Predicting the Wimbledon 2025 Champion: A Machine Learning Approach Without Scikit-Learn

Vatsa Rachuri

July 2025

1 Introduction

This is a simple implementation of using logistic regression to predict the Wimbledon champion.In this project, we explore the feasibility of predicting the winner of Wimbledon 2025 based on key features such as the number of past wins at Wimbledon, current form, and age. Rather than relying on high-level machine learning libraries like Scikit-Learn, we implement the core logic of logistic regression from scratch. Any machine learning algorithm requires certain amount of data for the simplicity I have created my own dataset consisting 10 players which includes their age, their total grand slam wins,their Wimbledon wins and their current ATP ranking. This is a complete depiction of usage of logistic regression from scratch. Even though the given dataset is very less and the number of parameters are very less yet it would give complete idea on logistic regression without using Scikit-learn.

2 Dataset

For any machine learning algorithm the basic requirement is the dataset. Generally a machine learning algorithm takes up the data from the dataset reads the data and predicts the outcome for a new data. For this project I have created my own dataset in MS Excel which contains the name of 10 players, their age, their ATP ranking, their wins at Wimbledon, and the total number of grand slams. Figure 1 depicts the dataset used for the model

Player	grandslams	age	ranking	wim wins	
Djokovic	24	38	6	7	
Alcaraz	5	22	2	2	
Sinner	4	23	1	1	
Zverev	0	28	3	0	
Fritz	0	27	4	0	
Ben	0	22	10	0	
jenson	0	24	99	0	
Matteo	0	29	36	0	
kyrgios	0	30	635	0	
jack	0	23	5	0	

Figure 1: Dataset

3 Basic Concepts

Machine learning is a statistical technique to give computer systems the ability to "learn" with data. Machine Learning can be classified into four main parts :

- Supervised Learning
- Semi-Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

3.1 Supervised Learning

When the input and output of the data is given then to predict the output of a new set of values we use supervised learning. Supervised Learning is of two parts:

- Regression
- Classification

3.1.1 Regression

Regression is a technique used to predict continuous numerical values.For example, we can predict the price using parameters such as area, bedrooms, bathrooms,story, main road, guestroom, basement, hot water heating, air conditioning, parking, etc. The common algorithms used are as follows:

- Linear Regression
- Polynomial Regression
- Support Vector Regression

3.1.2 Classification

Classification is a technique used to predict class labels. For example it predicts spam or not spam, yes or no, true or false,etc. Even in this model we use this technique to predict the champion of Wimbledon. The common algorithms used are as follows:

- Logistic Regression
- Decision Trees
- Random Forest

3.2 Unsupervised Learning

In unsupervised learning only the input of the data is given, from the data the given models will divide the data into categories. Some types of unsupervised learning are as follows:

- Clustering
- Dimensionality Reduction
- Anomaly detection
- Association Rule learning

3.2.1 Clustering

Clustering is an unsupervised learning technique that groups data points into clusters based on similarity, without any predefined labels. It is widely used in data mining, pattern recognition, and customer segmentation. Real-life examples are medical diagnosis,music recommendations,etc. K-means is the commonly used algorithm for clustering.

3.2.2 Dimensionality Reduction

In a dataset if the inputs are more this process will reduce the number of inputs therefore decreasing its dimension. It is used to simplify models, reduce computation and help visualize data especially when dealing with higher dimensional datasets.

3.2.3 Anomaly Detection

It is a process of identifying data points, patterns, or observations that deviate significantly from the normal behavior of a dataset. Figure 2 shows an image of an anomaly in the data.

3.2.4 Association Rule Learning

Association Rule Learning is a type of unsupervised learning that discovers interesting relationships (associations) between variables in large datasets. It is most famously used in market basket analysis to find products frequently bought together.

$$\text{Bread, Butter} \rightarrow \text{Jam}$$

For example in a market if someone buys bread and butter they are most likely to buy Jam. Therefore, this method is used for product recommendations.

3.3 Semi-Supervised Learning

Semi-Supervised Learning is a type of machine learning that uses a small amount of labeled data combined with a large amount of unlabeled data during training.
It bridges the gap between supervised learning and unsupervised learning .

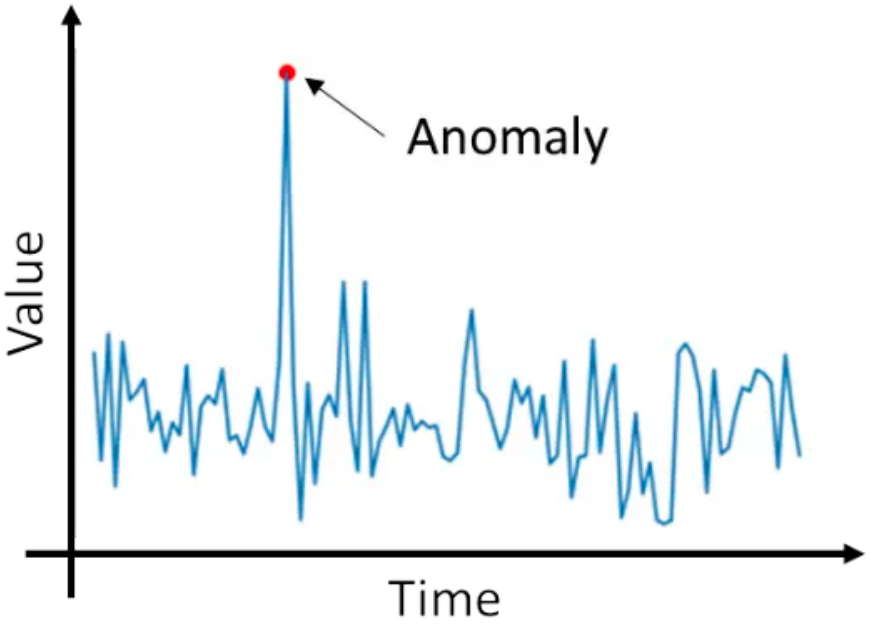

Figure 2: Anomaly Detection

3.4 Reinforced Learning

Reinforcement Learning is a powerful area of machine learning that focuses on how intelligent agents learn to make decisions in an environment to maximize a cumulative reward. Unlike supervised learning or unsupervised learning, RL learns through trial and error interactions.

4 Steps that are required to proceed

1. Collect Data
2. Pre-process Data
3. EDA(Exploratory Data Analysis)
4. Choosing a model
5. Training a model
6. Evaluate model
7. Optimize model

5 Logistic Regression

Logistic Regression is a supervised machine learning algorithm used for classification problems. Unlike linear regression which predicts continuous values it predicts the probability that an input belongs to a specific class. It is used for binary classification where the output can be one of two possible categories such as Yes/No, True/False or 0/1. It uses sigmoid function to convert inputs into a probability value between 0 and 1.

5.1 Types of Logistic Regression

1. **Binomial Logistic Regression:** This type is used when the dependent variable has only two possible categories. Examples include Yes/No, Pass/Fail or 0/1. It is the most common form of logistic regression and is used for binary classification problems.

2. **Multinomial Logistic Regression:** This is used when the dependent variable has three or more possible categories that are not ordered. For example, classifying animals into categories like "cat," "dog" or "sheep." It extends the binary logistic regression to handle multiple classes.

3. **Ordinal Logistic Regression:** This type applies when the dependent variable has three or more categories with a natural order or ranking. Examples include ratings like "low," "medium" and "high." It takes the order of the categories into account when modeling.

5.2 Sigmoid Function

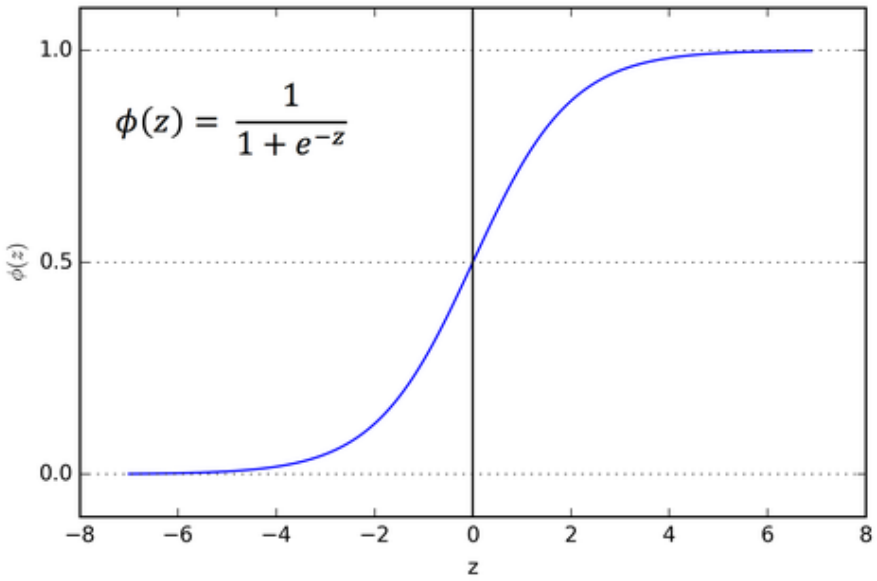

Figure 3: Sigmoid function

The sigmoid function is a important part of logistic regression which is used to convert the raw output of the model into a probability value between 0 and 1. This function takes any real number and maps it into the range 0 to 1 forming an "S" shaped curve called the sigmoid curve or logistic curve. Because probabilities must lie between 0 and 1, the sigmoid function is perfect for this purpose.

In logistic regression, we use a threshold value usually 0.5 to decide the class label.
If the sigmoid output is same or above the threshold, the input is classified as Class 1.
If it is below the threshold, the input is classified as Class 0.

The input features are represented in the given matrix:

$$X = \begin{bmatrix} x_{11} & \cdots & x_{1m} \\ x_{21} & \cdots & x_{2m} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nm} \end{bmatrix}$$

Since it is a binary value classifier we take an independent variable Y which takes 0 or 1 as its value.

$$Y = \begin{cases} 0 & \text{if } \textit{Class 1} \\ 1 & \text{if } \textit{Class 2} \end{cases}$$

$$z = \left(\sum_{i=1}^{n} w_i x_i \right) + b$$

Here x_i is the i^{th} observation of X, $w_i = [w_1, w_2, w_3, \cdots, w_m]$ is the weights or Coefficient and b is the bias term also known as intercept. Simply this can be represented as the dot product of weight and bias.

$$z = w \cdot X + b$$

z is a continuous value from the linear regression. Logistic regression then applies the sigmoid function to z to convert it into a probability between 0 and 1 which can be used to predict the class. Now we use sigmoid function where the input will be z and converts the data into probability

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

5.3 Derivation of Logistic Regression

1. Hypothesis Function:

The logistic regression hypothesis uses the sigmoid (logistic) function:

$$h_\theta(x) = \sigma(z) = \frac{1}{1+e^{-z}}, \quad \text{where } z = \theta^T x = w \cdot X + b$$

2. Sigmoid Function:

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

This maps any real value to the range $[0, 1]$, useful for binary classification.

3. Binary Classification Labels:

$$y \in \{0, 1\}$$

4. Cost Function (Log Loss):
The cost function for a single training example is:

$$J(\theta) = -\left[y \log(h_\theta(x)) + (1-y)\log(1-h_\theta(x))\right]$$

For m training examples:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log(h_\theta(x^{(i)})) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right]$$

5. Gradient Descent:
We update the weights using:

$$\theta_j := \theta_j - \alpha\frac{\partial J(\theta)}{\partial \theta_j}$$

The gradient of the cost function:

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j^{(i)}$$

So the update rule becomes:

$$\theta_j := \theta_j - \alpha\left[\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j^{(i)}\right]$$

6. Final Prediction Rule:

$$\hat{y} = \begin{cases} 1 & \text{if } h_\theta(x) \geq 0.5 \\ 0 & \text{otherwise} \end{cases}$$

6 Code in python for the dataset shown in Figure 1 using Logistic Regression

6.1 Importing the libraries

```
import numpy as np
import pandas as pd
```

Since we are not using Scikit-Learn we are going to use numpy and pandas. The alias np and pd is a convention — it makes code cleaner and shorter.

6.2 Reading the dataset

```
df=pd.read_csv("dataset.csv")
```

Here df stands for dataframe and in pandas library we are using the function

```
read_csv(" ")
```

to read the csv file that is "dataset.csv"

6.3 Extracting features

```
X=df[["age","wim wins","grandslams","ranking"]].values.astype(float)
```

Here X selects the features from the dataframe which is converted into numpy array by the function ".values" the data is then converted into as float datatype by the function ".astype(float)"

6.4 Normalization

```
X[:, 0] = 1 - (X[:, 0] - X[:, 0].min()) / (X[:, 0].max() - X[:, 0].min())
X[:,1]=(X[:, 1] - X[:, 1].min()) / (X[:,1 ].max() - X[:, 1].min())
X[:,2]=(X[:, 2] - X[:, 2].min()) / (X[:,2 ].max() - X[:, 2].min())
X[:, 3] = 1 - (X[:, 3] - X[:, 3].min()) / (X[:, 3].max() - X[:, 3].min())
```

Here we are normalizing the data because we are setting our data in such a way that lesser is the age the more fit they are, the more wimbledon wins they have the more is their experience to that field, and we are normalizing the ranking data also because the lesser is the rank the better is their chance to win.

6.5 Addition of a vector containing 1 to the feature matrix

```
X = np.hstack((np.ones((X.shape[0], 1)), X))
```

Here we add a column of 1 to the feature matrix so that we make sure when using dot product there would be a **bias term**

6.6 Assigning the target variable matrix to 0

```
y=np.zeros((len(X),1))
```

Here the function "np.zeros" makes sure that the array contains all elements as 0
Here we are assigning the winner should be represented as 1 in the target array so we are assigning an index to 1

```
y[1]=1
```

6.7 Defining sigmoid function

```
def sigmoid(a):
return 1 / (1 + np.exp(-a))
```

This returns the sigmoid function for a variable.

6.8 Defining Logistic Regression function

```
def logreg(X,y,lr=0.01,epochs=100000):
a,b=X.shape
theta=np.zeros((b,1))
for epochs in range(epochs):
    z=X @ theta
    y_hat=sigmoid(z)
    grad=(X.T @ (y_hat-y))/a
    theta -= lr*grad
return theta
```

Explanation of the code line by line

```
def logreg(X,y,lr=0.01,epochs=100000):
```

Here X is the feature matrix, y is the target matrix, lr is the learning rate for logistic regression it should be around 0.01 it also determines how big each step need to be in gradient decent , epochs is the total number of iteration

```
a,b=X.shape
```

This helps in initializing the weight vector and computing gradients

```
theta=np.zeros((b,1))
```

Here we initialize the weights to zero.

```
for epochs in range(epochs):
    z=X @ theta
    y_hat=sigmoid(z)
    grad=(X.T @ (y_hat-y))/a
    theta -= lr*grad
```

This loop is a training loop where we iterate this till the range of epochs In the first line of loop we are doing dot product of feature matrix with the weight vector.
In the second line of loop we apply sigmoid function to map z into probabilities
In the third line we compute gradient

$$X^T \cdot (\hat{y} - y)/a$$

which we divide with "a" which gives average gradient decent
In the fourth line we update the gradient decent
$\theta = \theta - \alpha \cdot \nabla J(\theta)$
Here α is the learning rate or lr which is 0.01

6.9 Calling the functions

```
theta=logreg(X,y)
prob=sigmoid(X @ theta)
```

Here we are calling the above functions and assigning them to a variable

6.10 Creating a new column

```
df["Probability"]=prob
```

Here we create a new column named Probability which contains the probability of the players to win the Wimbledon 2025

6.11 Sorting and printing the dataframe

```
df_sorted=df.sort_values(by="Probability",ascending=False)
print(df_sorted[["Player","age","ranking","grandslams","wim wins","Probabi
```

Here we are sorting the data in such a way that the values of probability are in a decending order. The more is the probability more likely to win the grand slam

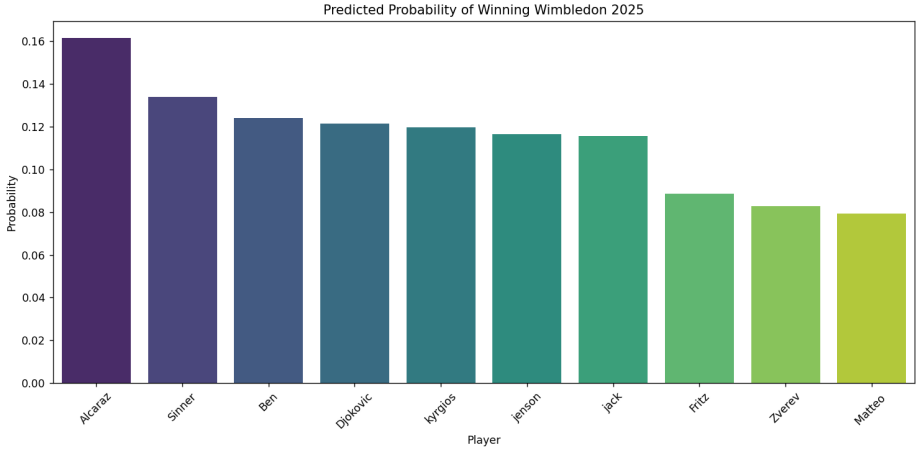

Figure 4: Bar Graph

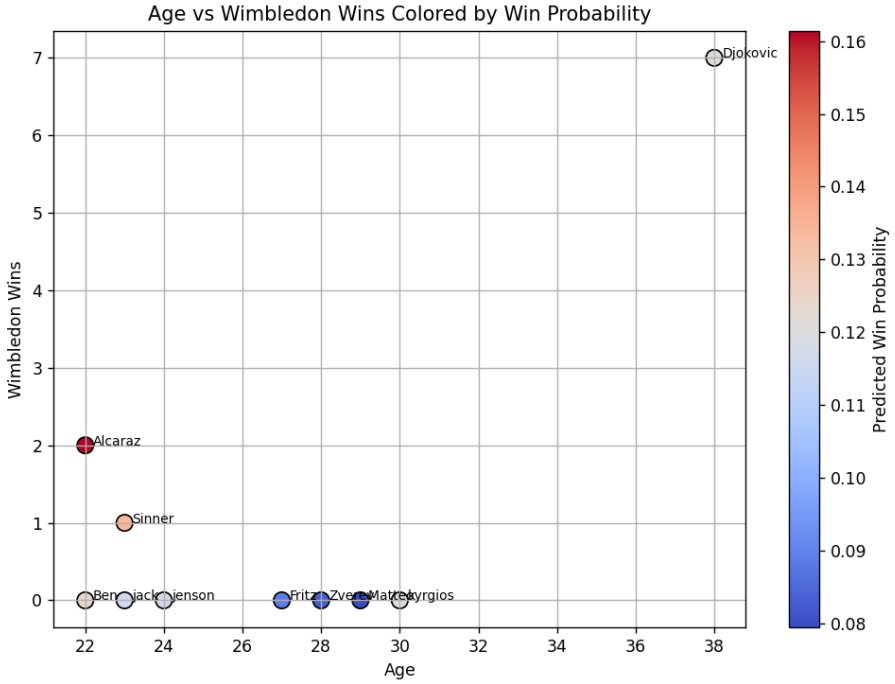

Figure 5: Age vs Wimbledon wins

```
PS C:\Vatsa\Coding\ML\tennis> python tennis.py
     Player  age  ranking  grandslams  wim wins  Probability
1   Alcaraz   22        2           5         2     0.536096
2    Sinner   23        1           4         1     0.189475
5       Ben   22       10           0         0     0.110934
9      jack   23        5           0         0     0.065520
0  Djokovic   38        6          24         7     0.059420
6    jenson   24       99           0         0     0.052385
8   kyrgios   30      635           0         0     0.011923
4     Fritz   27        4           0         0     0.007404
3    Zverev   28        3           0         0     0.004231
7   Matteo    29       36           0         0     0.002712
PS C:\Vatsa\Coding\ML\tennis> python tennis.py
```

Figure 6: Result