Is He In The Zone?

Team 242187

February 2024

1 Summary

Novak Djokovic, 23 time grand slam champion was almost destined to win wimbledon in 2023, but came up short, against the 1 time grand slam champion, and up and coming star Carlos Alcaraz in the final. Data is provided indicating a vast array of advanced analytics in tennis matches, from the third round of wimbledon to the finals. But the question remains how did Novak Djokovic lose a match up he seemed destined to win, how does the momentum of a tennis player change relative to time throughout a match. Using these advanced analytical statistics, it is possible to determine how momentum shifts throughout a match of tennis.

Through the dataset that was given, the data was divided into three groups, high, medium and low priority forms of data when determining momentum in a tennis match. It was also noted that momentum was not carried through rounds, just because a player won a specific way, their momentum would not be carried forward to the next round, its assumed that both players will have an equal amount of momentum. The approach that was taken to determine momentum swings was through the use of probability, the probabilities for each model the training set was iterated over and manually counted if the corresponding player won the point according to the specific arguments each variable took, for example, the probability of a player winning a point given that they served over 125 mph. The three models that were created to project momentum is the serve model, the set game model and the distance travelled model. Each of the models were tested on an evaluation dataset, where it was found that the serve and distance travelled model were correct in predicting who would win a point roughly 65 and 59 percent of the time respectively. While the set game model was able to correctly project, 49 percent of the time correctly.

Contents

2 Background

The 2023 Wimbledon tournament was full of surprises. The crowd and odd favored, Djokovic, was destined to win the tournament. During his last match against Alcaraz, Djokovic lost his momentum multiple times throughout the match, causing Alcaraz to regain momentum and win the final round. Within Tennis, the scoring system utilized can accurately reflect almost all events that take place on the court [1]. In professional single men's tennis matches during Wimbledon, there are only 2 players on the court at a time. A player wins a game upon reaching 4 points but they must win by 2 points. To win a set a player must win 6 games but players must win by 2 games, upon reaching 6-6 there is only one tiebreaker played [2]. Matches are the best of 5 sets which ultimately determines the winner.

With this format, probabilities at the micro and macro levels of tennis can be obtained and subsequently utilized to predict the flow of the game. The micro level of tennis covers more detailed aspects of the game, like serve depth, net points, server, serve speed, distance covered, etc. The macro level of tennis contains the sets won, games won, matches won, and the victors of games, sets, and matches. Change in momentum can be seen in both levels of the game but, is more prevalent in the macro level. [2]

Other work that has developed a model for a tennis match is outlined in Barrett's [3]. Where the only two parameters are the probabilities of each player winning a point based on their serve. The use of a markov chain to model a tennis game can be found throughout other literature due to tennis's inherent hierarchical scoring system. Within each round the points scored are all independent of one another [1]. Markov chains were also utilized to further develop new models and to gain insights into tennis's structure [4].

A model has been developed to capture the dynamics between both players in an ongoing round of tennis. It will be able to identify and quantify who has more momentum at a given time during the round. The model also assists in determining the effect of momentum on a round of tennis. Utilizing the supplied data from the 2023 Wimbledon tournament we can examine which factors influence crucial points during a game. Allowing us to predict pivotal moments in time during a round of tennis.

3 Assumptions

When developing our model, our first assumption was that momentum from a previous round of tennis does not carry over into the next game.[5] For example, Alcaraz winning in round 3 his momentum from winning that match would not carry over, as he is facing another player who just won a match to reach that round, so we can assume the momentum at the beginning of each match would be equal. This was done to make the rounds independent, simplifying our model and introducing some practicality. Also, as players can enter different mental spaces before entering their next round. This assumption also allows for the model to focus on the higher priority micro and macro levels of tennis. In doing so, it enables the model to be applied across various rounds and different players, including different genders.

The second assumption was that players have constant behavior under similar circumstances during a round of tennis.[6] This made the data easier to manage and interpret, as we assume players' behavior stays constant. In doing so, it allows for our model to focus on the different metrics of tennis; who has serve advantage, point number, games won, and sets won to predict the players performance at a given time. Without having to consider different strategies and play styles a professional tennis player might utilize during a round. With this assumption it may not cover all the intricacies of tennis but it simplifies the modeling process.

4 Solutions

Table 1: Priority

High Priority	Medium Priority	Low Priority
$P_{1,2,sets}$	Point Victor	Match ID
$P_{1,2,games}$	$P_{1,2,ace}$	Elapsed Time
$P_{1,2,score}$	$P_{1,2,breakpoint}$	$P_{1,2,PointsWon}$
Server	$P_{1,2,doublefault}$	$P_{1,2,Winner}$
Serve Speed	$P_{1,2,unforcederrors}$	Winner Shot Type
Total Distance	Rally Count	Net Points
—	Set Victor	Net Points Won
—	Game Victor	Serve Width
—	—	Serve Depth
—	—	Return Depth
—	—	Serve Number
—	—	Point Number
—	—	Set Number
—	—	Game Number

Categorizing all the variables into different priorities allowed us to approach this problem more strategically and focused. The variables were sorted to high, medium and low priorities. High priority variables directly impact the flow of the game and can help to depict which player is performing better at a given time. Medium priority variables still contribute to the flow of the game but with less influence. They are also previously covered by variables under high priority (i.e., P1, 2 Score covers P1, 2 Ace as when an ace occurs P1, 2 obtains the point, it is redundant). Low priority variables have minimal impact, provide irrelevant and new information but can be utilized in later stages of the round. The variables received from the data set have been subsequently organized into respective columns depicting their influence to the flow of the game. Shown in Table 1.

P1, 2 score, games, and sets are the major factors that influence the flow of the game. These quantities provide the fundamental insights into the current flow of the game at a given point in time. These variables can be utilized to determine the momentum of a certain player, when a player rapidly wins a succession of games they are more likely to have more momentum than their counterpart. If each player was performing equally, then their momentum would be equal, more so leaning on who has the serve advantage. Serve advantage and serve speed is under high priority as the player who is serving has a higher probability of winning the point/game and higher speeds equate to an increase in probability to obtain the point. Distance traveled is also listed as high priority due to its influence on the individual players throughout the entire round. Depending on the distance traveled a player's energy level can decrease, lessening his momentum. Listing these as high priority variables 1 allows for us to capture the most important aspects that affect a players performance at a given time.

The medium priority variables largely reflect other variables that have been previously covered by the high priority variables, but still contain relevant information about a player's performance. All the variables listed under medium priority, except rally count, is essentially saying the same information (who won the point). Rally count was listed under medium priority as when the count was even, player 2 wins the point, and odd count player 1 wins. Therefore similar to high priority variables but offers information about the length of exchange between both players.

Low priority variables contribute information about how the point was scored and are more generalized than others. All variables under low priority except points won and winner do not assist in the development of the model due to their lack of relevant information. P1, 2 Winner and P1, 2 points won was placed under low priority, as other variables indirectly indicated who won the point, game, set, and match. These variables include point number, P1, 2 score, and point victor.

The data was split up into two sets, the first being round 3 and quarter finals, which was utilized to calculate our probability models. The second data set contained the matches taking place during the semi finals, and final round.

To calculate the probabilities for each model the training set was iterated over and manually counted if the corresponding player won the point according to the specific arguments each variable took. For example, to calculate the probability of winning if the player of interest serves over 125 mph the number of points where the player won the point and served over 125 mph was divided by the total number of times the player served over 125 mph.

These models were tested against the evaluation data set by giving each model the appropriate values for each point and if the probability of winning a point was above 50 then the model predicts the player would win the point. The success rate of each model was calculated and the results were as follows.

The set-game model was correct 49.14 percent of the time. This was to be expected, as it is easy to tell by visual inspection of Figure 3 that there is no clear correlation between the set and game score and the probability of winning a point. A disadvantage of the set-game model was that the number of sets per match increased in the finals to 10. Therefore, any games occurring in those sets could not have a probability of winning a point assigned to them and thus the model fails in those instances.

The serve model was correct 65.76 percent of the time. This was also expected as there is a clear delineation in Figure 2 between the probability of winning the point and the speed of the serve. The serve model was the most accurate of the three tested, however fails to account for double fault scenarios as the player does not have a valid serve speed.

Finally, the distance travelled model was correct 58.91 percent of the time. Although appearing to have a high correspondence in Figure 1, this model fails to account for which player serves, which would affect the minimum distances between 0 and 1 meters. This minimum distance would most commonly be from the player serving from a corner of the court, moving to the centre of the court to receive and gaining an ace point.

5 Figures

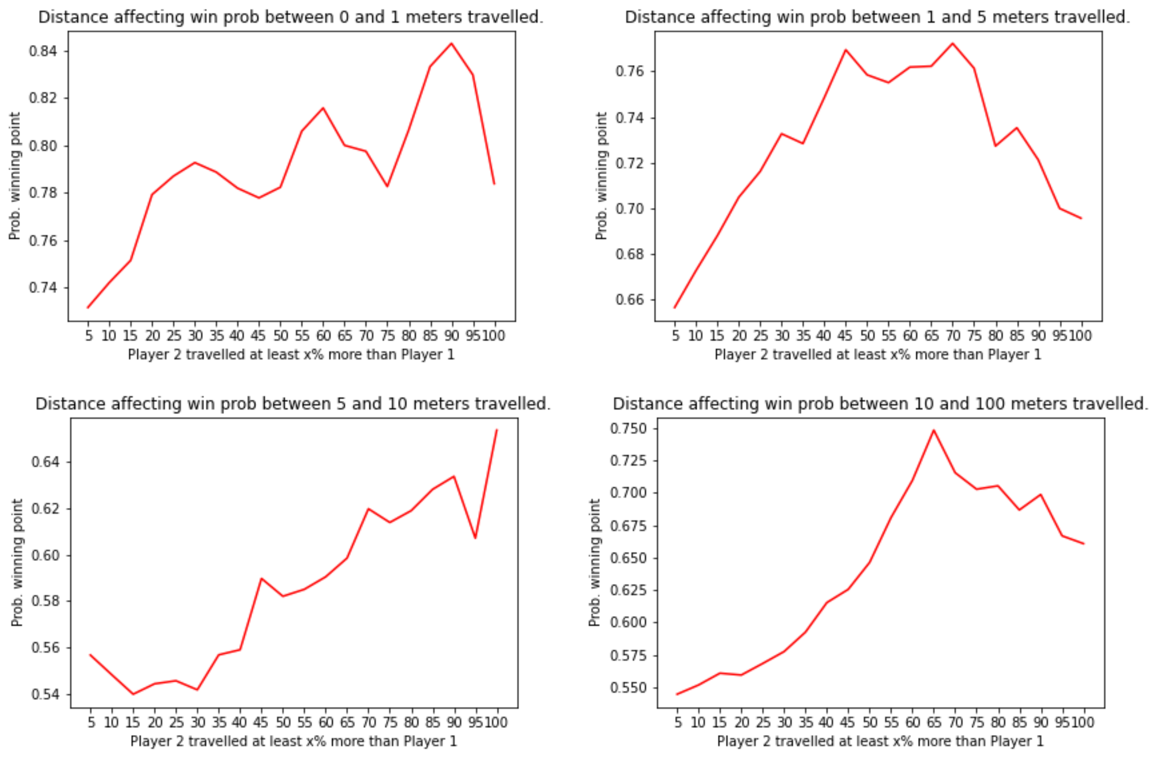

Figure 1: Distance ran graphs

Figure 1 displays the probabilities of player 1 winning a point when comparing the distance traveled between players in a single point. The distances range from 0-1 meters, 1-5 meters, 5-10 meters, and 10-100 meters. From our graphs, when player 2 travels a greater distance than player 1, player 1 is more likely to win the point. All graphs located in figure 1 have a positive correlation between the amount of distance traveled by player 1 when compared to distance traveled by player 2.

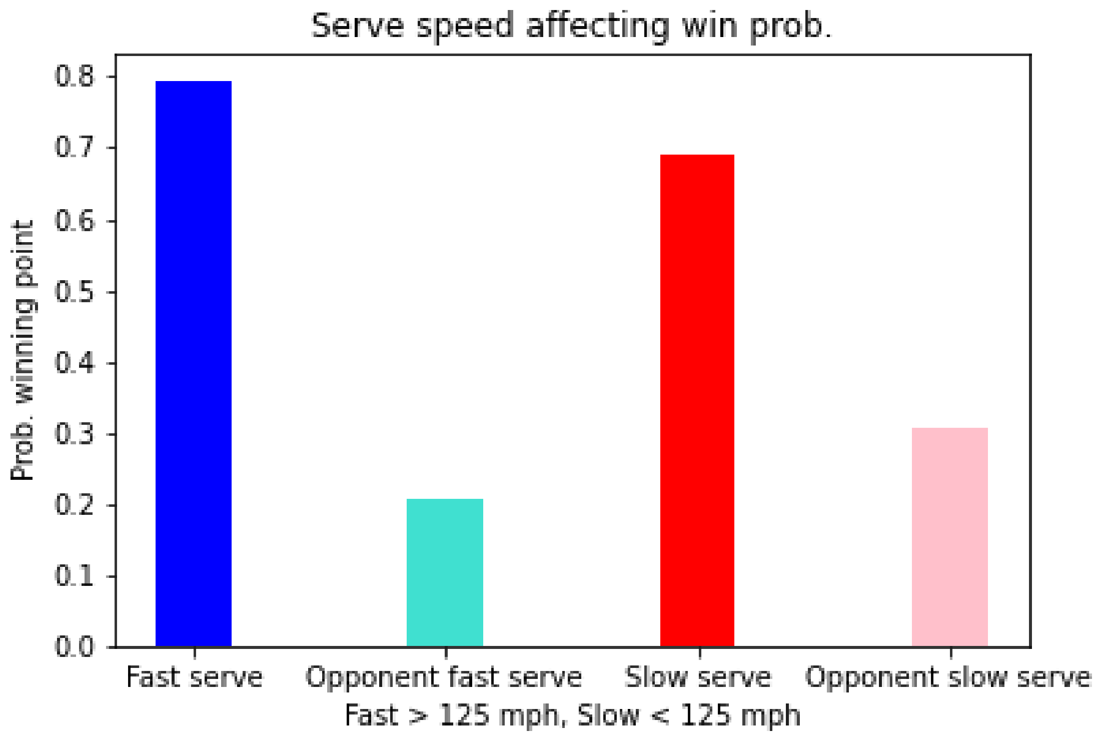

Figure 2: Serve graphs

Figure 2 displays how serve speed affects the probability of winning a point for player 1. From the graph it can be seen that when player 1 has serve and serves fast he has a probability of 0.8 to win the point, and 0.7 when serving slowly. When player 1 is receiving a fast serve he has a lower probability of winning the point than if it was a slow serve. This is in line with our prediction and further demonstrates how much of an advantage serving is.

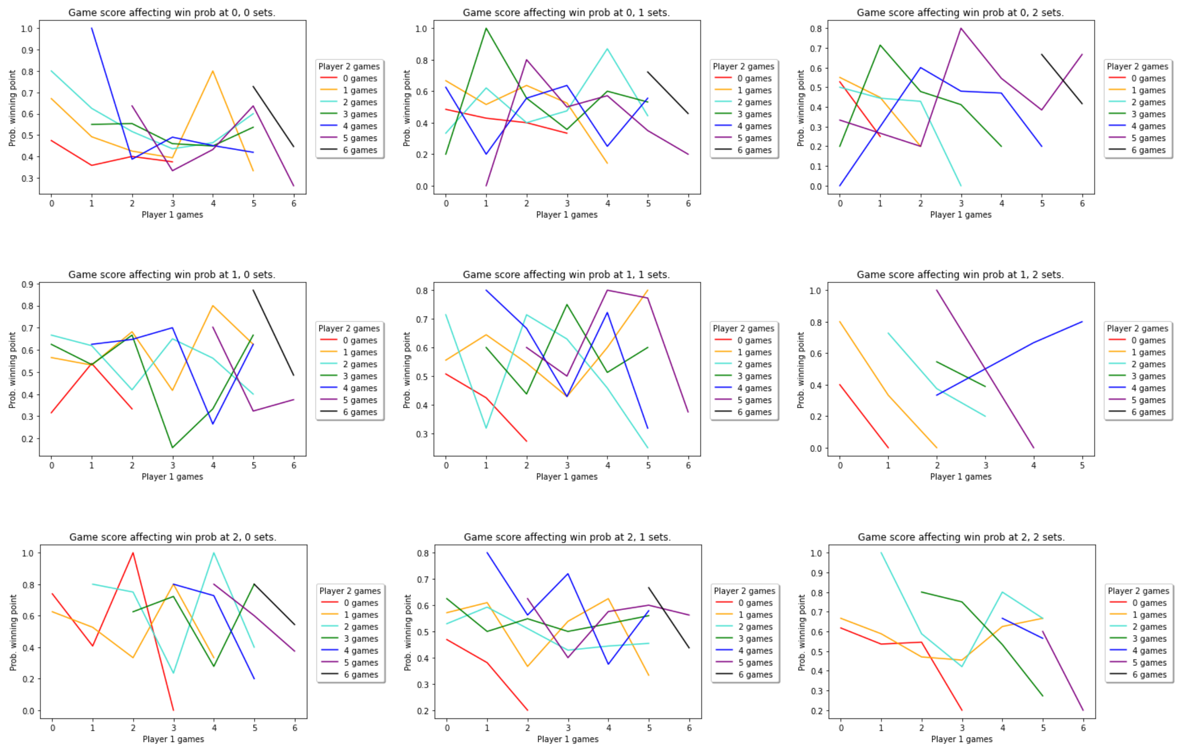

Figure 3: Set and game probability

Figure 3 displays the probability of winning a point depending on the set and game score. All possible combinations of set scores, ranging from (0-0) to (2-2) are shown. The amount of games currently won by player 1 is located on the x-axis, the probability of winning the point on the y-axis, and games currently won by player 2 is represented by the different lines, see legend on graphs. From Figure 3, there was no clear correlation between set score, game score, and the probability of winning the point.

6 Code

Model Evaluation code

```
import ast
import pandas as pd

# --------------------- EVALUATE MODEL PERFORMANCE --------------------
data = pd.read_csv("./Wimbledon_featured_matches.csv")

with open("score_chances.txt", 'r') as txt:
    scores = txt.readline().strip().replace('\'', '\"')
    score_chances = ast.literal_eval(scores)

with open("serve_speed_chances.txt", 'r') as txt:
    txt.readline()
    serve_speed_chances = ast.literal_eval(txt.readline().strip())

with open("marathon_chances.txt", 'r') as txt:
    marathon_chances = ast.literal_eval(txt.readline().strip())

eval_points = 0

score_correct = 0
score_outside_training = 0

serve_correct = 0

marathon_correct = 0

for player in [1, 2]:
    for index, row in data.iterrows():
        if int(row["match_id"][-4:]) >= 1500:
            eval_points += 1

            # ----------------- EVALUATE SET-GAME -----------------

            score = f"{row['p1_sets']}, {row['p2_sets']}, {row['p1_games']},
               {row['p2_games']}"

            try:
                if score_chances[score] > 0.5: # Model thinks we win point
                    if row["point_victor"] == 1: # We actually won the point
                        score_correct += 1
                else: # Model thinks we lose point
                    if row["point_victor"] == 2: # We actually lose the point
                        score_correct += 1
            except:
                score_outside_training += 1
                # print(f"SCENARIO {row} DID NOT OCCUR IN TRAINING SET")

            # ----------------- EVALUATE SERVE -----------------

            if row["server"] == player: # We serve

                if row["speed_mph"] > 125: # If we serve quickly
                    if serve_speed_chances[0] > 0.5: # Model thinks we win
                        point
                        if row["point_victor"] == player: # We actually won
                            the point
                            serve_correct += 1

                    else: # Model thinks we lose point
                        if row["point_victor"] != player: # We actually lose
                            the point
                            serve_correct += 1


                else: # If we serve slowly
                    if serve_speed_chances[1] > 0.5: # Model thinks we win
                        point
                        if row["point_victor"] == player: # We actually won
                            the point
                            serve_correct += 1

                    else: # Model thinks we lose point
                        if row["point_victor"] != player: # We actually lose
                            the point
                            serve_correct += 1

            else: # Opponent serves
                if row["speed_mph"] > 125: # Opponent serves quickly
                    if (1 - serve_speed_chances[0]) > 0.5: # Model thinks we
                        win point
                        if row["point_victor"] == player: # We actually won
                            the point
                            serve_correct += 1

                    else: # Model thinks we lose point
                        if row["point_victor"] != player: # We actually lose
                            the point
                            serve_correct += 1
                else: # Opponent serves slowly
                    if (1 - serve_speed_chances[1]) > 0.5: # Model thinks we
                        win point
                        if row["point_victor"] == player: # We actually won
                            the point
                            serve_correct += 1

                    else: # Model thinks we lose point
                        if row["point_victor"] != player: # We actually lose
                            the point
                            serve_correct += 1

            # ----------------- EVALUATE MARATHON -----------------

            if player == 1:
                if row["p1_distance_run"] > 0:
                    if (row["p2_distance_run"] / row["p1_distance_run"]) > 1:
                        if row["point_victor"] == 1:
                            marathon_correct += 1
            else:
                if row["p2_distance_run"] > 0:
                    if (row["p1_distance_run"] / row["p2_distance_run"]) > 1:
                        if row["point_victor"] == 2:
                            marathon_correct += 1


    print(f"SCORE PREDICTION GOT {round(score_correct / eval_points, 4)} CORRECT
        AND FAILED TO EVALUATE {score_outside_training // 2} POINTS OUT OF
        {eval_points // 2}.")

    print(f"SERVE PREDICTION GOT {round(serve_correct / eval_points, 4)}
        CORRECT")

    print(f"MARATHON PREDICTION GOT {round(marathon_correct / (eval_points //
        2), 4)} CORRECT")
```

Model Calculation Code

```
import pandas as pd

# -------------------- CALCULATE PROBABILITY DATA --------------------

# ---------------------------------------
#          Set and game score
# ---------------------------------------

data = pd.read_csv("./Wimbledon_featured_matches.csv")

print("(P1 SETS - P2 SETS, P1 GAMES - P2 GAMES) | set + game (occurrences)")

score_chances = {}

for p1_sets in [0, 1, 2]:
    for p2_sets in [0, 1, 2]:
        for p1_games in [0, 1, 2, 3, 4, 5, 6]:
            for p2_games in [0, 1, 2, 3, 4, 5, 6]:

                # Don't track games that can't occur (6-0, or 6-4 for example)
                if ((abs(p1_games - p2_games) < 2) or p1_games < 6) and \
                    ((abs(p1_games - p2_games) < 2) or p2_games < 6):

                    set_game_point_wins = 0
                    set_game_occurances = 0


                    for index, row in data.iterrows():
                        if int(row["match_id"][-4:]) < 1500:

                        # Probability of winning a point given a set state and
                            game state

                            if [row["p1_sets"], row["p2_sets"],
                                row["p1_games"], row["p2_games"]] == [p1_sets,
                                p2_sets, p1_games, p2_games]:
                                if row["point_victor"] == 1:
                                    set_game_point_wins += 1

                                set_game_occurances += 1

                    if set_game_occurances > 0:
                        set_game_chance =
                            round(set_game_point_wins/set_game_occurances, 4)
                        score_chances.update({f"{p1_sets}, {p2_sets},
                            {p1_games}, {p2_games}": set_game_chance})
                    else:
                        print(f"DIDNT OCCUR {p1_sets} - {p2_sets}, {p1_games}
                            - {p2_games}")

with open("score_chances.txt", 'w+') as txt:
    txt.write(f"{score_chances}")

# ---------------------------------------
#           serve speed > 125
# ---------------------------------------

print("fast_serve_chance (occurrences) | slow_serve_chance |
    marathon_chance")

fast_serve_point_wins = 0
fast_serve_state_occurances = 0

slow_serve_point_wins = 0
slow_serve_state_occurances = 0

for player in [1, 2]:
    for index, row in data.iterrows():
        if int(row["match_id"][-4:]) < 1500:

        # Probability of winning a point given a serve speed > 125 and P1 is
            serving

            if row["server"] == player and row["speed_mph"] >= 125:
                if row["point_victor"] == player:
                    fast_serve_point_wins += 1

                fast_serve_state_occurances += 1

         # Probability of winning a point given a serve speed < 125 and P1 is
             serving

            if row["server"] == player and row["speed_mph"] < 125:
                if row["point_victor"] == player:
                    slow_serve_point_wins += 1

                slow_serve_state_occurances += 1

# fast chance
```

```
if fast_serve_state_occurances > 0:
    fast_serve_chance =
        round(fast_serve_point_wins/fast_serve_state_occurances, 4)
else:
    fast_serve_chance = "NO OCCURRENCES"

# slow chance
if slow_serve_state_occurances > 0:
    slow_serve_chance =
        round(slow_serve_point_wins/slow_serve_state_occurances, 4)
else:
    slow_serve_chance = "NO OCCURRENCES"

with open("serve_speed_chances.txt", 'w+') as txt:
    txt.write("fast_serve_chance (occurrences) | slow_serve_chance\n")
    txt.write(f"[{fast_serve_chance}, {slow_serve_chance}]")

print(f"{fast_serve_chance} ({fast_serve_state_occurances}) |
    {slow_serve_chance} ({slow_serve_state_occurances})")

# ---------------------------------------
#      Distance traveled ratio
# ---------------------------------------

marathon_chances = {}

for distance in [[0, 1], [1, 5], [5, 10], [10, 100]]:
    for ratio in range(105, 205, 5):

        if ratio % 25 == 0:
            print("PROCESSING RATIO", distance, ratio / 100)

        marathon_points = 0
        marathon_occurances = 0

        for player in [1, 2]:
            for index, row in data.iterrows():
                if int(row["match_id"][-4:]) < 1500:

                # Probability of winning a point given a distance traveled
                    and ratio

                    if player == 1:
                        if (distance[0] < row["p1_distance_run"] <=
                            distance[1]) and (row["p2_distance_run"] /
                            row["p1_distance_run"]) > (ratio / 100):
```

```
                            if row["point_victor"] == 1:
                                marathon_points += 1

                            marathon_occurances += 1
                    else:
                        if (distance[0] < row["p2_distance_run"] <=
                            distance[1]) and (row["p1_distance_run"] /
                            row["p2_distance_run"]) > (ratio / 100):
                            if row["point_victor"] == 2:
                                marathon_points += 1

                            marathon_occurances += 1

        r = ratio / 100
        marathon_chances.update({f"{distance}, {r}": round(marathon_points /
            marathon_occurances, 4)})

with open("marathon_chances.txt", 'w+') as txt:
    txt.write(f"{marathon_chances}")

print(marathon_chances)
```

Graphing Code

```
import ast
import matplotlib.pyplot as plt

# --------------- READ PROBABILITY DATA AND CREATE GRAPHS ---------------

with open("score_chances.txt", 'r') as txt:
    scores = txt.readline().strip().replace('\'', '\"')
    score_chances = ast.literal_eval(scores)

with open("serve_speed_chances.txt", 'r') as txt:
    txt.readline()
    serve_speed_chances = ast.literal_eval(txt.readline().strip())

with open("marathon_chances.txt", 'r') as txt:
    marathon_chances = ast.literal_eval(txt.readline().strip())

print("CREATING SCORE GRAPHS")

fig_num = 1

for sets in ["0, 0", "0, 1", "0, 2",
             "1, 0", "1, 1", "1, 2",
             "2, 0", "2, 1", "2, 2"]:

    probs = [[], [], [], [], [], [], []]

    for score, prob in score_chances.items():
        # print(score, score[6:7])
        if score[:4] == sets:
            for p2_score in range(0, 6 + 1): # If p1 has 0-6 points
                if int(score[9:12]) == p2_score:
                    probs[p2_score].append([int(score[6:7]), prob])

    plt.figure(fig_num)

    fig, ax = plt.subplots()

    opp_0 = plt.plot([x[0] for x in probs[0]], [x[1] for x in probs[0]],
        c="red", label="0 games")
    opp_1 = plt.plot([x[0] for x in probs[1]], [x[1] for x in probs[1]],
        c="orange", label="1 games")
    opp_2 = plt.plot([x[0] for x in probs[2]], [x[1] for x in probs[2]],
        c="turquoise", label="2 games")
    opp_3 = plt.plot([x[0] for x in probs[3]], [x[1] for x in probs[3]],
        c="green", label="3 games")
    opp_4 = plt.plot([x[0] for x in probs[4]], [x[1] for x in probs[4]],
        c="blue", label="4 games")
    opp_5 = plt.plot([x[0] for x in probs[5]], [x[1] for x in probs[5]],
        c="purple", label="5 games")
    opp_6 = plt.plot([x[0] for x in probs[6]], [x[1] for x in probs[6]],
        c="black", label="6 games")

    handles, labels = ax.get_legend_handles_labels()
    ax.legend(handles, labels)

    ax.legend(bbox_to_anchor=(1.02, 0.8), fancybox=True, shadow=True,
        title="Player 2 games")

    plt.xlabel("Player 1 games")
    plt.ylabel("Prob. winning point")

    plt.title(f"Game score affecting win prob at {sets} sets.")

    plt.savefig(f"{sets}_graph", bbox_inches="tight")

    plt.show()

    fig_num += 1

print("CREATING SERVE GRAPH")

plt.figure(fig_num + 1)

fig, ax = plt.subplots()

x_labels = ["Fast serve", "Opponent fast serve", "Slow serve", "Opponent
    slow serve"]
probabilities = [0.7942, 1 - 0.7942, 0.6913, 1 - 0.6913]
bar_colors = ['blue', 'turquoise', 'red', 'pink']

ax.bar(x_labels, probabilities, color=bar_colors, width=0.3)

plt.xlabel("Fast > 125 mph, Slow < 125 mph")
plt.ylabel("Prob. winning point")

plt.title("Serve speed affecting win prob.")

plt.savefig("serve_graph")

plt.show()
```

```
print("CREATING MARATHON GRAPHS")

for distance in ["[0, 1]", "[1, 5]", "[5, 10]", "[10, 100]"]:

    probs_xs = []
    probs_ys = []

    for i in range(5, 105, 5):
        probs_xs.append(i)

    for dist_ratio, prob in marathon_chances.items():
        if dist_ratio.split('],')[0] + ']' == distance:
            probs_ys.append(prob)

    plt.figure(fig_num + 1)

    fig, ax = plt.subplots()

    dist_t = plt.plot(probs_xs, probs_ys, c="red", label="Distance
        travelled")

    ax.axes.set_xticks(probs_xs)

    plt.xlabel("Player 2 travelled x% more than Player 1")
    plt.ylabel("Prob. winning point")

    plt.title(f"Distance affecting win prob between {distance.split(',
        ')[0][1:]} and {distance.split(', ')[1][:-1]} meters travelled.")

    plt.savefig(f"{distance}_graph")

    plt.show()

    fig_num += 1
```

7 References and Appendices

References

[1] A. M. Madurska, “A set-by-set analysis method for predicting the outcome of professional singles tennis matches, https://www.doc.ic.ac.uk/teaching/distinguished-projects/2012/a.madurska

[2] N. S. YUVAL CALEV, “Simulating tennis matches using markov model, https://www.math.wustl.edu/ feres/math350fall2012/projects/mathproj02.pdf,” 2012.

[3] T. J. Barnett, “Mathematical modelling in hierarchical games with specific reference to tennis, https://strategicgames.tripod.com/phd.pdf,” 2006.

[4] M. L. F. Rothe, “Simulation of tennis behaviour using finite markov chains,” in *10th Vienna International Conference on Mathematical Modelling MATHMOD 2022: Vienna Austria, 27–29 July 2022*, **55**, pp. 606–611, 2022.

[5] B. A. Barnett, T. and S. Clarke, “Developing a tennis model that reflects outcomes of tennis matches, https://researchbank.swinburne.edu.au/file/e2452bdc-effc-46f3-af0e-db259274cf98/1/pdf

[6] M. V. Ana Šarčević, Damir Pintar and A. Gojsalić, “Modeling in-match sports dynamics using the evolving probability method,” in *Applied Sciences. 2021*, **1**, p. 4429, 2021.