Machine Learning for Predicting Tennis Match Outcomes

Tobin Otto

December 2024

Abstract

Tennis is an international sport with millions of fans. The margins of any given tennis match can be very thin and as such predicting tennis matches using machine learning can give a statistical-backed reason why a player came out on top. In the work we test this idea using two machine learning algorithms: first a simple logistic regression, and second a neural network. We then tested these models on professional tennis matches and resulted with high accuracy scores for both models. We found that our models statistically similar and performed just as well as one another.

1 Introduction

Tennis is the world's 4th most popular sport and garners fans and players from many different nations. Statistically, tennis is an interesting phenomena. For every point in tennis a multitude of statistics can be observed. The margins for winning a match can be very slim. These tight differences across a wide range of match statistics can make it hard for a human to understand and predict who the winner of a given match will be.
While it may be hard for a human to make sense of this data, machine learning algorithms work well with vast amounts of statistics. In an effort to see how machine learning algorithms work in classifying tennis match outcomes, we trained two model: a Logistic Regression model, and a Neural Network model to see how accurate they were at predicting match outcomes.

2 Background and Related Work

Tennis is unique compared to other modern sports. While factors like no time element, convoluted scoring, and multiple surfaces might deter fans of more linear sports, tennis creates interesting statistics that no other sport can replicate. Keyly, tennis is played between two players in a series of points. This means that the end result will always be one player winning a point and one player losing. This creates many symmetric statistics that we had to deal with in our

model building.
Using machine learning to predict tennis matches is not a new phenomena. The many statistics measured for a given tennis match can be alluring for anyone capable of implementing basic machine learning techniques. Yet consensus on the best model to predict tennis matches is still very much in develop. Recent attempts have concluded that a simple neural network is the best model [1]. However, novel approaches to support vector machines have achieved similar prediction accuracy scores [2]. Thus it seemed that training and testing multiple models could lead to more informative results.

3 Problem

The task of tennis match predictions may seem simple from the outside. Given two players what is the likelihood one or the other will win. Yet this becomes more difficult when viewing tennis from a statistically perspective. Because tennis can be broken down into a series of points, played between two players, the end results is always that one player won a given point while the other player lost. This sets the stage for symmetric statistics: if Player 1 won $x/total$ points, then Player 2 must have won $total - x$ points. This meant that when collecting data for tennis matches much of what was initially collected was correlated data. To properly train our models we needed to create independent variables.
We started by collecting raw datasets from Kaggle for ATP (Association of Tennis Professionals) for the 2010 to 2020 seasons [3]. Each row of the dataset represented a different match from the given season. Each of the 49 initial columns represented a variable for a given match. Many of these variables were deemed unmeasurable or unrelated to match prediction and so were dropped from our training set. After filtering these columns were left 21 variables. Three of these variables were universal statistics for the given match: surface, tournament level, and best of how many sets (3 or 5). The remaining 18 were 9 statistics for both the winner and loser of the match. The attributes for the winner are list below:

winner name
winner aces
winner double faults
winner total serve points
winner 1st serves in
winner 1st serve points won
winner 2nd serve points won
winner break points saved
winner total break points faced

Fig. 1: Raw statistics for match winner

Given these integer-valued statistics we needed a way of comparing them across other matches and players. using total service points and total break

points faced we refactored each of these statistics into the following percentages:

winner ace %
winner double fault %
winner 1st serve %
winner 1st serve points won %
winner 2nd serve points won %
winner break point conversion rate %

Fig. 2: Percentage statistics for match winner

and did the same for the loser's statistics.
To properly train our models we needed our data in the form:

$$instance =< attributes, label >$$

To to this we first randomly assign the winner and loser of each match as Player 1 or Player 2. Next we create labels were the the player that is assigned to the winner is listed. This will be the target we train our models to predict. Next, to create non-symmetric data, we create single independent attributes in the form:

$$attribute = percentage_{player1} - percentage_{player2}$$

This represents the original statistics from our raw dataset, without having the delineate between our winners and losers. Thus with our data full processed we represent each match as a list of the following attributes with a label of either player1 or player2.

ace %
double fault %
1st serve %
1st serve points won %
2nd serve points won %
break point conversion rate %

Fig. 3: Complete statistics for match (represents both winner and loser)

4 Solution

We then used this processed data to train our Logistic Regression and Neural Network models. We merged each season of our data into one larger dataset. Additionally, we encoded our categorical variables (surface, tournament level, best of) using onehot encoding without first dropping. Thus our final dataset prior to training and test splitting had a total of **15** attributes. We then split our dataset into a training and test set with an **80**% **20**% split, and shuffled our data with a seed of **2200**.

4.1 Feature Selection

Before training our models for testing, we conducted a feature selection to determine which models were actually significant for classification for each model. After reviewing Sipko's own paper on tennis match prediction I was aware of the influence specific factors like surface could have on results and thus wanted to account for these differences [4]. Using an automated feature selection process we found that **13** of our original 15 attributes were significant for our neural network model, whereas only **3** attributes were significant for our logistic regression model. The 13 features utilized in our neural network can be attributed to the fact that we did not drop the first binary variable made after hot encoding, which is not needed due to the ability to represent categorical variables with $n-1$ binary variables, where n is the number of discrete values for our original categorical variable. The significant filtering in attributes for our Logistic Regression model could be partially attributed to logistic regression's reliance on a linear regression model, which can become over complicated with too many features. The attributes used for each logistic regression are then listed below.

Hardcourt
Grass
Grand Slam
Masters 1000
Bo3
Bo5
ace %
double fault %
1st serve %
1st serve points won %
2nd serve points won %
break point conversion rate %

Fig. 4: Selected attributes for Logistic Regression

and for our neural network.

1st serve points won %
2nd serve points won %
break point conversion rate %

Fig. 5: Selected attributes for Neural Network

4.2 Model Overview

Logistic Regression works by first building a linear regression model. The output of the linear regression model, a real-valued number is then the input to the **Sigmoid** function which translates the value to a decimal between 0 and 1. The classification of our data is then decided where values over 0.5 are labeled

as player1 and values equal and below 0.5 are labeled as player2.
Our Neural Network was trained using the same split as before but now with with an **80**% training, **20**% validation split. We then conducted a Gridsearch to determine the optimal hyperparameters for our network, namely **number of hidden neurons** and **optimizer learning rate**. After comparing validation accuracy we found that **32** hidden neurons and a learning rate of **0.01** gave the best performance. Additionally, we used Cross Entropy as our loss function.

5 Experimental Setup

5.1 Performance Measures

To measure the performance of our models on our tasks, we utilized the following functions.

Classification Accuracy: A proportion of correct predictions out of all predictions.

```
# calculates the accuracy of the predictions of a given
    classification model
def calculate_accuracy(model, X, y):
    if isinstance(model, torch.nn.modules.container.Sequential):
        # make predictions for the given X
        X = torch.from_numpy(X.values).float()
        y = torch.from_numpy(y.values).long()

        if torch.cuda.is_available():
            device = torch.device('cuda')

            X = X.to(device)
            y = y.to(device)
            model = model.to(device)

        softmax_probs = model(X)
        predictions = torch.argmax(softmax_probs, dim=1)

        # the calculate accuracy of those predictions
        accuracy = sum(predictions == y) / len(predictions)
    else:
        predictions = model.predict(X)
        correct = sum(predictions == y)
        accuracy = correct / len(X)

    return predictions, float(accuracy)
```

Notably accuracy is calculated differently for our neural network than for our logistic regression.

5.2 Solution Implementations

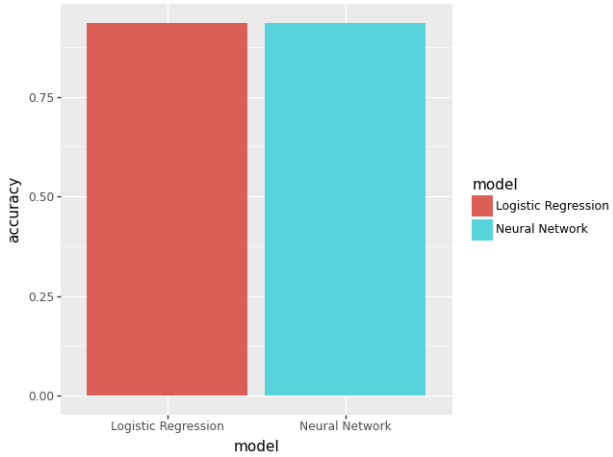
To build our models and to measure their performance we utilized the Python programming language, including a number of modules:

- `Sci-Kit Learn`
- `Pytorch`
- `Plotnine`

6 Results

Using these implementations we tested our logistic regresion model and our neural network model on our test sets to achieve the following performance.

6.1 Performance



Here we can see that both our logistic regression and neural network model performance with high accuracy on our test set. To understand whether there is any difference in utilizing our neural network model over our logistic model we calculated a confidence interval around both model's performance.

6.2 Confidence

Logistic Regression	(92.89% , 94.22%)
Neural Network	(92.28% , 94.13%)

Fig. 6: Confidence Intervals for model's performance

Here we can see that our logistic regression is not statistically significant from our neural network model. From our experiment it can be concluded that feature selection plays a large role in predictive accuracy. While our logistic regression model performed poorly with all 15 of our attributes, it improves to similar performance as our neural network when only using 3.

7 Conclusions and Future Work

Tennis is a sport that captivates a vast range of people from around the world. The unique structure of the sport make it hard for anyone person to characterize the winning qualities of a given match. Thus, we set up and experiment to test the capabilities of machine learning algorithms to predict tennis match outcomes. Using ATP data from the last ten years we created a dataset that proportionally represented each match, and their labeled outcome. Using this dataset we trained two machine learning model: a logistic regressor, and a simple neural network. Using features selection and gridsearch we could optimal variations on these two models and tested them on the rest of our dataset. We achieved high accuracy for both of our models and found that they were statistically the same even though their specifications are very different.
Tennis as a game is changing rapidly. With the likes of new players like Carlos Alcaraz and Jannik Sinner pushing the envelope of what is possible on a tennis court, new statistics and indicators of performance will surely arise. Thus, there is much that could be investigated in the future. Of the popular machine learning models not considered in this work, Random Forest and Decision Tree algorithms stand out as potential candidates for more specific predictive experiments. Additionally, the research done by Yilin Lei on training models to recognize key factors like momentum in a match work to include machine learning models like support vector machines. There really is no limit to what can be learned as tennis boasts numerous statistics and routes to victory. While our results are a good primer into machine learning's capabilities in predicting match outcome, in the future I hope to utilize more novel approaches to under significant factors for tennis match outcomes.

8 References

[1] Sgarra, Claudia, Montemurro, Andrea. (2022). Tennis match prediction using machine learning.

[2] Lei, Yilin. (2024). Rhythms of victory: predicting professional tennis matches using machine learning. IEEE Access.

[3] https://www.kaggle.com/datasets/afafathar3007/dataset/data

[4] Sipko, Michal. (2015). Machine learning for the prediction of professional tennis matches. Imperial College London.