Tennis Analytics - Predictive Modeling of ATP Match Data

Grace Flitsch

Abstract

In this paper, I conduct exploratory data analysis and machine learning modeling on data from ATP tennis matches from the years 2003-2017. I aim to predict the win or loss of a match based on 10 predictors in the original dataset. Some of these predictors include age, rank, double faults, games served, and breakpoints saved. Through modeling, I find that a support vector machine with a cost of 15.6969 and rbf_sigma of .0146 performs the best, with an f-score of .809 and accuracy of .81. I also find that the most important predictors to win status are the amount of first sets won, amount of breakpoints saved, amount of serve points won, and amount of first serves that are in play.

Introduction

In recent years, the field of sports analytics has exploded, specifically in baseball. Statistics are available for every possible moment in a game, and statisticians are employed full time to find ways to maximize scoring for teams. Some sports lend themselves more to having robust statistics, specifically sports where games are composed of discrete events that are easily quantified. A sport that has not experienced this rapid analytics growth to the same extent is tennis. Tennis presents a difficult problem to statistics in that it is hard to quantify most of the play that is happening. It would be impossible to label every kind of shot there is in the game, and points can be won often because of another player's mistake, instead of a perfectly placed winning shot. This paper takes a dataset of commonly recorded tennis statistics and attempts to draw insightful conclusions about playing style and characteristics of winning players through statistical analysis and machine learning.

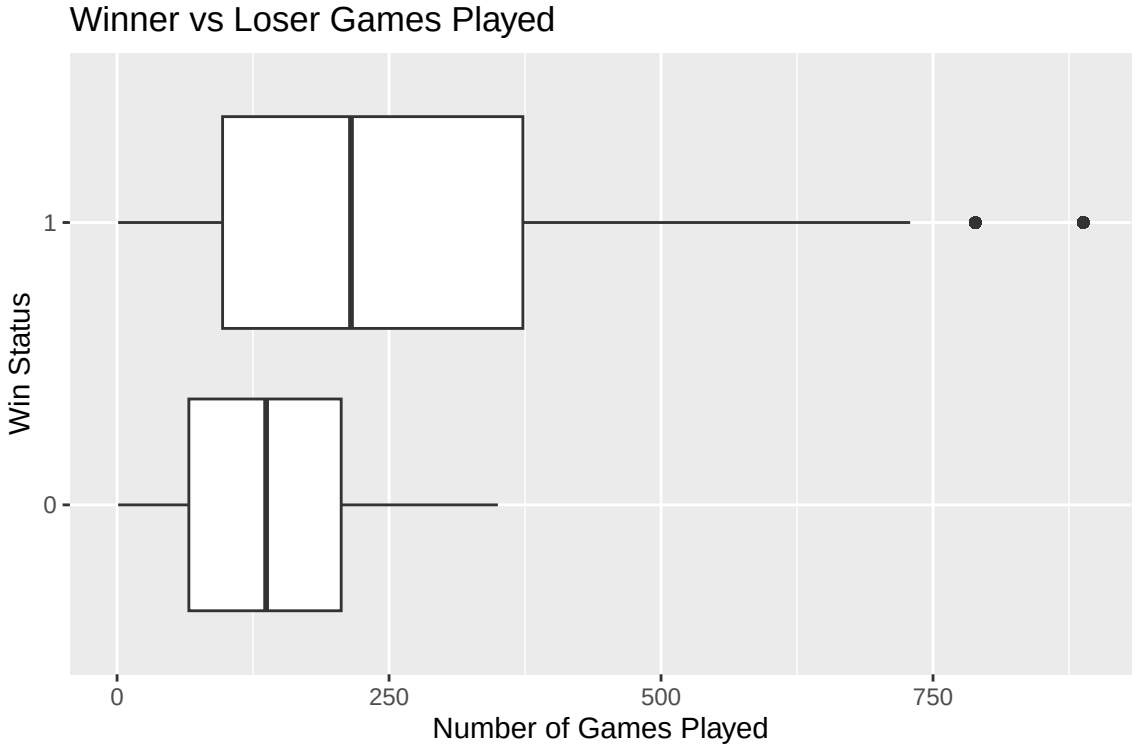
The dataset comes from Kaggle (https://www.kaggle.com/datasets/gmadevs/atp-matches-dataset) and contains data from men's ATP (Association of Tennis Professionals) matches from 2003 to 2017. The raw data contains rows of match by match data, with statistics given about both the winning and losing player. I manipulate the dataset in two different ways depending on the section of the paper it is used in. For the purposes of clustering, I create a new dataset that has every player in it once, with each column holding career averages for the given statistic of that column. The last column is a win percentage, which is the players' win percentage over all time. For modeling, I manipulate the dataset to have rows for every players' statistics from every match. This means that for each match in the original dataset, there are two rows in the modeling one, one for the winner and one for the loser. For the purposes of modeling, I select a random subet of 1000 player/match rows to work with. I aim to use this dataset to predict if a player will win or lose a match given a single matches' statistics. The variables in the dataset are the player's age (age), ranking (rank), number of aces per match (ace), number of double faults per match (df), number of service points won (svpt), number of first serves in play (onein), points won on first serve (onewon), points won on second serve (secwon), number of games serves per match (svgms), number of break points saved (bpsav - a breakpoint save is a point in which the player could lose a single game but wins the point to stay alive in that game), win percentage over all time (win_pct), and whether the matches described were won (won). In this paper, I refer to top-ranked players as "lower ranked" since their rankings are closer to 1, and vice-versa for players ranked higher.

In the first section of the paper, I conduct exploratory data analysis including clusterings of the players in an attempt to profile several different "types" of tennis player. In the second section, I create four models

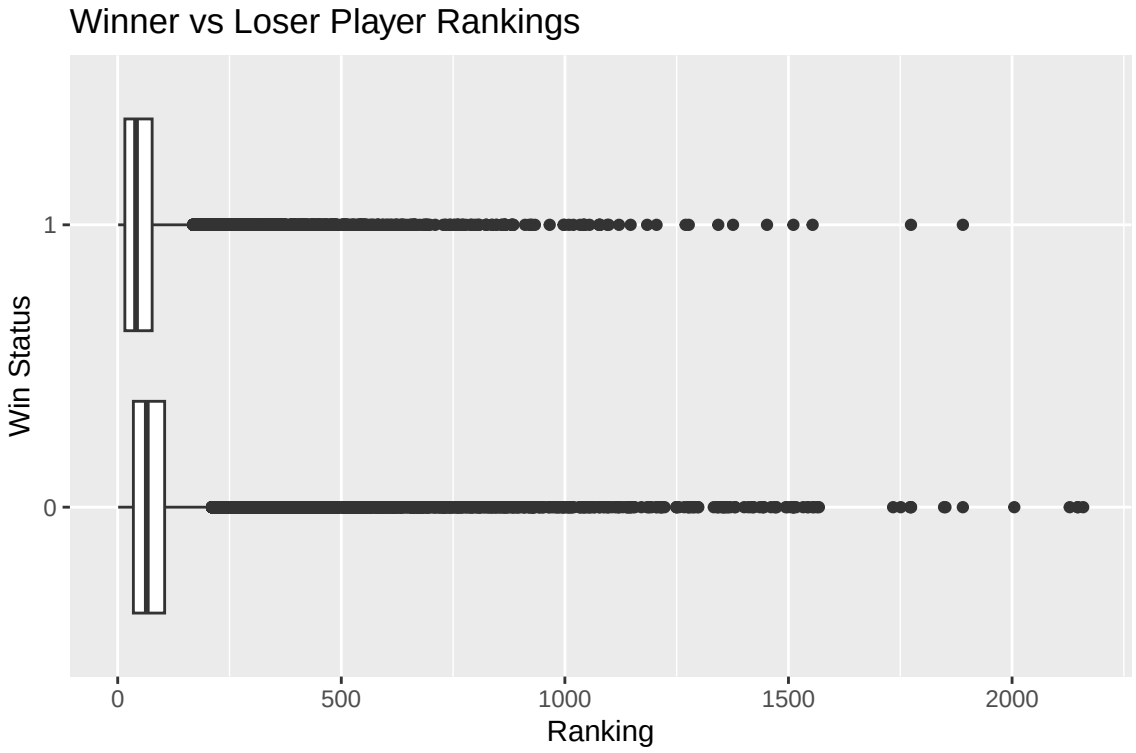
to predict the win/loss outcome of a match given a player's statistical profile from one match. The purpose of this paper is to use common sports machine learning techniques on sports data that is not as commonly analyzed to investigate whether the insights can be gained from available tennis data.

EDA

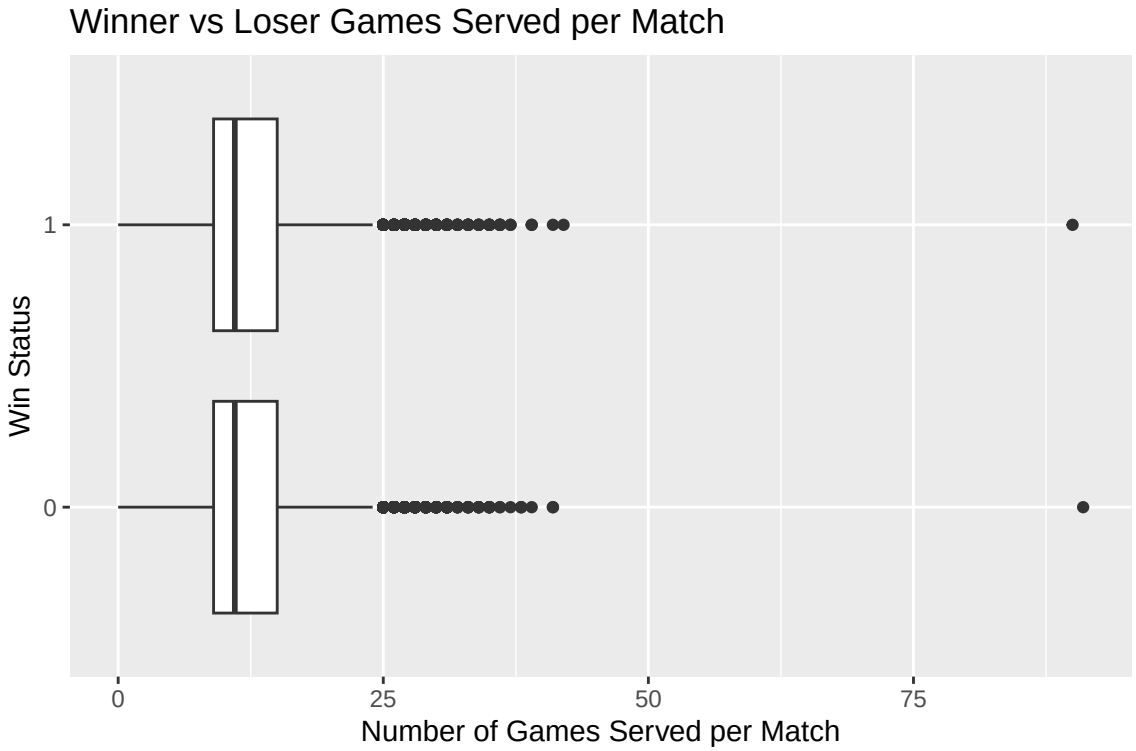
In my exploratory data analysis, I create visualizations of the relationships between variables that I find to be particularly important to the data, or surprising to me. Each plot informed in some way my modeling of the matches, and shows why I may have included or not included certain variables in my models.



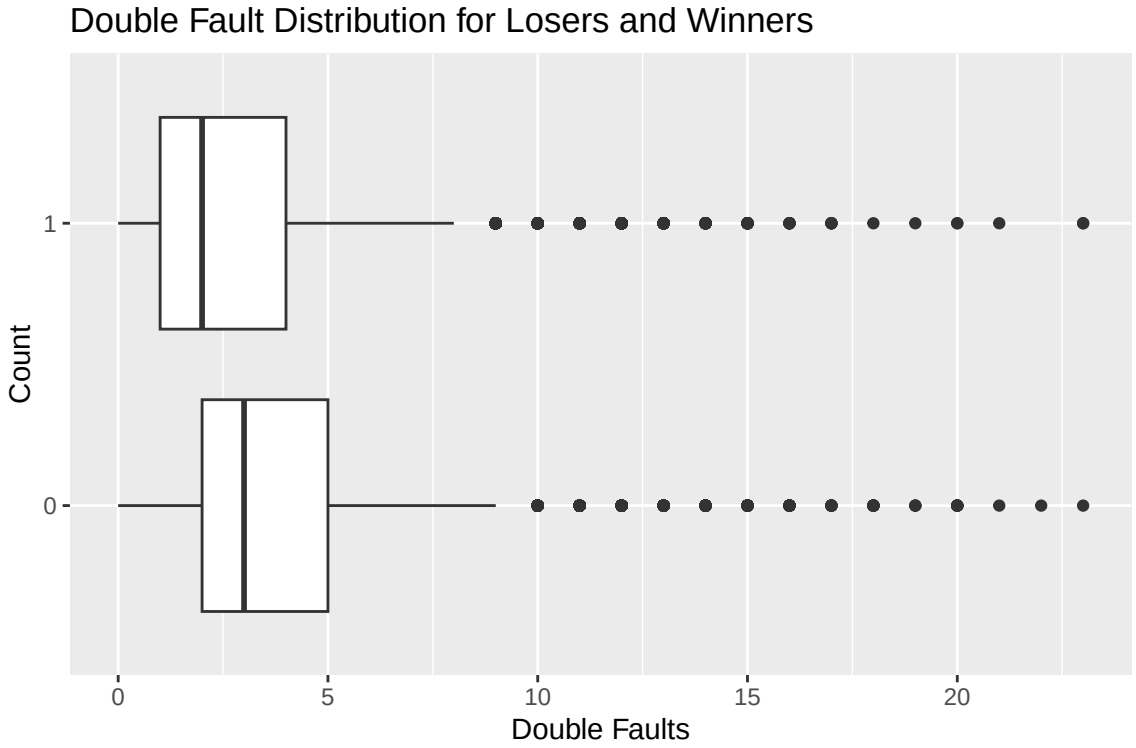
This plot shows that the more games a player has played, the more likely they are to win a match. This plot tells me that this variable is very helpful to the model, since the medians seems to be well split. The problem with this statistic is that it uses information that should not be available to the model while predicting. The number of games played is a total that each player has, which tells how many games that player will play over all time before they drop out of the dataset, or before 2017. Players who win more will naturally play more games, since you need to win games to continue advancing in tournaments. I need to not include this variable as a predictor since it is not something that would be available to a user of the model if they were to input real time data.



These boxplots show a comparison between ranks of players and whether the match was won or lost. As expected, the top boxplot, which is the one containing all of the winners, has a lower median ranking. A lower ranking means that the player has won more matches against more difficult opponents than higher ranked players. The interesting thing about this boxplot is the amount of high outliers in ranking for both winners and losers. I attribute this to the fact that most players in the tournaments in the dataset have low rankings, since they play professionally. However, some higher ranked players are able play via a wildcard draw, or by winning in an ATP qualifier tournament. It is harder to get into the tournaments as a highly ranked player, but it still happens, as can be seen in the plot. Since there are so many outliers on the right, I will take the log of this variable to improve the model. This ranking variable could possibly be a valuable predictor in my models, as there seems to be a slight difference between the distributions of rankings between winners and losers of matches.

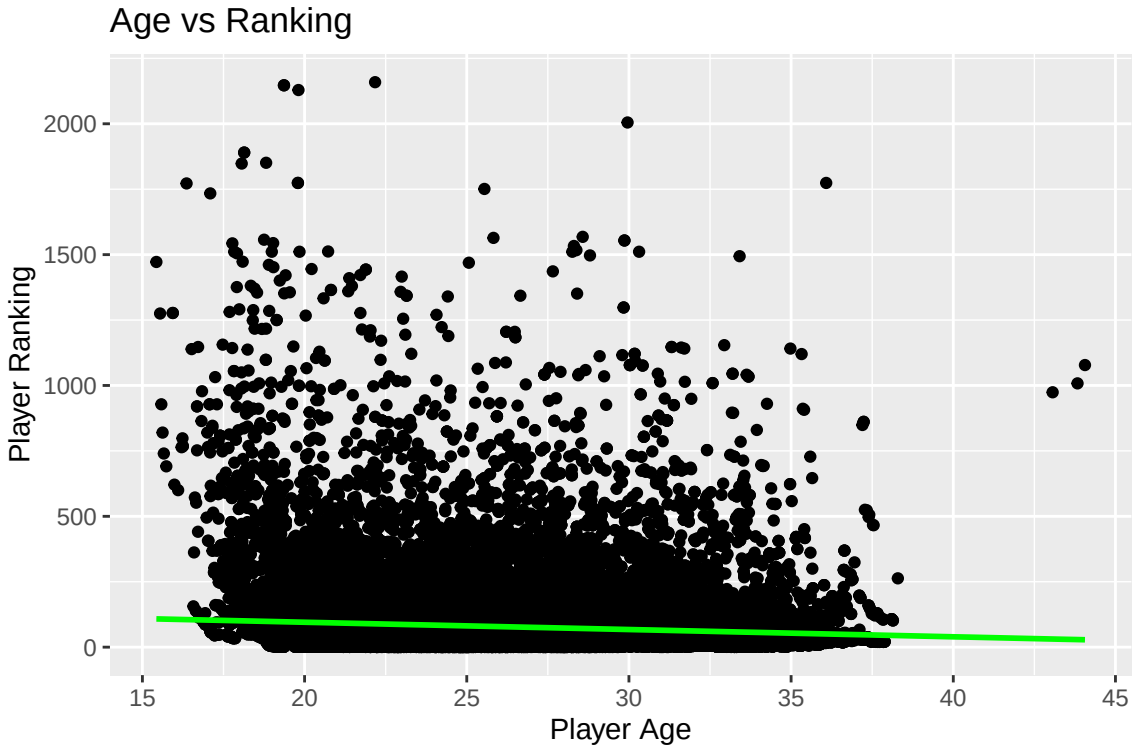


This plot suggests that there is not much of a difference in win status based on number of games served per match. This surprised me since in my own experience with tennis, it is my understanding that it is advantageous to serve more, since you can gain the upperhand before the other player hits the ball.



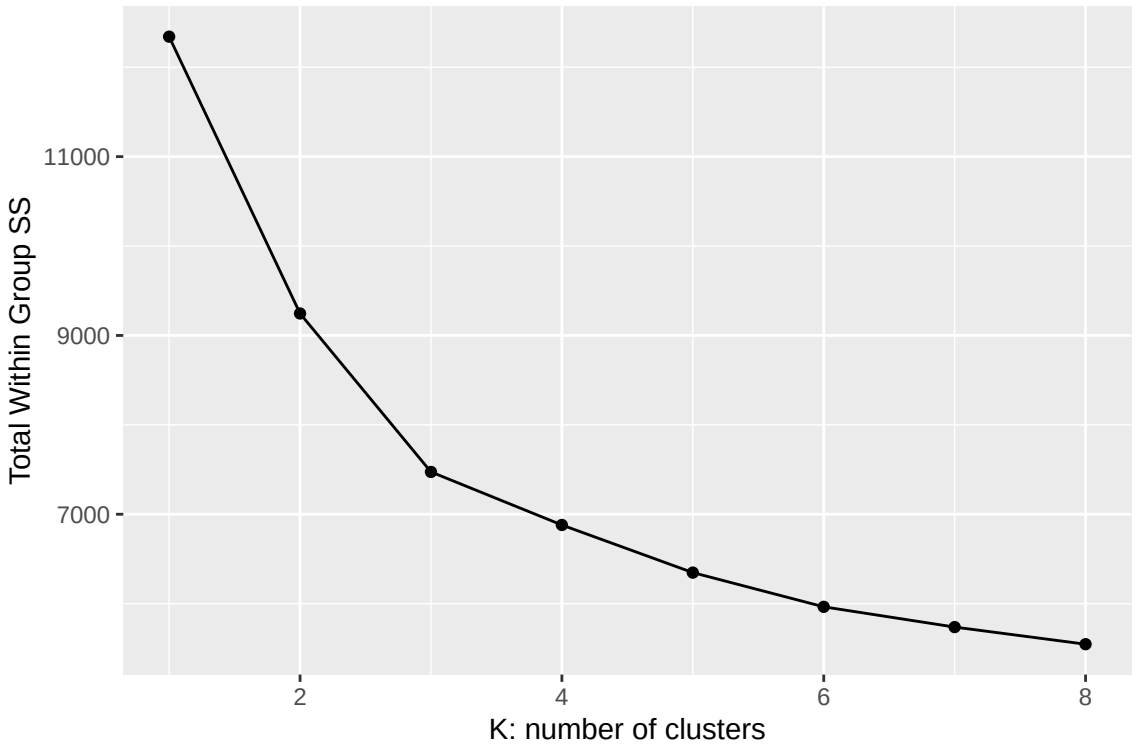
The double fault variable distribution does not vary greatly between winners and losers, but the losers seemed to be a bit more evenly distributed, with wider ranges of amounts of double faults falling on either side of

the median than the winners. Just from these plots, it looks like there are a few more losers double faulting more often than the winners, but modeling is necessary to see if this is an important predictor. There are low outliers, but it is not as heavily skewed as rank, so I am not going to take the log of this variable for sake of interpretability.



I was wondering of there was a relationship between rank and age, and it looks like there is not a strong one. Since most players retire once they can't play at a high level, it makes sense that their rank wouldn't fall for long before they retire. The regression line of these points is almost completely horizontal, showing that there is not much change in ranking as a player increases in age. There are some high outliers of young players with high ranks, showing that it is more likely for an exceptional player to be young than old.

Next, I conduct a clustering to categorize players by the style that their average statistics align with. First, I use the elbow method to select the optimal amount of clusters. I have removed the num_games statistic since it has unreachable information to the real time user of a clustering.



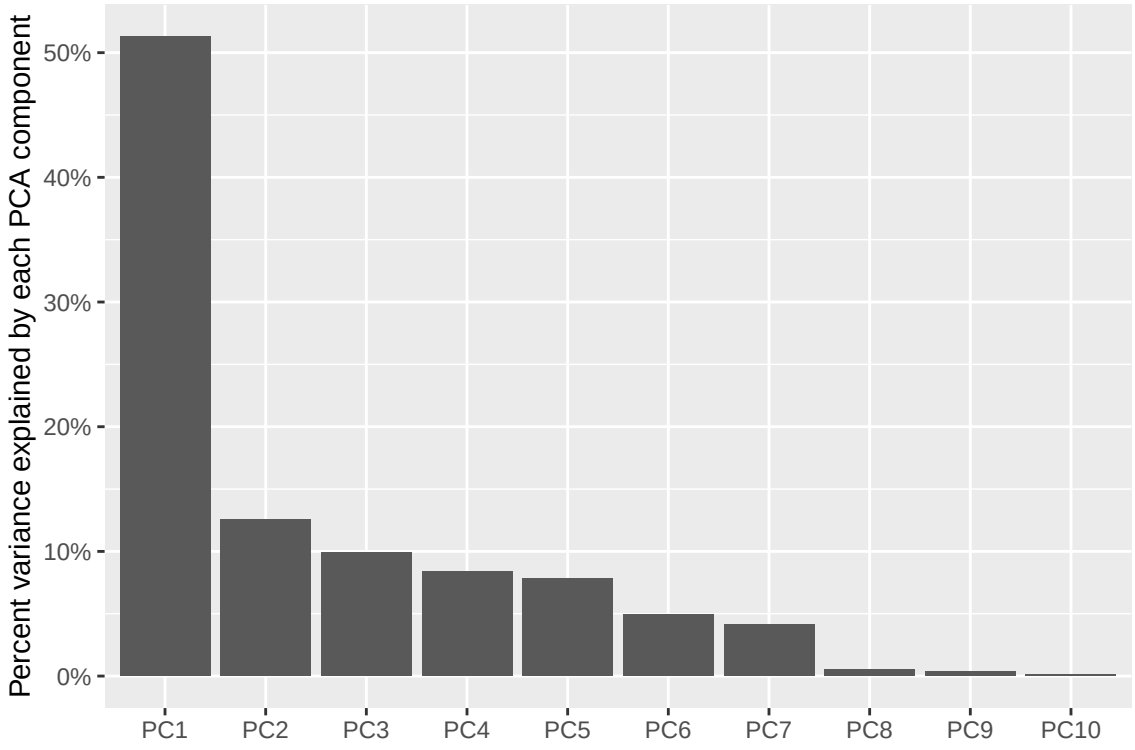
From this plot, it looks like the ideal number of clusters is 3, but there is not a clear elbow so it is an arbitrary choice. After running a clustering with four groups, the results are summarized below.

After splitting the players into three clusters, there are 107 in cluster 1, 729 in cluster 2, and 287 in cluster 3.Since each cluster has a relatively large sample size, I think that this is a reasonable way to split the players into groups. Looking at these three clusters, there are distinct differences if I consider win_pct. Cluster 1 and 2 both have win percentages between 20 and 35%, neither of which is a great record, but they both perform better than cluster 3. The rankings (which reflects the log of the original ranking) are relatively similar across the clusters, which indicates to me that splits between clusters cannot be made based on rankings. Cluster 3 performs much worse than all other groups, with a win percentage of only 6.4%. Players in this cluster have the least double faults, which makes me think that more double faulting may actually be an indicator of a strong server. This cluster performs poorly in all other areas, and also serves the least games on average. Interestingly, cluster 1 has the best overall statistics, winning the most first and second sets, serving the most games, saving the most breakpoints, getting the most aces, and serving the most first serves in, but their win percentage is not as good as the second cluster. This confuses me a bit, but a possible explanation is that there are other variables not present that may influence win probability. From this clustering, I can move into modeling with a better idea of what variables maybe helpful to my model - most likely, rank and double faults will not contribute like I thought they would, while statistics that vary a lot from cluster to cluster such as svpt, onewon, onein, and ace might contribute more to explaining the outcome variable.

Methods

I now create models to predict the categorical variable "won," which indicates either the win (1) or loss (0) of a match. I will be using the dataset that I made that has individual player data from every match in the original ATP dataset. I create recipes to build a KNN model, tuning the nearest neighbors and distance power, a boosted tree with the tree_depth tuned, an rbf support vector machine with the cost and rbf sigma parameters tuned, and a KNN model that also includes principle component analysis as a step in the recipe. In the preprocessing of the nearest neighbors model, I normalize the numeric predictors to

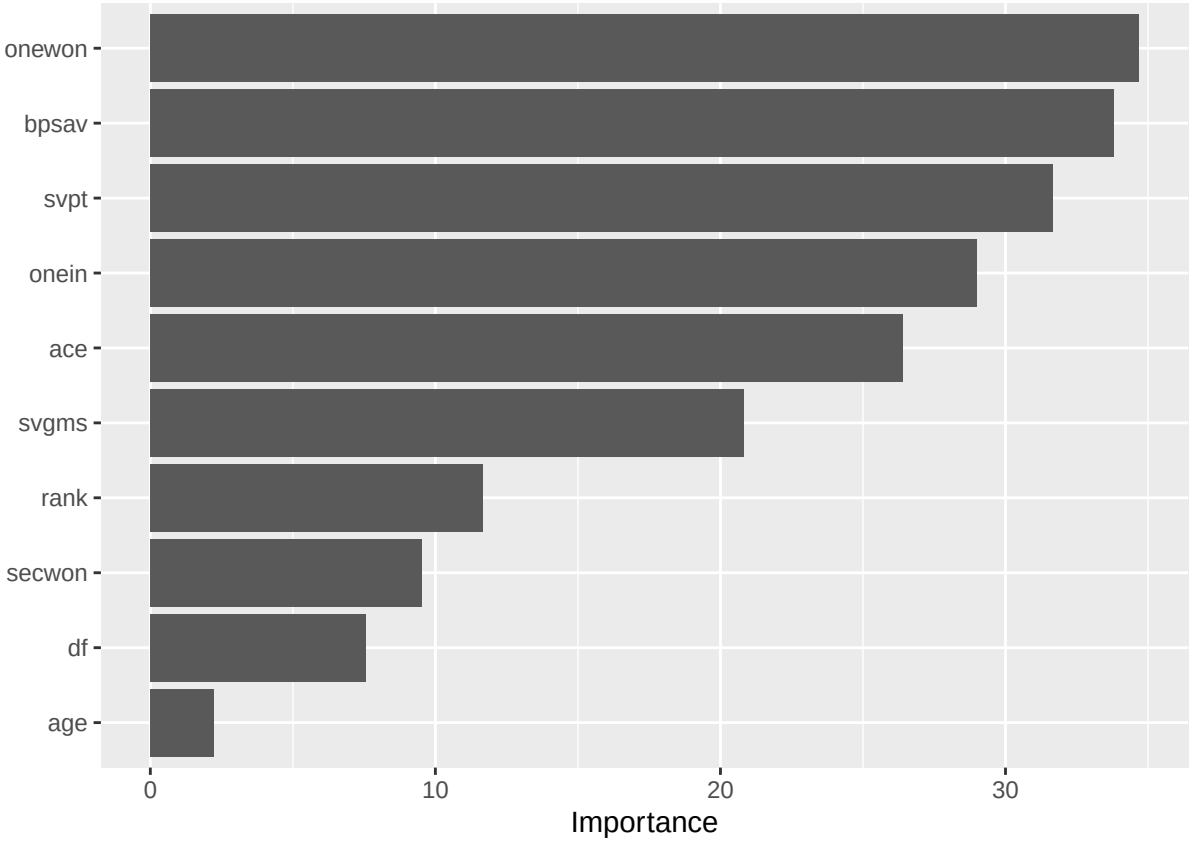
keep features on a similar scale. I also do this step for the support vector machine. For the boosted tree, I do not do any preprocessing, as it is not necessary for tree based methods. Finally, for the second KNN I include a step_pca to test if this results in better results for the model. I use 7 principal components, with the reasoning explained below under the scree plot. I tune all of the models in a workflow set with 10 fold cross validation and a grid size of 5. I evaluate the models according to their F-score because it accounts for precision and recall, which is important in a 0/1 classification. Specificity and sensitivity are not as important in this project since there are about 50% in each class, so predicting too many false negatives or positives will already be reflected in a low F-score. The resulting F-scores of the various models are described in the next section. The two following plots display information about the PCA conducted for my KNN PCA model.



This scree plot displays the percent variance accounted for by each principal component. PC1 accounts for about 51% of the variance, while the following 6 combine to account for about 48.5% of the variance. I select 7 components because my goal is to increase accuracy, or the f measure statistic, so I am willing to have more dimensionality in my model in order to have it make better predictions.

Results

After tuning and running all of the models, the best model is a support vector machine with cost tuned to 15.6969 and rbf_sigma tuned to .0146. This model performs with an f-score of .809 and accuracy of .81 on the cross validation set. I fit the model to the testing set and achieve a f-score of .786 and accuracy of .79. This is great because it means that my model is not overfit to the training data. The KNN model ends up tuning to 13 neighbors and a dist_power of 1.71 to achieve an f-score of .681 and an accuracy of .685. The KNN model with PCA performs best with 10 neighbors and a dist_power of 1.6. This model has an f-score of .6492 and an accuracy of .657. Finally, the boosted tree tunes to a tree depth of 8, which results in an f-score of .7511 and an accuracy of .756. Below is a plot showing the variable importance scores of all of the predictors in my model.



Conclusion

After testing four models and clustering the data, I have found some helpful insights about ATP matches and win prediction in tennis. Firstly, I created a support vector machine with relatively high power to predict the winner of a match given a single player's match statistics from one match. This model has an f-score of .809 and an accuracy of .81, and was tuned to a cost of 15.6969 and rbf_sigma of .0146. From the variable importance plot, I see that the most significant predictor was "onewon" which was the variable that gave the number of points won on the first serve for the given player. Other important variables were "bpsav," which is the number of break points saved during a given match, "svpt" which is the number of serve points won in the match, "onein" which is the number of first serves in, and "ace" which is the number of aces in a match. The least important predictors are age and double faults. From this model, players can learn that getting their first serve into play is very important, as winning a point off of the first serve is the most important predictor to winning. Since double faults do not affect match outcome much, players should serve aggressively and not worry too much about double faulting on occasion.