Real Time Tennis Match Prediction using Machine Learning

Entov, Gabriel
Robotics Engineering
Worcester Polytechnic Institute
Worcester MA, USA
gentov@wpi.edu

Lanotte, Nicholas
Robotics Engineering
Worcester Polytechnic Institute
Worcester MA, USA
njlanotte@wpi.edu

Mazza, Laurie
IMGD
Worcester Polytechnic Institute
Worcester MA, USA
lmazza@wpi.edu

Santos, Adam
Robotics Engineering
Worcester Polytechnic Institute
Worcester MA, USA
asantos@wpi.edu

***Abstract*— This paper discusses a supervised learning algorithm trained to predict tennis matches by combining both historical match data and real-time statistics. Different models for predicting matches before they begin were explored and compared to each other. A separate algorithm for predicting a match as it is being played was created by testing and comparing multiple algorithms and data structures. With both of these algorithms established, they were combined to produce an algorithm with better performance overall than either of the two algorithms alone. Sets were found to be the most dominant and predictive feature for matches, with other standout features also revealed, such as service points, specifically on the first serve. Finally, the project outlines the future work needed to help combat the problems of the classifier such as predicting upsets and comebacks.**

Keywords—Machine Learning, Tennis, Past, Real, Time

I. INTRODUCTION

Machine learning is exceptional at finding patterns in data. As more and more data for different sports is being collected, it is only natural to apply machine learning to this data in hopes of getting accurate predictions about an events outcome. With the copious amounts of sports betting happening every day [15], an accurate prediction could prove very powerful. Having a classifier that can put weight to different factors also gives coaches and club managers the ability to cater their efforts to the factors that contribute most to the events outcome.

This paper focuses on using supervised learning to predict the outcome of a tennis match, as the match goes on. Similar techniques described in this paper can be used to predict other sports as well. Predicting the outcome of a tennis match or a season in tennis is useful in enriching the entertainment value for spectators and is useful for people who invest in betting. Predicting the outcome of a match and the main factors contributing to that outcome could help the player or coach in creating strategies for improving their chances of winning [5]. This kind of research is also helpful to both players and coaches, as it allows coaches to see which attributes of a tennis match are most critical to winning a tennis match. Additionally, this research could affect how the brackets are created for different tournaments, to make it fairer for the players, or more entertaining to watch. The main problem with predicting match outcomes is obtaining sufficient amounts of data, including past match outcomes, player stats, point-by-point data, etc., to train an algorithm. Organizing the data in such a way that is useful for an AI to learn from is another challenge because, as discussed in this paper, certain feature sets produce poor results compared to others.

The problem of predicting an outcome of a tennis match has been attempted many times before, so the goal of this paper is to improve upon previous methods to obtain a better accuracy in predicting match outcomes. To solve this problem supervised learning was used. Supervised learning is where the output of an input is known, and coefficients are tuned in order to correlate certain features of the input with the output. If the algorithm is fed enough training data, it will be able to predict the output of the input, with a reasonably high accuracy. Two different models were created for this project. A model used to predict match outcomes based on match statistics, and another model used to predict matches before any information about them is known. The algorithms explored for both the real-time and pre-match predictions were K Nearest Neighbors (KNN), Logistic Regression, and Support Vector Machines.

II. BACKGROUND

There have been other implementations of real time tennis match predictors, such as a study done by Stanford students in 2017 [2]. These students used supervised learning techniques and found that using historical data was the best method of predicting a match, with real-time data worsening the

accuracy. They found that total points won and return points won, were the most important features when predicting a match using real time data [2] but overall, players exhibit some sort of "memory" making in-game stats irrelevant overall [2]. One item of future work they pointed out was to see if historical data combined with real-time data with an updated feature set would yield better results, which is something this paper builds upon.

Another study by Stanford students was conducted, with a larger focus on using tennis match prediction for betting [4]. This project implemented a random forest model that makes a prediction on who will win a match, generated betting odds and generates a betting strategy to maximize the betting reward [4]. The researchers believed that the model could be greatly improved with a richer feature set, use of different classification algorithms and more powerful computational hardware [4].

Another study in 2012 used a model that combines probabilistic equations for predicting tennis matches with a model that analyzes player performance against a common opponent [7]. The model the study started with was developed by O'Malley and expresses the probability of a player winning a game, set or match based on the player's probability of winning a point while serving [7]. This model finds the probabilities in reference to the average opponent which makes them biased since good players typically advance and face other good players rather than average. Their model attempts to remove the bias by comparing players in a match against common opponents from previous matches and adjust their prediction. The result was a 3.8% improvement over the methods that were built on, but the researchers believe there is still room for improvement by implementing more sophisticated strategies for calculating probabilities [7].

III. Methods

This section discusses the various supervised learning algorithms explored in this paper and how they were applied to this solution to the problem.

A. *K Nearest Neighbors*

K Nearest Neighbors classification is a simple algorithm that stores the data used to train it in the form of a point in an n-dimensional space, where n is the number of features each data vector contains [12]. For example, if the algorithm needed to classify an object based on length, width, height and color, then the number of dimensions would be 4 since there are 4 distinct features for this data. That data point is associated with a specific class. When a new data point is entered to be classified, the algorithm finds the k nearest points, where k is user specified, tallies up the number of times each class appears for those points and classifies the data based on the class with the most tallies [12].

B. *Support Vector Machines*

Support Vector Machines (SVMs) were used for the real-time prediction engine because SVMs are designed for binary classification problems. Imagine that there are two clusters, a winning and a losing cluster. SVM tries to draw a "decision boundary" to separate and generalize the data into two distinct categories. It tries to maximize the distance (or margin) between the decision boundary and the two data vectors closest to each other in opposing categories. In the example below, the vectors that the two red lines go through on either side of "direction 2" are known as support vectors. As shown in Fig. 1, both "direction 1" and "direction 2" are decision boundaries, but direction 2 is a substantially better one [8].

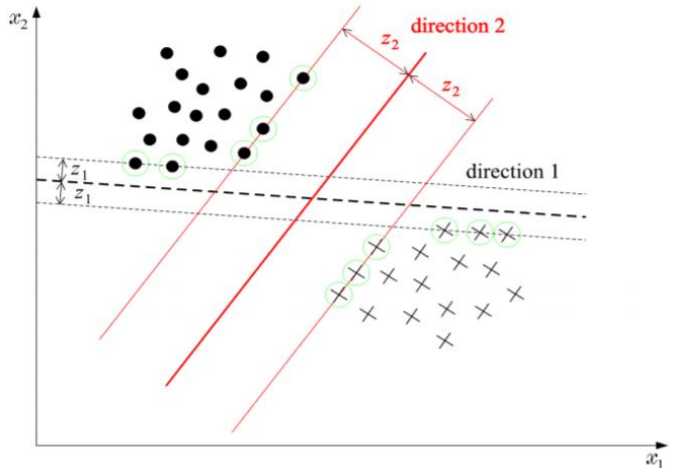

Fig. 1. Visual example of Linear SVM, demonstrating two possible decision boundaries. The dots are one class, and the pluses (+) are another. "Direction 2" leaves a larger margin than the margin created by "Direction 1".

Mathematically:

$$w^T x \geq C, if\ y = 1 \quad (1)$$

And

$$w^T x \leq C, if\ y = 0 \quad (2)$$

Where w is the feature weight parameters, x is training sample, C is some threshold, and y is the target class.

These equations come from intuition. If the projection of w^T (which is perpendicular to the median value) onto x is greater than some median value (consider direction 2 in the example above), then it is a positive class, otherwise it is a negative class [8].

Mathematically, SVM tries to minimize w (the parameters) such that the distance between the support vectors is maximized.

SVM minimizes w because it tries to maximize the width of the street. If the dot product is taken of the difference between a vector to a positive sample and a vector to a negative sample (Xp - Xn) and a unit vector w (which is perpendicular to a support vector), the width of the street is calculated. Fig. 2 helps illustrate this point [8].

$$width = (X_p - X_N) * \frac{\hat{w}}{\|w\|} \quad (3)$$

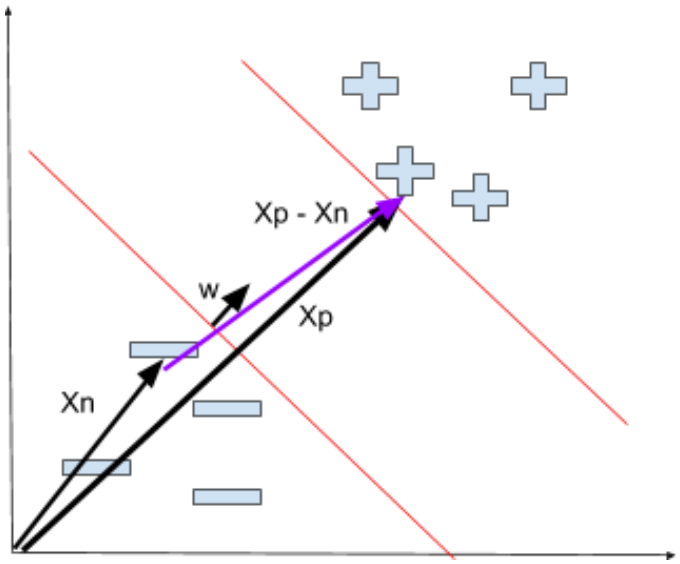

Fig. 2. Visual Explanation as to why SVM tries to minimize *w* [8] The pluses (+) are positive samples, and the minuses (-) are the negative samples. Xp is a positive vector, and Xn is the negative sample vector.

By minimizing *w* SVM will maximize the margins [13].

C. *Logistic Regression*

Logistic Regression is a modification of linear regression. Linear regression uses a function called a hypothesis which is a linear function where data samples above the line are classified as one class and data samples below the line are classified as another [11]. Logistic Regression is similar except the hypothesis function is a sigmoid function, which means the results from this function are always between 0 and 1. This makes predictions that are less than 0.5 classified as one class while predictions that are greater than 0.5 are classified as the other class [11]. The general form of the logistic function is shown in (4) where z is the linear function used in linear regression.

$$h_\theta(x) = \frac{1}{1 + e^{-z}} \tag{4}$$

The z function is essentially the dot product of the feature vector and a vector of feature weights [8] The feature weights are determined when the logistic regression model is fitted, and they determine how much each vector affects the prediction.

Fig. 3 shows the general shape of a sigmoid function where z is plotted on the x-axis and the certainty of prediction is plotted on the y-axis. The function converges to 0 as z approaches negative infinity and converges to 1 as z approaches positive infinity.

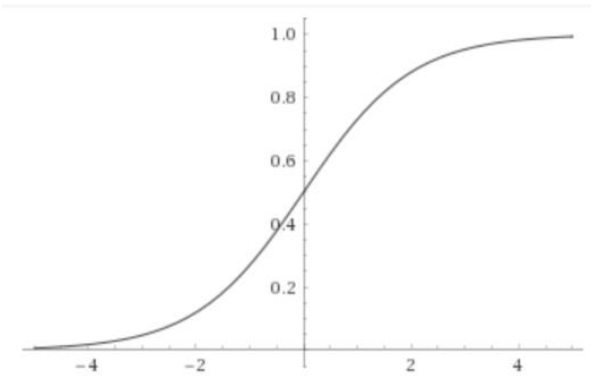

Fig. 3. General Shape of Sigmoid Function. Z is plotted on the x-axis and the certainty of the prediction is plotted on the y-axis

To fit the model, a cost function is needed to determine the model fitness. One method is a cost function that is to be minimized, where the minimized value is the best-fit model for the given data set. The cost function is shown in (5) where m is the number of samples in the training data and i is the current sample being evaluated [8].

$$J(w) =$$

$$\frac{1}{m}[\sum_{i=1}^{m} -ylog\big(h_\theta(x_i)\big) - (1-y)\log\,(1-h_\theta(x_i))] \tag{5}$$

To minimize J(w), a method called gradient descent is used. The equation for gradient descent is found in (6). θ_j is the feature weight being fit and α is a user specified learning rate [8].

$$\theta_j \coloneqq \theta_j - \alpha\,\frac{1}{m}(\sum_{i=1}^{m}(h_\theta(x_i) - y_i) * x_i) \tag{6}$$

The intended use of this function with J(w) is to first find the cost of the training data, perform one iteration of gradient descent and then measure the new costs with the updated feature weights. Each feature weight must be updated simultaneously. The difference in the costs is found and these steps are done in a loop for either a certain number of iterations or until the difference of the costs is below a certain threshold. This prevents the algorithm from running indefinitely [8].

Once the feature weights are found, predicting a new piece of data is performed by plugging the data into the sigmoid function. The result is a number between 0 and 1. Values close to 0.5 show that the algorithm is uncertain of the class while values close to 0 or 1 show the algorithm is certain the data belongs to the corresponding class [8]

The data the project uses comes from three main sources: Kevin Lin's "Serve and Volley" project, which contains match scores and match statistics for 2017, Jeff Sackmann's "tennis_slam_pointbypoint" repository, which contains point by point player statistics, as well as from the ATP Tour website. How this data is parsed and processed for training is discussed in the Methods section. Twitter is a social

media website where users are constantly sharing information about what is going on, including sports updates. This paper looked into taking advantage of the many accounts sharing tennis match information in real-time by grabbing the data from their tweets. Unfortunately, there is no common hashtag to look at for specific matches so the data would have been collected through grabbing tweets from a combination of relevant hashtags and tweets containing either of the player's names or twitter handles. This process resulted in irrelevant information needing to be filtered out before starting the process of filtering out duplicate information. Since this process was slow due to limitations on the twitter API and the amount of filtering, along with the risk of getting inaccurate information, it was decided to not use data pulled from twitter.

D. Real-Time Prediction

The "Serve and Volley" project was used for the real-time prediction engine and contains match statistics and match scores, which were combined by pulling the match data from the corresponding match URL. Combining data from both match scores and statistics allowed for a very large range of features. Eventually, these features were reduced to the most widely recorded and most predictive features. This allowed the team to test a few matches that had live statistics taken from Google's live tennis scores. The few matches that were tested in real-time are taken from the ATP Nitto Finals tournament, and Jeff Sackmann's point-by-point data was used for more rigorous testing.

Training data is entirely composed of post-match stats, but we intended to use this data for intra-match predictions. As a result, there is an innate issue with all stats that increment with progression of the game, such as points and serves, and they will always be higher at the end of a game than at the start. This creates a split between the nature of data samples used to train the real-time model, and the samples used for testing. To resolve this issue, all 'count'-based features were normalized into percentages (such as games, serves, points won, etc.). This converts the absolute values into relative ones, where a player's performance can be more closely analyzed across different points in a match. For example, instead of saying "Player 1 has won 4 games, and player 2 has won 8 games" the data is transformed into a sentence that would read: "Player 1 has won 33% of the games, and player 2 has won 66% of the games".

Our data, which originally contained an entire match for both players' statistics on a single line, was broken down into two vectors: one for winner statistics and another for loser statistics. This way, the target value for each sample was either a loser or a winner, and the features were the player's normalized statistics for that game.

This same principle was applied to matches twice, once where the percentages of sets won was known, and another when the percentages of sets won was not known. This way, before a player has won a set, the real-time prediction can still make a prediction, but once a player has won a set, the prediction confidence for that player can increase. The two models are averaged when a prediction is made.

To test the real-time engine, point by point data was needed. As there were very few matches in the time frame of this project, historic point-by-point was taken from Jeff Sackmann's "tennis_slam_pointbypoint" repository. The raw features reported by this repository did not match particularly well with the data that the classifier was trained with, but some of the data could be used to properly calculate the match statistics that were of interest. Statistics were filled in with zeros if they could not be derived from the point by point data. In this way, many more matches were able to be used to test the performance of the algorithm.

E. Pre-Match Prediction

The ATP match data was used for training the pre-match prediction algorithm. Originally the match data was used in a KNN algorithm to cluster matches together based on player names. One issue was that to use Scikit Learn's KNN method, the player names need to be encoded to a number. This means that players with names encoded by numbers close to each other would be clustered together which doesn't make sense for the problem, since each player should be equally distant from any other. This also means the algorithm could predict a player not in the match would win. To combat this, a custom KNN algorithm was created which treated every player name as equally distant from any other. When predicting a match, it would search out previous matches with the two players as well as matches containing only one of the players and decide on a winner based on amount of matches each player has won from that set of matches. This method proved to be inaccurate which led to the exploration of logistic regression and SVM algorithms for a pre-match prediction. Rather than use player names, the algorithms take in each player's rank, points won, wins and loses so far in that year, forming an 8-column vector to fit the sigmoid function. These features were chosen because they can be easily found on the ATP website for each player in the current season and were available in the ATP data used to train the model. Since the player stats are the only thing known about a match before it begins, the model had to be limited to stat data in order for future predictions to be possible. Over 200,000 matches from the ATP match database from 2011-2019 were used to fit the model.

In order to get an accurate prediction, the latest data was collected for the players from the ATP website. The pages to gather data from were derived by combining the base elements of the URL with the player's name and player id. The page was then collected as a string using the python requests library and then converted into an HTML tree. Every player's page shares a standard format that allowed for the use of established XPaths to traverse to the desired data within the tree which every player's page shares a standard format. The XPath is created by mapping out the identifier for the data point, with each data point needing a different XPath. In some cases, the identifier is a unique id that can be immediately jumped to while others were mapped out by identifying the correct index in each generation to follow until the data is reached. These points are then written into a CSV file along

with the player's identifying information to then be used in calculations.

F. Combining Algorithms

A goal of the project was to use the pre-match prediction as a baseline accuracy at the start of the match while the real-time prediction corrects the overall outcome as the match progresses. What was not clear was how much to weigh the two algorithms in the overall prediction. Various combinations of weights were tested on the point-by-point data and bar charts were generated, showing how early in the match the algorithm predicted the outcome correctly. These tests were used to decide on an optimal weight distribution between the 2 algorithms.

IV. RESULTS

A. Pre-Match Prediction

A custom KNN algorithm, a logistic regression algorithm, a support vector machine with a linear kernel and a support vector machine with a RBF kernel were tested for use with the pre-match predictor. The SVM algorithms were quickly eliminated as potential algorithms as the time complexity to train them are the square number of samples. Since 200,000 training examples from 2011 to 2018 were available this would mean it could take potentially days to train SVM. The results for logistic regression and KNN are in Table 1. They were tested with 3600 matches from 2019.

TABLE I. ACCURACY OF CLASSIFIERS FOR PRE-MATCH PREDICTION. THE LEFT COLUMN SHOWS THE CLASSIFIER CHOOSEN TO FIT THE DATA. THE RIGHT COLUMN SHOWS THE ACCURACY OF THE CLASSIFIER.

Model	Accuracy
KNN	46.0%
Logistic Regression	66.6%

The train and test data included ATP main season matches, matches from qualifying matches and futures matches. When only including matches from the ATP main season (23,000 training matches from 2011-2018, 645 test matches from 2019), the accuracy was 79.8%. This may suggest that the main season matches are more predictable.

B. Real-Time Prediction

In order to choose an algorithm for the real-time prediction, KNN, Logistic Regression, and SVM were tested. The two classifiers were trained with and without sets, and the following tables show the accuracy of each classifier. The data was trained on 3,684 matches. For the training data in which sets were included, all classifiers performed extremely well. In fact, all classifiers achieved an accuracy of over 99%. This result was verified using 10-fold cross validation.

TABLE II. ACCURACY OF CLASSIFIERS FOR REAL-TIME PREDICTION GIVEN SET INFORMATION. THE LEFT COLUMN SHOWS THE CLASSIFIER CHOOSEN TO FIT THE DATA. THE RIGHT COLUMN SHOWS THE ACCURACY OF THE CLASSIFIER. THIS DATA INCLUDES INFORMATION ABOUTS SETS FOR EACH PLAYER.

Model	Accuracy
KNN	99.7%
Logistic Regression	99.5%
SVM Linear	99.8%
SVM RBF	99.8%

Similarly, the data trained without sets was evaluated. Most of the algorithms performed well, but SVM with a linear kernel classified the most accurately and was ultimately chosen for the real-time prediction.

In practice, combining both classifiers proved more effective than using only one predictor. Although the pre-match predictor was able to correctly predict 74.5% of matches in the point-by-point data set based on player stats. The results of this test can be found in Fig. 4 It should be noted that the point-by-point data contains many of the same matches used to train the pre-match algorithm which may be why the prediction was correct more often than in the initial testing.

TABLE III. ACCURACY OF CLASSIFIERS FOR REAL-TIME PREDICTION BY END OF MATCH NOT GIVEN SET INFORMATION. THE LEFT COLUMN SHOWS THE CLASSIFER CHOOSEN TO FIT THE DATA. THE RIGHT COLUMN SHOWS THE ACCURACY OF THE CLASSIFIER. THIS DATA DID NOT INCLUDE ANY INFORMATION ABOUT SETS FOR EACH PLAYER.

Model	Accuracy
KNN	92.6%
Logistic Regression	95.2%
SVM Linear	95.7%
SVM RBF	94.7%

However, only ATP main season matches were used in the point-by-point data which are more easily predicted by the pre-match algorithm. Additionally, since the accuracy is not close to 100%, it suggests that even if these matches are the same as the test set, the algorithm has generalized the data. If this were a case of overfitting of the training data, the correct predictions would be much closer to 100%.

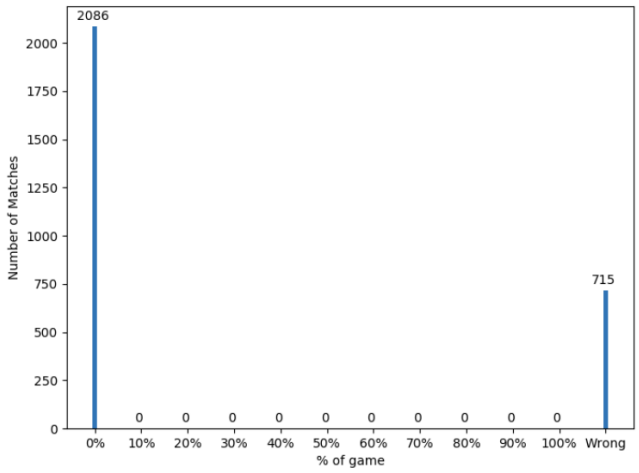

Fig. 4. Number of matches predicted correctly using the pre-match prediction. The x-axis is the percentage of the match based on current game over total games in the match. Since the algorithm only predicts the match at the start, it either got the prediction correct at the start or wrong.

On the other hand, the real-time prediction engine had its own weaknesses. Using only the real-time prediction model, without any knowledge of the pre-match prediction, the model did well to correctly identify the winner of the match but could not do so very early in the match.

The real-time prediction misclassified 18 matches, and correctly predicted 67.5% of the matches before the first half of the match The remainder of the matches were predicted close to the end of the match, which is expected since the outcome of a match is more predictable as the end of the match is reached Overall, the real-time classifier has less misclassifications at the end than the pre-match classifier, but the pre-match classifier has more correct at the beginning of the match.

C. *Combining Algorithms*

Combining the two classifiers, allowed for a balance of the strengths and weaknesses of both. By allowing the pre-match prediction to influence the real-time prediction and vice-versa, incorrect classifications made initially by the pre-match prediction could be corrected by the events of the match. With this balanced model, 2.8% of matches are classified incorrectly by the end of the match.

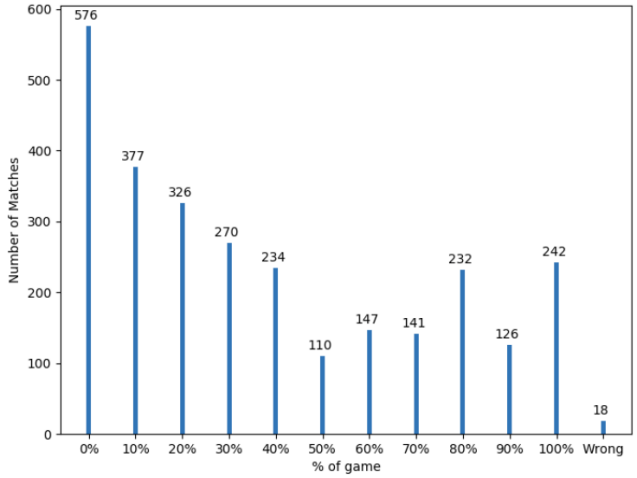

Fig. 5. Number of matches predicted correctly using only the real-time prediction.The x-axis is the percentage of the match based on current game over total games in the match. The percentage of the game where the final prediction was correct are rounded up to the nearest 10%.

This is a vast improvement over the 25.5% misclassification using the pre-match classifier alone. Although the real-time classifier misclassified 0.6% of the matches, this combined classifier predicts much earlier. In fact, 21.8% are predicted correctly in the second half of the match, as opposed to 31.7% using only the real-time prediction.

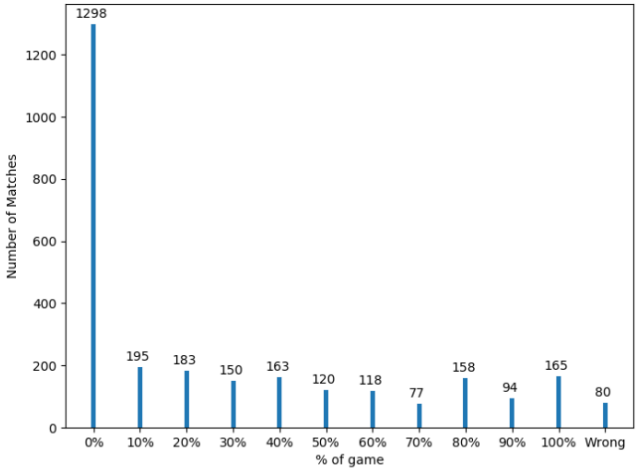

Fig. 6. Number of Matches Predicted Correctly using Combined Classifier. This combined classifier uses 50% of the Pre-Match prediction, and 50% of the Real-Time prediction.

TABLE IV. PERCENTAGE OF MATCHES CORRECTLY CLASSIFIED AT DIFFERENT TIME STAMPS IN A MATCH

Percentage of Games Played	Percentage of Matches Correctly Classified
0%	46.34%
30% (First Set)	65.19%
50%	75.29%
70%	82.25%
100%	97.14%

D. *Case Studies*

In addition to examining how well the classifier performed on a large scale, two specific matches were used as case studies to demonstrate key behavior of the model. The combined model does not have a strong ability to correct for major upsets due to the pre-match prediction adding a constant bias to the real-time prediction. This bias proves to be beneficial in most matches but for upsets, it prevents the player originally predicted to win to have a win percentage below the weight of the pre-match prediction.

In major upsets, our pre-match algorithm gives one player a near 1.0 confidence and a near 0 to the other player. As a result, games where one player has much stronger past match statistics than their opponent will result in a high win confidence for that player. In these cases, we find that each player ends up confined to a respective 'winner' and 'loser' channel. For the player favored by the pre-match algorithm, their win confidence value will never drop below ~0.5 and conversely the other player will be trapped below ~0.5 no matter their in-match performance. A 2013 match, shown in Fig. 7, demonstrates this behavior:

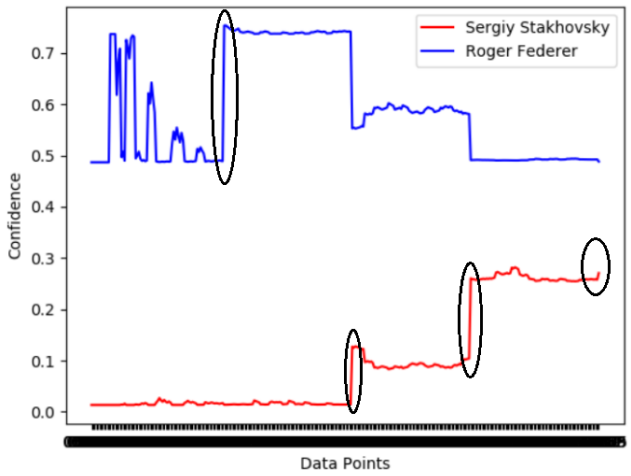

Fig. 7. Prediction Confidence for 2013 Wimbledon, point by point. Roger Federer (rank 3) was given a very high pre-match prediction, while Stergiy Stakhovsky (rank 136) was predicted to have a near zero chance of winning. Stakhovsky won the match, so the combined algorithm was unable to correct the prediction of this match. The circles are points where sets are won. Stakhovsky won more sets.

This is the model's confidence plot for the game as the match progressed. The higher confidence value corresponds to the predicted victor of the match. During the 2013 Wimbledon tournament, ATP rank #3 Roger Federer played against rank #136 Sergiy Stakhovsky. This game represents the behavior that our model is likely to exhibit due to the enormous disparity between the players' historical statistics.

The pre-match prediction algorithm nearly maximized Federer's chances of winning, while Stakhovsky was given a near-zero chance to win. Federer also won the first two sets in this match [9], exacerbating this already heavy favoring for Federer by giving him 100% of the sets won.

This game eventually would become known as a 'comeback for the ages', as Stakhovsky ended up scoring three sets in a row to defeat Federer. The prediction algorithm, however, favored Federer so heavily from earlier performance that it became impossible for the real-time prediction to assign enough confidence to Stakhovsky for him to ever be the predicted winner - even after the match when he had ⅗ of the sets and by definition enough to win the game.

These games represent a small minority of all tennis games, and overall the heavy pre-match bias results in more accurate predictions. The following match demonstrates the model's ability to correct a prediction given additional information from the game in progress, even given a heavy pre-match disparity between the players.

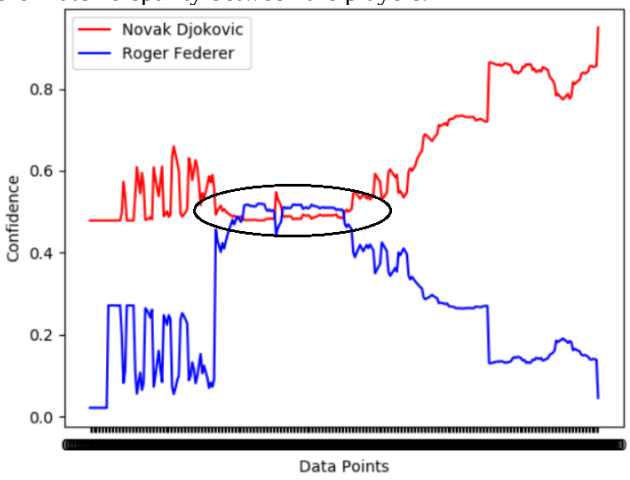

Fig. 8. Prediction Confidence for 2011 US Open Semi final, point by point. Roger Federer was given a substantially low prediction compared to Novak Djokovic. Roger Federer was beating Novak Djokovic in the first half of the match and the combined algorithm was able to correct the prediction. In the second half, Novak caught back up and beat Federer in the end. The circled portion shows how the combined algorithm was able to correct the original prediction based on match events.

The above game is the confidence plot of two high-ranking players in the 2011 US-Open. While the pre-match prediction between them still heavily favors one player, the plot shows how the players' performance can equalize the model's confidence in each player. At about a quarter of the way into the match, Federer wins the first set and becomes expected to win by a small margin. The graph of the prediction then splits modestly once Djokovic wins a set himself and the difference in the predictions only grows after a clear lead was established with a second set for Djokovic [10].

V. CONCLUSIONS

Using a pre-match classifier and a real-time classifier independently lead to poor results compared to combining them. The pre-match classifier was better at predicting more matches earlier on than the real-time classifier, but the real-time classifier got almost all matches correctly classified by the end. Combining the classifiers allowed for a prediction

accuracy of 97.15% before the end of the match with 75.3% of matches predicted correctly at the half-way point.

Combining the classifiers comes with advantages and disadvantages. An advantage is that it merges past data and present data within its prediction while maintaining high accuracy. This improves upon a Stanford study that found predictions to be worse when combining real-time information with predictions based on historical data. A disadvantage of this method is that it does not perform well in cases of major upsets where a player who is heavily favored to win loses to the other player making a comeback. It is possible that this could be more accurately predicted by factoring an element that is currently not being factored into the calculations.

Each classifier was tested using data from 2801 ATP matches of various cases ranging from matches where the highly favored to win player won, to matches where an underdog player comes back from behind to win. Only 80 matches were misclassified by the combined classifier.

The results showed that the most important factor to determine match outcome is the number of sets won. This is not surprising, as the player with the majority of sets will win the match. Training the classifier without set data showed that service points won are almost two times as important as return points won. Additionally, points won on a player's first serve are substantially more important than points won on a player's second serve in contributing to their overall match outcome. Therefore, coaches and trainers should focus on aggression of first serves and ending points quickly.

For future work, factoring in a "human element" is an important next step not considered in this paper. All players are human, and thus can have good or bad days. Emotions can get the better of people and impact how they perform. In the future, pre-match interviews could be used to help improve the performance of the pre-match prediction. Additionally, future work could consider the surface type as a factor in the prediction, as well as using the players' current momentum in a match to help predict if they are going to keep winning or losing. Finally, as our classifier struggled with major upsets and comebacks, having a focus on comebacks is something to be considered in the future. The ATP website provides some data on the percentages of matches for which a player has won after losing the first set of the match, so perhaps this data can become useful in the future. A final area of future work to consider that was not explored in this paper is creatively manipulating the data differently as it could reveal new information. An example of this would be to use the slope of confidence as a player's momentum to improve the prediction.

REFERENCES

[1] "Official Site of Men's Professional Tennis." Association of Tennis Professionals. https://www.atptour.com/en.

[2] Y. Chen, Y. Tian, and Y. Zhong, "Real Time Tennis Match Prediction Using Machine Learning," Stanford, 2017.

[3] A. Cohen. "IBM's Coach Advisor Will Track Players at U.S. Open, United Soccer League Signs New ESPN Deal." https://www.sporttechie.com/ibm-coach-advisor-player-tracking-tennis-us-open-united-soccer-league-espn/

[4] A. Cornman, G. Spellman, and D. Wright, "Machine Learning for Professional Tennis Match Prediction and Betting," Stanford, 2017.

[5] Bunker, R. P., & Thabtah, F. A machine learning framework for sport result prediction. Science Direct, 2017 https://www.sciencedirect.com/science/article/pii/S2210832717301485

[6] T. O. Evans. "How much is a set worth in tennis?" http://www.mathistopheles.co.uk/maths/how-much-is-a-set-worth/

[7] W. J. Knottenbelt, D. Spanias, A. M. Madurska. "A Common-opponent stochastic model for predicting the outcome of professional tennis matches," Science Direct, 2012.

[8] D. Korkin, "CS534 Lecture notes," Worcester Polytechnic Institute, 2019.

[9] K. Mitchell. "US Open 2011: Roger Federer struggles to accept Novak Djokovic defeat." The Guardian. https://www.theguardian.com/sport/2011/sep/11/us-open-2011-federer-djokovic.

[10] P. Newbery. "US Open 2011: Novak Djokovic beats Roger Federer in five sets." BBC Sport. https://www.bbc.com/sport/tennis/14868433.

[11] A. Ng, "CS229 Lecture notes," Stanford, 2018.

[12] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825--2830, 2011.

[13] [10] J. Sackmann. Grand Slam Point-by-Point Data [Online] Available: https://github.com/JeffSackmann/tennis_slam_pointbypoint.

[14] P. Winston. "Learning: Support Vector Machines," *YouTube*, Jan 10, 2014.https://www.youtube.com/watch?v=_PwhiWxHK8o. [Accessed: 11 3, 2019].

[15] P. P. Betfair, "Online revenue of selected sports betting / daily fantasy sports companies in the United States in 2017 (in million U.S. dollars)*," ed. Statista: Paddy Power Betfair, 2018.