I have defined a dictionary **ECO\_map** that maps opening moves in chess (called ECO codes) to a win rate for the player who makes the first move (usually referred to as "white"). The win rates are precomputed and the dictionary is created using train\_datset.

# pseudo code

# df['winner']= df['Result'].apply(lambda x: 'white' if x == 1 else 'black')

# df\_ = pd.crosstab(df.ECO, columns = df.winner, normalize = 'index')

# df['ECO']= df['ECO'].map(dict(df\_['white']))

The purpose of this dictionary is to be later used to encode the ECO codes as numerical values that can be fed into a machine learning algorithm for predicting the outcome of chess games.

To apply this encoding, one could use the following code:

*df*['ECO']=*df*['ECO'].map(ECO\_map)

The **transform\_data** function takes a pandas dataframe **df** as input and performs a series of transformations on it.

1. The missing values in the dataframe are filled in with the mean value of the respective column using the **fillna** method with **inplace=True**.
2. Any remaining missing values are filled with the mode (most common value) of the respective column using the **fillna** method with **inplace=True**.
3. The values in the 'ECO' column are mapped using the **ECO\_map** dictionary.
4. The 'MatchID' column is dropped from the dataframe using the **drop** method with **axis=1**.
5. A new column 'prob\_white\_win' is added to the dataframe, which represents the probability of white winning the game based on the Elo ratings of the players. This is calculated using the formula **1/(1+10^((BlackElo-WhiteElo)/400))**.
6. The dataframe is filtered to keep only the columns 'WhiteElo', 'BlackElo', 'ECO', 'eval', and 'prob\_white\_win' using the **filter** method.
7. The data in the filtered dataframe is scaled using the **MinMaxScaler** from the **sklearn.preprocessing** module, which scales the data to be between 0 and 1.
8. The function returns the scaled dataframe.

The purpose of these transformations is to prepare the data for use in a machine learning model, by filling in missing values, mapping categorical data, dropping irrelevant columns, adding a new relevant feature, and scaling the data to ensure that all features have similar ranges of values.

Why prob\_white\_win feature is added?

* After some research on internet I got to know that ,

probability(A win) = **1/(1+10^((Rtaing\_B-Rating\_A)/400))**.

* So, This value is a better choice for our feature. As it clearly depicts the rating importance.

The generate\_model(x) function :

1. Trains the model on Training data using LGM Classifier and return the model
2. Why LGM Classifier ?
3. LGM Classifier gave the highest accuracy of 0.73 while other model (Random Forest Classifier , MLP , Logistic Regression) remained below 0.63 .
4. Also I read a article on internet which mentioned LGM Classifier very good for chess winner predictions .

Model finally trains on X= ['WhiteElo','BlackElo','ECO','eval','prob\_white\_win']

How features affect Result?

