ATP Singles Match Prediction

Kevin Nguyen
knguyen2525@gmail.com

October 13, 2017

ABSTRACT

Over the years, with the rise of online betting, different approaches have been taken to model and predict professional tennis matches in hopes of making a quick dollar. Many past approaches use a hierarchical Markov model, passing in estimates to solely predict whether a player will win a certain point on serve based on past information to predict the outcome of tennis matches. Rather than focusing on this approach, I attempt to use a less prevalent supervised machine learning approach that similarly uses a player's historical performance, but allows for greater creativity and flexibility when modeling a tennis match. Drawing off my own tennis experiences, I use the aggregated statistics of players and other generated features to create a model based on the ensemble modeling technique of stacking, resulting in a stacked model with impressive predictive power when tested on major recent tournaments.

Keywords

Tennis, Machine Learning, Classification, Stacking

1. INTRODUCTION

Tennis is one of the most popular and well-known sports today. It has not only become a prominent spectator sport around the world, but also a mainstay in the online sports betting community. Betting participants can wager on a large variety of outcomes during a tennis match, which include the distribution of score amongst games or sets, and the actual outcome of the match. There has been a recent surge in tennis betting volume, making it one of the top sports for online betting. This rise in tennis betting markets combined with data availability has sparked an interest in the topic of tennis modeling and prediction algorithms. Motivation to create these algorithms mainly stems from the possible profit to be made, but also comes from the fact that the structure of a tennis match is quite susceptible to being modeled when compared to other sports. This is primarily because of the small limit of 2 or 4 players in one match, and player match statistics that can be found readily available online.

Many previous approaches for tennis match prediction use a hierarchical Markov model to predict the probability of winning points on serve and sets based on historical data. These approaches solely focus on determining the probability of a player winning a point on serve or on return at any point during a game in a match (0-0, 15-0, 0-15, etc.). Since these methods treat each point as identical and independent, they fail to capture the dynamics of a tennis match accurately as matches often have changes in momentum and lapses in player mentality. Like many sports, tennis's mental components make it tough to model in a way that captures the many subtle changes that can occur during a match.

The goal of this project is to move away from a point-by-point analysis and focus on aggregating a player's historical performance to predict the outcome of that player's future matches by following a supervised machine learning approach. The scope will be narrowed to focus solely on ATP singles matches, as a two player match is simpler to model and ATP rankings have a tendency to not fluctuate as frequently as WTA rankings.

The rest of the paper will follow as such: I will describe the base dataset, discuss the methodology I used for feature generation and modeling, and present experimental results.

2. DATASET

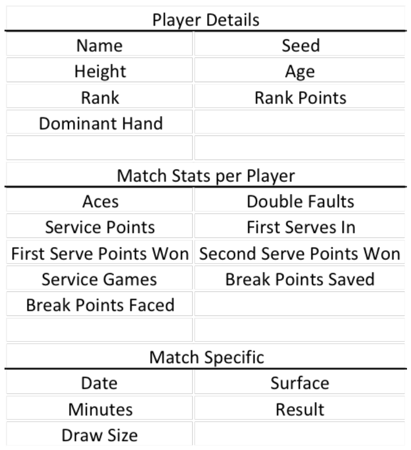
My data source is taken from the work of Jeff Sackmann [7] who has gathered ATP match information throughout the years for various categories of tennis tournaments. For this project, I will be focusing on data taken from ATP singles matches from the years 1990 to 2017. I could not take advantage of previous years because there was a lack of technology to capture the necessary statistics of each match.

Each match has general basic data for each player and for the match in a comma separated fashion. Table 1 outlines the different statistics I used from each match.

2.1 Data Cleaning

Sackmann describes matches from the perspective of the winner, meaning that for each statistic seen in Table 1, his data has the winner's value for that statistic and the loser's value. Cleaning this data focused on four objectives:

- Removing any matches that contained an empty statistic for either player.
- Removing any matches with statistics that did not make sense. This is mostly to remove matches that have a 0 for statistics that will never be 0 such as number of service games played.
- Removing any matches that did not end normally. Usually from a medical retirement, walkover, or default. These matches do not provide accurate information as a player could be the favorite but lose the match due to injury or rule violation.



Player Details	
Name	Seed
Height	Age
Rank	Rank Points
Dominant Hand	
Match Stats per Player	
Aces	Double Faults
Service Points	First Serves In
First Serve Points Won	Second Serve Points Won
Service Games	Break Points Saved
Break Points Faced	
Match Specific	
Date	Surface
Minutes	Result
Draw Size	

Table 1: Statistics for each match that were considered for prediction. Broken down into player specific details, match stats details, and match specific details.

- Creating outcome labels for each match. For a supervised learning model, it's ideal that there be a relatively equal number of positive labeled rows and negative labeled rows to result in an even class balance. To account for this, the labels of winner and loser are generalized to $player_x$ and $player_y$ respectively, and for a given match, a result column is added to determine the winner of the match. For half of the matches, the match is considered in the order of $player_x$ versus $player_y$, while for the other half of the matches, the match is considered in the order of $player_y$ versus $player_x$. For any match in the form $player_1$ versus $player_2$, the results column reflects whether $player_1$ won that match. By inverting $player_x$ and $player_y$ and their statistics, I can more or less guarantee an even split between positive labeled rows and negative labeled rows.

Cleaning the data from 1990 to 2017 reduces the dataset to 69940 matches from 90286 matches.

3. FEATURE GENERATION

This section will provide details on how I aggregated player match statistics and generated features for model training.

3.1 Match Outcome

There are no draws in tennis, allowing tennis to easily be viewed as a classification problem. Given a tennis single match of $player_x$ vs $player_y$, the outcome of a match is labeled 1 if $player_x$ wins and labeled 0 if $player_x$ loses.

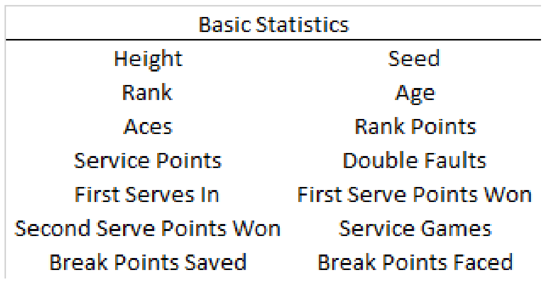
3.2 Basic Statistic Features

Given a match between $player_x$ and $player_y$, basic statistics are those seen in Table 2. These statistics are those explicitly given in the base dataset.
The feature for a basic statistic is simply:

$$statFeature_s = stat_{xs} - stat_{ys} \tag{1}$$

Where:

$statFeature_s$ = feature value for statistic s
$stat_{xs}$ = the basic statistic s for $player_x$
$stat_{ys}$ = the basic statistic s for $player_y$



Basic Statistics	
Height	Seed
Rank	Age
Aces	Rank Points
Service Points	Double Faults
First Serves In	First Serve Points Won
Second Serve Points Won	Service Games
Break Points Saved	Break Points Faced

Table 2: Basic statistics given in dataset.

For each basic statistic s, the difference between the value s for $player_x$ and $player_y$ is found to create the feature value for s. There are other ways to represent a basic statistics such as listing the values for both players in the feature array instead of finding the difference. This would preserve the original values and potentially make the models more accurate. However, by combining the two values by taking the difference, I remove the opportunity for the model to assign higher importance to one player's statistic than the other [13].

3.3 Match Time Decay

Like all sports, tennis is a dynamic game that evolves over time. These evolutions are evident in racket technology, surface alterations, and player fitness, experience, and mentality. As a result, a match that a player participated in a decade ago is less informative on the current performance of a player than a match in his or her most recent tournament, meaning that more recent matches should be weighted more heavily than past matches. This idea of match time decay can be modeled by providing a weight factor to the feature values of a match based on a time difference:

$$W(t) = h^t \tag{2}$$

Where:

h = a hyper parameter that determines the decay rate of a match's significance
t = the time difference between the current match and a future match

3.4 Statistic Aggregation

Tennis matches are a culmination of a player's past match experiences. For a given match between two players, both players past matches must then be considered when determining the current match's feature vector. This can be modeled by taking the weighted average of the statistics of a player's past matches using the time weight discussed in section 3.3. Given a match date and a player, gathering that player's aggregated statistics and finding the weighted average or the player's statistics can be seen in Algorithm 1.

Given a player p and the date of a match m, look through that player's past matches. If a match is found with a date less than match m, gather the statistics for that match for player p. For each statistic, multiply the statistic by a time weight and keeping a running sum consisting of weighted statistics. After iterating through all of player p's matches, find the weighted average of the statistics and return these weighted averages. Feature vector creation is described in Algorithm 2.

Algorithm 1 Statistic Aggregation

```
1: procedure AGGREGATE(date, player)
2:     $stats \leftarrow$ slot for each stat
3:     $weightSum \leftarrow 0$
4:     for match in player.matches do
5:         if match.date < date then
6:             $weight \leftarrow getTimeWeight(date-match.date)$
7:             for stat in match.player.stats do
8:                 $stats.stat \leftarrow stats.stat + weight * stat.$
9:             $weightSum \leftarrow weightSum + weight.$
10:    $average \leftarrow$ slot for each stat
11:    for stat in stats do
12:        $average.stat \leftarrow stat/weightSum.$
13:    return $average$
```

Algorithm 2 Feature Vector Creation

```
1: procedure FVECTORCREATION(match, p1, p2)
2:     $p1Stats \leftarrow aggregate(match.date, p1)$
3:     $p2Stats \leftarrow aggregate(match.date, p2)$
4:     $fVector \leftarrow$ slot for each stat
5:     for stat1, stat2 in p1Stats, p2Stats do
6:         $fVector.stat1 \leftarrow stat1 - stat2$
7:     $weight \leftarrow getTimeWeight(testDate-match.date)$
8:     return $fVector$
```

Aggregate the statistics for p1 and p2 using Algorithm 1. Fill the feature vector by iterating through all statistics and taking the difference between each weighted average statistic for p1 and p2. Return the difference of each of their statistics as values in the feature vector.

3.5 Hand Preference

A player's hand preference has shown to be impactful at all levels of play. Left-handed players have a natural advantage as other players have fewer opportunities to practice against these opponents. Playing a left-handed player causes a player to adjust to different spins, bounces, and change his or her strategy to attack the generally weaker backhand side. Given a match with $player_x$ versus $player_y$, a feature for hand difference is generated:

$$hand_x = \begin{cases} 1 & hand_x = Right \\ 2 & hand_x = Left \end{cases}$$

$$hand_y = \begin{cases} 1 & hand_y = Right \\ 2 & hand_y = Left \end{cases}$$

$$handDiff = hand_x - hand_y \tag{3}$$

It does not make sense to consider a time decay when dealing with hand preference since players almost never switch dominant hands during their career.

3.6 Head to Head Aggregation

Tennis players have various play styles, resulting in some styles being more effective against others. Although play style can be difficult to model, a look at two players' head to head record can hint at who has the advantage if they are equally matched in terms of skill. For a given match m with $player_x$ versus $player_y$, a head to head feature can be created by checking all of their past matches before match m and counting the wins of each player against the other. A head to head difference feature is then:

$$h2hDiff = wins_x - wins_y \tag{4}$$

3.7 Surface Effect

Certain players are known for their prowess on a surface. Rafael Nadal, given the title of "King of Clay", performs exceptionally well on clay surfaces due to his topspin heavy play style, while others, such as Dustin Brown, only play on a few surfaces. For a tennis player to be complete, he must perform well on all surfaces. Comparing the effectiveness of two players on a given surface is a good way to indicate the outcome of a match. Given a match m with $player_x$ versus $player_y$, determining a feature for surface effect is similar to generating a feature for a basic statistic. Matches before match m that are played on surface s are considered. The number of wins and matches played on s are counted and the feature is created by dividing the number of wins by number of matches on s. If a player has no matches on s, he is given a value of 0.5. The feature created is then:

$$surfDiff = \frac{winSurface_{xs}}{matchesSurface_{xs}} - \frac{winSurface_{ys}}{matchesSurface_{ys}} \tag{5}$$

3.8 Service Game Aptitude

ATP singles matches are usually decided on how well a player holds his service and breaks his opponent's service; both first and second serves allow a player to initially dictate a point giving him a clear advantage at the beginning of every point during a service game. Typically, if a player has more points on his serve, he is finding it difficult to win points even with the initial advantage provided by a serve; this allows the opponent more chances to break a player on serve and eventually win the match. A player's service game aptitude can then be modeled using various basic statistics:

$$svgAptDiff = svgApt_x - svgApt_y \tag{6}$$

Where:

$$svgApt = \frac{firstWon + secondWon - bpFaced}{svpt}$$

A player has a stronger service game if a player wins more points off his or her first or second serve but is penalized if his or her opponent is allowed a break point. If a player wins service games with fewer service points, that player has won the majority of his or her points on serve resulting in a stronger service game.

3.9 Feature Summary

Creating these features makes a total of 18 feature columns for model training and testing. Table 3 summarizes all features considered.

4. MODELING AND TRAINING

In the previous section, I described the various features extracted and created from Sackmann's dataset. In this section, I will describe the techniques I used for modeling and training a classifier.

4.1 Dataset Division

Features	
heightDiff	seedDiff
rankDiff	ageDiff
acesDiff	rankPointsDiff
servicePointsDiff	doubleFaultsDiff
firstServesInDiff	firstServePointsWonDiff
secondServePointsWonDiff	serviceGamesDiff
breakPointsSavedDiff	breakPointsFacedDiff
handDiff	h2hDiff
surfaceEffectDiff	serviceGameAptDiff

Table 3: Features generated.

After cleaning the dataset, I have 69940 matches that need to be split into training and testing datasets. Using a standard 70/30 split of the data leaves me with two sets of data:

- Training - 70% of the cleaned data is used for training. This dataset contains 47765 matches ranging from 1990 to part of 2007. This dataset will be used to train my model to create the best mapping between feature vectors and the outcome of a tennis match.

- Testing - 30% of the cleaned data is used for testing. This dataset contains 20472 matches ranging from 2007 to 2017. Testing data is not touched when creating and training a model and is strictly set aside to judge the effectiveness and accuracy of the trained model.

Although this data division is used for training and testing the efficacy of my model, 100% of the dataset is used for training when wanting to predict the outcome of a future tennis match.

4.2 K-Fold Cross Validation

K-fold cross validation is a common technique used in the training of a predictive model. K-fold cross validation splits a training dataset into k folds of data [9]; for each fold, models are trained on the other k-1 folds of data and tested on the remaining fold of data often called the validation set. The main goals of k-fold cross validation are to reduce variability and overfitting of the model. In addition, it allows a person the opportunity to test the efficacy of multiple models without touching the testing set, and pick the model that best fits the biases presented by the dataset via a criteria such as prediction accuracy on the validation set. These accuracies can be averaged over the k-folds to determine the best model.

In the context of tennis modeling, the usual k-fold cross validation is not sufficient as tennis is chronological in player development and growth. It does not make sense to train on a player's future matches in order to predict a past match. To circumvent this problem, a ”rolling” k-fold cross validation is done. The idea is to use past data to predict only the next set of the data in the data folds provided. Given that k is 5, I can divide the dataset into 6 folds and perform k-fold cross validation as follows:

- train models on fold [0], test models on fold 1

- train models on fold [0, 1], test models on fold 2

- train models on fold [0, 1, 2], test models on fold 3

- train models on fold [0, 1, 2, 3], test models on fold 4

- train models on fold [0, 1, 2, 3, 4], test models on fold 5

4.3 Feature Selection

Feature selection is a common step to filter and keep important features. I used univariate feature selection to decide the most correlated features to the outcome column of my training data. This is done during the k-fold cross validation step by finding the features with the largest average F value throughout all the folds. I found that keeping 16 of the 18 original features yielded the best average accuracy for my base models during k-fold cross validation. The two features excluded are firstServesInDiff and firstServePointsWonDiff as seen in Table 3. Feature selection is always performed before actual model training.

4.4 Stacking

Ensemble modeling is a useful technique that tries to take advantage of several models. Ensemble modeling works by blending the different characteristics of several base models together in order to achieve a more general and accurate model that takes advantage of the inherent biases of each base model [15] [5]. For this project, I used type of ensemble modeling known as stacking, which is technique that uses the output predictions of several base models as part of the feature vector input of a final stacked model or classifier. For each base model, predictions are generated on the training set; these predictions are then used as additional feature vector columns when training the final stacked model. These base model predictions are created during the k-fold cross validation step where predictions are made for the current test fold and kept to be used as features later on.

In regards to stacking, I used two levels of models: base models and a final stacked model. I use my base models to train on the current training folds then make predictions on the validation fold. These predictions are saved in a column for each base model and appended to my feature vectors described in section 3. After predictions are made for each test fold for each base model, the training set for my stacked model is ready. To create the new testing set with the new feature columns created by base model predictions, the base models are trained on the entire original training dataset (without the predictions previously made with the base models) and predictions are made on the testing dataset to create the necessary features to test my stacked model. These predictions are appended as additional columns to the feature vectors generated for the testing dataset. The stacked model then trains on the training data with the new base model prediction features and tests its accuracy on the testing dataset with the new base model prediction features.

4.5 Models

4.5.1 *Logistic Regression and Base Models*

Most tennis modeling approaches make use of logistic regression to classify the outcome of a match. Because of this, I also chose to use logistic regression as my final stacked model to make my work more comparable to other approaches. Logistic Regression is a classification algorithm that constrains real number inputs to a value between 0 and 1 using a logistic function:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{7}$$

Similar to linear regression, logistic regression trains by trying to optimize the β parameters of each feature to best fit the result value of all feature vectors. Logistic regression does this by minimizing the logistic loss function. [13] does a nice job adapting this loss function to the context of tennis:

$$Loss(p) = -\frac{1}{N} \sum_{i=1}^{N} p_i \log(y_i) + (1 - p_i) \log(1 - y_i) \tag{8}$$

Where:

N = number of training matches
p_i = predicted probability of a win for match i
y_i = actual outcome of match i (0 for loss, 1 for win)

I used an exhaustive grid search approach to tune the hyper parameters of the logistic regression model.

For base models, I tried various classification algorithms during the k-fold cross validation step and found that the three that performed best in terms of average accuracy on the training folds and validation folds are the random forest, gradient boosting, and perceptron classifiers.

5. RESULTS

In this section, I describe the experiments I ran for testing model accuracy. For all results, I ran my model with k as 10 in k-fold cross validation. I used a perceptron, random forest classifier, and gradient boosting classifier as the base models in the stacking process, and logistic regression as the stacked model.

As mentioned previously, a 70/30 split is used to create training and testing datasets. After training, the model achieved an accuracy of 68.36% on the testing dataset.

The model is further tested by training on matches from 1990 to 2016 and then predicting the outcome of matches for each round of grand slam tournaments. Grand slam tournaments are the four largest and most prestigious tournaments of tennis played each year on various surfaces (hard court, grass, and clay). For each round of each tournament, the number of matches predicted, number of correct predictions, and accuracy are recorded. These results are shown in the following tables.

Table 4: Australian Open 2017 Prediction Results

Round	Predictions	Correct Predictions	Accuracy
R128	30	22	73.33%
R64	19	13	68.42%
R32	15	13	86.67%
R16	8	7	87.50%
QF	4	4	100.00%
SF	2	2	100.00%
F	1	1	100.00%

Table 5: French Open 2017 Prediction Results

Round	Predictions	Correct Predictions	Accuracy
R128	36	26	72.22%
R64	21	16	76.19%
R32	12	8	66.67%
R16	7	6	85.71%
QF	4	3	75.00%
SF	2	1	50.00%
F	1	1	100.00%

Table 6: Wimbledon 2017 Prediction Results

Round	Predictions	Correct Predictions	Accuracy
R128	33	25	75.76%
R64	22	17	77.27%
R32	12	8	66.67%
R16	8	7	87.50%
QF	4	2	50.00%
SF	2	2	100.00%
F	1	1	100.00%

Table 7: US Open 2017 Prediction Results

Round	Predictions	Correct Predictions	Accuracy
R128	30	22	73.33%
R64	18	14	77.78%
R32	11	7	63.64%
R16	6	4	66.67%
QF	3	1	33.33%
SF	2	1	50.00%
F	1	1	100.00%

6. DISCUSSION AND FUTURE WORK

Overall, the results are quite promising. The 68.36% accuracy on the testing dataset needs to be considered carefully because a 70/30 split of the dataset splits the matches into matches from 1990 to September in 2007 and September of 2007 to the end of 2017. Since tennis is based on player evolution overtime, inaccuracies are expected when using training data to predict matches that happen almost a decade later in the testing dataset.

A more adequate test for the model is attempting to predict the outcome of recent tournament bracket matches in 2017 represented by Tables 4 to 7 by training on data from 1990 to 2016. Oftentimes, in large tournaments, many new players make it past the qualifying stages to the earlier rounds. These players do not have any past matches so it is impossible to aggregate statistics for these players. As a result, any matches that include these players must be excluded from the matches to be predicted as there is no basis for their skill level; this often leaves fewer matches predicted than expected given a round (e.g. R128 should have 64 matches but only 30-40 are predicted in these results). Overall, the prediction accuracy for these tournaments are quite impressive.

In tournaments, higher ranked players tend to easily move through the earlier rounds with one-sided scores, resulting in higher seeded players playing in later rounds. As a result, the one-sided difference in player statistics generated in earlier rounds of a tournament tend to be more indicative of the outcome of a match, while outcomes of matches in later rounds tend to trend more towards a coin-flip that depend on which player is having a "better day". As a result, the accuracy of later rounds in a tournament can be erratic. For the results recorded, the model actually had a relatively high accuracy in predicting the later rounds, but it could have easily been the complete opposite. For future work, a good way to accommodate this trend would be to remove matches that go beyond a certain round when cleaning the original dataset as the feature vectors for these matches may not prove to be accurate.

I have only scratched the surface, as there is much potential for further refinement as well as other facets of a tennis match that can be considered. The choice of time decay function can definitely be further explored by more closely monitoring how time affects the evolution of a tennis player. More work can also be done with the features themselves to account for other important factors like injury, fatigue, surfaces, weather, backhand type, and common opponents. Moving away from physical aspects, feature creation that focuses on capturing the dynamic nature of tennis throughout a match is key to predicting its outcome such as the mental state of the players and the momentum each player has at certain parts of a match. Other aspects can include considering the betting odds presented for a match as well as factoring in how close or one-sided a match actually is between two players.

As far as the actual modeling is concerned, future work may want to explore other commonly used machine learning models and other classification methods such as neural networks. An exhaustive grid search on base model hyper parameters may also improve accuracy. but was avoided in the interest of time. It would also likely be worthwhile to experiment with other ensemble methods such as bagging. In terms of metrics, I focused on accuracy, which is probably more relevant to practical use of our model to predict match outcomes, but future models may want to output a probability of victory and use AUC (area under curve of ROC). Alternatively, they might opt to predict the match score via regression and use root mean square error (RMSE) or mean absolute error (MAE) as the metric.

Finally, the scope was limited to ATP matches in this paper, but future work can extend the scope to predict WTA matches and doubles matches as well.

7. RELATED WORK

7.1 Hierarchical Markov Model Approaches

Past approaches to predicting tennis matches are mostly based on a hierarchical Markov approach. Oftentimes, these hierarchical Markov models try to estimate a player's probability of winning a point on serve as the main parameter; these estimates are based on past match statistics with the assumption that they will not change during the match to be predicted. Huang [6] uses a hierarchical Markov model to determine the score of tennis matches in real time with just the input of live in-play online betting odds of the match. Knottenbelt [8] presents another hierarchical Markov model that uses the idea that tennis is a transitive sport, meaning that if Player 1 is better than Player 2 and Player 2 is better than Player 3, then Player 1 should be more likely to win against Player 3 even if no past matches have been played. Knottenbelt introduces the idea of common opponents that take advantage of opponents that both players in a match have faced in the past. Works by Agnieszka [1] propose a set-by-set analysis to predict the outcome of tennis matches, which analysis the changes in the players themselves throughout the match based on past score-lines to see how these changes affect the outcome of sets. While past models have assumed that probabilities of players winning points on serve are fixed or that data from the first set is enough to predict the match, Agnieszka tries to capture the dynamic nature of tennis and how changes in momentum or mentality in players can affect a match set by set. Beve [3] approaches tennis prediction with the idea that player's often have "bad days". Basing a model too strongly on historic data can introduce a chance to overestimate or underestimate a player's probability to win a match. Beve tries to find the right balance between historic and current data to better assess the input parameters of a Markov model. As most approaches to tennis modeling have been made using a hierarchical Markov model, I wanted to try a more uncommon supervised machine learning method. I believe that a machine learning method will allow a more flexible and creative approach as it allows for feature design that can capture more facets of a tennis match rather than solely focusing on a player's probability of winning a point on his service game.

7.2 Machine Learning Approaches

Although tennis modeling seems very attractive to machine learning approaches, these approaches have been overshadowed by the more popular hierarchical Markov model methods. Most documented machine learning approaches use logistic regression such as Clarke and Dyte [4] who made use of logistic regression to model the difference in ATP ranking points of two players to predict the outcome of a set, which was extended to predict the outcome of several tournaments with some success. Ma, Liu, Tan, and Ma [11] also used logistic regression to fit 16 variables they believed were key components to deciding tennis matches. These variables fell into the categories of environmental conditions, match conditions, player strategies, and personal characteristics. Somboonphokkaphan and Phimoltares [2] breaks from the logistic regression trend and approaches modeling tennis with a multi-layer neural network. They use both statistical and environment data within a few years of the match to be predicted. The most recent breakthrough in tennis modeling via machine learning is presented by Sipko [13]. Sipko aggregates player information over the years and weighs statistics of each players based on several factors. These include betting odds, time, match surface, and common opponents. Sipko also considers other factors such as player fatigue, injury, and head-to-head of two players. Sipko uses basic statistics taken from a match combined with his own created features mentioned above and passes these parameters into a logistic regression based model as well as a neural network. My approach is similar to that of Sipko, which is a combination of easily accessible tennis statistics with my own experiences in the sport to create features I believe are impactful to determining the outcome of a match. In addition to these features, I have taken steps to bolster the modeling and training parts of the process by introducing the ideas of ensemble modeling via stacking and a time series based k-fold cross validation to the modeling of tennis matches. This results in a more generalized model that takes advantage of the biases of multiple models while avoiding variability and overfitting.

7.3 Data Loss in Stacking

Few past machine learning approaches to the prediction of tennis matches mention the use of a validation set during model creation. Only [13] splits their data into one training, validation, and testing set; however, the validation set sits in between the training and testing set creating a disconnect from a tennis player's growth. My project adapts the k-fold cross validation process to account for the chronological nature of tennis evolution to avoid this time gap. In order to support k-fold cross validation that is conducive to tennis's chronological nature and stacking, k+1 folds must be created in the training dataset. This is to allow base models at least one data fold for training in order to predict the outcomes of matches in the current test fold. However, because the base models must train on data that is in the past of the current test data, the first fold of data used for training must be spliced from the entire training dataset, because there is no previous training fold to use as training data to make predictions on this first fold of data. For example, if I use 2 folds, I will generate k+1 folds [0, 1, 2]. I will then train base models on fold 0 to make predictions on fold 1, then train on folds 0 and 1 to make predictions on fold 2. However, there is no data to train on to make predictions on fold 0 so these feature vectors must be cut from the training dataset when considering the training set as a whole. The splicing of this data should not be detrimental as the only the earliest matches are removed; these matches will have the least, if any at all, influence on the prediction of more current matches due to the time decay weight parameter discussed in section 3. The loss of this outdated data is a small price to pay in order to allow for the use of k-fold cross validation.

7.4 Same-level Players

Feature generation takes advantage of the idea that when looking at a match between a better tennis player versus a worse tennis player, the difference between their match statistics will be significant. However, it fails to capture the case in which two players are of similar skill. For example, if two players ranked in the top ten were to play, the created feature vector may have smaller values because their skill levels, and the resulting match statistics for both players, are similar. However, if two players ranked around the top 100 play, a similar feature vector with small values may be created since their skill levels and corresponding match statistics are also similar. Despite this phenomenon happening, the model still produces impressive results, which I attribute to some features that are not based strictly on statistics of matches such as rank and rank points. I have also taken measures against this from happening by creating my own features such as head to head difference and service game aptitude. These features will more readily distinguish the better players through their past wins and their ability to more consistently take advantage of their serve.

7.5 Technologies

All data processing, cleaning, feature generation, and modeling are implemented in the Python [14] language. For all machine learning algorithms and models, I used the machine learning library scikit-learn [10]. For array operations, I used the NumPy [12] library.

8. CONCLUSION

Much research has been conducted in the prediction of tennis matches. Most prior work has based their methods on a hierarchical Markov model. In this project, I move towards a supervised machine learning approach. I show that by using basic match statistics, I am able to create a stacked model that can readily predict the outcome of recent tournament matches with adequate accuracy.

As I have only scratched the surface of tennis match prediction, I also elaborated on the various directions in which future work can build on my contribution. Namely, refining the time-decaying factor, fine-tuning models used, creating more relevant features, and increasing the scope to include other metrics all have the potential to lead to further insights into effective modeling and prediction of tennis matches. I hope that this project provides a good stepping stone for future work that can more accurately capture tennis match subtleties and match prediction.

9. REFERENCES

[1] M. M. Agnieszka. A set-by-set analysis method for predicting the outcome of professional singles tennis matches. 2012.

[2] S. P. A.Somboonphokkaphan and C. Lursinap. Tennis winner prediction based on time-series history with neural modeling.

[3] M. Bevc. Predicting the outcome of tennis matches from point-by-point data. 2015.

[4] S. Clarke and D. Dyte. Using official ratings to simulate major tennis tournaments. 2000.

[5] P. T. F. GuìĹnesÌğ, R. Wolfinger. Stacked ensemble models for improved prediction accuracy. 2017.

[6] X. Huang. Inferring tennis match progress from in-play betting odds. 2011.

[7] J. Sackmann. ATP Tennis Rankings, Results, and Stats. 2017.

[8] S. D. Knottenbelt J. William and M. M. Agnieszka. A common-opponent stochastic model for predicting the outcome of professional tennis matches. 2012.

[9] H. L. P. Refaeilzadeh, L. Tang. Cross-validation. 2008.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[11] C. L. S. Ma and Y. Tan. Winning matches in grand slam men's singles: an analysis of player performance-related variables from 1991 to 2008. 2013.

[12] G. V. S. van der Walt, S. Chris Colbert. The numpy array: A structure for efficient numerical computation. *Computing in Science and Engineering*, 13:22–30, 2011.

[13] M. Sipko. Machine learning for the prediction of professional tennis matches. 2015.

[14] G. van Rossum. Python tutorial. 1995.

[15] D. Wolpert. Stacked generalization. 1992.