Overview

In this project our group choose to work on Kaggle dataset European Soccer Database. Our goal is to extract data from the package, apply data cleaning and preprocessing technique to formulate trainable feature-target split. Then we implemented decision tree algorithm to make predictions on the encounter result between two chosen teams.

One significant challenge for this project, is to construct a GUI to facilitate data selection, model training and output display, instead of interacting with console window like before. And I am proud to say with a great deal of trial and error we managed to make that happen, and I learned a lot (PyQT5 package, coding technique, debugging, use of git) through the process.

Personal Contribution

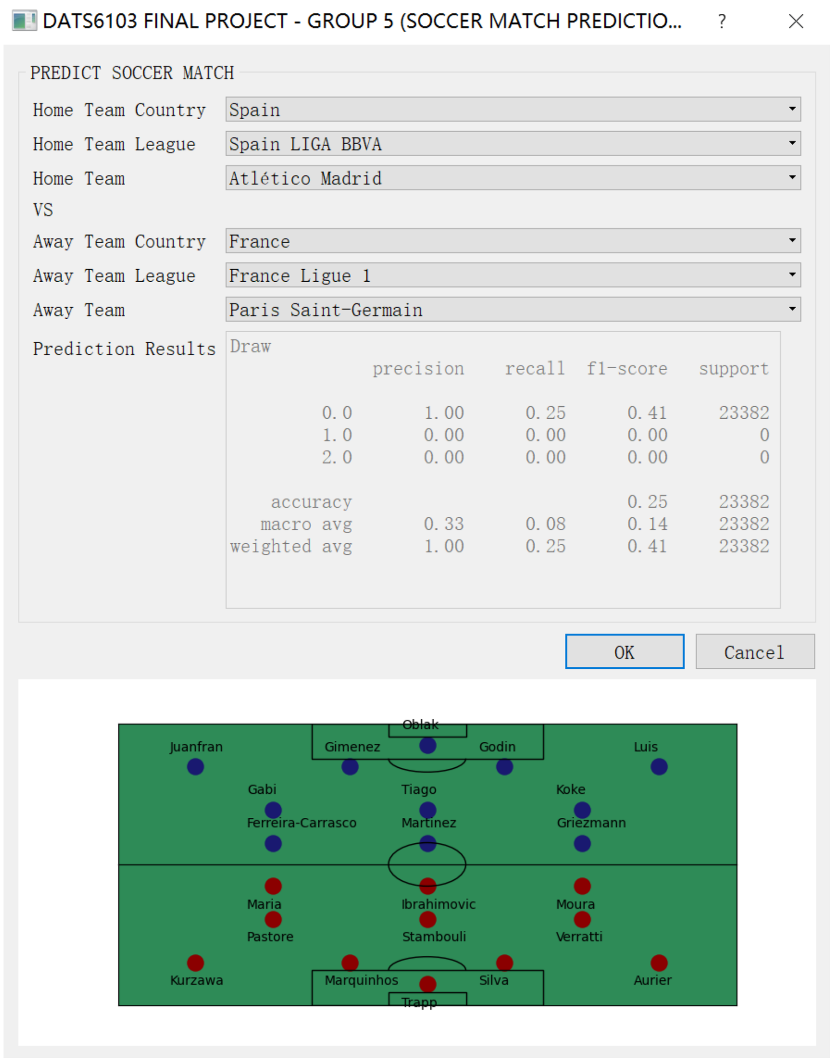
In this project I started working on model training, in the original match_analyzer file. After we compile the match analyzer into the "play_match_window" file, I worked with Salim to revamp the GUI to suit changes in development. I learned how to make room in the form and specify its size so that model output could be properly displayed.

Decision tree, as a linear classifier, attempts to separate the data with hyperlanes and plane. We used Gini index as metrics to initiate the model and installed default parameters in Sklearn module. The training algorithm will take in the match features from the full dataset from season 08 to 16 to train the tree for win/lose/draw result. The match features are the play statistic such as a team's passing and number of shots. The model will then use the feature from selected teams to predict outcome of an encounter.

Result Discussion

Model:

The model will generate prediction together with training accuracy. The model generally yield reasonable results with nothing to our surprise. However, the model is trained on the "average" performance of teams through the years and could not capture the most recent and relevant changes of teams. Obviously the model is giving a rough estimate of relative strength of teams and could not tackle "close calls" between two competitive teams, like the following:



DATS6103 FINAL PROJECT - GROUP 5 (SOCCER MATCH PREDICTIO...

PREDICT SOCCER MATCH

Home Team Country	Spain
Home Team League	Spain LIGA BBVA
Home Team	Atlético Madrid
VS	
Away Team Country	France
Away Team League	France Ligue 1
Away Team	Paris Saint-Germain

Prediction Results

```
Draw
              precision    recall  f1-score   support

         0.0       1.00      0.25      0.41     23382
         1.0       0.00      0.00      0.00         0
         2.0       0.00      0.00      0.00         0

    accuracy                           0.25     23382
   macro avg       0.33      0.08      0.14     23382
weighted avg       1.00      0.25      0.41     23382
```

OK Cancel

This is an example of “close call”, it features with a “draw” as prediction, and low accuracy scores. It shows the trained model has rather poor fit to past encounters, and the tree could not tell better than coin-tossing.

Coding:

In the earlier builds of the project, we wanted to implement multiple shallow models including SVM, Naïve Bayes, and Random Forest to see if more models could provide us with better confidence. I designed a GUI dropdown menu (QcomboBox) for model kernel selection, and link the choice to training function through a dictionary. However, it did not work as intended and give RAM overflow errors frequently. We had to call it off and proceed with decision tree model solo.

Also, earlier queries runs slow and it take several minutes to traverse the entire database. So I designed the analyzer function to train with user specified number of past matches between chosen teams and wrote GUI for that input. This was no longer needed when Ashwin improve the query speed so we could train with the full dataset.

Compared to what we've done right, we often learn more from what's done wrong. For me personally, debugging took me more effort than writing code lines and I took quite a few lesson in mind during the process.

Limitations and Improvements:

Through the entire course of this project, we have kept in mind that this model is as limited as it could be. It is using the entirety of 8 years data, so the most recent development to team formation and tactics could not be captured. It could be improved by assigning weights to more recent matches, but how to design the weights becomes another issue as it is heavily prone to human discretion. Decision tree as a linear classifier has its trouble dealing with class imbalance, as well as its overfitting issues. Soccer is a science that we are only beginning to quantitatively understand, and lots of other features were not included by the dataset, like pitch condition, weather, could become game changer in a non-linear system.

Code Percentage:

The code defining Widgets (QComboBox, QLabel, etc.) were learned from tutorials quoted below. Decision tree training codes were adopted from lecture. So around 20 lines were acquired from external sources and I modified 15 of them to suit our environment. So the proportion of external code will be:

$$\frac{20-15}{20+66} = 5.81\%$$

Reference:

PyQT5 Tutorial: https://build-system.fman.io/pyqt5-tutorial;

https://zetcode.com/gui/pyqt5/widgets2/