Predicting Tennis Match Results

Problem Statement

Tennis is an extremely competitive sport that involves physical skill, mental capability, and logic. But what exactly determines the winner of a tennis match? There are a multitude of different statistics in a tennis match, including number of aces, number of break points, number of double faults, and many more. What effect do these factors have on the winner of a tennis match? This is the question that this project will explore. More generally, the problem to be solved is determining what statistics or areas are most important in determining the winner of a tennis match. In addition, machine learning models will be created to predict the outcome (win/loss) of tennis matches.

There are many possible clients who could benefit from this project. Most obvious would be the tennis players and coaches. Based on a predictive model, they could practice/teach the areas of the game that were found to be most important in the model. Other possible clients include tennis bettors for obvious reasons; namely, to determine which player to bet on. Finally another possible client could be sponsors, such as Nike or Wilson.

Data Wrangling

Kaggle has provided a dataset including detailed information of ATP (Association of Tennis Professionals, Men) matches from 2000 to 2019. The individual match information provided includes number of aces, number of double faults, 1st serve in percentage, break points saved, and more. While the data tables from each individual year only include a winner ID and loser ID, Kaggle has also provided a separate table including information (country, birth date, etc.) about the different players and can be referenced according to the IDs in the individual match tables.

There are a total of 21 csv files: one csv file for matches in each year from 2000-2019 and one csv file for players. The first step was to combine the 20 csv files containing match data into a single dataframe. This dataframe had 59,430 entries and 32 columns. However, each entry in this dataframe represented a single match, including statistics for both the winner and the loser in the same row. From this dataframe, I created a new dataframe where each row represented either a winner or a loser. Additionally, I added a column named ‘outcome’ where ‘1’ represents a win and

‘0’ represents a loss.

There was some missing data where when one statistic was missing, all the statistics were missing. These rows with missing statistics were dropped from the dataframe. Several columns were dropped as well. Columns with irrelevant information, including tourney_id and tourney_name, were dropped. Columns that were the same regardless of winner or loser, including surface, score, round and minutes, were dropped. Finally, the best_of column was dropped after eliminating all matches that were a best of 5 sets and keeping only the matches that were a best of 3 sets. Next, I supplemented my matches dataframe with player data, including the birthdate, name and country. From the birthdate, I extracted the age at the time of the tournament.

The final shape of the dataframe was 86,230 rows with 14 columns, 13 features and 1 dependent variable called ‘outcome’. The 13 features and their descriptions are listed below:

age: player age at time of tournament

rank_points: number of ranking points at time of tournament

rank: rank at time of tournament

bpFaced: number of break points faced

bpSaved: number of break points saved

SvGms: number of serve games

2ndWon: number of second-serve points won

1stWon: number of first-serve points won

1stIn: number of first serves made

svpt: number of serve points

df: number of double faults

ace: number of aces

hand_R: right-handed (1) or left-handed (0)

Exploratory Data Analysis

First, I created a heatmap, Figure 1, to show the correlations between the features

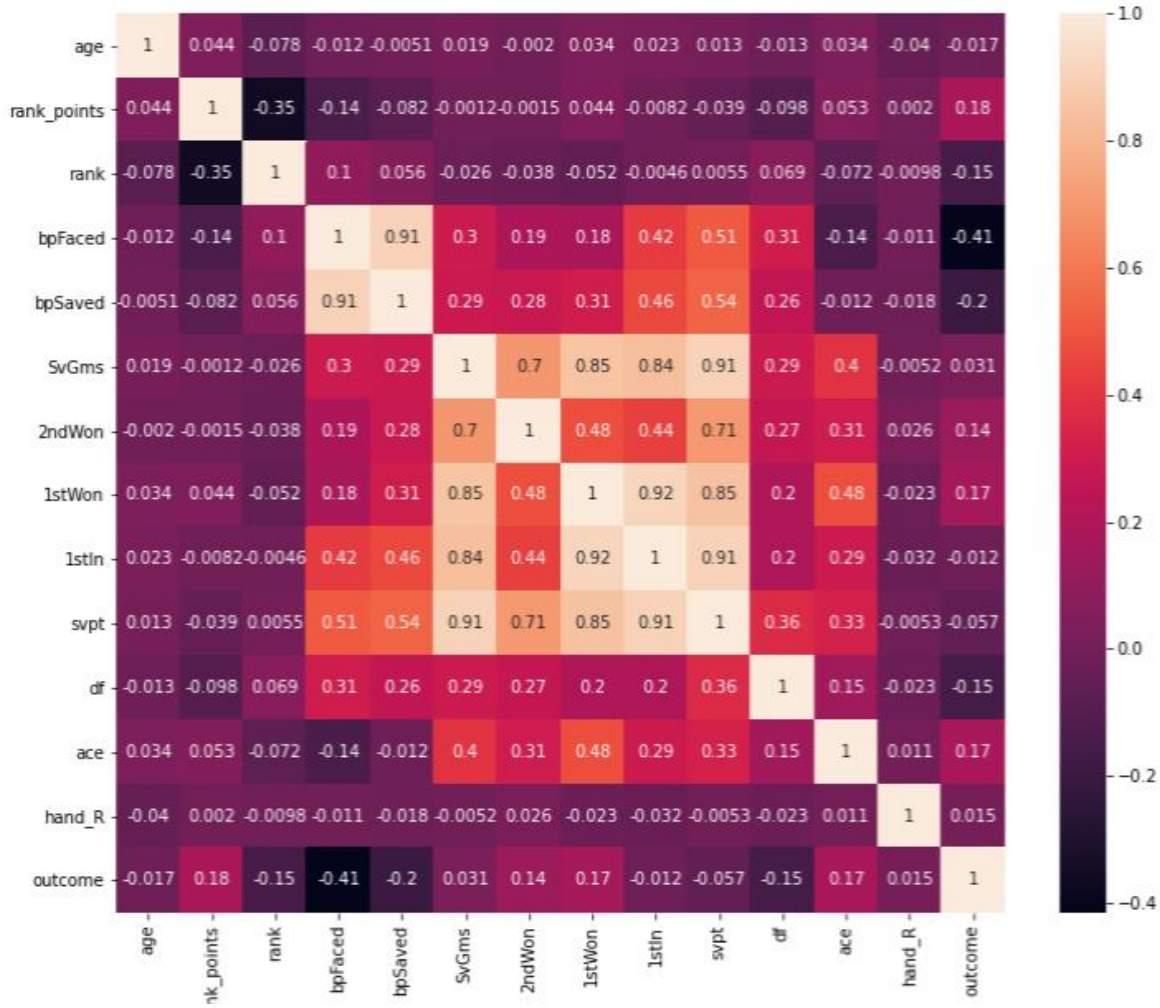


Figure 1: Heatmap of Features

Clearly, there are several features which appear to be highly correlated, in that they have correlation coefficients greater than 0.7. In order to visualize these correlations, I plotted the features that are correlated. Figure 2 and Figure 3 show two examples of these visualizations.

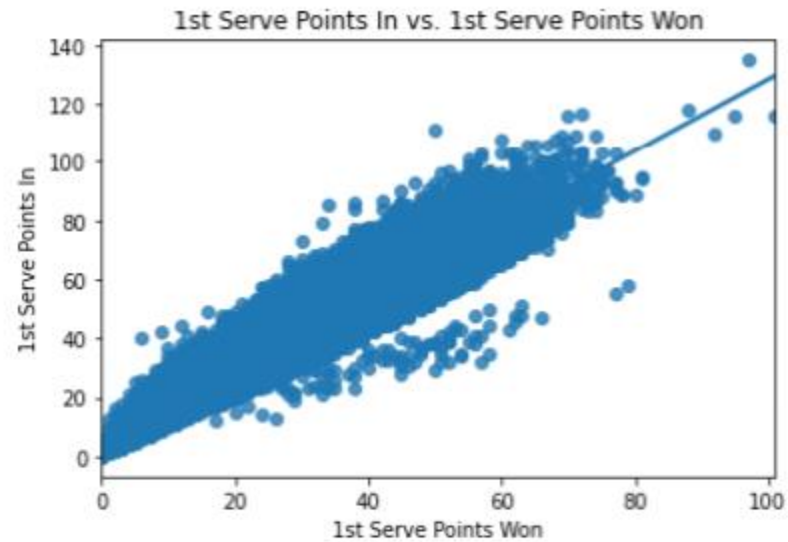


Figure 2: Correlation between 1st serve points in and 1st serve points won

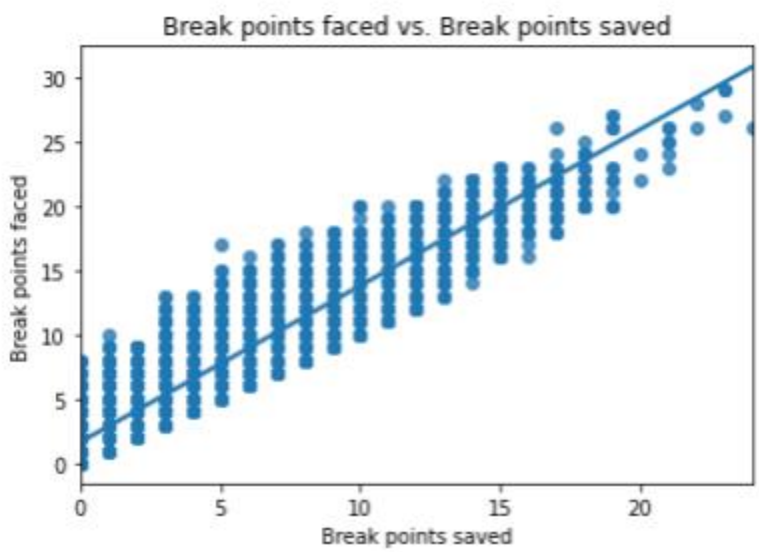


Figure 3: Correlation between break points faced and break points saved

There is an obvious positive correlation between 1st serve points in and 1st serve points won. The more 1st serve points in, the more 1st serve points won. Additionally, there is a positive correlation between break points saved and break points faced. The more break points faced, the more break points saved. There were several other positive correlations that are not shown

Next, I sought to visualize how some of the features affect the outcome (win/loss). Figures 4 and 5 show two example boxplots. Figure 4 shows that the median value of break points faced is significantly higher for the losses as opposed to the wins. On the other hand, Figure 5 shows that the median value of aces is significantly higher for the wins as opposed to the losses.

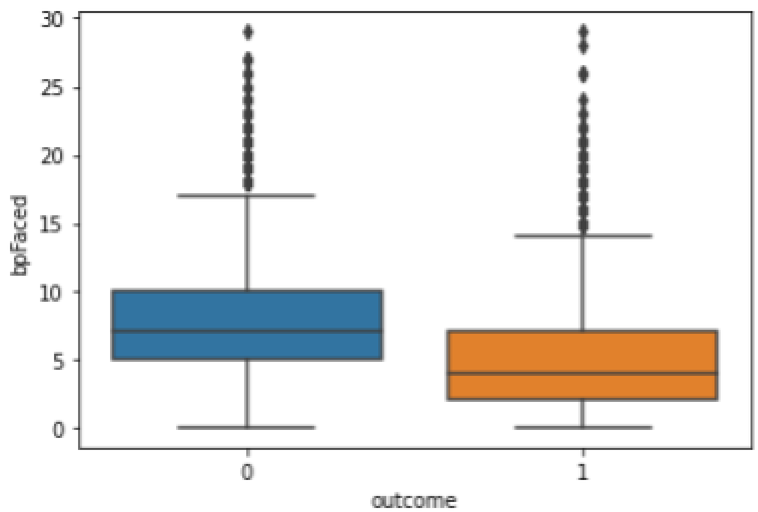


Figure 4: Effects of number of break points faced on outcome

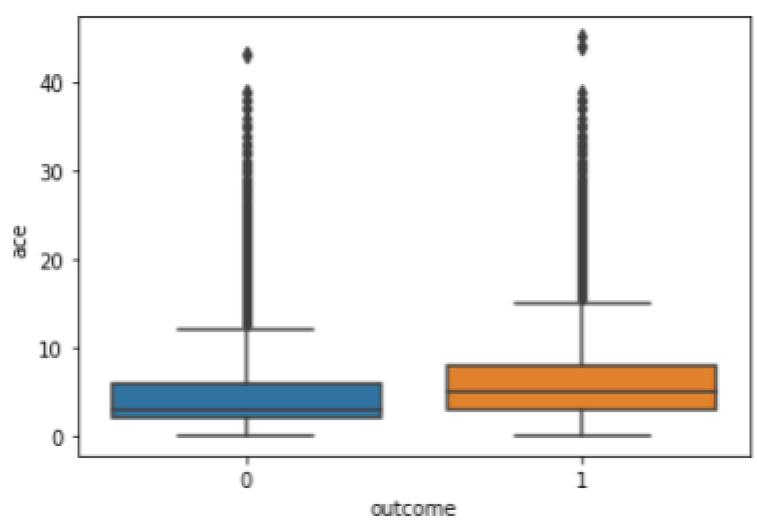


Figure 5: Effects of number of aces on outcome

Preprocessing:

Exploration of the feature histograms shows that the features need to be standardized to zero-mean and unit-variance. Additionally, some of the features had long right tails, so they were log-transformed. Figure 6 shows the result of log-transforming and standardizing the 'ace' feature.

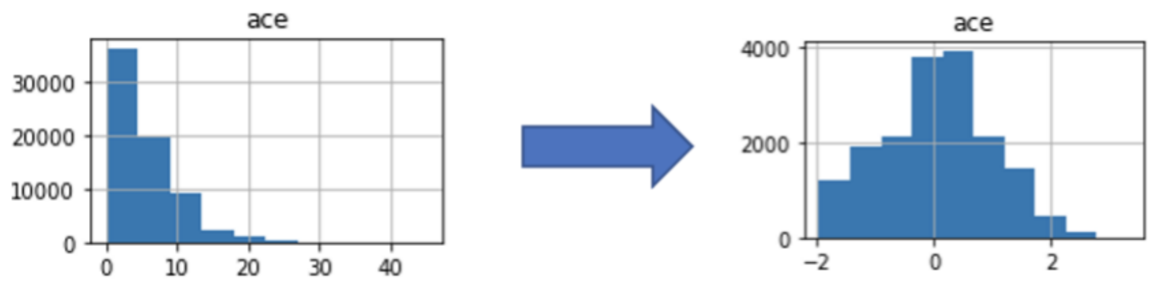


Figure 6: Log transformation and standardization of 'ace' feature

Next, the dataset was split into a training set and a testing set, giving X_train, X_test, y_train, and y_test. The testing set size was 20%, giving X_train a shape of 68,984 rows by 13 columns and X_test a shape of 17,246 rows by 13 columns. To ensure the results would not be biased, value counts of the outcome were calculated for both y_train and y_test, as seen in Table 1 below. As you can see, there are approximately an equal number of both outcomes in both the training and testing sets.

Table 1: Value Counts of Outcome for Training and Testing Sets

	y_train	**y_test**
Outcome = 0	34,429	8,607
Outcome = 1	34,555	8,639

Modeling and Parameters:

The goal of this work was to construct a model capable of predicting tennis match results (win/loss). Therefore, this is a classification problem. I used the following classification models for prediction:

- Logistic Regression
- K-Nearest Neighbor (KNN)
- Support Vector Machine (SVM)
- Random Forest
- Naïve Bayes
- Gradient Boosting

Logistic Regression: A simple loop was used to determine the best value for the regularization parameter, C. The best value for C was found to be 0.01.

K-Nearest Neighbor: The first step here was to determine the ideal number of neighbors. Figure 7 below shows that 3 is the ideal number of neighbors.

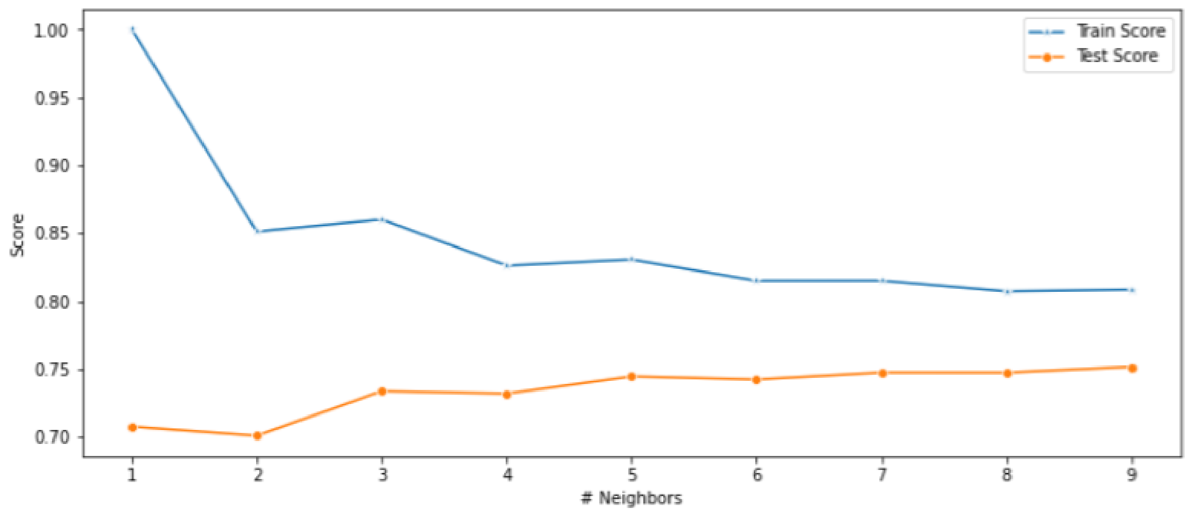


Figure 7: Number of neighbors vs. score for KNN

Model Selection Metrics

The optimal model was selected based on both primary and secondary performance metrics. The primary performance metric was the model accuracy score, which is the number of correct predictions made divided by the total number of predictions made. One secondary performance metrics was the time taken to train the model and make predictions. A second secondary performance metric was the ROC-AUC train and test scores from 5-fold cross-validation.

Results

The primary performance metric, the model accuracy score, is seen in Table 2 for each of the 6 models and is plotted in Figure 8. The SVM model has the highest model accuracy, but the Logistic Regression, Random Forest, and Gradient Boosting models are all within 1% of the SVM model accuracy score. Therefore, the secondary performance metrics were used to determine the best model.

Algorithm	Model Accuracy Score
Logistic Regression	0.798736
KNN	0.733677
SVM	0.803897
Random Forest	0.793633
Gradient Boosting	0.794329
Naive Bayes	0.715354

Table 2: Model Accuracy Scores

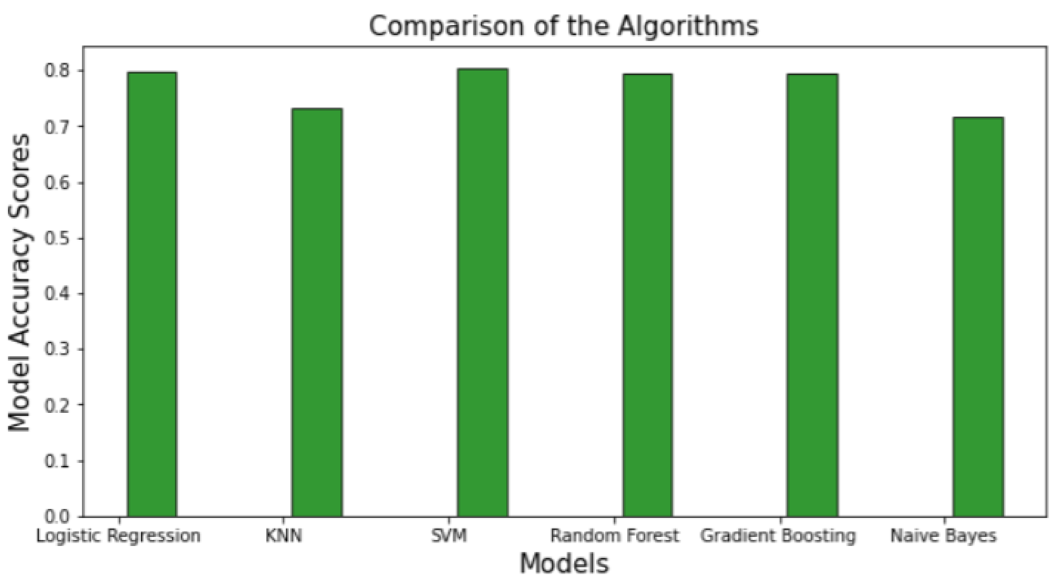


Figure 8: Model Accuracy Scores

The 'ROC-AUC' scores for both the training set and the test set were used as secondary performance metrics. Table 3 shows the tabulated scores, and Figure 9 shows the scores graphically. Similar to the model accuracy score, the ROC-AUC train and test scores are the highest for SVM, but the Logistic Regression, Random Forest, and Gradient Boosting are all within 1%. Therefore, the time to train the model and make predictions was used to determine the best model.

Table 3: Model ROC-AUC Train and Test Scores

Algorithm	ROC-AUC Train Score	ROC-AUC Test Score
Logistic Regression	0.871921	0.870983
KNN	0.790149	0.779892
SVM	0.894223	0.893458
Random Forest	0.888146	0.881503
Gradient Boosting	0.887809	0.888341
Naive Bayes	0.786312	0.789141

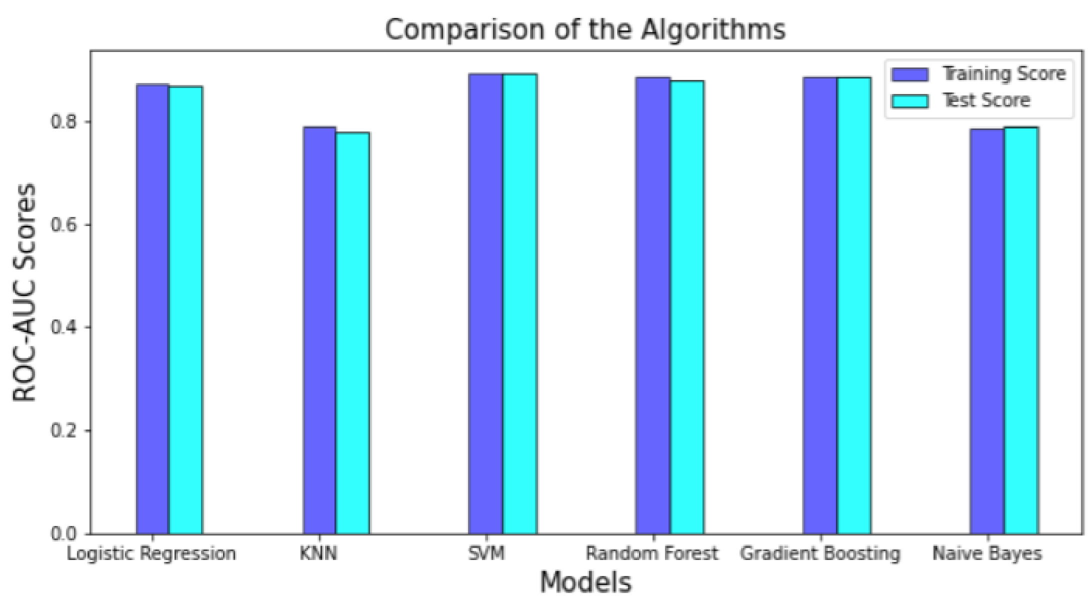


Figure 9: Model ROC-AUC Train and Test Scores

The times to train the model and make predictions are seen in Table 4 for each of the 6 models. Although SVM had the highest accuracy score, it also had the longest train and predict times by far. The Logistic Regression model had one of the lowest train times and the lowest predict time. Therefore, based on efficiency and model accuracy score, we can conclude that the Logistic Regression model is the optimal model.

Table 4: Train and Predict Times for Each Model

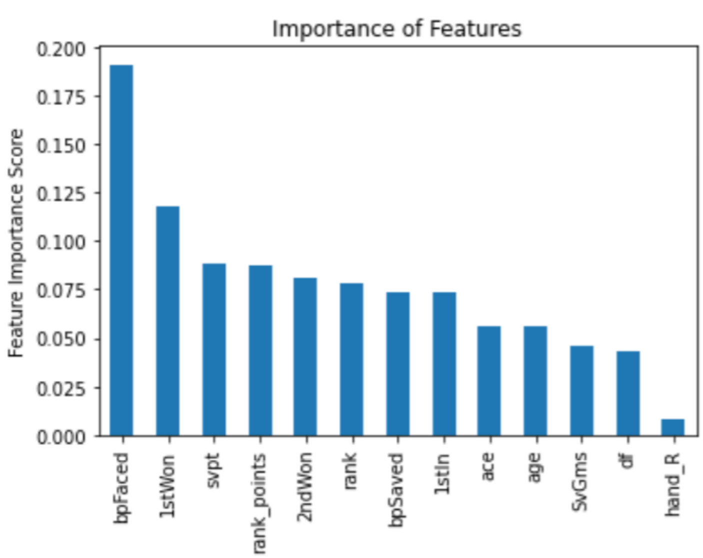
Model	Train Time (s)	Predict Time (s)
Logistic Regression	0.1546	0.0028
K-Nearest Neighbor	0.2018	3.6273
Support Vector Machine	81.6413	8.9006
Random Forest	7.4065	0.3300
Gradient Boosting	6.9962	0.0343
Naïve Bayes	0.0230	0.0052

Hyperparameter Tuning

A grid search was used to determine the optimal hyperparameters for the Logistic Regression model. The optimal combination of hyperparameters was found to be C=1 and solver='newton-cg'. The model accuracy score for this combination was 79.42%.

Feature Importance

Finally, in order to look at feature importance, I looked at the random forest model. Figure 10 below shows the feature importance for each of the 13 features. The two most important features were break points faced and 1st serve points won. I also plotted the ROC curve for the random forest model, as seen in Figure 11.



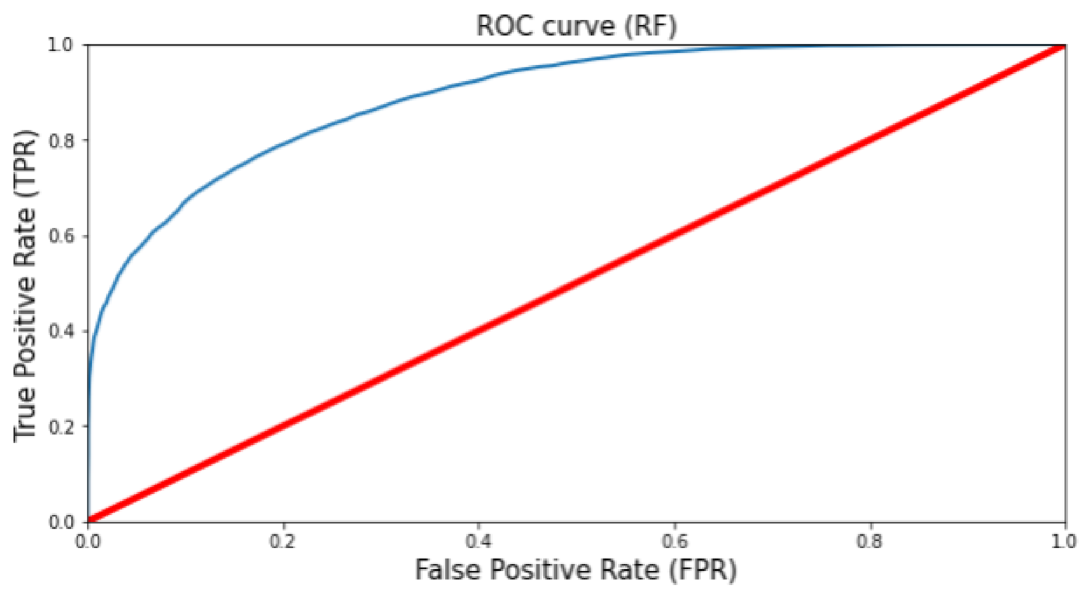


Figure 11: ROC Curve from Random Forest Model

Conclusion

This project aimed to predict tennis match results (win/loss). There was significant data wrangling involved. Each entry in the raw dataframe represented a single match, including statistics for both the winner and the loser in the same row. From this dataframe, I created a new dataframe where each row represented either a winner or a loser. Additionally, I added a column named 'outcome' where '1' represents a win and '0' represents a loss.

Exploratory data analysis showed that many of the features are correlated with each other. Additionally, boxplots showed that some features had a clear effect on the outcome. Preprocessing involved standardizing the data and splitting the data into a training set and a testing set.

Six machine learning models were evaluated. Based on the model accuracy scores and the train and predict times, the Logistic Regression model was the optimal model. Hyperparameter tuning for the Logistic Regression model found the optimal combination of hyperparameters to be C=1 and solver='newton-cg', with a model accuracy score of 79.42%.

From the Random Forest model, I was able to extract feature importance. The two most important features were break points faced and 1st serve points won. This is extremely important for my clients to know. It tells them that in order to have a higher chance of winning a tennis match, they need to work on preventing break points and winning their 1st serve points.

Future Improvement

Here I have used data from the years 2000-2019, but data is available starting from 1968. The model could possibly be improved if I use data from earlier years. Hyperparameter tuning was unsuccessful for the best model, but I could perform more hyperparameter tuning for each of the individual models. I could also investigate model stacking, which utilizes multiple learning algorithms. Finally, I used all 13 features in my models. I could define an importance cutoff and use features only with an importance higher than the cutoff. This could improve the model accuracy.