Chess players result Prediction | ML Course Project

Shilo ben natan-314717125

Introduction

The goal in this project is to build a model that, given chess player his opponent elo, his color(white or black, his elo, his age. will predict whether the player is in will win or not. Using this

Player around the world can look on this and estimate the player chance to win the game. This model, will train only on 12 players, but given the right data will be able to estimate his right chance to win

Dataset

The dataset for this project obtained from Kaggle. It was provided by google.com [(link).](https://www.kaggle.com/datasets/liury123/chess-game-from-12-top-players) This data set contains a huge number game by the top players

information including pre-conditions. The analysis of this project will not necessarily apply

for other players; However, it will give a general picture that serves the purpose of the

model.

**Features:**

1. player: the name of the player that we focus on

2. color: know if it black or white

3. opponent: the name of his opponent

4. playerELO: the player rating by elo method

5. opponentELO: the player rating by elo method.

6 . result: the result of the player game win,lose,draw

7. event: the event where the game append

8. site: the location of the event

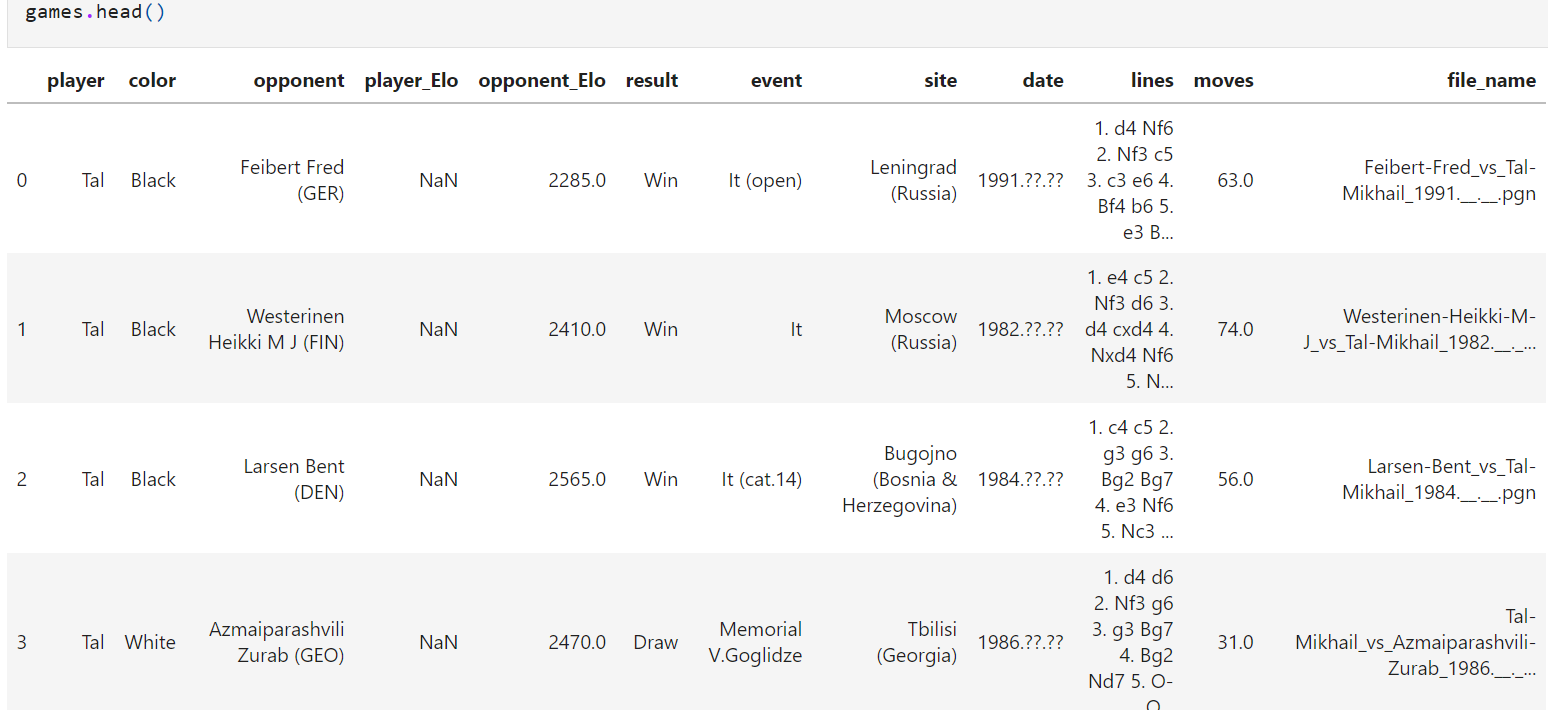
9.date .

10.lines: record of each line of the game

11.moves: the numbers of total moves of the game.

12. file\_name:the name of the file the data was take from.

My data head:



One of the significant challenges we will have to deal with is that the data is un clean – some of the players has null data or wrong date( the player elo in the database is sometimes 0 and this cant be true by the definition of elo)

Some of the player I was have to delete them because they don’t have a lot of date.

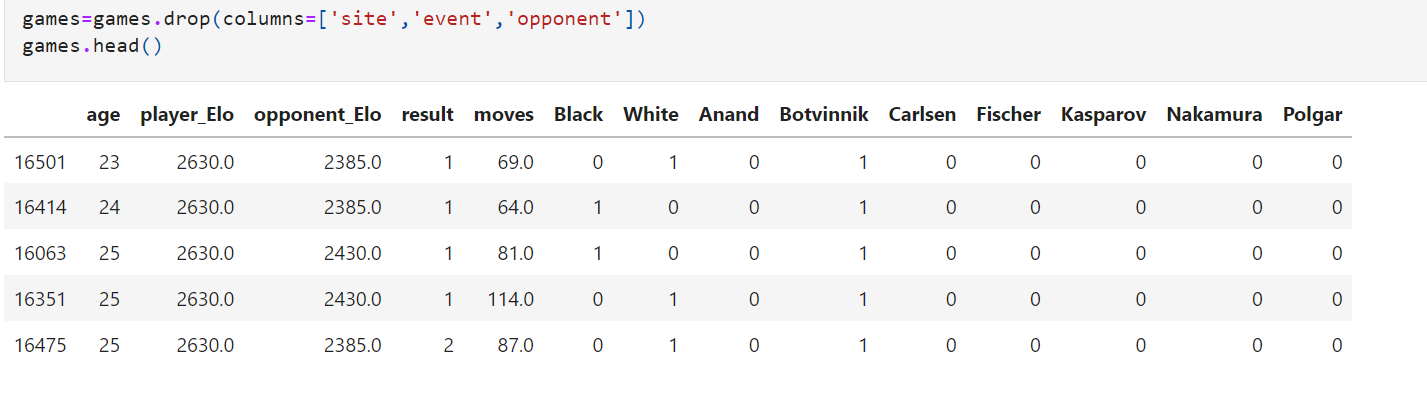
So in the first move of my work I delete the players: Alekhine, Capablanca,Tal,and Morphy. And stay with Botvinnik Carlsen Nakamura Caruana Polgar Kasparov Fischer Anand

My next problem was the null and zero elo of the opponent. So I remove all the game that don’t hava the opinent elo( wrong elo can give us very wrong data and effect all the dataset.

To deal with the player elo I replace all the null to zero and order the data by date and for each player I save the closest update elo and put it instead of the zero value.

In the next I turn the color and the players to dummies and make each game of each player is age on the game(the year of game minus the date of birth of the player from Wikipedia – may effect the game result)

I drop the non values columns and stay with a clean data set



The algorithms that will used to create a classifier for the problem mentioned above are:

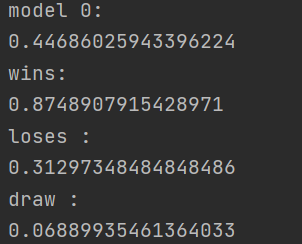
1) svm

2)K-Nearest neighbors

3)decisions tree

4)random forest

First for compere I use model 0 the if the the player elo in higer than 10 from is opponent he win and if less the 10 he will lose and otherwise draw.



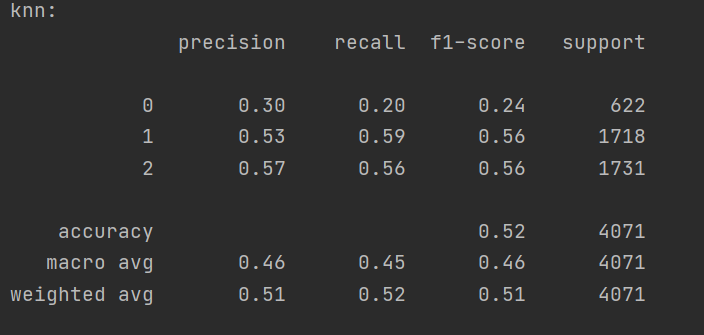
K-Nearest Neighbors:

The K-Nearest Neighbours is non-linear classifier in which an object is classified to the class most common

among its k nearest neighbours in the train set. The hyperparameters for the KNN algorithm are the value of k

and the distance function that calculates the distance between two objects

in my project I use 7 n to the knn.



U can see that in total the knn is more currect in draw and loses. Model 0 is decide the wrong answer.

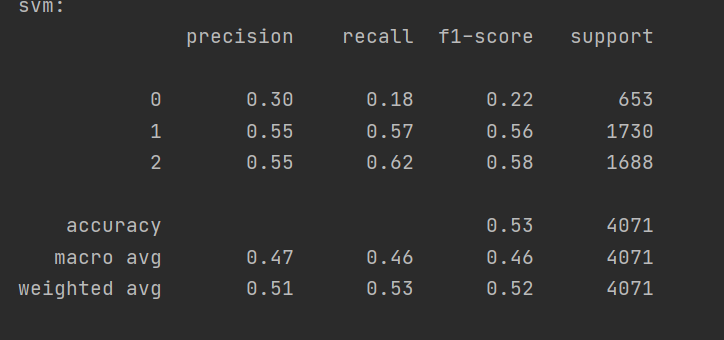
Svm:

SVM is a classifier that separates data points using a linear hyperplane with the largest amount of margin

between the two classes in the train set. SVM can efficiently perform a non-linear classification using what is

called the kernel trick, implicitly mapping the inputs into high-dimensional feature spaces.

I used the “sklearn.svm” library to run the SVM algorithm on the data



Decision tree:

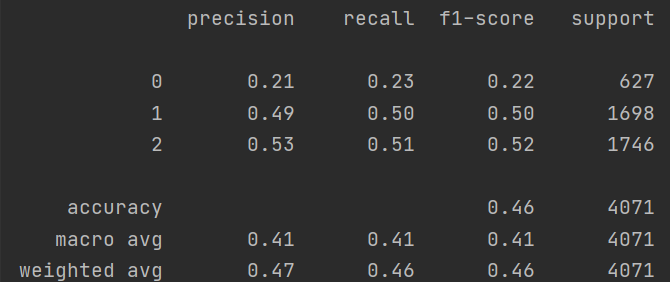
A decision tree is a flowchart-like structure in which each internal node represents a "test" on a feature, each

branch represents the outcome of the test, and each leaf node represents a class label. The paths from root to

leaf represent classification rules. The decision tree algorithm uses heuristic functionsfor selecting the splitting

criterion that partition data into the best possible manner.

I used the “sklearn.tree” library to run the decision tree algorithm on the data.



Random forest:

