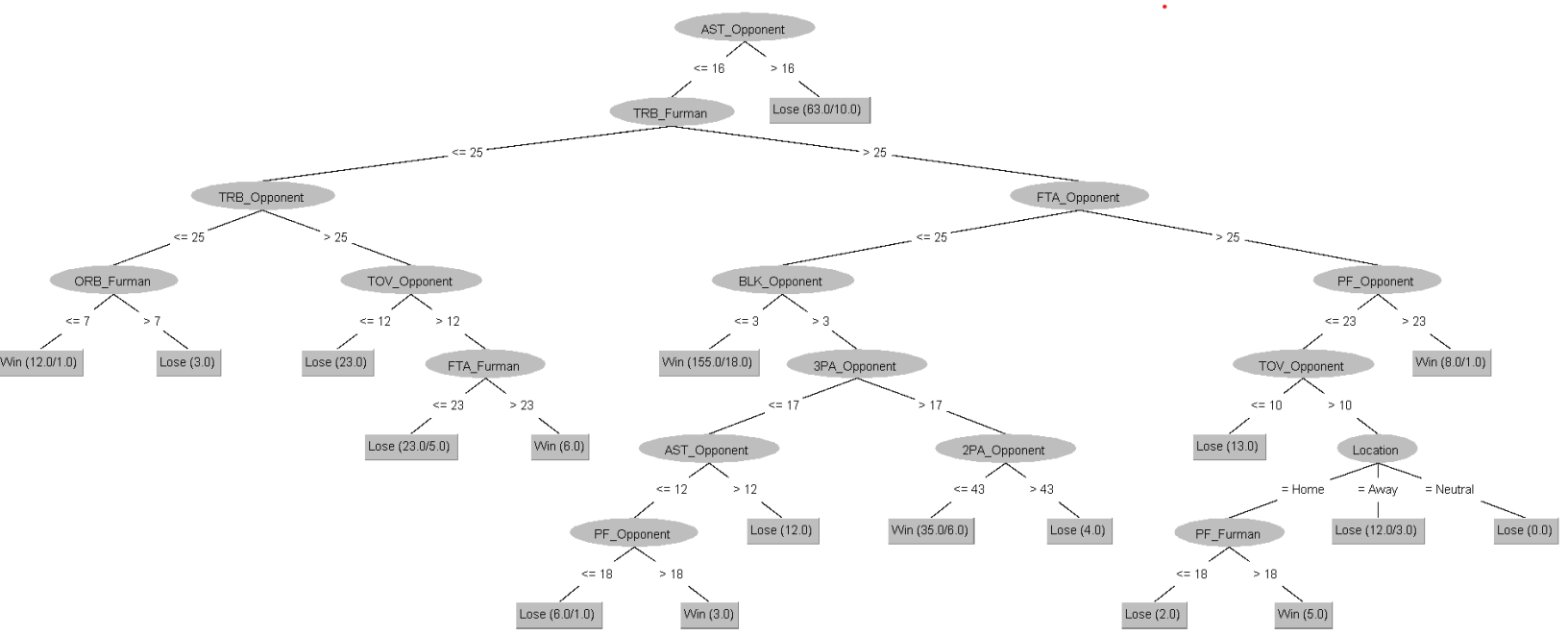
STL_Furman, BLK_Furman, FTA_Opponent, DRB_Opponent, and TRB_Opponent were the most predictive subset while having the lowest correlations with each other.

```
Selected attributes: 2,8,10,11,12,13,14,20,22,23,24,26,27 : 13
                     Location
                     FTA_Furman
                     DRB_Furman
                     TRB_Furman
                     AST_Furman
                     STL_Furman
                     BLK_Furman
                     FTA_Opponent
                     DRB_Opponent
                     TRB_Opponent
                     AST_Opponent
                     BLK_Opponent
                     TOV_Opponent
```

Our next step was to see if we could improve our Naive Bayes Model through attribute selection, as it had our highest accuracy in initial testing with a rate of 92.3077%. We first attempted to select attributes with the CfsSubsetEval. We removed all attributes except the ones selected as the most predictive subset. This did not give us our most accurate model, as our accuracy dropped to 75.9615%. Our confusion matrix values were 56 true positives, 23 true negatives, 11 false positives, and 14 false negatives. This shows these attributes were equally poor at predicting wins and losses.

We then wanted to see if we could improve our Naive Bayes model through manual attribute selection. We returned to our shadow data set and manually checked if removing any attributes could increase our accuracy. With our shadow dataset, our initial accuracy on the test data was 82.6923%, which was significantly lower than with the dataset containing score information. We first removed the attribute TRB_Furman, which increased our accuracy to 85.5769%. With the removal of other attributes, we were not able to improve our accuracy so this was our final version of our Naive Bayes Model. We then evaluated the **J48** algorithm. J48 selects the attribute with the highest information gain and begins building a decision tree starting from that node. From there, it splits the data in a way that leads toward predicting the class attribute. This process continues until the model can predict most of the instances in our dataset. We say “most,” not “all,” because we intentionally avoid overfitting the data.
We waited to run this algorithm until after the first round of attribute reduction to ensure the information we gathered was useful. When attributes related to the score were still included, the algorithm technically yielded the highest information gain; however, from a logical standpoint, we weren't learning much by predicting the class based on whether Furman’s score was higher than our opponent’s. When we ran the J48 algorithm, we found that using cross-validation within our training data resulted in a 78.961% accuracy in predicting the class attribute. However,

when we tested the model on our test data, the accuracy dropped slightly to 76.9231%. The first two nodes in our decision tree were assists by Furman, followed by assists by the opponent. We then attempted to refine this algorithm through CfsSubsetEval. With this collection of attributes, our accuracy dropped to 70.1923%. It was back to manual attribute selection for us. We found the first attribute to remove was AST_Furman, which brought our accuracy up to 78.8462%. We then removed the attribute STL_Furma, which increased our accuracy to 81.7308%. We were unable to increase the accuracy of our model any further through attribute selection.



With the accuracy of J48, we then wanted to include the ensemble learning algorithm **Random Forest** again. This was the algorithm we wanted to refine our attributes around. Our initial results with our shadow data set, before any manual attribute selection, was 83.6538% accuracy. We had 65 true positives and 22 true negatives, along with 10 false positives and 5 false negatives. We once again looked at our CfsSubsetEval and once again were let down. Our accuracy had dropped to 78.8462% and at this point were pretty fed up with Cfs. Manual attribute selection appeared to be the way to go. For the third and final time, we went through our attributes. We first found that removing AST_Furman increased our accuracy to 89.4321% when predicting wins and losses. After this, we removed OT_happen, which increased our accuracy to 92.3077%.

Finally, we found that removing DRB_Furman finished our attribute selection and increased our accuracy to 94.2308%. With this accuracy percentage came 69 true positives and 29 true negatives, with this came 5 false positives and 1 false negative.

Results/Discussion:

Model	Test Data Accuracy	Attributes Used
ZeroR	**67.31%**	**Full Dataset**
OneR	**73.08%**	**Full Dataset**
Ibk	**80.77%**	**Full Dataset**
Naïve Bayes	**92.31%**	**Full Dataset**
Random Forest	**91.35%**	**Full Dataset**
OneR	**70.19%**	**Shadow Dataset**
Naïve Bayes	**85.58%**	**Shadow Dataset without TRB_Furman**
J48	**81.73%**	**Shadow Dataset without AST_Furman and STL_Furman**
Random Forest	**94.23%**	**Shadow Dataset without AST_Furman and OT_happen and DRB_Furman**

Our Naive Bayes model was initially exciting as it had the highest accuracy percentage; however, once we reduced our attributes to the shadow dataset, which did not include scoring, it let us down. This drop in performance highlighted its reliance on the scoring attribute and raised concerns about its generalizability to new or incomplete data. Another challenge with this model was that we were not confident that our attributes were not correlated, so the independence assumption may not have been met.

Naïve Bayes		
	Win	Lose
Win	65	5
Lose	4	30

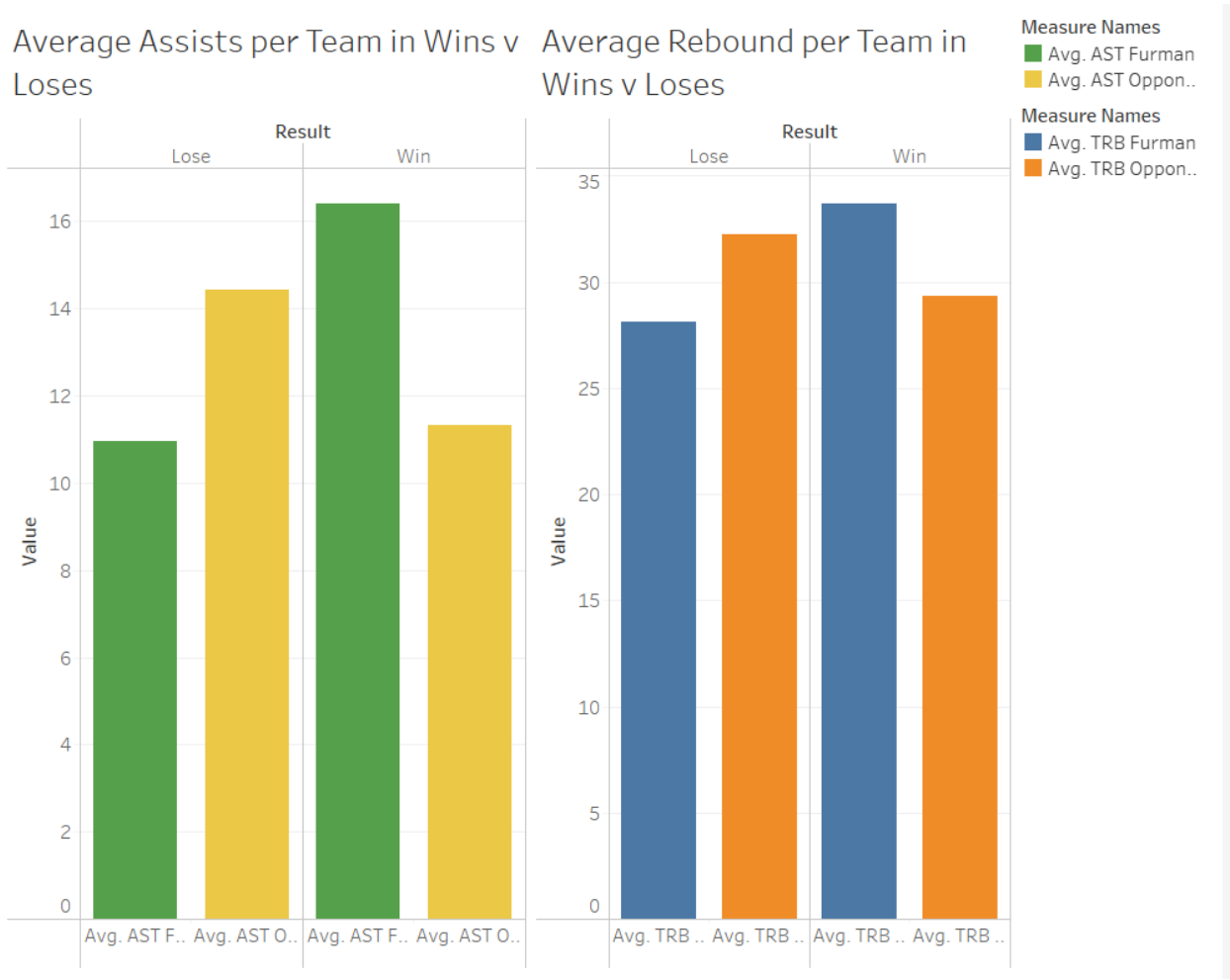
In terms of our best model, random forest was able to provide us with the highest accuracy at 94.23% on our testing data. This is not surprising to us, as it is a more complex ensemble learning technique. It is more complex and takes more time, but also is able to check more combinations of attributes than any of our other models. This model is highly accurate, and its interpretability is unique as it gives us a collection of attributes to focus on. Because of the nature of the algorithm, it does not tell us which of these collections is the most important.

Random Forest		
	Win	Lose
Win	69	1
Lose	5	29

In terms of interpretability, we found the J48 model to be the best. Our accuracy was significantly lower at 81.73%; however, because our nodes are split based on information gain, this could give coaches a clear picture of what needs to be focused on and when. The tree structure makes it easy to figure out decision paths and understand how specific attributes influence outcomes. Ultimately, while random forest offers higher accuracy, the insights from the J48 model make it a valuable tool for actionable decision-making. Our attributes with the highest information gain were opponents assists, total rebounds for Furman, and total rebounds for the opponents.

J48		
	Win	Lose
Win	62	8
Lose	11	23

As this became our most accurate and interpretable model, we wanted to visualize our three most predictive attributes of AST_Opponent, TRB_Furman and TRB_Opponent to compare the two rebounds together and assists of the opponent to Furman to see the difference.



As we can see, the results especially for assists were staggering on both sides. Furman averages a fairly wide margin of assists when winning, and averages a fairly wide margin of assists when losing. This along with our total rebounds makes a lot of sense to why J48 as predicted them as our best attributes.

Conclusion:

This project allowed us to explore manual data collection, data preparation, model selection, and model refinement. Our final results showed that the ensemble learning model Random Forest, was the most accurate at 94.23% on our test data, after manual attribute selection. Although it was not as interpretable as J48, its ability to explore interactions between attributes helped it outperform our other algorithms. Our J48 decision tree model was the most interpretable and had 81.73% testing data accuracy. The structure of the tree allows for clear, insightful paths, which could be very useful in real-world coaching scenarios.

One of the most useful parts of the project was attribute selection and reduction. It made us think about what variables actually contributed to the game outcome and helped us discover that some seemingly small things, like removing our OT_happen attribute, may increase our model accuracy significantly. Our manual attribute selection was more effective than our use of CfsSubsetEval. This reinforced the idea that it was important to know how to do this manually and not rely solely on machines.

We did face some challenges; data collection was tedious and time-consuming, but the sports-reference.com data was reliable and complete, so we thought this was worth it. We also had to use background knowledge of overfitting to make sure our models would be generalizable. Our shadow dataset came in handy to fix part of this problem. Another major point learned was the trade-off between a more complex model and a more interpretable model. Where it felt excellent to have a model with almost 95% accuracy, sometimes OneR was able to give us just as valuable insights.

Future work could explore player-level analytics. By breaking down statistics by individual players and adding information such as height and position, we could investigate whether players of certain sizes or profiles should focus on specific aspects of their game. For example, should a starter have a different statistical focus or mindset compared to a sixth man? Additionally, there is potential for a comparative study with other universities. Are the attributes we found predictive for Furman basketball unique to the Paladins, or are they consistent across other programs? Since our dataset only covers Furman games, it would be naive to assume the same trends apply to all NCAA teams. Expanding the dataset to include all NCAA games over the same timeframe could allow for broader, macro-level predictions about which attributes are most predictive nationally. It would also be valuable to analyze trends at the conference level: do certain conferences place greater emphasis on the three-point shot, faster pace, or slower, more methodical styles of play? A more robust dataset could uncover deeper trends across different regions and styles of college basketball.

Overall, this project enhanced our understanding of data mining tools and how we can learn from it. We were challenged at many stages from data collection to attribute selection and overall are happy with the insights on Furman basketball we were able to find.

Data Source

Example of a single season (2024-2025)

https://www.sports-reference.com/cbb/schools/furman/men/2025.html

Example of a single game/instance (Furman vs North Texas)

https://www.sports-reference.com/cbb/boxscores/2025-03-19-20-north-texas.html