Sean Kook
Professor Trok
Data 119
3-7-2025

Predicting ATP Tennis Match Outcomes Using Logistic Regression and k-Nearest Neighbors

Introduction

Tennis match outcomes depend on a variety of factors, including player performance metrics, historical statistics, and game conditions. In this project, I developed a predictive model for tennis match outcomes using ATP match statistics from 2017. The goal was to explore and compare different models, focusing on logistic regression and k-nearest neighbors (kNN), to identify key predictors, tune hyperparameters, and optimize classification accuracy.

Data Exploration

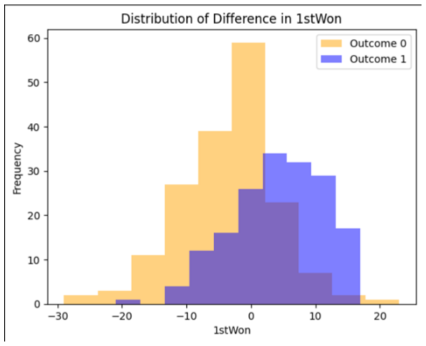
We began by cleaning and exploring the ATP match data, focusing on match statistics for winners and losers. The dataset contained numerical features such as player rank, first serve percentage, break points faced, and service game wins. While categorical variables like player names and tournament details were present, they were not the primary focus of this project.

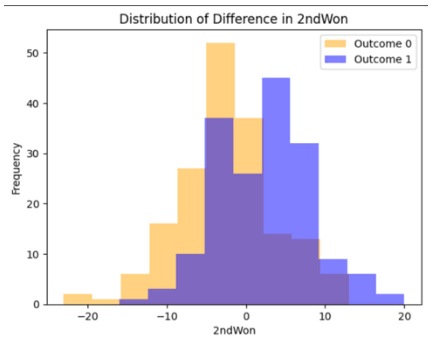
To facilitate logistic regression modeling, I created a binary response variable: 1 if player_w won and 0 if player_w lost. To introduce randomness and avoid biases in the dataset structure, I generated a list of random floats between 0 and 1 of the same length as the dataframe and converted it into a Boolean list by setting values greater than 0.5 to True. For rows where this Boolean value was True, I swapped the statistics of the winner and loser and set the response column to 0.

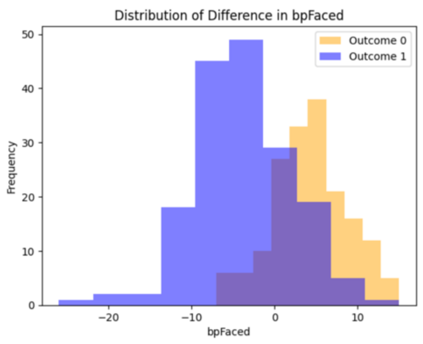
Lastly, I transformed my dataset by computing the difference between corresponding statistics between winner and losers, rather than keeping them separate in their own columns to streamline the modeling process, and stored them in a new dataset (diff_df). I then added back the date and response columns to the diff_df dataframe as they will be useful later for modeling purposes.

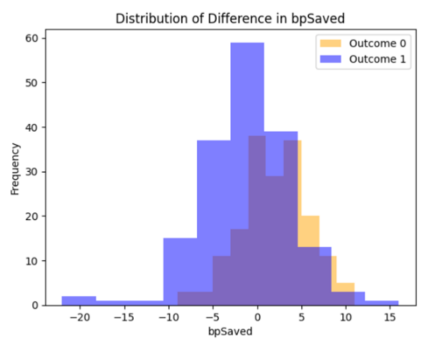
Feature Selection

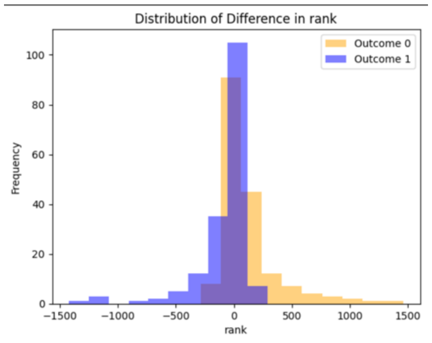
Using this new dataframe, I plotted histograms for every predictor variable that I deemed to have potential in contributing to a good model. These variables were 'age', 'rank', 'ace', 'df', 'svpt', '1stIn', '1stWon', '2ndWon', 'SvGms', 'bpSaved', and 'bpFaced'. I looked for features with clear separation between winners and losers (color coded), meaning little overlap between two colors, and favored those that deviated further from zero, as this would suggest a stronger correlation. Based on this criteria, I initially found that the most promising predictors were 1stWon (first serve points won), bpFaced (breakpoints faced), svpt (serve points), 2ndWon (second serve points won), Ace, and svGms (service games).

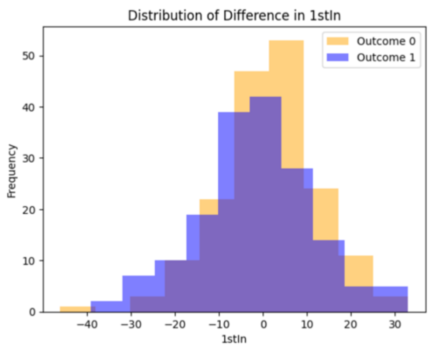


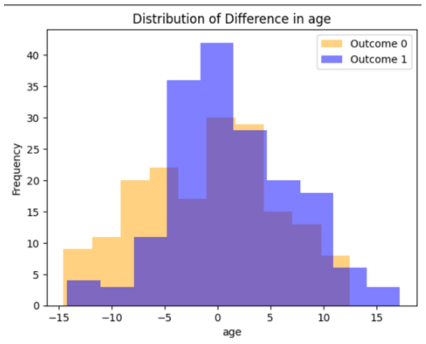


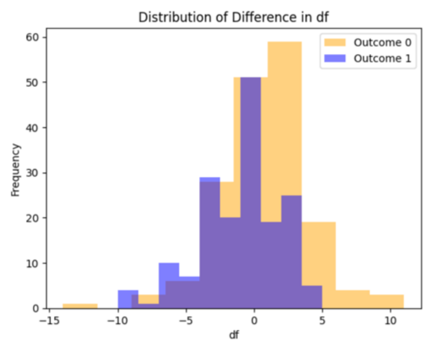


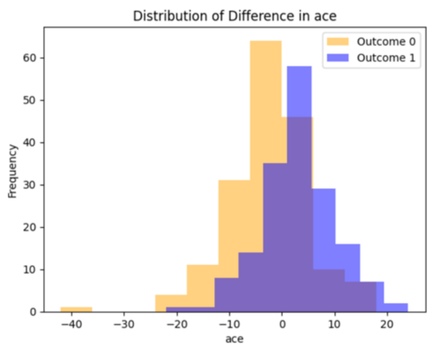


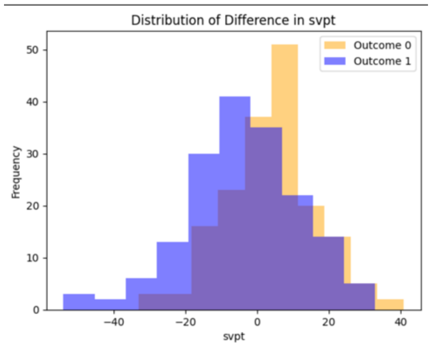












However, upon analyzing the correlation matrix, I noticed that bpFaced and svpt were collinear (r=0.7169). Since bpFaced had higher correlation with response, I decided to drop svpt. Of the variables not included in this initial model, 'rank' (or rather difference in ranks) had the strongest correlation with response(r=-0.38). Moreover, analyzing the coefficients for my initial logistic model showed that Ace had a pretty small coefficient, and didn't contribute much to the accuracy of the model. Taking all this into account, ultimately the set of predictors I compared were the following:

1. ['1stWon', '2ndWon','SvGms', 'bpFaced']
2. ['rank', '1stWon', '2ndWon','SvGms', 'bpFaced']
3. ['rank', '1stWon','SvGms', 'bpFaced']
4. ['ace', '1stWon', '2ndWon','SvGms', 'bpFaced']

These variables were scaled using the StandardScalar function.

Train-Test Splitting

Since tennis matches follow a time series structure, it was essential to split the dataset accordingly. I used data from before and after a set date, ensuring that test data comprised unseen matches. The dataset was split into a 63% training set and a 37% test set. The training set included data from 2017-01-02, 2017-01-09, and two-thirds of 2017-01-16, while the test set contained the remaining one-third of 2017-01-16 and all of 2017-02-03. As a note, these were the only 4 dates included in this csv data.

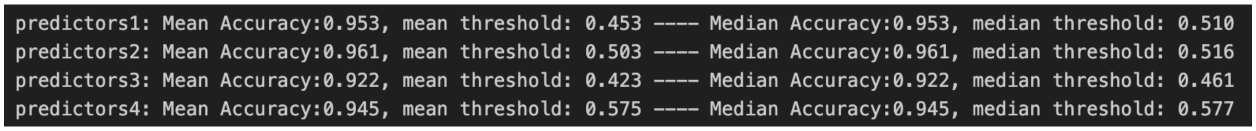
Logistic Regression Model

To build a robust logistic regression model, I employed LogisticRegressionCV that tried out 10 different C values with cross-validation via K-Fold with 5 splits to determine the optimal regularization parameter (C). The best C was 0.359. Additionally, I implemented a function to determine the best probability threshold for classifying wins and losses. This function performed k-fold validation within the training set, calculating thresholds across folds and returning the mean and median optimal thresholds. In greater detail, this process entailed fitting the sub training data into a logistic regression model using best C; then determining the predicted probability from the X validation data; and finally using the roc_curve function to identify the threshold that maximizes the difference between TPR (True Positive Rate) and FPR (False Positive Rate). This threshold was appended to a list of thresholds, and when the k-fold cross validation process came to an end, I returned the mean and median of this list. Both yielded the same accuracy score despite minor variations in cutoff points.

The best logistic regression model used predictors: ['rank', '1stWon', '2ndWon', 'SvGms', 'bpFaced'], achieving:

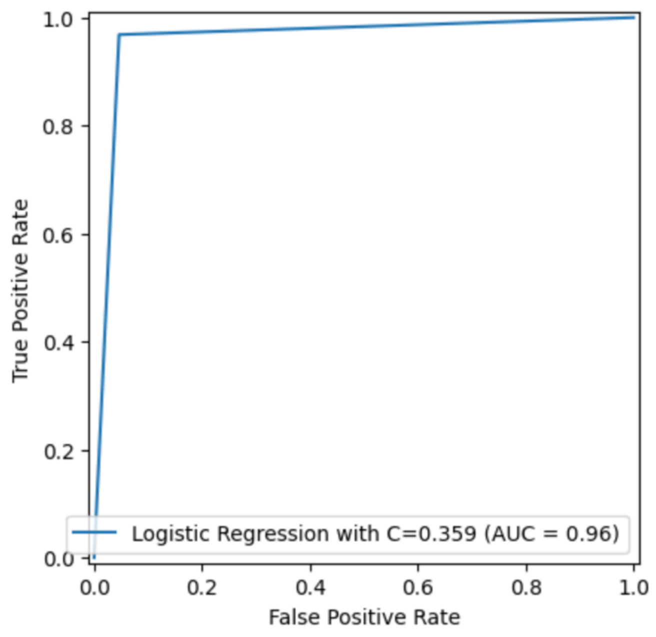
- **Mean Accuracy:** 0.961, **Mean Threshold:** 0.503
- **Median Accuracy:** 0.961, **Median Threshold:** 0.516

My general results for all predictors are shown below, with the predictors number corresponding to the index values above in the “Feature Selection” section.



```
predictors1: Mean Accuracy:0.953, mean threshold: 0.453 ---- Median Accuracy:0.953, median threshold: 0.510
predictors2: Mean Accuracy:0.961, mean threshold: 0.503 ---- Median Accuracy:0.961, median threshold: 0.516
predictors3: Mean Accuracy:0.922, mean threshold: 0.423 ---- Median Accuracy:0.922, median threshold: 0.461
predictors4: Mean Accuracy:0.945, mean threshold: 0.575 ---- Median Accuracy:0.945, median threshold: 0.577
```

My logistic regression model's predictive capability can also been seen by its ROC curve's AUC. ROC AUC Curve is a visual measure of the model's true positive rate against its true negative rate. The AUC refers to the area under this curve, which is between 0 and 1. My best model had an ROC AUC of 0.96.



Coefficient Interpretation

The logistic regression coefficients provide valuable insights into feature importance:

The logit function for the best performing predictors was:

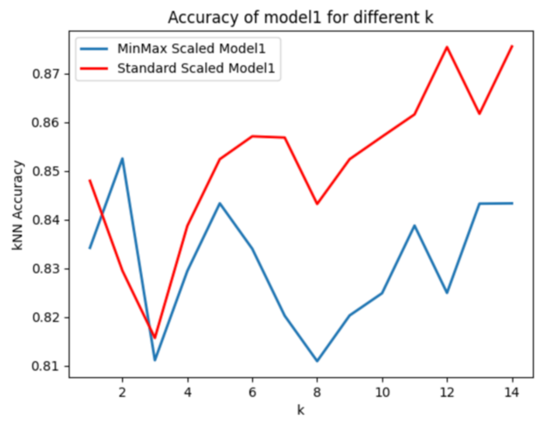
$\log(p/(1-p)) = -0.014 + -0.517(\text{rank\_scaled}) + 1.664(\text{1stWon\_scaled}) + 1.413(\text{2ndWon\_scaled}) + 0.375(\text{SvGms\_scaled}) + -1.776(\text{bpFaced\_scaled})$

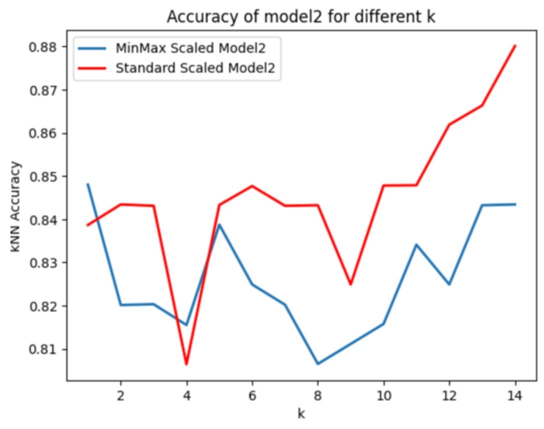
- **rank (-0.517):** The odds that player_w beats player_l was e^-0.517 = 0.6 times as large for every 1 greater difference in the rank between the two players, holding everything else constant. Note a player with a 'lower' rank means they are better since rank 1 is the best, followed by rank2, and so on. This is why if rank of player_w – rank of player_l > 0, this suggests player_l is considered a better player, reducing player_w's odds of winning by 0.6 times per every incremental difference.

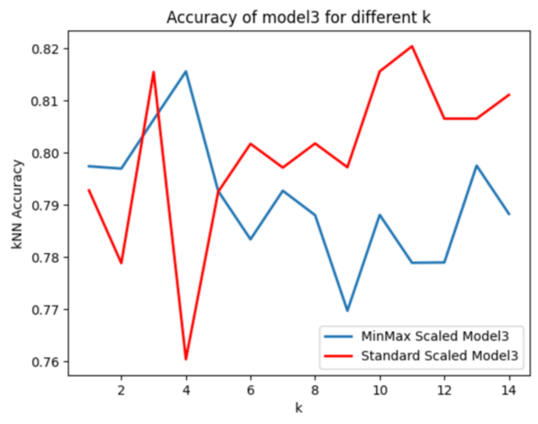
- **1stWon (1.664):** The odds that player_w beats player_l is e^1.664 = 5.28 times as large per every first serve points won by player_w compared to player_l, holding everything else constant.
- **2ndWon (1.413):** The odds that player_w beats player_l is e^1.413 =4.11 times as large per every second serve point player_w won compared to player_l, holding everything else constant.
- **SvGms (0.375):** The odds that player_w beats player_l is e^0.375 = 1.45 times as large per every one more serve game player_w played than player_l, holding everything else constant.
- **bpFaced (-1.776):** The odds that player_w beats player_l is e^-1.776 = 0.169 times as large per every one more breakpoint player_w faced compared to player_l, holding everything else constant. In other words, the odds that player_l beats player_w is 5.917 times as large for every one less breakpoint player_l faced compared to player_w.

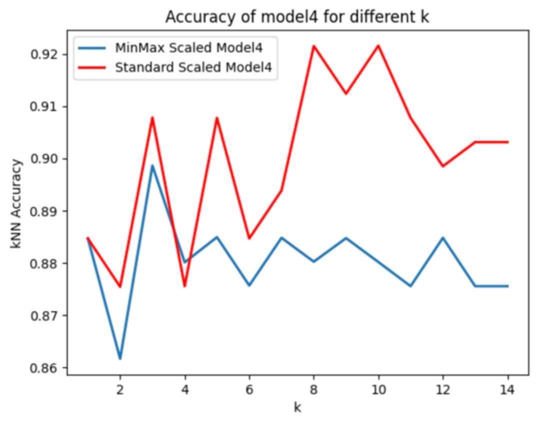
k-Nearest Neighbors Model

For kNN, I experimented with different values of k (from 1 to 15) for multiple sets of predictors and compared MinMax scaling with StandardScaler. Standard scaling consistently outperformed MinMax scaling, as can be seen below (model number has the same correspondence as before).









The best predictors for kNN were that of model4 ['ace', '1stWon', '2ndWon', 'SvGms', 'bpFaced'], with optimal k=8, 9, or 10. (9 doesn't seem optimal here but due to variability, 9 was sometimes the optimal k).

- **Training Set Accuracy:** 0.922 (k=8), 0.916 (k=9), 0.922 (k=10)
- **Test Set Accuracy:** 0.548 (k=8), 0.539 (k=9), 0.539 (k=10)

Despite promising results on training data, kNN performed poorly on the test set, suggesting logistic regression's superior performance on this dataset.

Model Comparison

A consistent metric was needed to compare logistic regression and kNN. Instead of using ROC-AUC only for logistic regression and accuracy for kNN, I calculated accuracy for both. Logistic regression significantly outperformed kNN:

- **Logistic Regression Test Accuracy:** 0.961
- **kNN Test Accuracy:** 0.5438

kNN's accuracy selloff in the test set might have been due to its sensitivity to overfitting.

Conclusion and Future Work

This project demonstrates the effectiveness of logistic regression in predicting ATP match outcomes. The model achieved high accuracy and provided interpretable coefficients for key match statistics. In contrast, kNN performed poorly in out-of-sample testing.

Future Extensions

- **Live Predictions:** Implementing a real-time prediction system for upcoming ATP matches.
- **Recent Match Trends:** Using a player's last 10 matches rather than data from 2017.
- **Alternative Models:** Exploring tree-based models like Random Forest or Gradient Boosting to improve predictive performance.

Overall, logistic regression, with carefully tuned regularization and feature selection, proved to be the most effective model for this dataset.