Feature Importance Analysis: Predicting Tennis Match Outcomes

Liz Garcia Ovalles

Abstract

In today's data-driven world, tennis offers extensive player and match statistics that can be utilized for match outcome predictions, benefiting both betting markets and training programs. However, studies differ on whether player rank or serve strength alone suffices for accurate predictions. This study evaluates XGBoost and Random Forest models trained on match and player statistics, rank alone, serve-related features alone, and rank with service-related features only. Results show that both models achieved at least 80% accuracy using all features or serve-related features, compared to around 60-65% when using rank alone. Adding player rank to serve-related features did not significantly improve model performance. These findings highlight serve strength as the most critical predictor of outcomes in men's singles tennis matches on the ATP circuit.

1. Introduction: Problem and Questions of Interest

Nowadays, there is plenty of data collected about tennis players to quantify and analyze their performance. For example, for men's tennis, there are ATP rankings which are based on the points earned by the players in official ATP-certified men's singles or doubles matches over the preceding 52-week time frame [5]. There is also data regarding velocity of serves, number of aces, valid first or second serves, among many others. **Problem:** There is a discrepancy among studies on whether players' rank or serve strength data alone is enough to accurately predict the outcomes of tennis matches. **Questions:** The questions this project aims to answer are the following. What are the most important features needed to accurately predict tennis match outcomes? More specifically, is the rank or serve strength of a player enough to make accurate predictions on tennis match outcomes? Could including both players' rank and serve strength data for the training of machine learning models (such as Random Forest and XGBoost) achieve better results than existing studies have obtained? **Relevance:** These questions are valuable not only for individuals invested in the betting industry but also for players and coaches in learning about the key factors needed to win a tennis match.

2. Data and Variables

The proposed project will collect ATP data[1] on men's singles matches from 2000 to 2024, sourced from Jeff Sackmann's repository on GitHub. This data contains 73,247 rows (matches) and 49 columns (variables). From this dataset, we will extract key variables: year, tournament name, surface, tournament level, round, draw size, match duration (in minutes), winners' and losers' handedness, height, age, number of aces, number of double faults, number of serve points, number of valid first serves, number of first-serve points won, number of second-serve points won, break points saved, break points faced, and most recent rank.

To minimize issues of multicollinearity while still using as much information as possible, we will derive predictors from combinations of the aforementioned variables. This is an approach employed by Gao and Kowalczyk [2]. As a result, the final dataset for model training will include: year, tournament name, tournament level, minutes, surface type, draw size, round, winners' and losers' handedness, height, age, rank, ratio of aces over double faults, percentage of first/second serves in, percentage of first/second serves in and won, and percentage of break points saved.

These variables will be utilized to predict the outcome of tennis matches based on the players' attributes (e.g., age, number of aces) and match statistics (e.g., surface type, duration).

3. Relevant Literature

This project is motivated by four different studies which employed machine learning methods to predict tennis match outcomes based on players' skills and match statistics. The following is a summary of these studies.

The study by Wilkens [1] trains logistic regression, neural network, random forest, gradient boosting machine, and support vector machine models to predict tennis outcomes of singles matches. **Data:** The primary data used for this study are records of the major men's singles ATP (Association of Tennis Professionals) and women's singles WTA (Women's Tennis Association) matches, as well as records from the four Grand Slam tournaments. **Features/predictors:** There were fifteen features used in the training of the models as explanatory variables for tennis match prediction. These were: average bookmaker odds for the favorite / longshot player, spread between best and average odds for the favorite / longshot player, gender, type of tournament (e.g., Grand Slam), tournament round, difference in ranking between longshot and favorite, difference in ranking points between favorite and longshot, difference in age between favorite and longshot, player's preferred hand, home advantage (if tournament is held in a player's home country), surface advantage, record of performance against opponent in the last three years, and player's average ranking on a log-scale over the previous six months minus his or her current ranking. For a detailed description of these features, please see Table 2 of the study [1]. The **target variable** of this study was the probability that the favorite player wins the match. As a **baseline**, this study uses the players' rankings to form a

[1] https://github.com/JeffSackmann/tennis_atp

straightforward definition of a meaningful baseline model. The player with the highest ranking (favorite) always wins. Besides this, a bookmaker-implied baseline (without an explicit model) is calculated by using the betting odds to determine the probability of the favorite player winning, thus deriving the outcome of the match. In terms of **evaluation metrics**, the Area-Under-the-ROC-Curve (AUC), which has values between 0.5 and 1.0, summarizes the performance of the classification models across different classification thresholds. Additionally, the accuracy, precision, recall, specificity, and F-score (harmonic mean of precision and recall) were also calculated. **Training models:** As aforementioned, this study trains and evaluates logistic regression, neural network, random forest, gradient boosting machine, and support vector machine models. For training and prediction, sliding windows were utilized, with three-year calibration windows used to forecast the probabilities for the subsequent year. Starting with the period 2010 through 2012 for training and predicting for the year 2013, the setup resulted in seven calibrations (2010-2012 through 2016-2018) and prediction periods (2013 through 2019) in total. The time span of the sliding windows is a compromise between a too-long window that ignores potential changes in player composition, gameplay, and betting markets over time, and a too-short one that does not allow for statistically robust calibrations. **Results:** All models led to moderate AUC figures of around 0.7. The accuracy of the models were about 70% during the calibration and **69%** during the prediction. The baseline model has an accuracy of about **66%**, equal to the proportion of wins by the favorite player across matches. The bookmaker-implied benchmark showed a performance no different from that of the models. All models performed similarly in precision with values around 71-72%; for the baseline model, the precision is by design equal to the accuracy, 66%. In terms of recall, specificity and F-score, all models performed relatively the same. The conclusion of this study is that the official player rankings and bookmaker odds together encompass most of the information to predict match outcomes. Thus, the addition of historical match and player data such as tournament series and round, age difference between opponents, and home advantage do not really add any additional explanatory power. Besides this, a second finding was that the various machine learning models employed performed closely the same with prediction accuracy typically not exceeding 70%, and as such not outperforming the model-free bookmaker odds alone. The models also did not do much better than the simple baseline model which only used the current rankings to determine the match outcome, which has an accuracy of around 65%.

In contrast to the previous study which identified ranking as the most significant predictor, the study performed by Gao and Kowalczyk [2] employed random forest to identify serve strength as the key predictor of tennis match outcomes. **Data:** The dataset used in this study was collected from numerous other datasets from the ATP (Association of Tennis Professionals), including the ATP World Tour which comprises the ATP World Tour masters 1000, ATP World Tour 500 series, ATP World Tour 250 series, and the ATP Challenger Tour. The final merged dataset ranges from 2000 to 2016 and contains data about environmental factors, general match information, match results, and the average of betting odds from major betting companies. Missing data values were filled with the median value for each feature to

allow for the full use of the dataset without having these values be affected by skewed data and lead to a detrimental impact on the performance of the predicting model. **Features/predictors:** This study employed 18 features to predict match outcomes: player's height, age, rank points, court surface, percentage of ace over double faults, previous percentage of games won, numbers of championship, games won before in the same round, games played in past 12 months, percentage of games won in the same tournament, percentage of games won against player with same handedness as the current opponent, percentage of games won on the same type of surface, percentage of games won against the same opponent, percentage of making the first/second serve, percentage of making the first/second serve and winning, percentage of break points saved, and lastly current round number. For a detailed description of these features, please see Table 1 of the study [2]. Similar to the previous study, the **target variable** is the probability that a player (with a given set of characteristics) will win the match. As a **baseline** model, the betting odds information is used to evaluate the predictive performance of the trained models. **Evaluation metrics:** Model accuracy was assessed based on test accuracy using the random train/test split and 10-fold cross validation. To compare the performance of the trained models with the betting odds, the following formula was used to calculate scores: ***Score = w * (p - 0.5).*** The p is the probability of winning according to either betting odds or probabilities from machine learning predictions. The w is an indicator variable representing whether a player won or lost the match (w = 1 indicates the player won, while w = -1 indicates that the player lost). A high score (up to 0.5) indicates that a player was predicted to win a match with high certainty and the player did win the match, or that a player was predicted to lose a match with high certainty and the player did lose. Alternatively, a low score (down to -0.5) indicates that a player was predicted to win a match with high certainty and the player lost the match, or that the player was predicted to lose a match with high certainty but actually won. Scores close to zero indicate low prediction confidence, with positive values indicating correct low-confidence predictions and negative values indicating incorrect low-confidence predictions. **Trained models:** To predict match outcomes, this study employed a support vector machine with a radial basis function kernel, logistic regression, and random forest classification, since these allow for easy interpretability and fast training times that facilitate inclusion of additional data as it becomes available. **Results**: Overall, the random forest model showed the highest prediction accuracy at **83.18%** compared to accuracy of 69.04% when using predictions based on betting odds probabilities. However, betting odds and the support vector machine model show higher scores compared to the random forest model and the logistic regression model, where the former predict with higher confidence but with lower accuracy than the latter models. This study identified serve strength (the proportion of first and second serves for which a player won a point) as the key factor to accurately predict match outcomes. This may be due to the fact that serve strength is likely to encapsulate a wide range of information about overall player performance. Furthermore, the study found that external factors such as tournament round and court surface, as well as psychological factors such as proportion of break points saved and record against the current opponent, have little impact on overall prediction accuracy. Similarly, physical player

characteristics such as player age and height improved model accuracy, but not as much as serve strength. An important finding that is key for the present study is that features related to player skill, such as rank and winning record, were observed to have relatively little impact on prediction accuracy.

4. Other relevant studies

The study by Yue et al. [3] is relevant for the proposed project because it supports the findings of the study by Wilkens [1] which found **a player's rank** is the most significant variable for predicting the outcome of a tennis match. The study by Yue et al. [3] employs logistic regression, support vector machine (SVM), neural network, and lightGBM (light gradient boosting machine). Using a dataset containing data on 20 years of Grand Slam single matches from 2000 to 2019 for men and women, 49 variables were considered for prediction which included players' demographic information (such as residence, age, year their career started, preferred hand) and the ranking data (rank and points). Similar to the study by Wilkens [1], this study by Yue et al. [3] utilizes ranking as the baseline model, where it always predicts as the winner the player with the highest ranking. Additionally, like the study by Wilkens [1], this study evaluated prediction performance by comparing accuracy and the area under the curve (AUC) between the trained models and the baseline model and observed a prediction accuracy of approximately **70%** by its models.

On the other hand, the study by Yu and Wang [4] supports the findings of the study performed by Gao and Kowalczyk [2] in their conclusion that **serve strength** (specifically the scoring rate of the first serve, the scoring rate of the second serve, and the scoring rate of the receiving serve) is the most important factor in accurately predicting the outcome of a match. Using data collected from the official ATP website for men's single matches in 2021, this study employed logistic regression to determine the feature importance of the variables available. The core three indicators related to serve strength were deemed the most critical and thus were used in the logistic regression model to predict the probability that a player wins the match, with the threshold (split point) being 0.5. The overall accuracy of the final model was **82.1%**.

5. Anticipated results

Given that the studies by Wilkens [1] and Yue et al. [3] found **rank** to be the most critical predictor, and the studies by Gao and Kowalczyk [2] and Yu and Wang [4] found that **serve strength** was the most significant predictor, it might be that the rank and serve strength of a player are highly correlated. Thus, combining these two variables might not lead to higher accuracy. However, since only study [2] includes both rank and serve strength data to predict the outcome of tennis matches, it is of interest to train a model with both of these predictors and verify whether it would help improve the predictive performance.

Additionally, note that studies [2] and [4] trained logistic regression and/or random forest to predict tennis match outcomes, identifying **serve strength** as the most significant predictor

and achieving overall accuracy of approximately 82-83%. In contrast, studies [1] and [3] trained logistic regression and/or random forest to predict tennis match outcomes, identifying **rank** as the most important predictor and achieving an overall accuracy of approximately 70%. This implies that out of the two, **serve strength** might be the best predictor. To investigate this discrepancy and verify the results found in the study by Gao and Kowalczyk [2], which includes both rank and serve strength as predictors, this study will train and evaluate a random forest model. For comparison, an XGBoost model will also be trained and evaluated. The inclusion of an XGBoost model is of particular interest as there is a scarcity of research employing this technique for predicting tennis match outcomes based on players' characteristics and match statistics (including rank and serve strength data).

Overall, the primary objective of this study is to determine whether rank, serve strength, or the inclusion of both can be regarded as the most effective predictor of outcomes in men's singles ATP matches.

6. Exploratory Data Analysis

Table 1 below is a data dictionary of the variables selected from ATP data[1] on men's singles matches from 2000 to 2024, sourced from Jeff Sackmann's repository on GitHub. It contains the data type and a description for each variable.

Table 1: Selected Variables from ATP data on men's singles matches from 2000 to 2024

Variable	Data type	Description
year	numerical	Year of the date of the tournament during which the match takes place
tourney_name	categorical	Name of tournament to which the match belongs
surface	categorical	Surface type of tennis court
tourney_level	categorical	'G' = Grand Slams, 'M' = Masters 1000s, 'A' = other tour-level events, 'C' = Challengers, 'S' = Satellites/ITFs, 'F' = Tour finals and other season-ending events, 'D' = Davis Cup
round	categorical	Stage of the tournament in which the match takes place, typically denoted by abbreviations like "R32" (Round of 32), "R16" (Round of 16), "QF" (Quarterfinals), "SF" (Semifinals), and "F" (Final). The number represents the number of players remaining in the draw at that stage. For

		example, "R32" refers to the Round of 32, meaning 32 players are still in the tournament at that point. This notation indicates the progression of the match within the overall tournament structure, which is determined by the draw size
draw_size	numerical	Number of players in the draw, often rounded up to the nearest power of 2. (For instance, a tournament with 28 players may be shown as 32.)
minutes	numerical	Duration of match, in minutes
winner_hand, loser_hand	categorical	R = right, L = left, U = unknown. A = for ambidextrous players, this is their serving hand
winner_ht, loser_ht	numerical	Height in centimeters
winner_age, loser_age	numerical	Age, in years, as of the date of the tournament
w_ace, l_ace	numerical	Number of aces
w_df, l_df	numerical	Number of double faults
w_svpt, l_svpt	numerical	Number of serve points
w_1stIn, l_1stIn	numerical	Number of first-serves made in
w_1stWon, l_1stWon	numerical	Number of first-serve points won
w_2ndWon, l_2ndWon	numerical	Number of second-serve points won
w_bpSaved, l_bpSaved	numerical	Number of break points saved
w_bpFaced, l_bpFaced	numerical	Number of break points faced (saved or missed)
winner_rank, loser_rank	numerical	ATP rank, as of the tournament date, or the most recent ranking date before the tournament date

Missing Data

Upon inspecting the missing values of the original dataset obtained from GitHub, we found columns `loser_entry, winner_seed, loser_seed, and winner_entry`

had more than half of their total values missing. Therefore, we excluded these columns. The winner/loser entry indicates the manner in which a player enters the tournament (e.g., wild card, qualifier). The remaining columns have quite low percentages of missing entries, with a mean percentage of about 3.96% and a median percentage of around 1.38% of missing data in a given column. Of the variables that remained, we selected the ones shown in **Table 1** since they provide relevant information regarding player's skills and characteristics (e.g., age, height, number of aces and double faults) as well as match information (e.g., surface type, round, tournament name/level) that could affect player performance and thus would affect predictions for match outcomes. Specifically, we excluded `w_SvGms`, `loser_id`, `winner_ioc`, `match_num`, `loser_ioc`, `winner_name`, `winner_id`, `score`, `loser_rank_points`, `tourney_date`, `l_SvGms`, `winner_rank_points`, `best_of`, `loser_name`, `tourney_id` (see the data dictionary[2] provided on GitHub for the definition of these variables). Most of these variables are identifiers or general information unnecessary for making match outcome predictions based on players' performance. Upon selecting variables, we identified and removed 1 duplicate row and converted variables to the correct data types. Numerical variables were either converted to `float64` or `int64`, while categorical variables were assigned the `category` data type.

To clean the data of the remaining selected variables, we filled in missing data for numeric values using the median of each respective feature based on year and tournament name. For groups based on year and tournament name that are empty, we use the median of the feature column. As the study by Gao and Kowalczyk [2] note, taking the median allows us to use as much data as possible while assigning values that are unaffected by potentially skewed data and would not have a detrimental impact on model fit. For categorical variables, we excluded all rows that contained missing data since these variables contain critical information that is unique to each match (e.g., tournament name) or player (e.g., age) and thus cannot be replaced. This is also acceptable given that the percentage of data excluded by removing such rows is only around 0.14%.

Code 1: Handle Missing Numeric or Categorical Entries

```
# Fill in missing values with the median for numeric columns for each
group based on tourney_name and year

# Group data by 'tourney_name' and 'year'
# observed = True includes the categories that appear in the data,
# which avoids creating empty groups for categories that do not appear in
the data
grouped = df.groupby(['tourney_name', 'year'], observed=True)
```

[2] https://github.com/JeffSackmann/tennis_atp/blob/master/matches_data_dictionary.txt

```
# Iterate through numeric columns and fill missing values with the median
for each group
for col in numeric_cols:
 col_median = df[col].median()
 df.loc[:, col] = grouped[col].transform(lambda x: x.fillna(x.median()) if
not x.isna().all() else col_median)
```

```
# Remove all rows where categorical variables contain missing values
df = df.dropna(subset=categorical_cols)
```

Outliers

(Please note, we employ the convention of a postfix 'A' to indicate the figure can be found in the **Appendix** section.)

We exclude outliers before combining variables as predictors for model training and evaluation. This is because outliers in the individual features can disproportionately affect the derived feature, such as creating misleading or invalid ratios in the presence of extreme values.

Before outlier removal, let's inspect the distribution of numeric variables. The match data distribution across all years is relatively the same, although there is a slight decrease towards the more recent years. There is also a sudden decrease around the year 2020, which may be due to the Covid-19 pandemic that likely impacted the number of matches played that year (**Figure 4A)**. The distribution of winner and loser heights is approximately normally distributed, with an overall bell-shaped curve (**Figure 5A)**. The distribution of age of winners and losers is approximately normally distributed, but a bit skewed to the right (**Figure 8A)**. The distribution of the feature `minutes` (duration of matches) is approximately normally distributed, but contains extreme outliers (**Figure 6A)**. The distribution of aces, double faults, break points saved and faced, number of serve points, first-serve points in, and first-serve/second-serve points won for both winners and losers is approximately normal and/or skewed right. Moreover, the distribution of winner and loser rank is significantly skewed to the right, which makes sense since top-players (with low ranks) participate in ATP tournaments and matches (see figures in **Outliers section of Colab notebook)**. It is clear from these observations that there are extreme outliers that need to be removed for model training and evaluation.

To exclude outliers for numeric variables, we employed the Interquartile Range (IQR) method, which identifies data points that are significantly different from the majority of the data. The lower bound is calculated as `Q1 - 3.0 * IQR`, which determines the threshold below which data points are considered outliers. Similarly, the upper bound is calculated as `Q3 + 3.0 * IQR`, which determines the threshold above which data points are also considered outliers. At first, we used `1.5 * IQR`, but this removed around 50% of the data, eliminating completely the category D of tournament levels from our dataset. Therefore, we modified the outlier

threshold to `3.0 * IQR` which only leads to a decrease in the data by around 13%. Specifically, after numeric outlier removal, there remain 63,693 rows out of 73,145 (with 31 columns). To verify the validity of the remaining data after outliers were removed, we inspect the summary statistics of the numeric variables.

Code 2: Outlier Removal of Numeric Variables

```
#Reference:
https://blog.devgenius.io/data-quality-implementation-a-comprehensive-guid
e-243b6e8e4bda

# Exclude outliers per column for numeric variables

def exclude_outliers_iqr(df, numeric_cols):
   df_no_outliers = df.copy()
   for col in numeric_cols:
       Q1 = df[col].quantile(0.25)
       Q3 = df[col].quantile(0.75)
       IQR = Q3 - Q1
       lower_bound = Q1 - 3.0 * IQR
       upper_bound = Q3 + 3.0 * IQR
       df_no_outliers = df_no_outliers[
            (df_no_outliers[col] >= lower_bound) & (df_no_outliers[col] <=
upper_bound)
       ]
   return df_no_outliers
```

Summary Statistics

We performed sanity checks by inspecting the summary statistics of the data. For instance, the year column contains entries with a minimum of 2000 and a maximum of 2024, which makes sense since we selected variables from 2000 to 2024. Additionally, the height and age of winners is relatively the same as that of losers, which is expected. The average and median age of winners and losers is around 26. The minimum value of `winner_rank` and `loser_rank` is 1 and the maximum value of winners and losers is approximately the same. The median rank of winners is approximately 1.6 times smaller (better) compared to the median rank of losers. Moreover, the `draw_size` minimum value is 2 and the maximum value is 128, which is accurate as draw size ranges from 2 to 128 and is always a power of 2. More importantly, we observe that the average and median number of first-serve and second-serve points won by winners is greater than that of losers by a difference of 2–4 points. Similarly,

although closely the same, the average and median number of aces by winners is greater than that of losers by 1–2 points. The number of break points saved, double faults, total serve points, and first-serve points made in is similar between winners and losers. However, the number of break points faced by the loser is twice as large as that of winners on average and by the median. The maximum value for the duration of a match in minutes is 267 (~4.45 hours), the minimum is 0 minutes, the median is 99 minutes (~1.7 hours), and the average is 104 minutes (~1.7 hours), which seems reasonable. According to a New York Times article published in July 2023, the average length of a men's Grand Slam match is 2 hours and 54 minutes [6]. Our dataset includes different types of ATP tournaments, including Grand Slam tennis tournaments.

Patterns Between Categorical and Numeric Variables

Next, let's inspect the distribution of categorical and numeric variables. We observe that the distribution of tournament names is quite skewed, showing that most of the match data belongs to a select group of tournaments (**Figure 3A)**. This should not be an issue since match outcomes in a tournament are independent of match outcomes from other tournaments. As expected, the distribution of winners' and losers' handedness shows being right-handed is significantly the most frequent category, followed by being left-handed, and a few being unknown (**Figure 2A)**. The distribution of rounds is in the shape of a bell-curve, with R32 being the most common (the mode) (**Figure 7A)**.

Figure 1 below shows the duration of tennis matches do not differ significantly by surface type. The median duration of matches for all four surface types (carpet, clay, grass and hard) is approximately around 90 to 100 minutes.

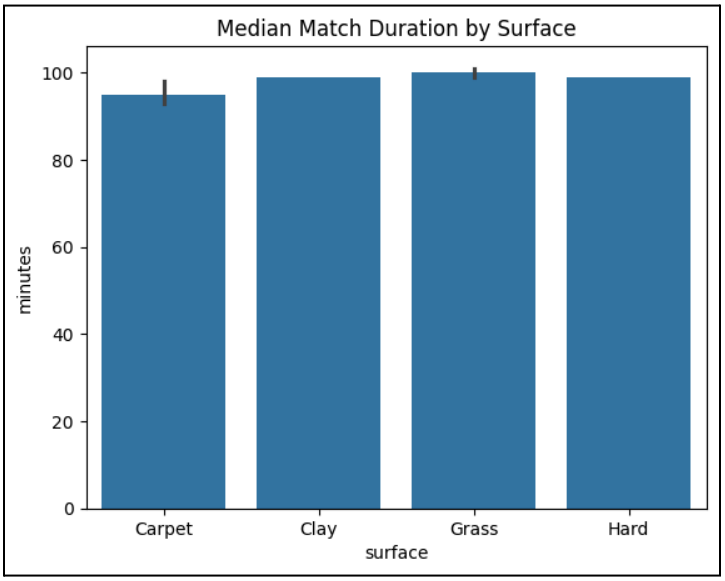


Figure 1: Median Match Duration by Surface Type

Additionally, **Figure 2** shows the same general trend for winners and losers regarding the median number of aces on different surface types. Overall, players have the highest median number of aces on grass, followed by carpet and hard, and last clay. Inspecting the distribution of surface shown by **Figure 1A**, we see that the most common surface to play tennis on is hard, followed by clay, then grass, and lastly carpet. This indicates that returning first-serves on grass might be difficult since the highest median number of aces is achieved on grass despite grass being fairly uncommon compared to tennis matches on hard and clay surface types.

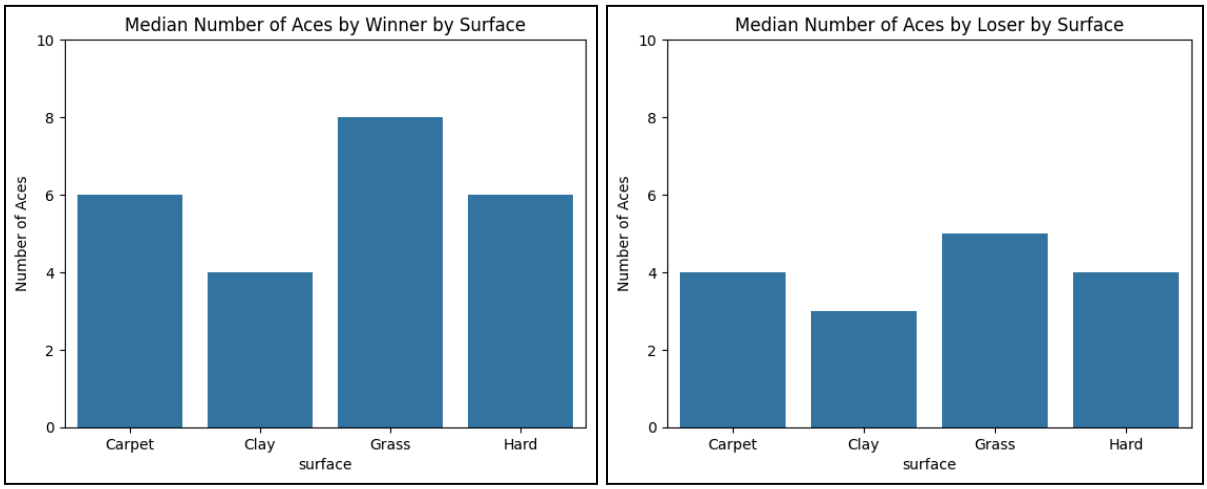


Figure 2: Median Number of Aces of Winner/Loser by Surface Type

Figure 3 below shows a slow increase in the median match duration as the year progresses from 2000 to 2024. From years 2000 to 2015, the median match duration is around 85 minutes. There is a spike in the median match duration around 2015 and a relatively constant increase from 2018 onwards.

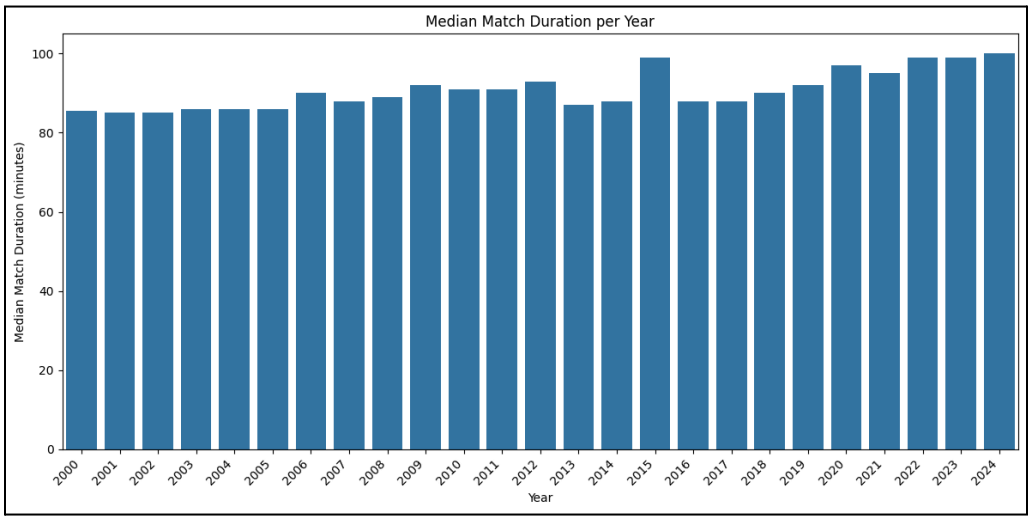


Figure 3: Median Match Duration per Year

Figure 4 below visualizes the median match duration by tournament level. Please see **Table 1** for the definition of letters along the x-axis. From this figure, we observe that there is not a significant difference in the median match duration among tournament levels, except for G which clearly has a higher duration. This pattern may be explained by the fact that the Grand Slams are considered the most prestigious and competitive tennis tournaments. These tournaments offer the most ranking points and attract the best players in the world, which can lead to longer matches.

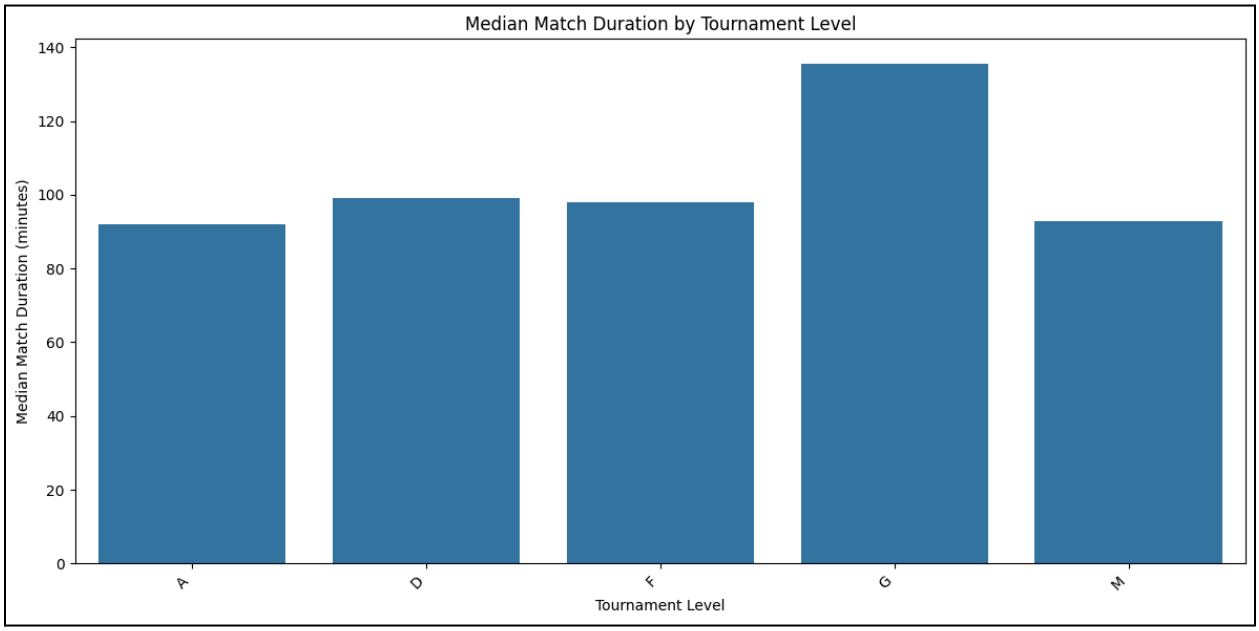


Figure 4: Median Match Duration by Tournament Level

Correlation between Numeric Variables (Outliers Removed)

Figure 5 below shows the correlation among the numeric variables of **Table 1**. Absolute values closer to 1 indicate higher correlation, suggestive of a positive or negative linear relationship between two variables. It is important to note that variables with low correlations (absolute values close to zero) might have non-linear relationships between them that are not expressed by the correlation matrix.

The duration of matches (minutes), number of serve points, number of first-serve/second-serve points made in as well as those won, and the number of break points saved or faced are all highly correlated with each other for the most part. The number of a player's aces is subtly correlated with their double faults, and moderately correlated with their number of first-serve points won. Year, height, age, rank, and draw size have low correlation with most other variables.

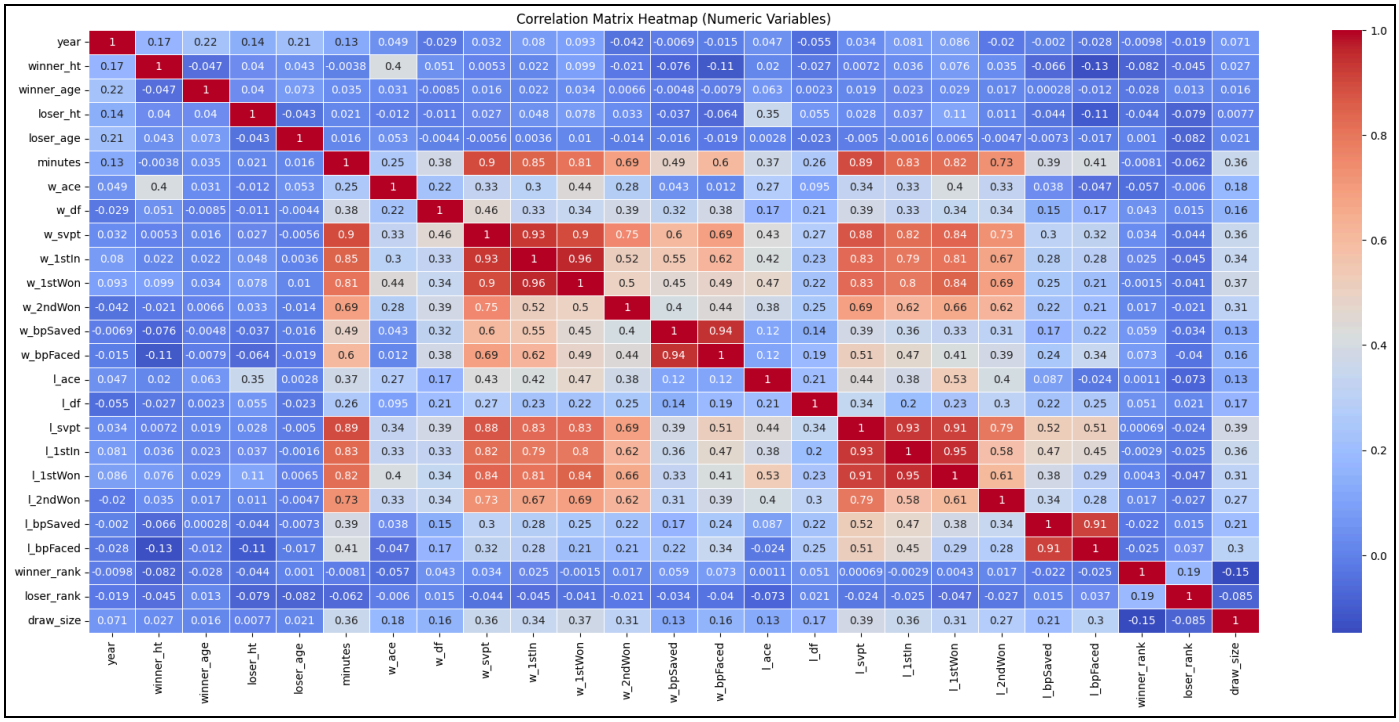


Figure 5: Correlation Matrix of Numeric Variables

Combined Variables as Predictors

Table 2 below describes the variables planned for use during model training and evaluation, including how certain variables are calculated by combining variables from **Table 1**. The final categorical variables were encoded using `labelEncoder` from the `sklearn.preprocessing` module.

Table 2: Predictors for Model Training and Prediction of Tennis Match Outcomes

Variable	Rationale	Calculation
year	Can take into account match variations through time	N/A
tourney_name	Players might be better at certain tournaments	N/A
tourney_level	Experience and skills of players differ by tournament level	N/A
minutes	Some players might play better than others in long	N/A

	matches	
surface	Some players are better on particular courts/surface types	N/A
draw_size	Needed to make sense of round	N/A
round	Players may play differently depending on the round	N/A
winner_hand, loser_hand	Players might have difficulty playing an opponent with the same/different handedness	N/A
winner_ht, loser_ht	Height provides a major advantage during serves	N/A
winner_age, loser_age	Younger players have better physical endurance and fewer injuries	N/A
winner_rank, loser_rank	Lower values correspond to top-players, which can reflect the quality of their tennis skills	N/A
w_ace/df, l_ace/df	Higher value means higher serving speed and accuracy	# aces / # double faults
w_1stInPerc, l_1stInPerc	Higher value represents greater first-serve accuracy	# first-serves in/ # serves made
w_1stWon/1stIn, l_1stWon/1stIn	Higher value indicates first-serve effectiveness and control over points initiated with first-serve	# first-serves won/ # valid first-serves
w_2ndInPerc, l_2ndInPerc	Higher value represents greater second-serve accuracy	# second-serves in/ # serves made
w_2ndWon/2ndIn, l_2ndWon/2ndIn	Higher value indicates second-serve effectiveness and control over points initiated with second-serve	# second-serves won/ # valid second-serves
w_bpSaved/bpFaced, l_bpSaved/bpFaced	Higher value indicates greater ability to defend critical	# break points saved/ # break points faced

	points under pressure as well as maintain serve stability	

Correlation of Numeric Predictors upon Combining Variables

Figure 6 below shows the correlation between the numeric features after combining variables from **Table 1** in order to minimize multiple collinearity issues. From the correlation matrix, observe the ratio of winners' and losers' second-serve points made in (`w_2ndInPerc` and `l_2ndInPerc`) is strongly negatively correlated with their ratio of first-serve points made in (`w_1stInPerc` and `l_1stInPerc`). This is expected, given that if a player has consistent first-serves, they will not need to play second-serves and thus the number of second-serve points decreases, and vice versa. Additionally, the ratio of first-serve points won out of first-serve points made in (`w_1stWon/1stIn` and `l_1stWon/1stIn`) is positively correlated with the ratio of aces over double faults (`w_ace/df` and `l_ace/df`) for both winners and losers. This makes sense since both metrics are indicative of the efficiency of players' first-serves. Another interesting observation is that the height of players (`winner_ht` and `loser_ht`) is positively correlated with the ratio of first-serve points won out of the first-serves made in (`w_1stWon/1stIn` and `l_1stWon/1stIn`), but not significantly with the ratio of second-serves won out of the second-serves made in (`w_2ndWon/2ndIn` and `l_2ndWon/2ndIn`). Moreover, the players' height is positively correlated with their ratio of aces over double faults, which reflects a serve advantage in being taller.

It is important to note that unlike serve data (`*_1stInPerc, *_2ndInPerc,*_ace/df, *_bpSaved/bpFaced`), rank has no moderate or substantial correlation with any other variable besides the rank of the opponent. This suggests that between rank and serve strength, serve strength might be the stronger predictor for tennis match outcomes. In this study, serve strength is measured by the percentage of valid first-serves (or second-serves), the ratio of aces over double faults, and the percentage of breakpoints saved. For all these metrics, the higher the value, the stronger the serve strength of the player.

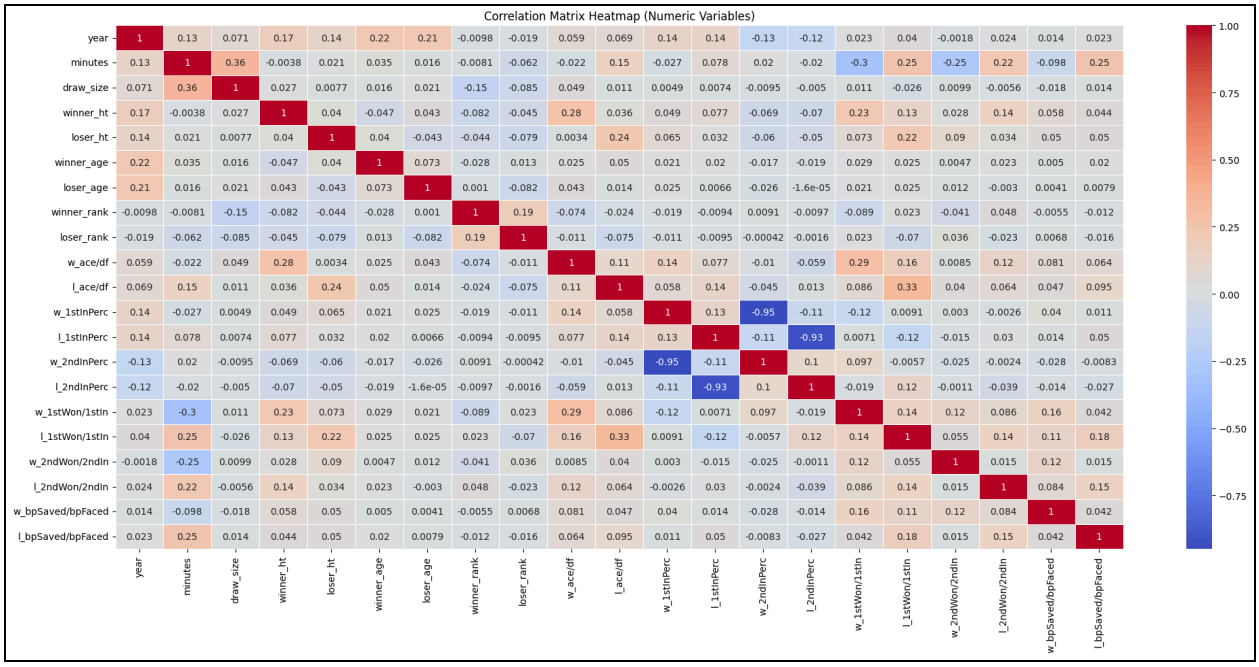


Figure 6: Correlation of Final Numeric Predictors

7. Methods

Models

XGBoost and Random Forest are the two models chosen to predict tennis match outcomes and perform feature analysis in this study. Both of these models are tree-based and display excellent performance in capturing complicated patterns within data [7]. Random Forest was selected as one of the models for the present study because it was used in the two main papers [1] [2] that inspired the problem and questions of this research. We include XGBoost due to its numerous advantages over Random Forest and to facilitate a comparison that evaluates the consistency of the obtained results and findings.

Random Forest is a bagging model that trains multiple trees in parallel. It samples the dataset on both rows and columns to create random subsets of the original dataset. The filtration procedure ensures that each feature makes it to at least one of the subsets. Individual decision trees are trained for each subset, and their outputs are collected. As observed in **Figure 7** below, the output class obtained from most trees becomes the final prediction [7].

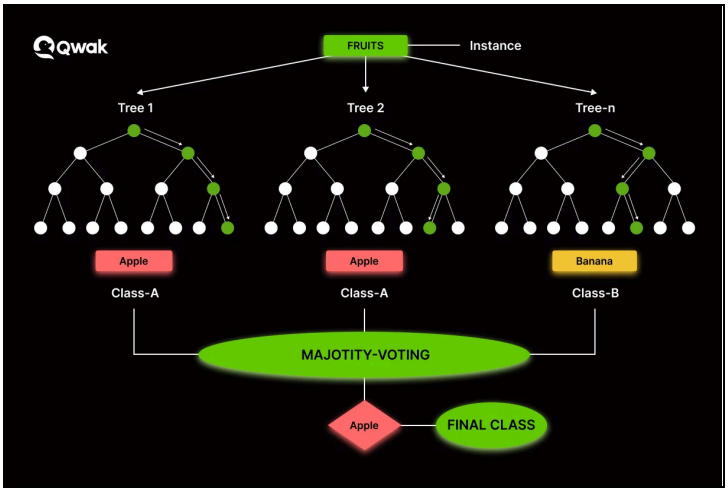


Figure 7: Visualization of Random Forest [7]

On the other hand, instead of training decision trees in parallel, XGBoost creates a sequential ensemble of tree models, all of which work to improve each other and determine the final output (as shown by **Figure 8**). The XGBoost algorithm employs advanced regularization techniques to suppress weights, mitigate overfitting, and enhance performance in practical applications. Additionally, its implementation supports data caching and multicore processing, enabling faster computations. These features have established XGBoost as one of the most widely adopted machine learning algorithms in recent years [7].

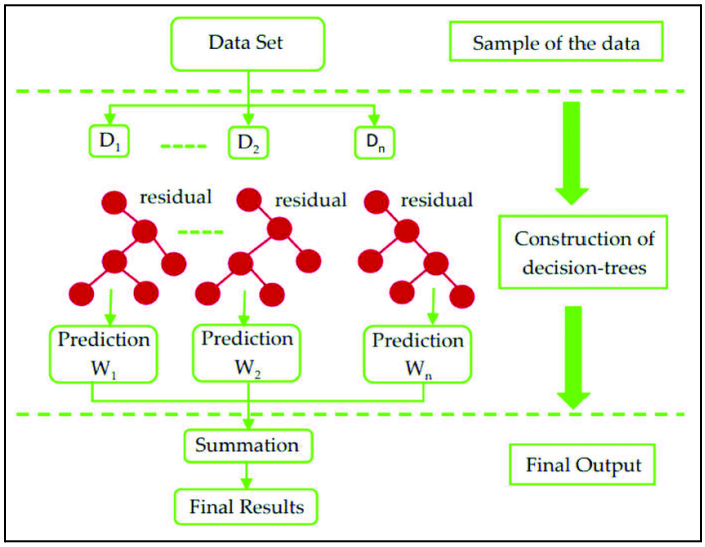


Figure 8: Structure of an XGBoost model [8]

Random Forest became popular due to its high accuracy, but this comes at the cost of overfitting when sampled data contains similar data points to each tree. XGBoost mitigates the risk of overfitting by pruning trees when it considers the gain obtained from building further nodes to be minimal. In addition to this, XGBoost performs better than Random Forest when we have a class imbalance since its algorithm iteratively learns from the mistakes of previous trees.

For instance, if a tree fails to predict a class (such as the imbalanced class), the processing tree will give more weight or importance to this sample. In contrast, Random Forest does not have any such mechanism to handle data imbalance. Lastly, another benefit of XGBoost over Random Forest is that the hyperparameters for XGBoost are set only for the first tree; the rest of the trees adjust themselves with every iteration using the loss of the preceding tree and carrying out gradient descent. This dynamic adaptability allows for better predictive performance on data that displays high variation, unlike for Random Forest which has fixed parameters for the entire ensemble [7].

Training Model Configuration

XGBoost and Random Forest classification models were trained with the variables outlined in **Table 2** of **Section 6** (Exploratory Data Analysis). We used the `train_test_split` function from the `sklearn.model_selection` module to randomly split our data into training and test subsets, allocating 80% of the data for training and 20% for testing. The default parameters for XGBoost and Random Forest were used for training, except for `random_state` = 42 for reproducibility. Default parameters are a practical starting point because they are designed to work reasonably well for general cases. The default parameters for XGBoost include: `booster`= 'gbtree' (tree-based models), `eta` = 0.3 (learning rate, or step size for each boosting step), `n_estimators` = 100 (number of boosting rounds, or trees), `max_depth` = 6 (maximum depth of each tree), `objective` = binary:logistic (logistic regression for binary classification), `eval_metric`= 'logloss' (evaluation metric for binary classification) [12]. The default parameters for Random Forest include: `n_estimators` = 100, `max_depth` = None (nodes are expanded until all leaves are pure – meaning they contain samples from a single class – or until they contain fewer than min_samples_split samples), `min_samples_split` = 2 (the minimum number of samples required to split an internal node), and `criterion` = 'gini'[13]. Gini, or Gini impurity, is a measurement of the likelihood of incorrect classification when a randomly selected data point is assigned a class label based on the distribution of classes in a particular node. It ranges from 0 to 0.5, where 0 indicates a perfectly pure node (all instances belong to the same class) and 0.5 signifies maximum impurity (an equal distribution of classes). Decision trees use this criterion to select the optimal split by identifying features that result in more homogenous subsets of data [14].

Note: See **Appendix** for code implementation of training, prediction and evaluation of the XGBoost and Random Forest models.

Evaluation Metrics

Since the current task at hand is a classification problem, where given player and match statistics we predict the outcome of a tennis match (output which player is the winner), we employ various classification evaluation techniques to assess the quality of classification

obtained from the two models. The evaluation metrics include: **accuracy, ROC-AUC, precision, recall, F1 score, and specificity.** Besides this, we plot a heatmap of the confusion matrix to visualize false positives and false negatives.

Accuracy is the proportion of all classifications that were correct, whether positive or negative. Mathematically, it is defined as in **Formula 1**, where TP = true positives, TN = true negatives, FP = false positives, FN = false negatives. A perfect model would have zero false positives and zero false negatives, resulting in an accuracy of 1.0, or 100% [9].

$$\text{Accuracy} = \frac{\text{correct classifications}}{\text{total classifications}} = \frac{TP + TN}{TP + TN + FP + FN}$$

Formula 1: Accuracy [9]

The ROC curve is a visual representation of model performance with respect to the true positive rate (TPR) and false positive rate (FPR). The area under the ROC curve (AUC) represents the probability that the model, given a randomly chosen positive and negative example, will rank the positive higher than the negative. For a binary classifier, a model that does exactly as well as random guesses or coin flips has a ROC that is a diagonal line from (0,0) to (1,1), with AUC of 0.5 representing a 50% probability of correctly ranking a random positive or negative example. The model with greater area under the curve is generally the better one (see **Figure 9**) since this indicates there are more true positives than false positives [10].

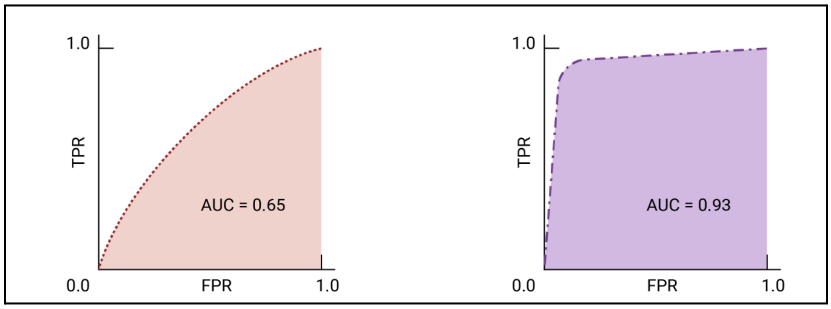


Figure 9: ROC and AUC of two hypothetical models. The curve on the right, with a greater AUC, represents the better of the two models [10]

Precision is the proportion of all the model's positive classifications that are actually positive. A hypothetical perfect model would have zero false positives and therefore a precision of 1.0 (or 100%) [9].

$$\text{Precision} = \frac{\text{correctly classified actual positives}}{\text{everything classified as positive}} = \frac{TP}{TP + FP}$$

Formula 2: Precision [9]

Recall (sensitivity or true positive rate) is the proportion of all actual positives that were classified correctly as positives. A hypothetical perfect model would have zero false negatives and therefore have a recall (TPR) of 1.0 (100% detection rate) [9].

$$\text{Recall (or TPR)} = \frac{\text{correctly classified actual positives}}{\text{all actual positives}} = \frac{TP}{TP + FN}$$

Formula 3: Recall [9]

Similar to recall, specificity is the proportion of all actual negatives that were classified correctly as negatives. Specificity represents the accuracy of the model's negative rate, while recall represents the accuracy of the model's positive rate [11].

$$\textit{Specificity} = \frac{\text{TN}}{\text{TN + FP}}$$

Formula 4: Specificity

Lastly, the F1 score is the harmonic mean (a kind of average) of precision and recall, where the metric balances the importance of these two metrics. The F1 score is preferred over accuracy when dealing with imbalanced datasets. When precision and recall both have perfect scores of 1.0, the F1 score will also have a perfect score of 1.0. When precision and recall are far apart, the F1 score will be similar to whichever metric is worse [9].

$$\text{F1} = 2 * \frac{\text{precision * recall}}{\text{precision + recall}} = \frac{\text{2TP}}{\text{2TP + FP + FN}}$$

Formula 5: F1 Score [9]

8. Analysis of Results

For both XGBoost and Random Forest models, we trained the model first with all predictors (as defined in **Table 2**), then with only rank, next with only serve-related features, and finally with rank and serve–related features alone. Regarding serve-related features, the percentage of points won after a valid first or second serve is excluded, as it heavily depends on factors beyond the serve itself. Conversely, the percentage of break points saved is included, as it strongly measures serve strength by reflecting serve consistency under high-pressure situations.

Model performance with all features

XGBoost obtained a test accuracy of 0.94, ROC-AUC of 0.99, precision of 0.93, recall of 0.94, F1 score of 0.93, and specificity of 0.94. Overall, XGBoost shows strong performance across all metrics. In the context of the confusion matrix shown below (**Figure 10**), along the first column we have the true negative (first row) and false negative (second row). Along the second column, we have false positive (first row) and true positive (second row). A true positive means that the model correctly predicted Player 1 as the winner, while a false positive means the model incorrectly predicted Player 1 as the winner when Player 0 actually won. On the other hand, a true negative means the model correctly predicted Player 0 as the winner, while a false negative means the model incorrectly predicted Player 0 as the winner when Player 1 actually won. **Figure 10** shows there are substantially more true positives and true negatives than false positives or negatives, which signifies XGBoost had a strong classification performance. Upon inspection of the feature importance output by the model (**Figure 11**), we see the most important predictors are: the players' percentage of winning a point having had a valid first serve, players' percentage of saving a break point, and percentage of winning a point having had a valid second serve. Although with a lower gain, these top predictors are followed by the players' percentage of valid first serves and players' ratio of aces over double faults. This aligns with the results found by the study performed by Gao and Kowalczyk [2], as they note that removing serve-related features caused a decrease in accuracy and therefore were included as significant features; these features included: ratio of aces over double faults, percentages of valid first serves, and percentages of points won having had valid first and second serves. However, while we obtained 'percentage of break points saved' as a top predictor, their study did not, as when this feature was removed they found their model's accuracy actually increased.

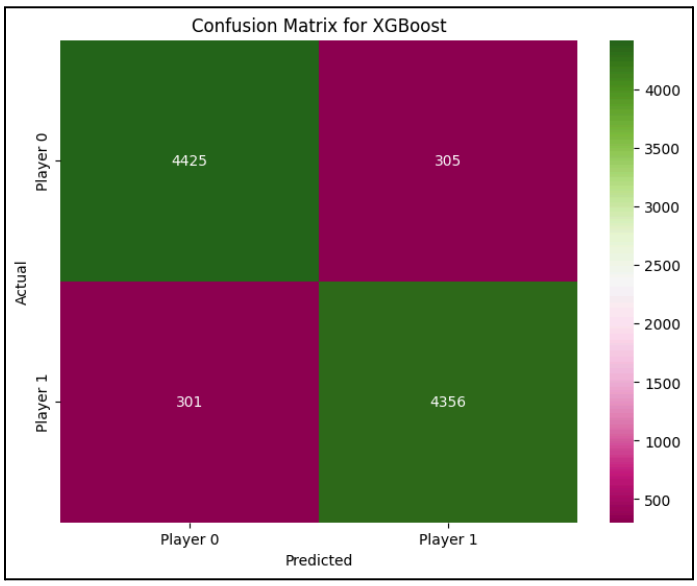


Figure 10: Confusion Matrix for XGBoost Trained with All Features

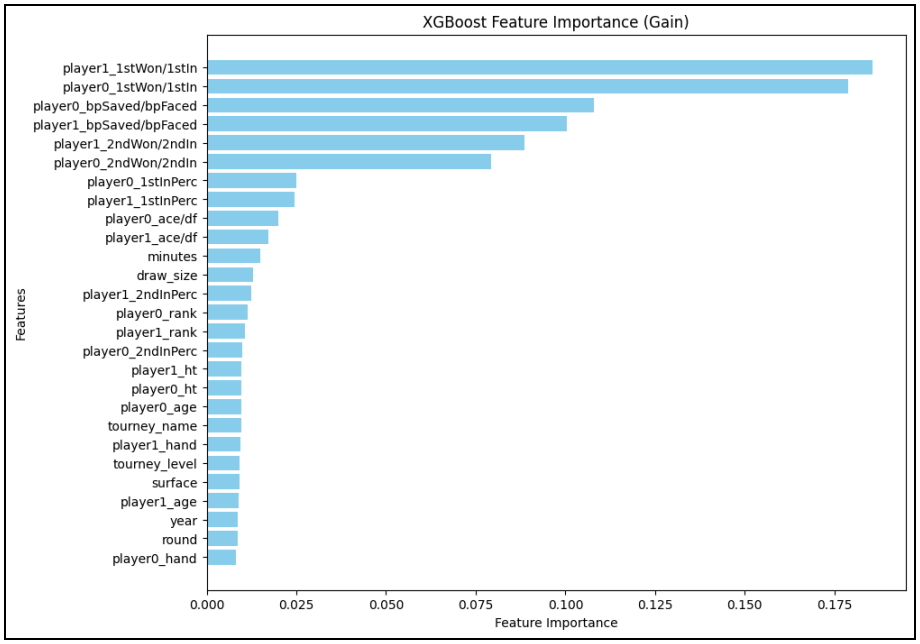


Figure 11: Feature Importance for XGBoost using All Features

Now, let's examine the performance of Random Forest trained with all features from **Table 2**. Random Forest obtained a test accuracy of 0.93, ROC-AUC of 0.98, precision of 0.93, recall of 0.92, F1 score of 0.93, and specificity of 0.94. All metric values indicate strong classification performance. The confusion matrix heatmap shown in **Figure 12** illustrates the strong classification performance of the model, as there are substantially more true predictions than false. The feature importance plot in **Figure 13** shows the same top three most important features as XGBoost (**Figure 11**), although not in the same order or magnitude; these features are: players' percentage of points won having had a valid first serve, players' percentage of points won having had a valid second serve, and players' percentage of break points saved. Like XGBoost (**Figure 11**), the next most significant feature (but with noticeably lower importance) are the players' ratio of aces over double faults. In contrast to **Figure 11** which only had player 0's percentage of valid second serves below the players' rank, the percentages of valid first and second serves for both players are below the players' rank in **Figure 13**.

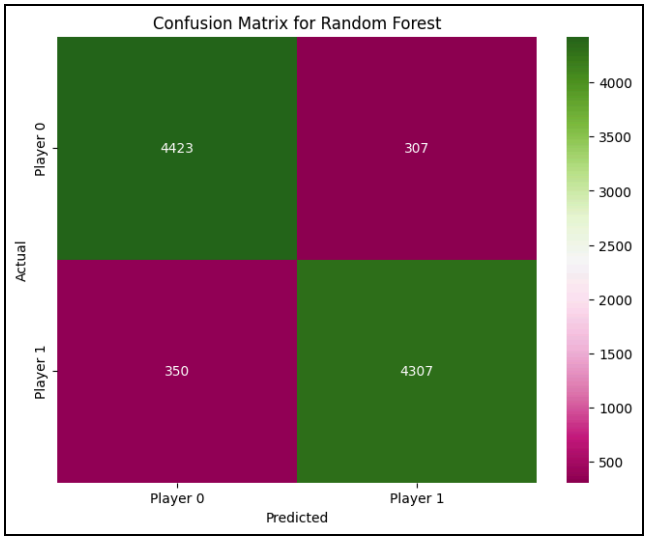


Figure 12: Confusion Matrix for Random Forest Trained with All Features

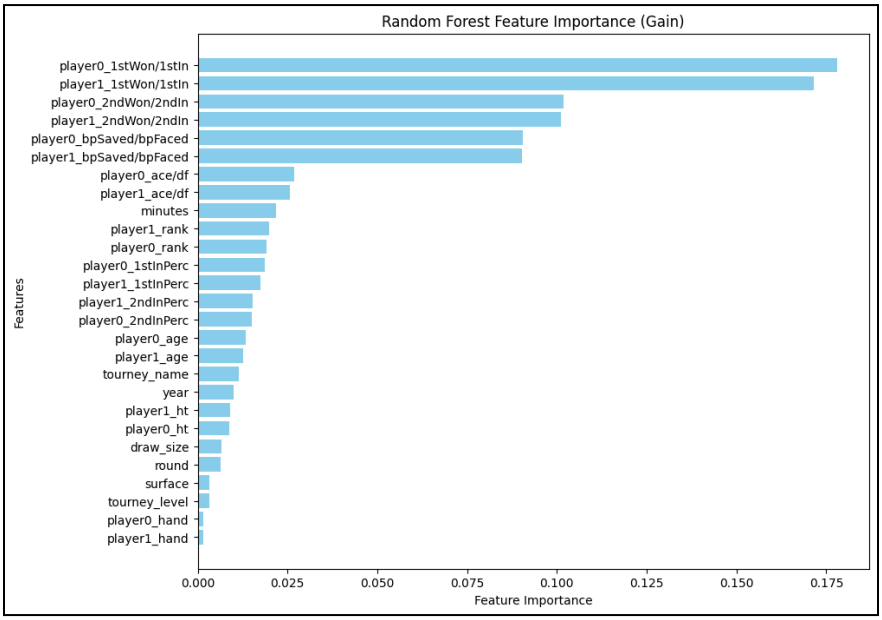


Figure 13: Feature Importance for Random Forest using All Features

Model performance with only rank

Let's examine the performance of XGBoost after training with only rank. The model obtained an accuracy of 0.63, ROC-AUC of 0.69, precision of 0.63, recall of 0.62, F1 score of 0.63, and specificity of 0.65. As observed, the model's performance is much worse when training with only rank. The study performed by Wilkens [1] used a straightforward baseline model which predicted the player with the highest ranking (the favorite) as the winner, thus only accounting for rank as a predictor. This model obtained an accuracy and precision of 0.65, similar to the value obtained by XGBoost for accuracy and precision, 0.63. **Figure 14** highlights

an increase in false negative and false positive predictions, illustrating a decline in classification performance.

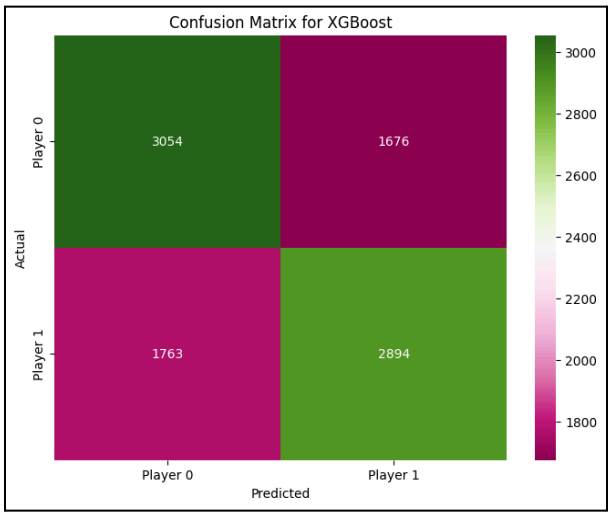


Figure 14: Confusion Matrix for XGBoost Trained with Rank

Next, let's evaluate the performance of Random Forest after training with rank as the sole predictor. The model obtained accuracy of 0.59, ROC-AUC of 0.63, precision of 0.59, recall of 0.58, F1 score of 0.58, and specificity of 0.60. Although slightly lower, these metric values are closely the same as the values obtained with XGBoost. Therefore, we conclude Random Forest also performs poorly when trained only with rank. The confusion matrix shown in **Figure 15** depicts this diminished classification quality, as the number of true predictions is comparable to the number of false predictions made by the model.

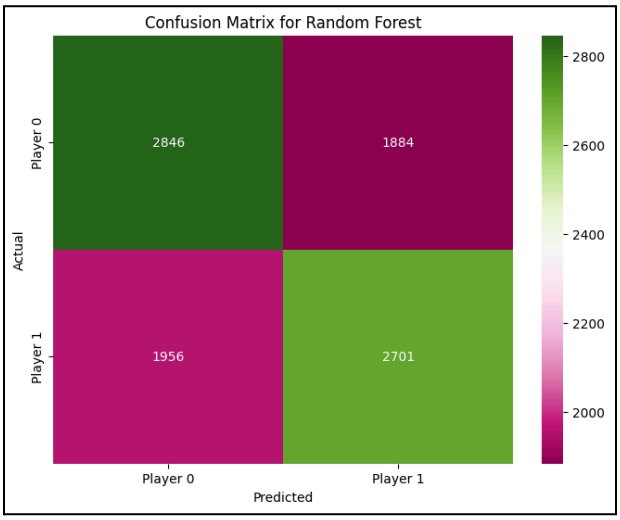


Figure 15: Confusion Matrix for Random Forest Trained with Rank

Model performance with only serve-related features

Unlike rank, training the XGBoost model with only serve-related features resulted in relatively strong classification performance comparable to when it was trained with all features. Having trained with only serve-related features, the XGBoost model obtained an accuracy of 0.84, ROC-AUC of 0.92, precision of 0.84, recall of 0.84, F1 score of 0.84, and specificity of 0.84. In the study performed by Gao and Kowalczyk [2], their Random Forest model obtained an accuracy of 0.83 (the highest out of any other model they employed), which is closely the same as the accuracy obtained by XGBoost in this study. Additionally, looking at the confusion matrix shown by **Figure 16**, we see that there are significantly more true negative and positive predictions compared to false negative and positive. Inspecting the feature importance plot in **Figure 17**, we see that the most influential serve-related feature was the players' percentage of break points saved. The latter is followed by the players' ratio of aces over double faults, then by players' percentage of valid second serves, and lastly the percentage of players' valid first serves. This supports the findings of the studies performed by Yu and Wang [4] and Gao and Kowalczyk [2] which found that features representative of players' serve strength are the most important in accurately predicting the outcomes of tennis matches. These results contradict the findings of studies performed by Yue et al. [3] and Wilkens [1] which found a player's rank to be the most significant predictor.

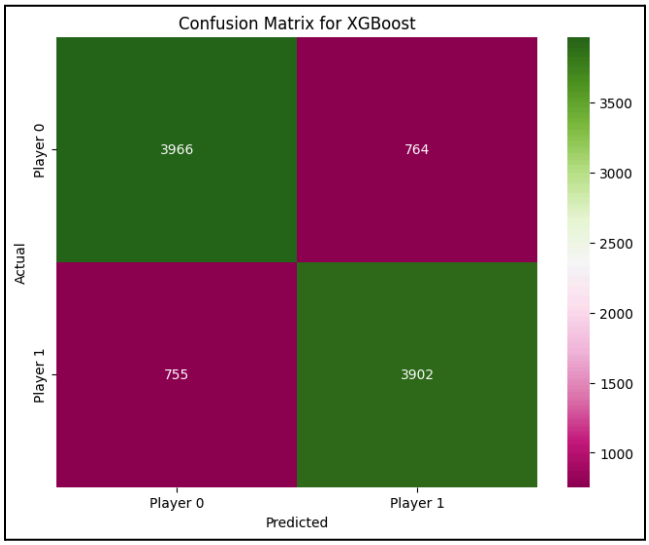


Figure 16: Confusion Matrix for XGBoost Trained with Serve-Related Features

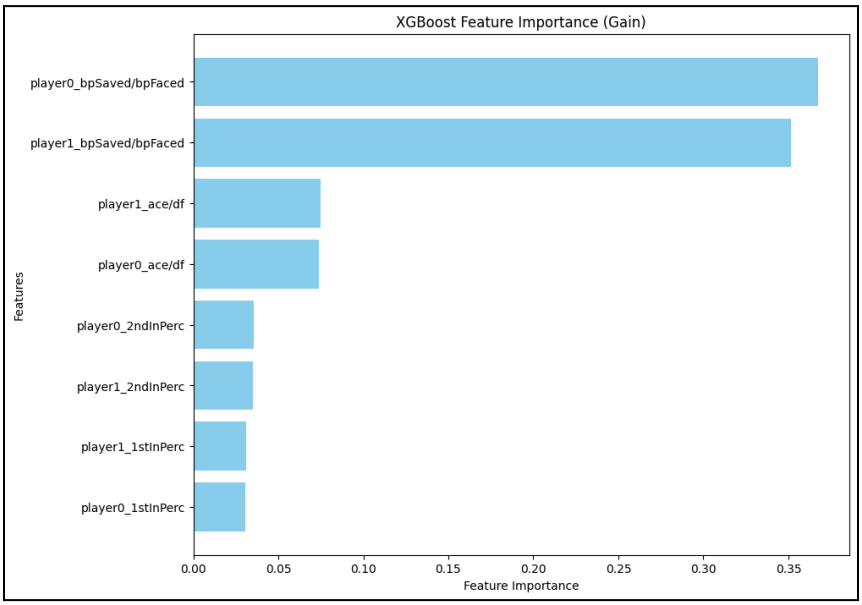


Figure 17: Feature Importance for XGBoost using Serve-Related Features

Similar to the findings with the XGBoost classifier, the Random Forest model performed much better when trained with only serve-related features compared to only rank. The Random Forest classifier received a test accuracy of 0.81, ROC-AUC of 0.90, precision of 0.81, recall of 0.80, F1 score of 0.80 and specificity of 0.81. Additionally, the confusion matrix in **Figure 18** shows significantly higher true predictions (green) compared to false predictions (pink). These metric values are comparable to those obtained from the model when trained with all features, indicating that most of the information needed for accurate predictions is retained by the serve-related features. The feature importance plot in **Figure 19** shows the players' percentage of break points saved are the most important among the serve-related features used. The feature importance of serve-related features is the same as the one obtained for XGBoost in **Figure 17**, except that Random Forest found the percentage of valid first serves to be more influential than that of second serves. Like XGBoost, these results support the findings of the studies performed by Yu and Wang [4] and Gao and Kowalczyk [2] in their conclusion that serve-related data is most important when predicting the outcome of tennis matches, and not rank as found in the studies by Yue et al. [3] and Wilkens [1].

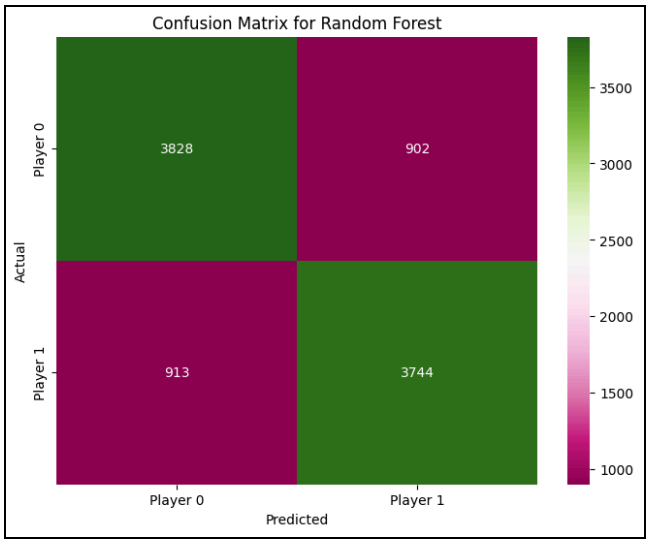


Figure 18: Confusion Matrix for Random Forest Trained with Serve-Related Features

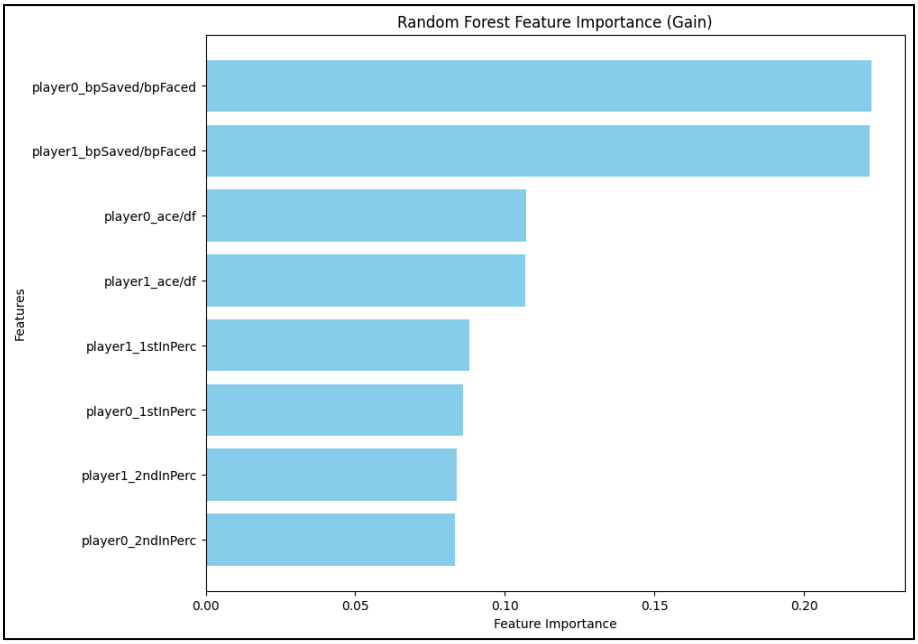


Figure 19: Feature Importance for Random Forest using Serve-Related Features

Model performance with rank and serve-related features only

To investigate the interaction between rank and serve-related features in accurately predicting tennis match outcomes, we trained and evaluated the performance of XGBoost and Random Forest using only rank and serve-related features as predictors.

With XGBoost, we obtained an accuracy of 0.85, an ROC-AUC of 0.93, precision of 0.85, recall of 0.85, F1 score of 0.85 and specificity of 0.85. Similarly, Random Forest obtained an accuracy of 0.82, ROC-AUC of 0.91, precision of 0.82, recall of 0.82, F1 score of 0.82, and specificity of 0.82. While these metrics indicate strong classification performance, they do not

reveal a substantial improvement over models trained solely on serve-related features. This suggests that serve-related features remain the most significant predictor of tennis match outcomes.

9. Conclusion

In this study, XGBoost and Random Forest were trained using match and player statistics to predict outcomes of men's singles matches on the ATP circuit from 2000 to 2024. Both models were evaluated with all features outlined in **Table 2**, with only rank, and with only serve-related features (`*_1stInPerc`, `*_2ndInPerc`,`*_ace/df`, `*_bpSaved/bpFaced`). When trained with all features, both models displayed strong classification performance, with evaluation metrics (accuracy, ROC-AUC, precision, recall, F1 score, and specificity) over 90%. When trained with only rank, the models' performance drastically reduced, with evaluation metrics being around 60-65%. Unlike rank, when trained with only serve-related features, the models performed closely the same as when trained with all features, with metric values being roughly 80-85%. The evaluation metrics did not significantly improve for either model when rank and serve-related features were both used as predictors, with metric values rising by a maximum of 2% compared to the model's metric values using only serve-related features.

The results obtained in this study support the findings of the research investigations performed by Yu and Wang [4] and Gao and Kowalczyk [2], where they conclude that features measuring the serve strength of players are most important in accurately predicting the outcome of tennis matches. In turn, the outcome of this study contradicts the conclusion made by Yue et al. [3] and Wilkens [1] that a player's rank is the most significant predictor. Furthermore, Wilkens [1] found that the various machine learning models employed for the classification task had a prediction accuracy typically not exceeding 70%. Among the models used, they included random forest and gradient boosting machine. In the present study, Random Forest obtained an accuracy of 0.93 (with all features) and 0.81 (with serve-related features), values which exceed 70%. Additionally, XGBoost – which is built on the same principles of traditional gradient boosting – also obtained accuracies exceeding 70% when trained with all features or with only serve-related features.

This research provides valuable insights for stakeholders in the betting industry, as well as players and coaches, by identifying the key factors that influence men's singles tennis match outcomes on the ATP circuit through data-driven analysis. Such insights enable informed decision-making, such as selecting players to bet for or against or designing training programs that prioritize the development of key skills for athletes. In the context of this research's findings, bettors can enhance the accuracy of their predictions regarding match outcomes by evaluating players' serve strength. Likewise, aspiring and professional tennis players can enhance their performance and boost their win rates by fortifying their serve strength, such as by maximizing serve consistency in high-pressure situations such as break points (one of the key predictive factors found in this study).

A lot of interesting questions and exploration remains in the field of tennis match predictions. Future work can explore the influence of time, such as considering trends and variations in player performance across seasons. Such work would allow examination of the progression of serve strength's influence on match outcomes across different player generations and can reveal how the sport's dynamics evolve. Furthermore, a comparative analysis of the impact of serve-related features on doubles versus singles matches could evaluate the generalizability of these findings and offer more targeted insights for players based on their specialization. It is also of interest to inspect the predictive power of rank and serve-related features on men's singles matches versus women's singles matches and assess if it differs.

References

[1] Wilkens, S. (2021). Sports prediction and betting models in the machine learning age: The case of tennis. *Journal of Sports Analytics*, 7(2), 99-117. https://doi.org/10.3233/JSA-200463

[2] Gao, Z., & Kowalczyk, A. (2021). Random forest model identifies serve strength as a key predictor of tennis match outcome. *Journal of Sports Analytics*, 7(4), 255-262. https://doi.org/10.3233/JSA-200515

[3] Yue, J. C., Chou, E. P., Hsieh, M.-H., & Hsiao, L.-C. (2022). A study of forecasting tennis matches via the GLICKO model. *PLOS ONE*, *17*(4). https://doi.org/10.1371/journal.pone.0266838

[4] Yu, L., & Wang, Y. (2023). Analysis of tennis techniques and tactics based on multiple linear regression model. *Applied Mathematics and Nonlinear Sciences*, *8*(2), 2061–2068. https://doi.org/10.2478/amns.2023.1.00310

[5] Tennis rankings: Everything you need to know. (n.d.). https://olympics.com/en/news/tennis-rankings-atp-wta-men-women-doubles-singles-system-grand-slam-olympics

[6] Eccleshare, C. (n.d.). *Men's Grand-slam matches are 25% longer than in 1999. does something need to change?* The New York Times. http://www.nytimes.com/athletic/4651272/2023/07/01/mens-grand-slam-matches-five-sets-three-sets/

[7] Lev, A. (n.d.). *XGBoost versus Random Forest*. JFrog ML. http://www.qwak.com/post/xgboost-versus-random-forest#xgboost-vs-random-forest-how-to-choose

[8] Jan, M. S., Hussain, S., e Zahra, R., Emad, M. Z., Khan, N. M., Rehman, Z. U., Cao, K., Alarifi, S. S., Raza, S., Sherin, S., & Salman, M. (2023). Appraisal of different artificial intelligence techniques for the prediction of marble strength. *Sustainability*, *15*(11), 8835. https://doi.org/10.3390/su15118835

[9] Google. (n.d.). *Classification: Accuracy, recall, precision, and related metrics*. Machine Learning Crash Course.

https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall
[10] Google. (n.d.). *Classification: Accuracy, recall, precision, and related metrics*. Machine Learning Crash Course. https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc
[11] Thieme, C. (2021, June 16). *Understanding Precision, Sensitivity, and Specificity In Classification Modeling*. Medium. https://towardsdatascience.com/understanding-common-classification-metrics-titanic-style-8b8a562d3e32
[12] *The defaults for XGBClassifier*. Kaggle. (n.d.). https://www.kaggle.com/discussions/general/237096
[13] *Random Forest Classifier*. scikit-learn. (n.d.). https://scikit-learn.org/1.5/modules/generated/sklearn.ensemble.RandomForestClassifier.html
[14] Singh, H. (2024, November 18). *Splitting decision trees with Gini impurity*. Analytics Vidhya. https://www.analyticsvidhya.com/articles/gini-impurity/

Appendix

Figure 1A: Distribution of Matches by Surface Type

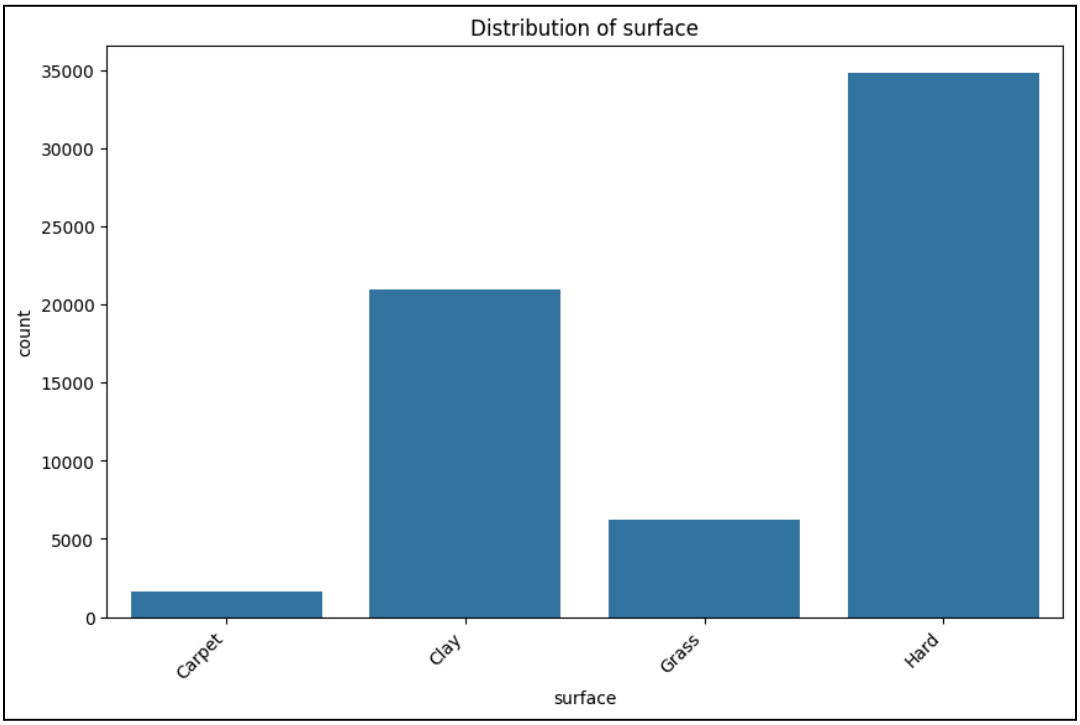
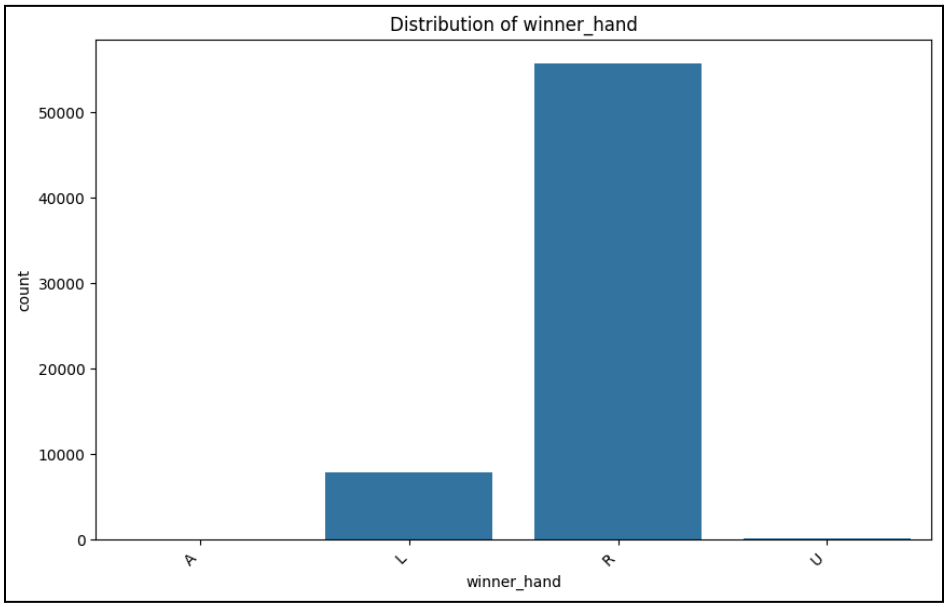


Figure 2A: Distribution of Handedness of Winners and Losers



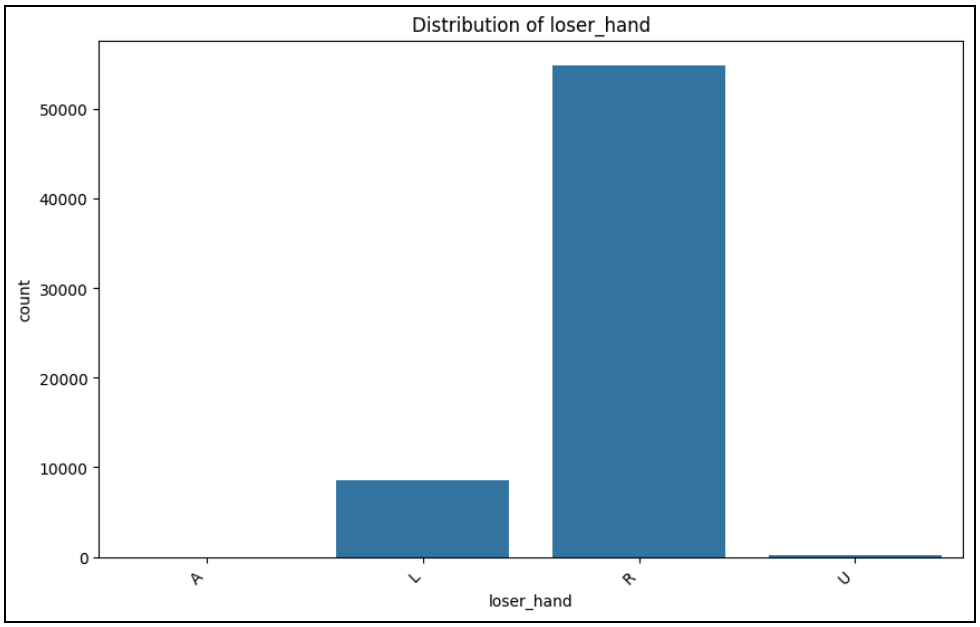


Figure 3A: Distribution of Tournaments

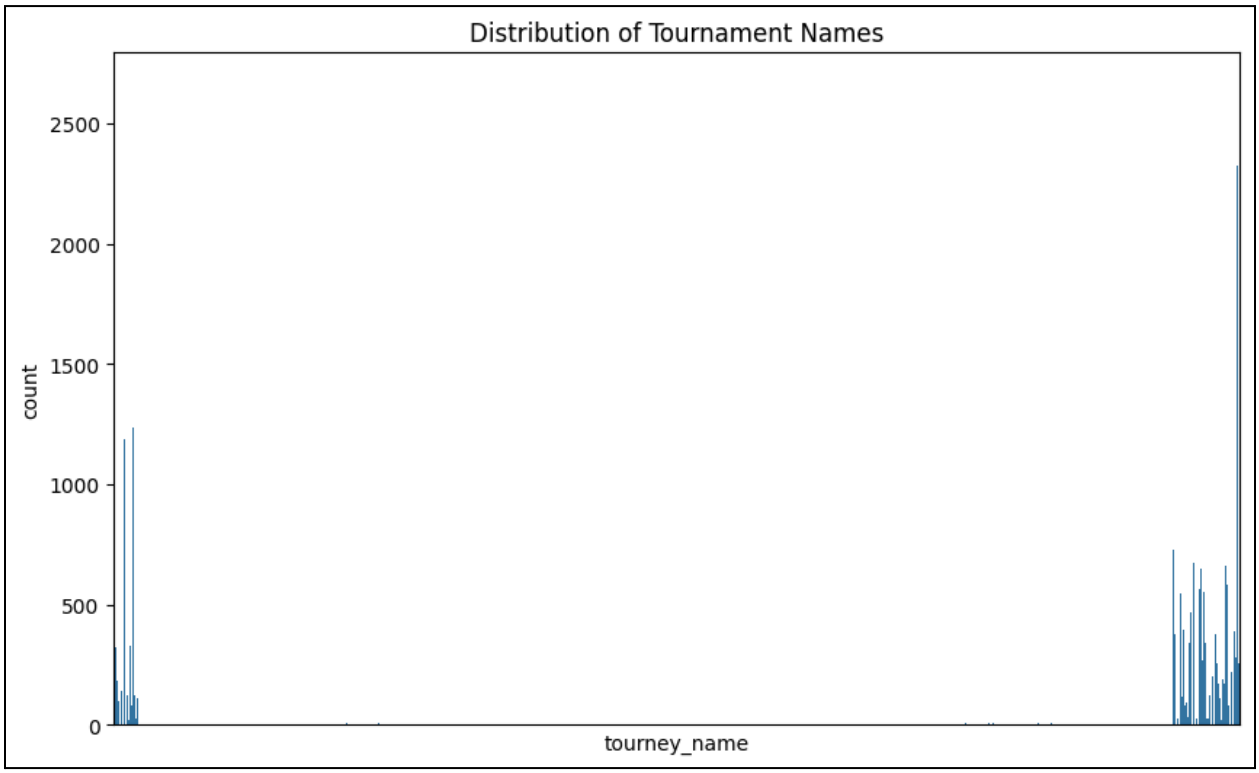


Figure 4A: Distribution of Matches by Year

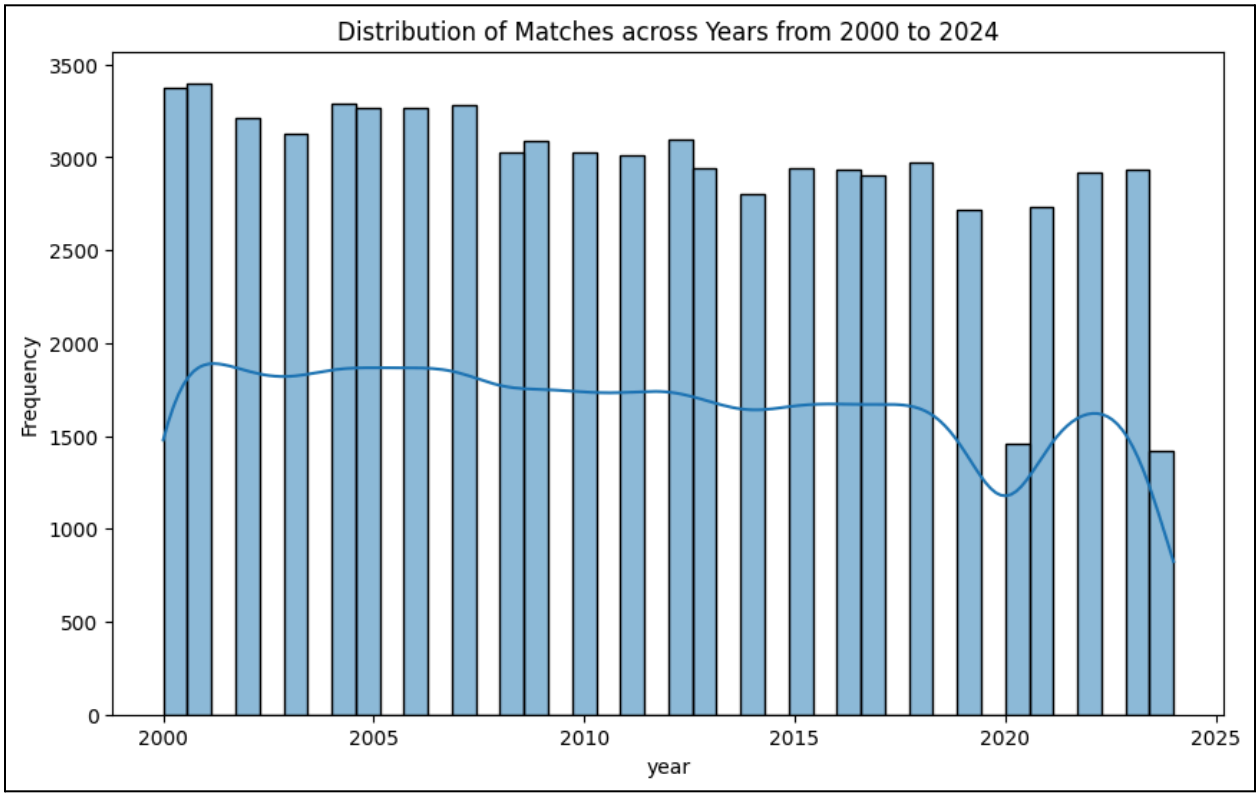
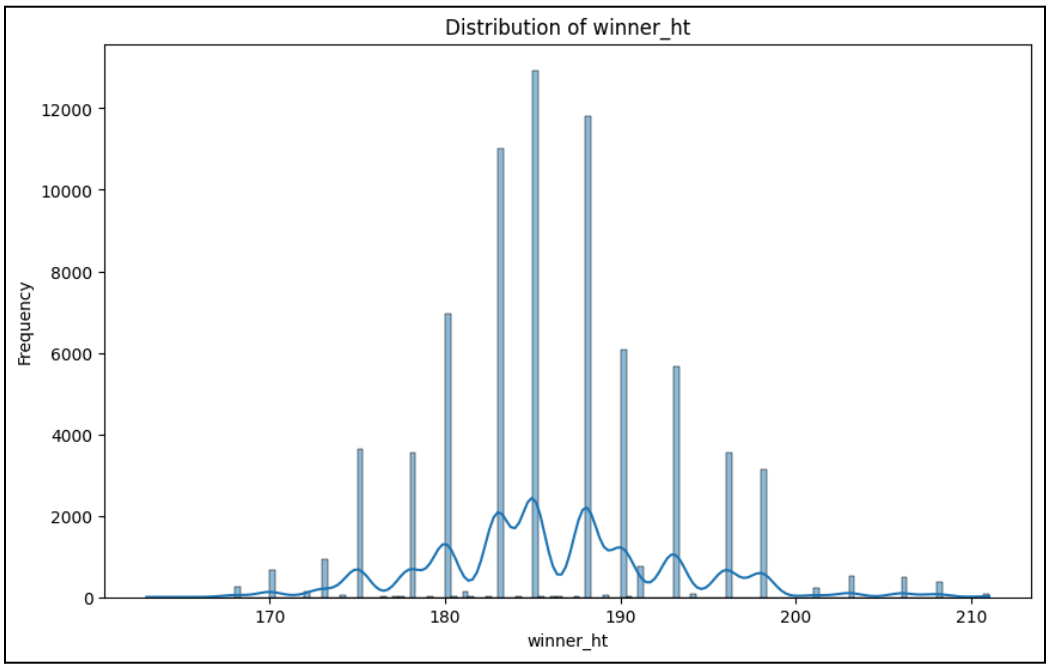


Figure 5A: Distribution of Heights of Winners and Losers



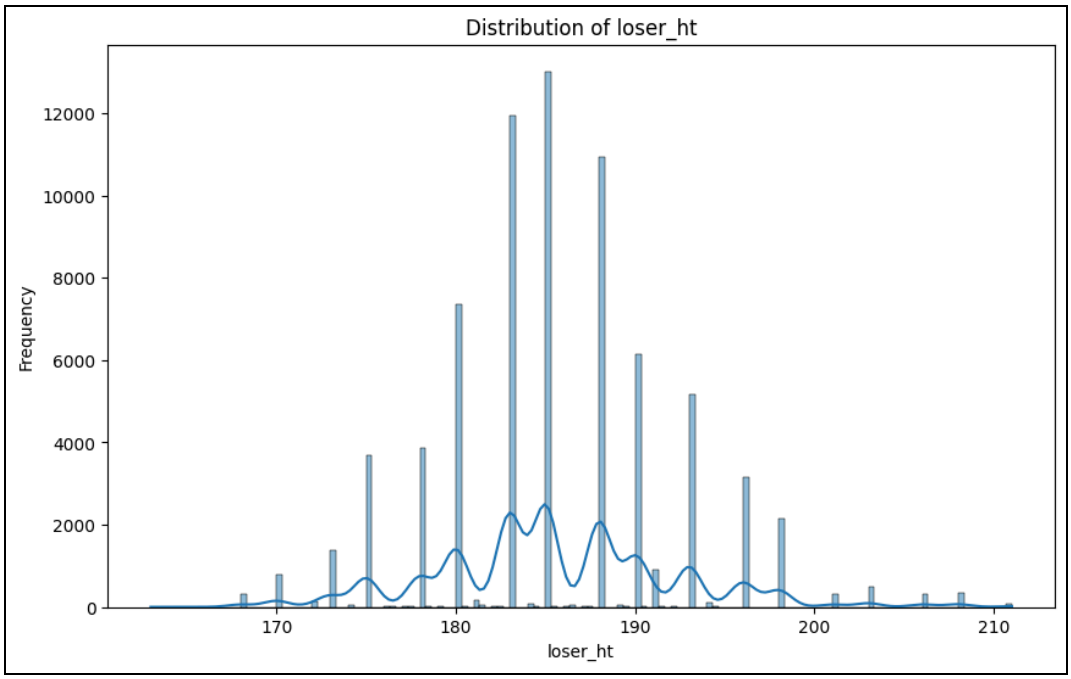


Figure 6A: Distribution of Match Duration in Minutes

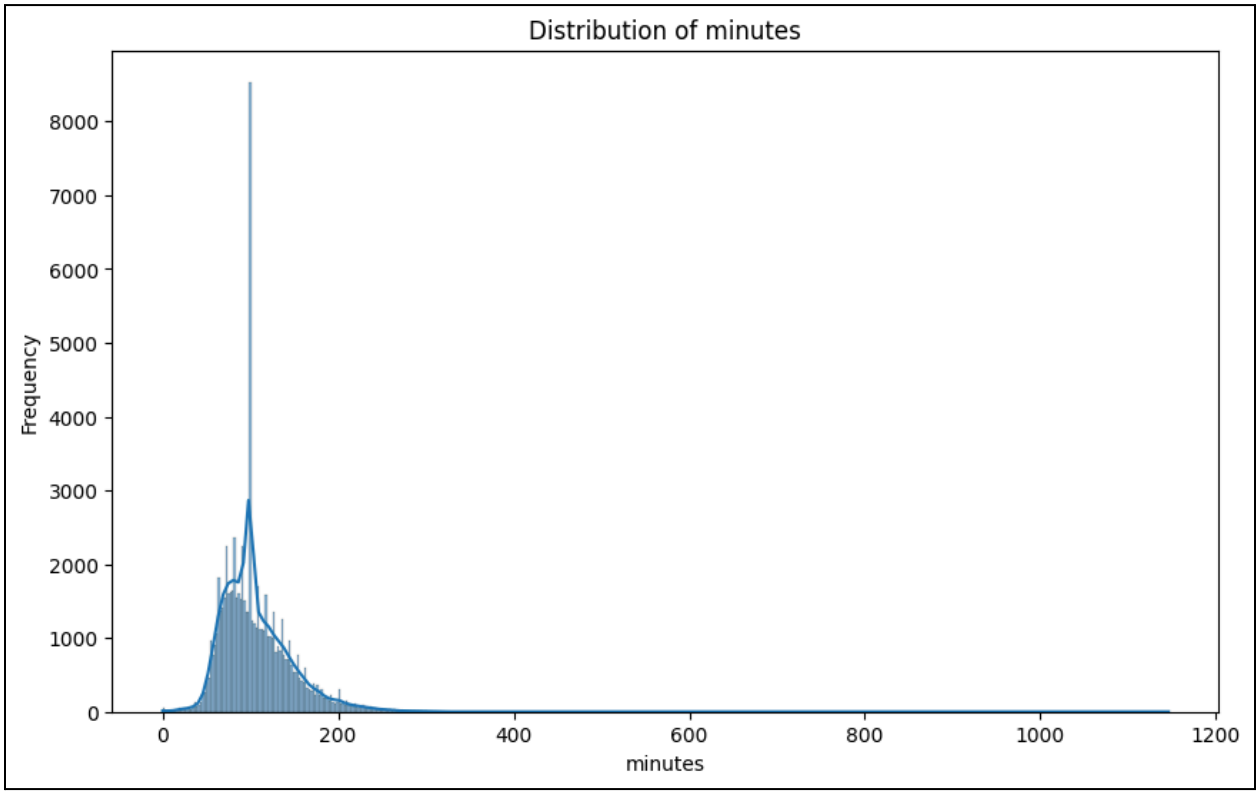


Figure 7A: Distribution of Matches by Round

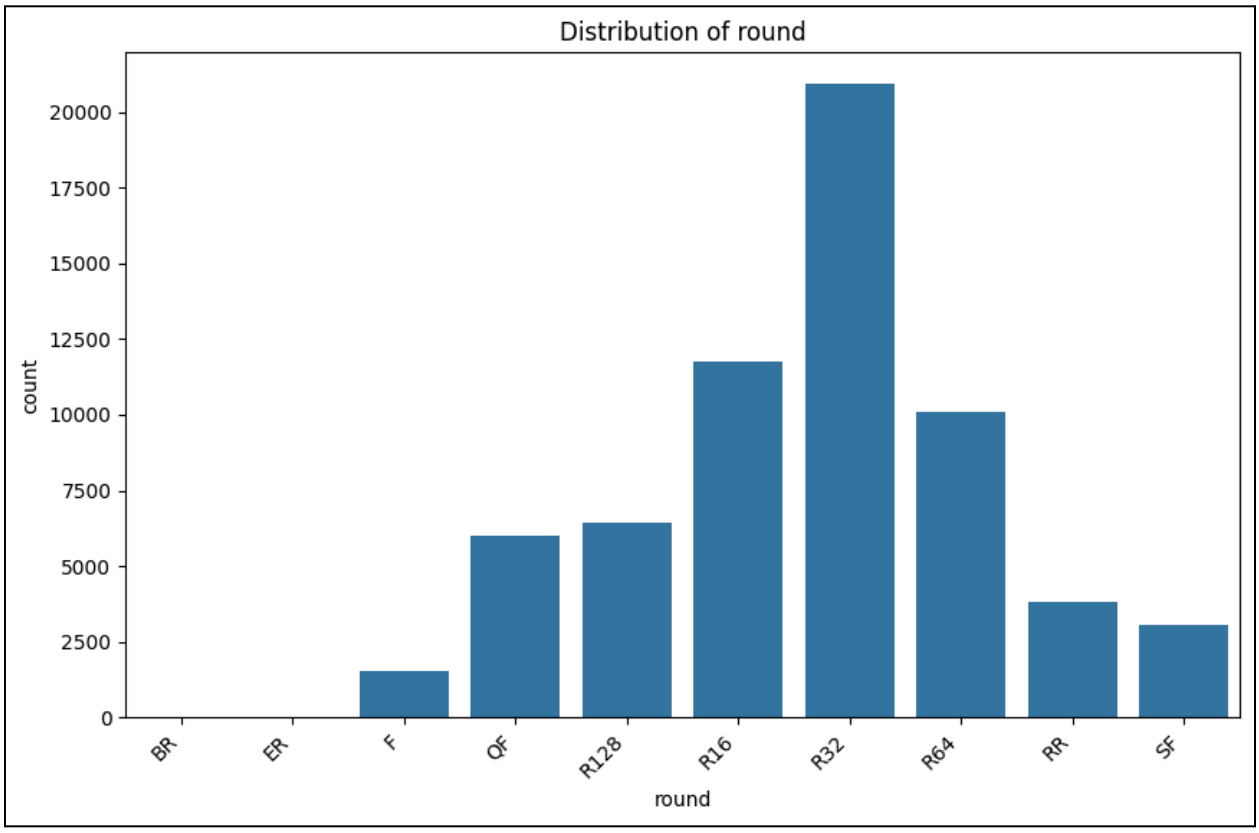
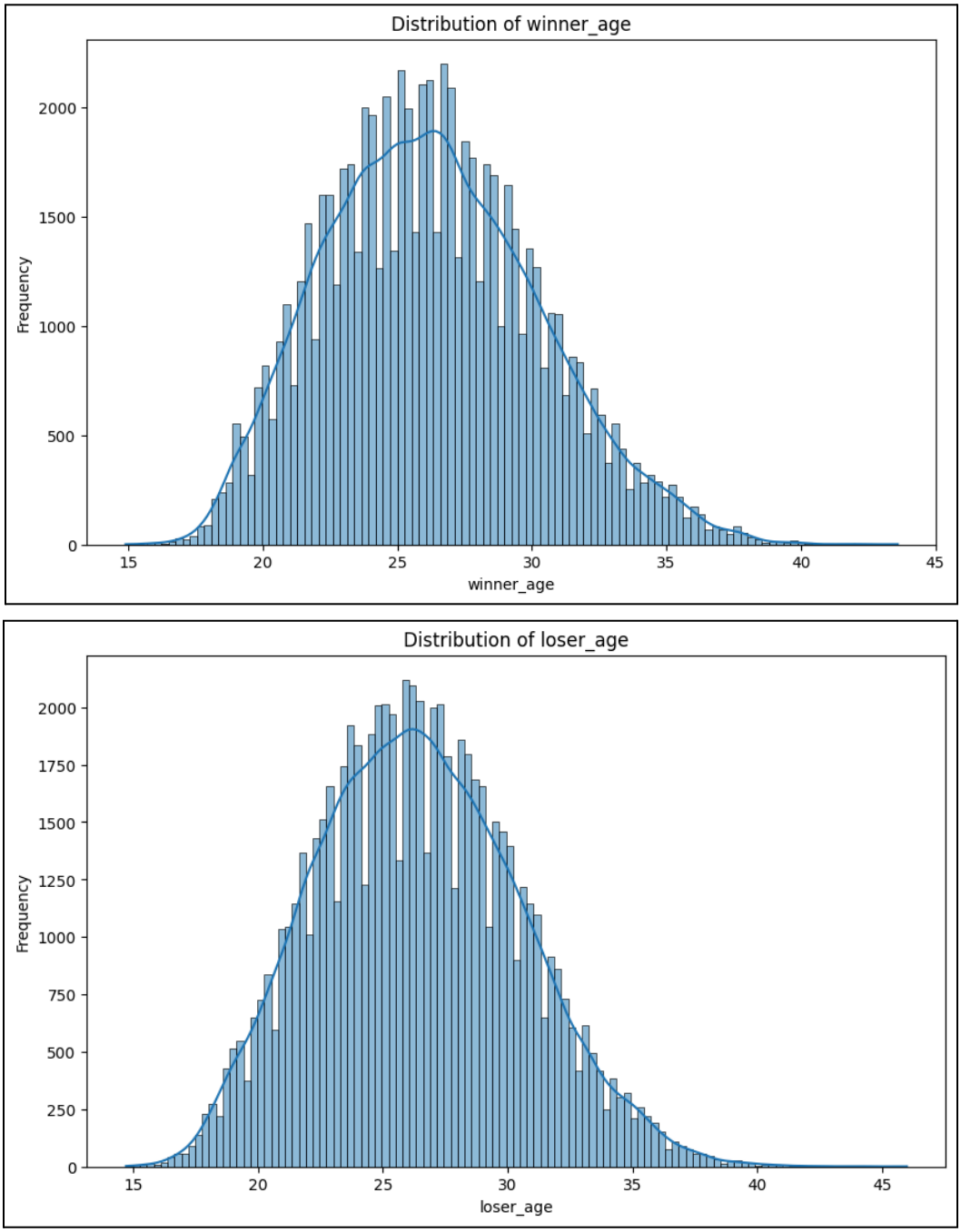


Figure 8A: Distribution of Matches by Age for Winners and Losers



Code 3: Train and predict function for XGBoost

```
from sklearn.model_selection import train_test_split
from xgboost import XGBClassifier

def train_predict_XGBoost(data):
 X = data.drop('winner', axis=1)
```

```
 y = data['winner']
 # Train-test split (randomly selects rows to split data)
 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

 # Initialize and train XGBoost model
 model = XGBClassifier(eval_metric='logloss')
 model.fit(X_train, y_train)

 # Predict
 y_pred = model.predict(X_test)
 # Get probability of predicting 1 (i.e., player 1 wins, player 2 loses)
 y_pred_prob = model.predict_proba(X_test)[:, 1]

 return [model, X_train, X_test, y_train, y_test, y_pred, y_pred_prob]
```

Code 4: Train and predict function for Random Forest

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier

def train_predict_RandomForest(data):
   # Separate features and target
   X = data.drop('winner', axis=1)
   y = data['winner']

   # Train-test split
        X_train,   X_test,   y_train,   y_test   =   train_test_split(X,   y,
test_size=0.2, random_state=42)

   # Initialize and train Random Forest model
   model = RandomForestClassifier(random_state=42)
   model.fit(X_train, y_train)

   # Predict
   y_pred = model.predict(X_test)

   # Get probability of predicting class 1
   y_pred_prob = model.predict_proba(X_test)[:, 1]
```

```
    return [model, X_train, X_test, y_train, y_test, y_pred, y_pred_prob]
```

Code 5: Evaluation function for XGBoost (note: evaluation function for Random Forest follows same pattern)

```
from sklearn.metrics import accuracy_score, roc_auc_score,
precision_score, recall_score, f1_score
from sklearn.metrics import confusion_matrix

def evaluate_XGBoost(model, X_train, X_test, y_train, y_test, y_pred,
y_pred_prob):

 # Accuracy
 accuracy = accuracy_score(y_test, y_pred)
 print(f'Accuracy: {accuracy:.2f}')

 # Area under the ROC Curve
 roc_auc = roc_auc_score(y_test, y_pred_prob)
 print(f'ROC AUC: {roc_auc:.2f}')

 # Train vs. Test Average Prediction Accuracy
 train_acc = model.score(X_train, y_train)
 test_acc = model.score(X_test, y_test)
 print(f"Train Accuracy: {train_acc}, Test Accuracy: {test_acc}")

 # Precision
 precision = precision_score(y_test, y_pred)
 print(f"Precision: {precision:.2f}")

 # Recall (sensitivity)
 recall = recall_score(y_test, y_pred)
 print(f"Recall: {recall:.2f}")

 # F1 Score
 f1 = f1_score(y_test, y_pred)
 print(f"F1 Score: {f1:.2f}")

 # Get confusion matrix values
 cm = confusion_matrix(y_test, y_pred)
 TN, FP, FN, TP = cm.ravel()
```

```
 # Calculate Specificity
 specificity = TN / (TN + FP)
 print(f"Specificity: {specificity:.2f}")

 # Create a heatmap of the confusion matrix
 plt.figure(figsize=(8, 6))
  sns.heatmap(cm, annot=True, fmt='g', cmap='PiYG', xticklabels=['Player
0', 'Player 1'], yticklabels=['Player 0', 'Player 1'])

 # Add labels and title
 plt.xlabel('Predicted')
 plt.ylabel('Actual')
 plt.title('Confusion Matrix for XGBoost')

 # Show the plot
 plt.show()

 # Plot feature importance based by gain
 importances = model.feature_importances_

 # Create a DataFrame for better readability
 feature_importance_df = pd.DataFrame({
     'Feature': X_train.columns, # feature names (predictors)
     'Importance': importances
 })

 # Sort features by importance in descending order
feature_importance_df = feature_importance_df.sort_values(by='Importance',
ascending=False)

 plt.figure(figsize=(10, 8))
plt.barh(feature_importance_df['Feature'],
feature_importance_df['Importance'], color='skyblue')
 plt.xlabel('Feature Importance')
 plt.ylabel('Features')
 plt.title('XGBoost Feature Importance (Gain)')
 plt.gca().invert_yaxis()  # To display the highest importance at the top
 plt.show()
```