Tennis Match Prediction

Guanchen Liu

December 13, 2022

MSML602

Abstract

In this project, we implement Machine Learning models to sports prediction by focusing on the data available of the statistics of previous ATP matches for each individual ATP player. We implement and train Neural Networks model in the classification of match results. The model is then used to predict current and future matches, and the predicted results is displayed onto a website.

1. Introduction

Sports prediction is a challenging problem as it incurs unpredictability and have use in areas such as betting, and entertainment. Traditional tennis prediction is heavily based on the 1st server winning percentage and combined with the seed, and the ranking of the player. This introduces shortcomings as with the limited features and the simplicity of the model, the model will miss other data that may show useful in prediction. However, as more and more sports are being modeled by machine learning algorithms, we will focus on predicting match outcomes based on machine learning. We will implement and train a neural network model for the classification of ATP match results and display the results onto a website.

2. Dataset and Features

2.1 Data Collection

For this project, we will be using two sources for our dataset.Many tennis match datasets require costly subscriptions to access. For this reason, this project will use tennis_atp opensource repository from GitHub(https://github.com/JeffSackmann/tennis_atp) to acquire previous match data. This dataset contains matches from current times all the way back to 1970s, and we will use 2022 matches as training set. The data is frequently updated and contains the most up to date and complete statistics of the matches that is available for free, with some matches missing statistics and players missing data points.

The second dataset that is used for the project is data for any current or future matches. The initial plan was to scrap the data from the internet, primarily from sportsbook sites, as there isn't any good location to acquire future matches as these tends to be grouped by tournaments compared to a masters list. However, these sites are all equipped with anti-scripting scrips, and required solving captchas while using selenium headless browser. As a result scraping was proved to not be an easy job to do in this case. While browsing match display sites, the site https://live-tennis.eu/en/atp-live-scores appears to be promising as it used a widget from another third party match display site https://tnnslive.com/widgets/allscores?host=ltr. However, these two sites are still unable to be straightforwardly scraped as they are anti-scraping as well as the match chart being written in JavaScript. By inspecting the network page of the website, we observe that the site is making API calls to acquire the match data. By copying the request link(https://tnnz.io/api_web?date={YYYY-MM-DD}&mode=matches_daily&timezone=America%2FNew_York&language=en&platform=web&version=100&subscribed=%7B%7D&favorites=%7B%7D&theme_settings=%7B%7D) and use it to access the API directly getting match data of the specific date.

Feature Building

The tennis_atp dataset contains 49 features, ranging from tournament data, player data, match statistics, and player's rank. For the training of our Neural Network, we will solely focus on player information and their match statistics. Any descriptive information, such as tournament name, player name, match types will be dropped as well as any duplicate features that can be inferred from other features.

Next, we sanitize the dataset. Any match entries that are missing player statistics will be dropped, as the model relies on these data for training and predicting, and we cannot recover or fill in these data. The dataset is unprocessed and includes features that are text entries. These only have a few possible options, and we will convert them into numerical values.

The dataset is organized in a way that separates the winners and losers into different sets of columns. If we were to send this straight into a machine learning model to train, it would not be able to classify any real world matches as it would aalways classify any player in the winner's column as the match winner. To solve this issue, we need to randomize the location of the winner/loser. We will create a new column called results and use the pseudo random function that numpy has(np.random) to assign the location of the players and assign numerical value 0 if the player located in the left columns lose and 1 if they win. This will ensure that the model weights do not bias towards a set of columns during training.

We were able to acquire data by sending GET requests to the API website. The site responded to the request call and data was received. This API endpoint contains raw data and that its format dynamically changes according to the date. However, since the two websites were able to display the data from the API call to a chart, then it must have some way of decoding the JSON response file such as a index that is stored in the response itself. We will need to reverse engineer the response and extract the information we need.

By examining the JSON file, labels were discovered in the first section of the JSON, which contains the description of the data included. We determined the indexes of the titles that we are concerned with and converted from integers of base 10 to alphanumeric values of base 36. With these indexes found, we were able to iterate through the JSON response and call the index to acquire the players participating in the match and its score.

However, some of the player names were not in plain text, but in codes of the format 'p:<alphanumeric numbers>'. Going through the JSON response, a section of the response contains names scattered throughout its list. It's found that the index of the player string matches the alphanumeric numbers that we obtained earlier. By converting the alphanumeric numbers of the player to integers and looping through the list, we were able to find names for all the players in the matches.

Now that a list of all current and future matches were available, we need to format the data to fit in to our Machine Learning model. We start by converting the names of the players into integer IDs, and new array was constructed with based on the player ids, filling in the rest of the features based on our training dataset. Features such as height, age, and handedness were referenced by the last match that the player played. The other match statistics data will be generated based on the statistics of the previous matches played. A weight of 0.4 was set for all previous matches while a weight of 0.6 was set for previous match against the same opponent. This will ensure that the previous matchup against the same opponent is influenced in the classification of winner.

For training of the machine learning model, we will rely solely on the tennis_atp match dataset, without considering current and future matches. This dataset is unique in that it is heavily relied on sequence, thus a standard randomized train-test-split would not be ideal, as the testing data will rely on previous match information. Instead, we will split the dataset in place, with a testing data to be 10% of the dataset. Before training, we will normalize the data using standard scaler function from scikit-learn. This will enable the model to make better prediction and improve performance. We will also store the scaler value as a binary file which will be used to normalize the data used to predict future matches.

3. Neural Network

For the machine learning model that will be trained by the tennis_atp match dataset, the model will be neural network. The neural network model will perform binary classification on the match data and is structured with four fully connected layers. The first layer is the input layer, which has neurons equal to the number of features in the training data with ReLU activation function. The next two layers are hidden layers with 37 neurons with ReLU activation function. The output and final layer will use sigmoid activation function which will output a value between 0 and 1.

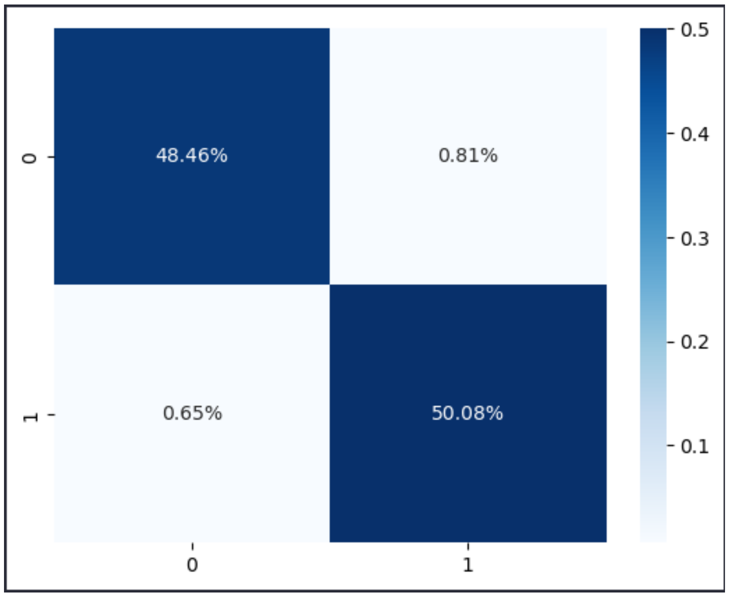
The model uses binary cross-entropy as its loss function to train the weights of the layers and optimizer with Adam. The output from the final layer will be the probability of the first player to win. We have set the threshold for the two classes to be 0.5, making the classification output 0 if the value is smaller than 0.5, and 1 if the value is greater than 0.5.

4. Results

For the training of the Neural Network, we will use a batch size of 20 and epochs of 100. F1-score will be used to determine the model's accuracy on the classification.

4.1 Testing model on testing data using original match statistics

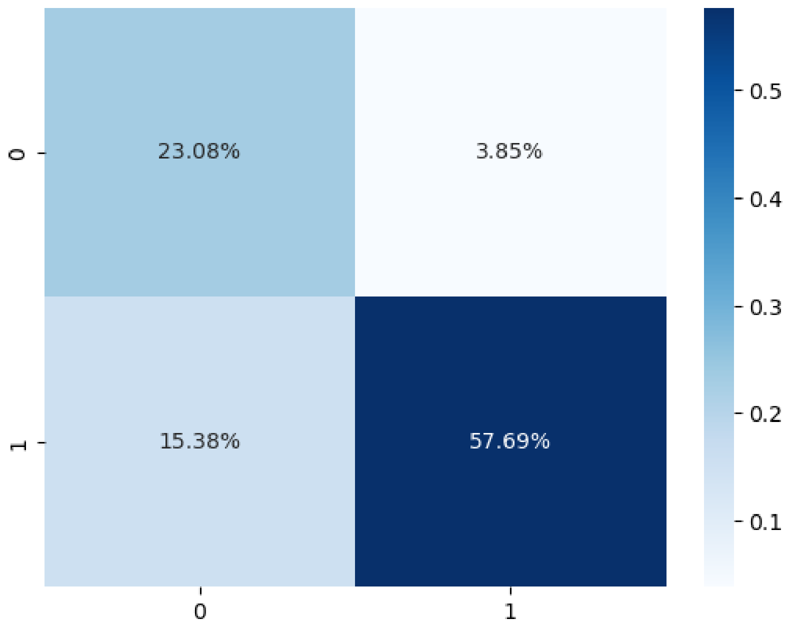
Before populating the testing data with data based on previous matches, its original match statistics is used in the testing of the robustness of the Neural Network model. By asking the model to predict the test data with no changes to the match statistics, we get a F1 score of 0.9856. The confusion matrix of the classification is:



The number of test sample is 1234 entries, with 628 actually valued at 1 and 606 valued at 2. The model can accurately predict on this balanced dataset.

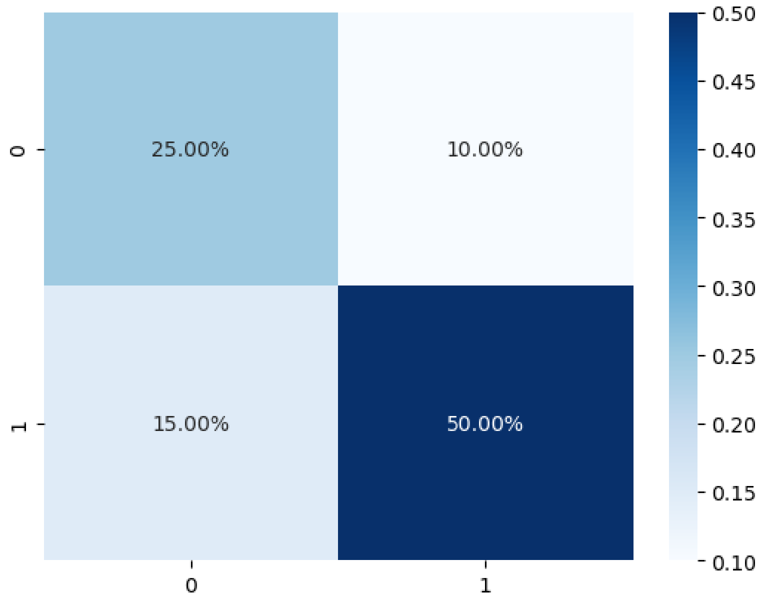
4.2 Testing model on testing data with generated match statistics

The model that was trained was able to accurately classify the results given the match statistics. What if the statistics were generated in reference to previous matches. By populating the model with generated statistics, we get a F1-score of 0.8571 with a confusion matrix of:

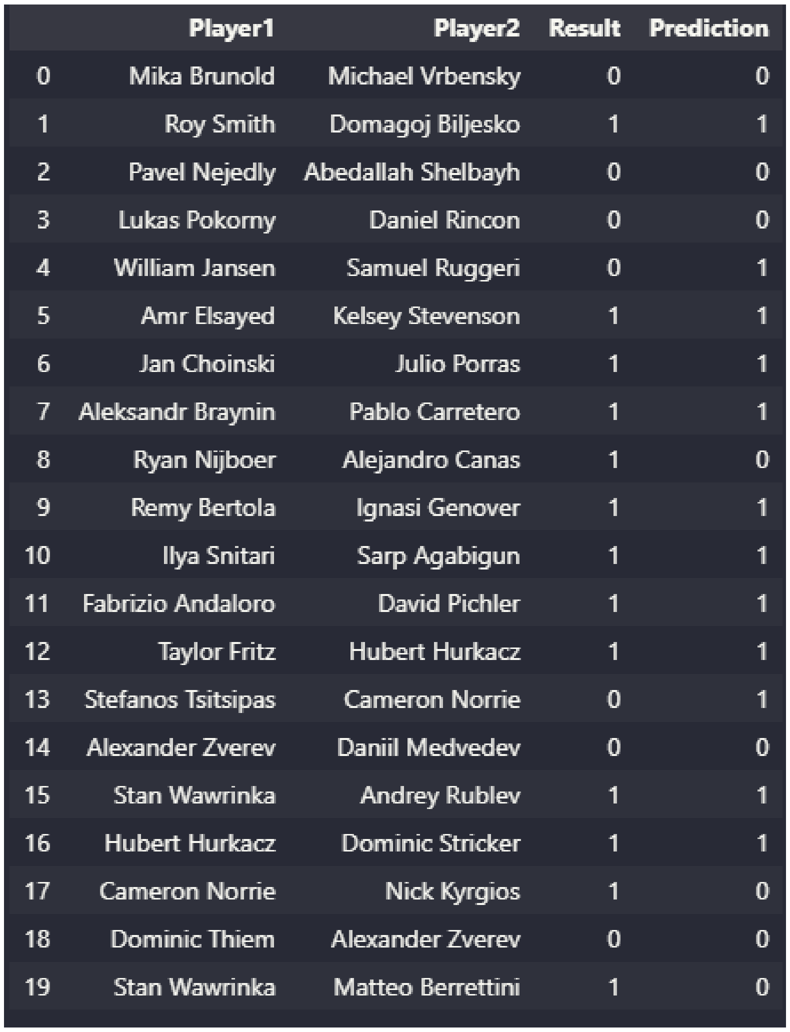


4.3 Testing model on future data

We will feed the model with the match data constructed from the API and populating the match statistics with previous matches. For testing purposes, we use the API data of the date 2022-12-08 as it includes number of matches from ATP men's category and these matches having concluded so we can perform comparison to the predicted result. From the model, we receive a F1-score of 0.8 with a confusion matrix of:



And the prediction results compared to actual results along with match player names:



	Player1	Player2	Result	Prediction
0	Mika Brunold	Michael Vrbensky	0	0
1	Roy Smith	Domagoj Biljesko	1	1
2	Pavel Nejedly	Abedallah Shelbayh	0	0
3	Lukas Pokorny	Daniel Rincon	0	0
4	William Jansen	Samuel Ruggeri	0	1
5	Amr Elsayed	Kelsey Stevenson	1	1
6	Jan Choinski	Julio Porras	1	1
7	Aleksandr Braynin	Pablo Carretero	1	1
8	Ryan Nijboer	Alejandro Canas	1	0
9	Remy Bertola	Ignasi Genover	1	1
10	Ilya Snitari	Sarp Agabigun	1	1
11	Fabrizio Andaloro	David Pichler	1	1
12	Taylor Fritz	Hubert Hurkacz	1	1
13	Stefanos Tsitsipas	Cameron Norrie	0	1
14	Alexander Zverev	Daniil Medvedev	0	0
15	Stan Wawrinka	Andrey Rublev	1	1
16	Hubert Hurkacz	Dominic Stricker	1	1
17	Cameron Norrie	Nick Kyrgios	1	0
18	Dominic Thiem	Alexander Zverev	0	0
19	Stan Wawrinka	Matteo Berrettini	1	0

5. Application and Deployment

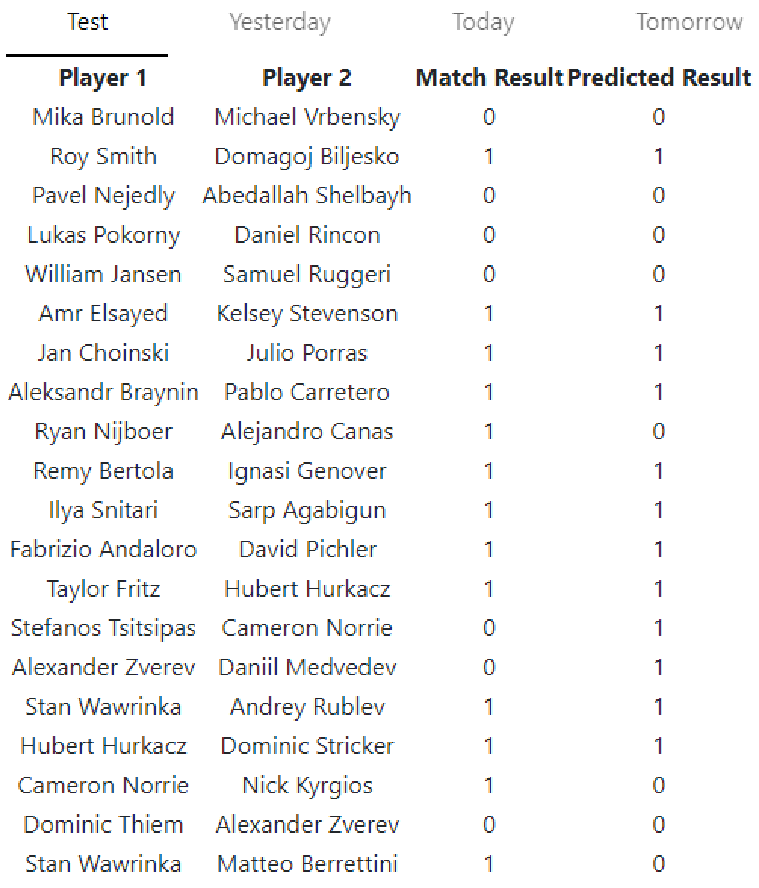
The goal of this project is to predict as well as display the prediction and update it automatically. To do this, the application is separated into two parts, frontend and backend. Due to time constraints, the development of the frontend and backend will follow the software engineering practice of MVP (minimum viable product) and may lack CI/CD practices such as testing and linting components.

5.1 Frontend

The frontend will be a ReactJS static website. It is a simple website that will perform API calls to get the prediction information and display it on a table. The website will have a tab of days that the backend will run to make predictions. It will include a test day, the day before, current day, and the next day. The table will include all matches that day that can be predicted, which entails both players information exist in the database and having ATP official match played in the current year. This will exclude any doubles matches, or any female matches as it is not included in our model. Match prediction will display alongside with the match result, if the match has not been played, or has been cancelled, the match result column will be blank. If no matches occur on a specific date, or no compatible matches is available, the website will display “No Matches Available”.

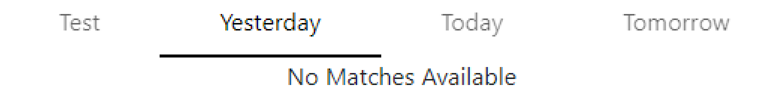
Below is the demo of the website running as of 2022-12-13.

Website displaying the test case:



Player 1	Player 2	Match Result	Predicted Result
Mika Brunold	Michael Vrbensky	0	0
Roy Smith	Domagoj Biljesko	1	1
Pavel Nejedly	Abedallah Shelbayh	0	0
Lukas Pokorny	Daniel Rincon	0	0
William Jansen	Samuel Ruggeri	0	0
Amr Elsayed	Kelsey Stevenson	1	1
Jan Choinski	Julio Porras	1	1
Aleksandr Braynin	Pablo Carretero	1	1
Ryan Nijboer	Alejandro Canas	1	0
Remy Bertola	Ignasi Genover	1	1
Ilya Snitari	Sarp Agabigun	1	1
Fabrizio Andaloro	David Pichler	1	1
Taylor Fritz	Hubert Hurkacz	1	1
Stefanos Tsitsipas	Cameron Norrie	0	1
Alexander Zverev	Daniil Medvedev	0	1
Stan Wawrinka	Andrey Rublev	1	1
Hubert Hurkacz	Dominic Stricker	1	1
Cameron Norrie	Nick Kyrgios	1	0
Dominic Thiem	Alexander Zverev	0	0
Stan Wawrinka	Matteo Berrettini	1	0

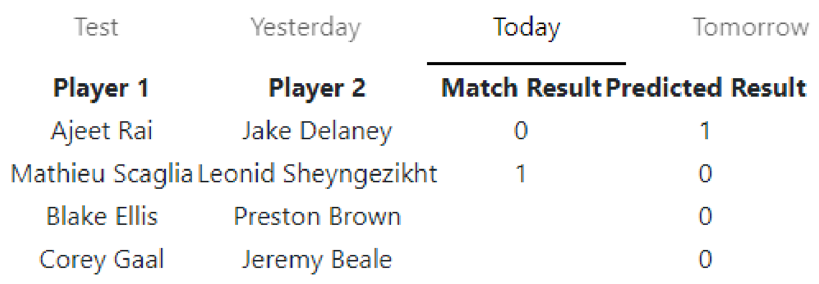
Website displaying yesterday’s match:



No Matches Available

There were no compatible matches available yesterday, as there were only matches from women's series and lower tier men's series.

Website displaying today's match:

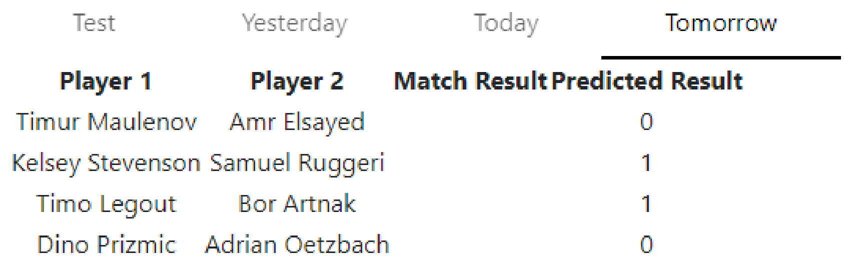


Test Yesterday Today Tomorrow

Player 1	Player 2	Match Result	Predicted Result
Ajeet Rai	Jake Delaney	0	1
Mathieu Scaglia	Leonid Sheyngezikht	1	0
Blake Ellis	Preston Brown		0
Corey Gaal	Jeremy Beale		0

Some matches were already played, thus the match results are up, while others have not taken place yet.

Website displaying tomorrow's match:



Test Yesterday Today Tomorrow

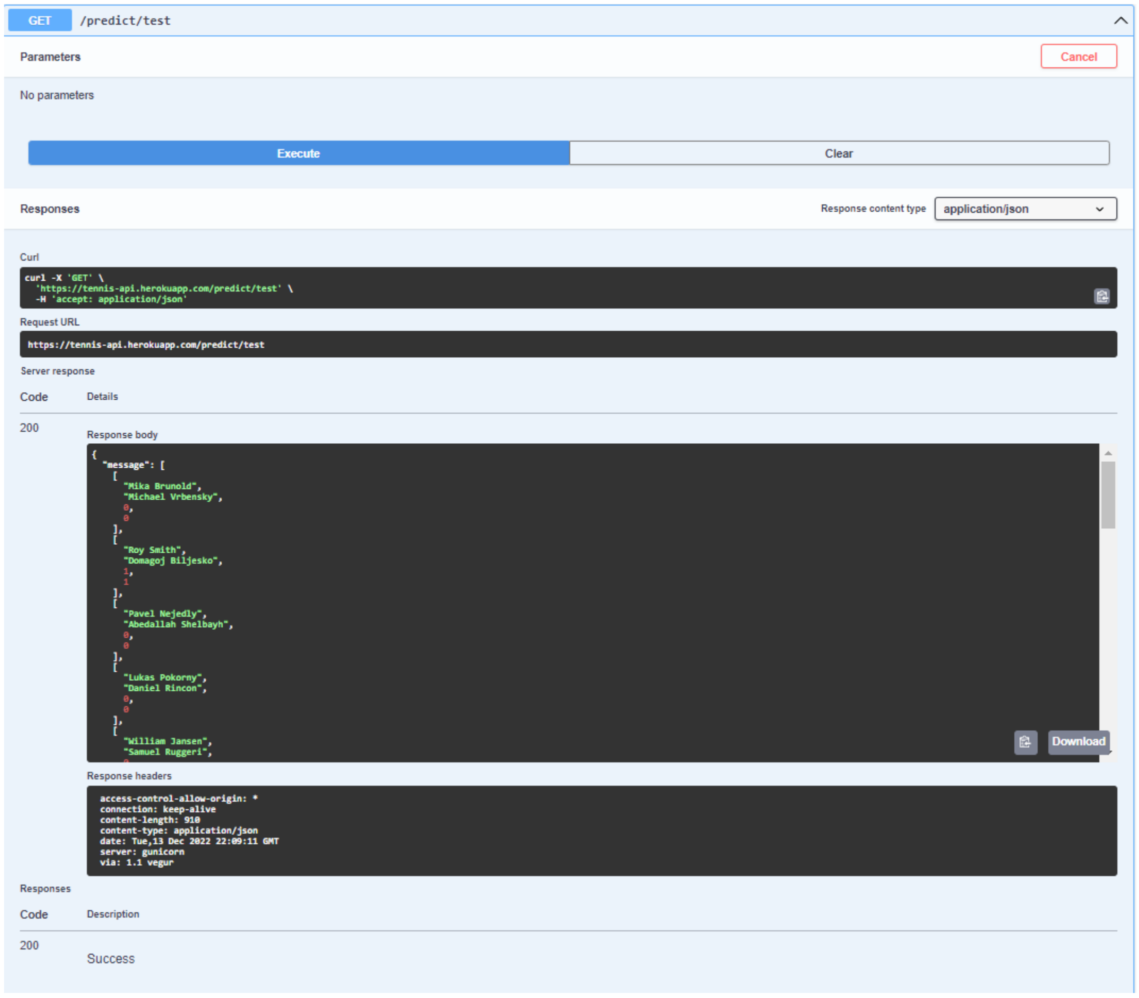
Player 1	Player 2	Match Result	Predicted Result
Timur Maulenov	Amr Elsayed		0
Kelsey Stevenson	Samuel Ruggeri		1
Timo Legout	Bor Artnak		1
Dino Prizmic	Adrian Oetzbach		0

5.2 Backend

The backend of the application will be a REST API implemented in Python using Flask-Restx. Previous machine learning code written in JupyterNotebook were refactored into .py files able to be called from the Flask app. The Flask app will have four prediction endpoints directed to test, yesterday, today, and tomorrow displayed on the website. The endpoints of the API can be accessed from the website as Swagger is built into Flask-Restx.

For the sake of runtime, and the quality of prediction, the model used has been pre-trained with the dataset and is stored as a file in the directory. Same thing goes for the standard scaler value, it is stored within the file directory and loaded and applied to the prediction dataset. These files are meant to be updated manually periodically to ensure the best predictions.

API response displayed on Swagger:



Curl

```
curl -X 'GET' \
  'https://tennis-api.herokuapp.com/predict/test' \
  -H 'accept: application/json'
```

Request URL

```
https://tennis-api.herokuapp.com/predict/test
```

Server response

Code Details

200 Response body

```
{
  "message": [
    [
      "Mika Brunold",
      "Michael Vrbensky",
      0,
      0
    ],
    [
      "Roy Smith",
      "Domagoj Biljesko",
      1,
      1
    ],
    [
      "Pavel Nejedly",
      "Abedallah Shelbayh",
      0,
      0
    ],
    [
      "Lukas Pokorny",
      "Daniel Rincon",
      0,
      0
    ],
    [
      "William Jansen",
      "Samuel Ruggeri",
```

Response headers

```
access-control-allow-origin: *
connection: keep-alive
content-length: 910
content-type: application/json
date: Tue,13 Dec 2022 22:09:11 GMT
server: gunicorn
via: 1.1 vegur
```

Responses

Code Description

200 Success

5.3 Deployment

Both the React and Flask app are being deployed onto Heroku.

Link to the backend Flask app: https://tennis-api.herokuapp.com/

Link to the frontend React app: https://tennis-heroku.herokuapp.com/

Due to resource constraints of Heroku's basic plan, both apps will be asleep after a short period of inactivity. Thus, to access both apps without issues, the backend app needs to be started first. Simply click the link belong to the backend app and wait until the Swagger UI boots up*. Then click the frontend link and wait until it boots up, then the matches for each day will be called and the API response will be displayed.

*Note: it may be a few minutes for either apps to boot up

6. Conclusion and Future Work

In conclusion, the model developed shows strong results on past and real time results. Achieving 80% or higher F1-score overall results shows that the model is able to correlate strongly in the comparison of players and their performance.

However, there are some drawbacks with the data used and the potential to improve the overall performance of the model dramatically. One area to consider is the relevancy of the information. Player performance varies a lot, and one year can be a long time that the stats for the older games are no longer relevant and representative to the performance of the player in current times. Therefore, in addition of having weights factor in whether the stats is with the same opponent, we can include a time series weight, which will weight the stats of recent games much higher than the previous ones.

In addition, there can be much to improve than using raw match statistics. Limitation to the data availability makes us unable to use percentages in our analysis. Instead of how many breakpoints are saved by a player, if we have the number of total breakpoints, we can know the breakpoint percentages that are being saved. If we populate the dataset with features like this, the models performance is set to increase. A confidence of performance feature can be added, which can provide insight into the number of matches the player has played in a certain period. If a player have incredible stats but has only played one game in the span of a year against a rookie, his stats cannot be compared to another player who has played dozens of games this year but have a slightly worse numbers.

Further research into different algorithms could improve the prediction accuracy as well. Instead of having to train the model on all matches played, we could explore Long Short-Term Memory (LSTM) and train the model on matches that only the two players that we are trying to predict has played. Though Neural Networks usually requires large sample size, due to the volatility of the sport, with a short memory of the network, the model may be able to provide accurate predictions.