Serving up Predictions: Using Machine Learning to Anticipate Tennis Match Outcomes

Rishi Mukundan

March 21st 2025

Introduction

My first memory of tennis was the 2008 Wimbledon Final, between Roger Federer and Rafael Nadal. Widely regarded as the greatest match of all time, I watched two gladiators of the sport trade jaw-dropping shots over 5 grueling sets. This was the moment when I aimed to make tennis a priority in my life. Ever since that day, I have continued to closely follow the sport, as well as play very frequently. As I have navigated my professional and academic career, I have tried to combine my skills in machine learning with this passion. This final project presents the perfect opportunity to analyze performance trends within the sport I value so much.

Data-driven analytics has continually become a bigger part of decision-making in all sports. Franchises are always searching for a deeper understanding of “the game within the game” to gain a competitive edge. With this in mind, I ask the question:

*Can we leverage machine learning techniques to predict tennis match outcomes based on performance metrics?*

The dataset chosen for this question was ‘atp\_matches\_2024.csv’, a publicly available dataset created by Jeff Sackmann. This meticulously gathered data contains information from every men’s tennis match played in 2024, including player demographics, tournament details, match score, and player statistics. Examples of statistics include 1st/2nd serve points won, double faults, match length, break points saved/faced, player height, age, handedness, rankings, etc. Each row contains all the information from a single match. The dataset contained 3076 observations and 49 predictors.

Methods

The first task was to properly transform the data to support a binary classification model. Rather than having both the winner and loser’s statistics contained within each row, it was more convenient to record each match occurrence twice – once for the winner and once for the loser. I also added a binary factor variable “outcome” which indicated whether the player won or lost.

Next, I created visualizations of key performance metrics from each match and grouped them by match outcome. This was done to identify important variables and clear trends that could predict success. For example, a density plot of 1st serve win percentage showed that match losers averaged around 65%, whereas winners averaged closer to 80%, which is a significant difference (Fig 1). I also created a boxplot showcasing the difference in break points faced, again grouped by match outcome. This showed that winners do a much better job in limiting break points faced, lending more strength to the idea that consistent serving is the biggest key to victory (Fig 2). Finally, mapping 1st serve win A graph with red and blue dots

AI-generated content may be incorrect.A graph of a number of points

AI-generated content may be incorrect.A graph of a graph

AI-generated content may be incorrect.percentage against ace count showed a clear boundary between winners and losers (Fig 3).

Figure 1 Figure 2 Figure 3

Now that I had a general sense of which variables to emphasize, I wanted to assign importance to each feature. So, I utilized a random forest model on the predictors. The random forest was chosen because it is easily scalable, adept at handling potentially non-linear trends, and is more resistant to overfitting. With these results, I was able to choose an upper limit of variables to train the model on.

The question then became “How do I strike a good balance between model simplicity and accuracy, while keeping my computation costs low?” To handle this, I iterated through my top features list, gradually increasing the number of features included in the training step, until I hit the upper limit. Each iteration consists of 3-fold cross validation. The data was split into 70% training and 30% testing. Random forest was initially chosen for this step, but computation time simply became infeasible. So XGBoost was chosen instead, as it possesses similar advantages while being more efficient. The number of features which yielded the highest accuracy value was then used to train the final model.

Finally, I tuned the model’s hyperparameters while holding the training data to our optimal number of variables. Again, the issue of computation time emerged here. I initially tried to tune through 3 values of each of xgboost’s 7 hyperparameters. This again took a very long time. To increase efficiency, I focused my tuning to “nrounds” (number of trees), “eta” (learning rate)”, and “max\_depth” (tree depth). Since these are the high priority hyperparameters for this specific model, I felt it was best to dedicate my time there. Iterating through 2 values for each of these metrics resulted in the task running quickly. With the optimal hyperparameters chosen, the final version of the model was trained and tested.

Results

The feature importance analysis results are summarized in the variable importance plot below:

A graph with numbers and lines

AI-generated content may be incorrect.

The top 3 most important variables are: 1st Serve Win Percentage, 2nd Serve Win Percentage, and Break Points Faced. This lines up well with our earlier visualizations, which showed a clear gap between winners and losers in these metrics. The large gap in MeanDecreaseGini values between the top 3 and the rest of the variables also highlights how essential serving efficiency is to success.

After the top 5 variables, a large majority of the predictors hover around a MeanDecreaseGini value of 50-80. Including all of these predictors in our model may introduce unnecessary complications without a tangible increase in accuracy. With this in mind, I elected for an upper limit of the top 10 features.

To decide how many of the top 10 features to include in my final model, I iterated through 3-fold cross validation, gradually increasing the number of predictors included in the training data. I started with the top 5 and increased by 1 until I reached the top 10. As mentioned earlier, the randomForest method could not handle this task, so XGBoost was used instead.

| **NumFeatures** | | **Accuracy** |
| --- | --- | --- |
|  |  |  |
| 1 | 5 | 0.7304845 |
| 2 | 6 | 0.7297236 |
| 3 | 7 | 0.7343241 |
| 4 | 8 | 0.7379078 |
| 5 | 9 | 0.7396946 |
| 6 | 10 | 0.7309946 |

A graph with a line

AI-generated content may be incorrect.Accuracy vs Number of Features Graph of Accuracy against Number of Features

The graph indicates that the top 9 predictors yield the highest accuracy. However, the accuracy only slightly deviates as we increase to 5 to 10, suggesting that our model strength is high regardless of the number of predictors we choose within this range. The top 9 predictors were chosen as the final amount to pass on to our model.

The last step was to tune our model’s hyperparameters. I iterated through “nrounds” values of 100 and 200, learning rates of 0.01 and 0.1, and “max\_depth” values of 3 and 6. The “gamma”, “colsample\_byTree”, “min\_child\_weight”, and “subsample” hyperparameters were fixed at 0, 0.8, 1, and 0.8, which are the default values. As mentioned earlier, this approach aimed to increase computational efficiency.

|  |  |
| --- | --- |
| Final Model Results | Value |
| Accuracy | 0.777 |
| Sensitivity | 0.774 |
| Specificity | 0.780 |

Discussion

The final model’s accuracy rate of 77.7% represents a 4-5% increase compared to pre-hyperparameter tuning, which validates the approach I took. When attempting to predict sports outcomes, there will always be inherent variation in outcome. With tennis, this effect is magnified, due to the individual nature of the sport. Given this, an accuracy value of 77.7%, with similar sensitivity/specificity values is a strong performance. It significantly outperforms chance guessing, while not falling into the risk of overfitting. However, this experimental setup is not without its limitations. 58% of the matches in the dataset were played on hard court, which could introduce court-based bias. Furthermore, this model assumes that as you sequentially add features to train the model, there is no negative interaction between them. To account for this, I could test different combinations of feature variables to see if that improves performance. Finally, the XGBoost method for feature selection took a fairly long time, so simpler models such as logistic regression may be considered for simplicity. In conclusion, the results from this experiment show a strong indication that performance metrics can dependably predict tennis match outcomes.