ARTIFICIAL INTELLIGENCE

Theme 3 - Supervised Learning

Practical Assignment 2 - Checkpoint

Stylianos Tsagkarakis // up201911231
Vasileios Konstantaras // up201911213

Presentation contents

01 | Dataset

European Soccer Database by Hugo Mathien

A soccer database that contains:

1. +25,000 matches
2. +10,000 players
3. 11 European Countries including their lead championship
4. Seasons 2008 to 2016
5. Players and Teams' attributes
6. Team line up with squad formation (X, Y coordinates)
7. Betting odds from up to 10 providers
8. Detailed match events (goal types, possession, corner, cross, fouls, cards etc...) for +10,000 matches

02 Specification of work

- Exploratory analysis of the dataset
- Examination of the data
  - pre-processing
  - transformation
- Parameterization of supervised learning algorithms

- Algorithms comparison
- Analysis of the confusion matrix
- Demonstration of comparison through appropriate graphs
- Obtain best precision, recall, accuracy, F-measure

03 Research

kaggle

Match Outcome Prediction in Football (link)

kaggle

EUROPEAN FOOTBALL DATA ANALYSIS (link)

ResearchGate

Exploring and modelling team performances of the Kaggle European Soccer database (link)

04 Tools and Algorithms

- Python
- Google Colab
- Scikit Learn
- Numpy
- MatplotLib
- Pandas
- Pipelines

- **Naive Bayes Classifier**
- **k Nearest Neighbors Classifier (kNN)**
- XGBoosts classifier (?)
- Gaussian Model
- **kNN with GridsearchCV**
- **MPL Classifier**

05 Organizing

Python notebook will be organized in 4 Sections:

Section A: Our Team

Section B: Introduction to the dataset

- Dataset description
- Data retrieval
- Remove empty values
- Label frequencies
- Over / under sample
- Split to test / train set

Section C: Baseline classification

- **kNN Classifier**
- Dummy Classifier
- **Naive Bayes Classifier**
- XGBoosts classifier (?)
- **MPL Classifier**
- Gaussian Model

Section D: Optimizing classifiers

- Pre-processing
- Balance dataset
- Standardization
- Variance Threshold
- Scaling
- GridSearchCV

06 | Work already implemented

NULL values in the dataset:

- removed samples (lines) from dataset
- other option: fill values with mean / most frequent values of feature

```
imr = Imputer(missing_values='NaN', strategy='mean', axis=0)
imr = imr.fit(df.values)
imputed_data = imr.transform(df.values)
```

LINUX:

```
cat data.csv | grep "?" |  wc -l
cat data.csv | grep -v "?" > nomissing.data.csv
```

Map important non-numeric values to numeric

```
Mapping = {'of':1, 'def':2, 'gk':3}
df['size'] = df['size'].map(mapping)
```

Keep only samples and features with numeric values

```
df.dropna(axis=1)   int : ['home_team_goal', 'away_team_goal']
                    Categorical: ['month', 'home_team_name', 'away_team_name',
                    'country_name', 'league_name']
```

06 Work already implemented

Reduce dimensions

```
selector = VarianceThreshold (threshold=0.5)
train_reduced = selector.fit_transform (C_trainData)
```

Normalize values

```
min_max_x = (x - np.min(x) )/ (np.max(x) - np.min(x))
```

Possible usage:

`imbalanced-learn` to over/under-sample dataset

`principal components analysis - PCA`

```
req_cols = ['overall_rating', 'crossing', 'finishing','heading_accuracy',
            'short_passing', 'volleys', 'dribbling', 'curve',
            'free_kick_accuracy','long_passing', 'ball_control', 'acceleration',
            'sprint_speed','agility', 'reactions', 'balance', 'shot_power', 'jumping',
            'stamina','strength', 'long_shots', 'aggression', 'interceptions',
            'positioning','vision', 'penalties', 'marking', 'standing_tackle',
            'sliding_tackle','gk_diving', 'gk_handling', 'gk_kicking',
            'gk_positioning','gk_reflexes']

data = player_data[req_cols]



data = player_data.drop(labels = ['id', 'player_fifa_api_id', 'player_api_id',
'date',
                                            'potential', 'preferred_foot',
                                            'attacking_work_rate',
                                            'defensive_work_rate'], axis = 1)
data.fillna(0, inplace=True)
#data.isnull().values.any()
data.corr()a
```

```
from sklearn.cross_validation import train_test_split
feature_cols = ['crossing', 'finishing', 'heading_accuracy', 'short_passing',
        'dribbling', 'curve', 'free_kick_accuracy',
        'long_passing', 'ball_control', 'acceleration', 'sprint_speed',
        'agility', 'reactions', 'balance', 'shot_power', 'jumping', 'stamina',
        'strength', 'long_shots', 'aggression', 'interceptions', 'positioning',
        'vision', 'penalties', 'marking', 'standing_tackle', 'sliding_tackle',
        'gk_diving', 'gk_handling', 'gk_kicking', 'gk_positioning',
'gk_reflexes']

x = data[feature_cols]
y = data.overall_rating

x_train, x_test,
y_train, y_test = train_test_split(x, y, test_size = 0.25, random_state = 42)
```

ARTIFICIAL INTELLIGENCE

Theme 3 - Supervised Learning

Practical Assignment 2 - Final Presentation

Stylianos Tsagkarakis // up201911231
Vasileios Konstantaras // up201911213

07 Dataset Presentation

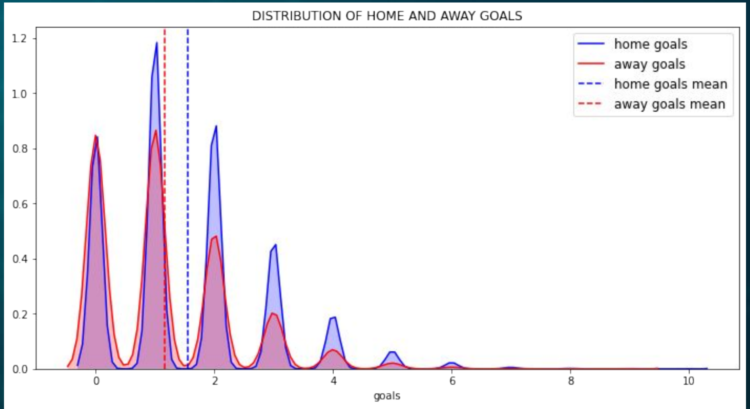
Thanks to this (link) work already implemented by Pavan Raj (link) were able to speed up the data analysis and double check the statistics of the dataset. We handpicked some specific cells that we were interested and we added them to our .ipynb.

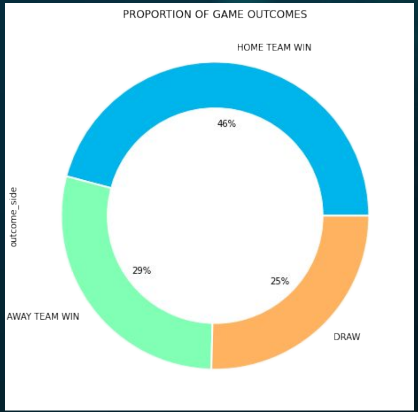
Goal distribution between:

- Home goals
- Away goals

Game outcomes

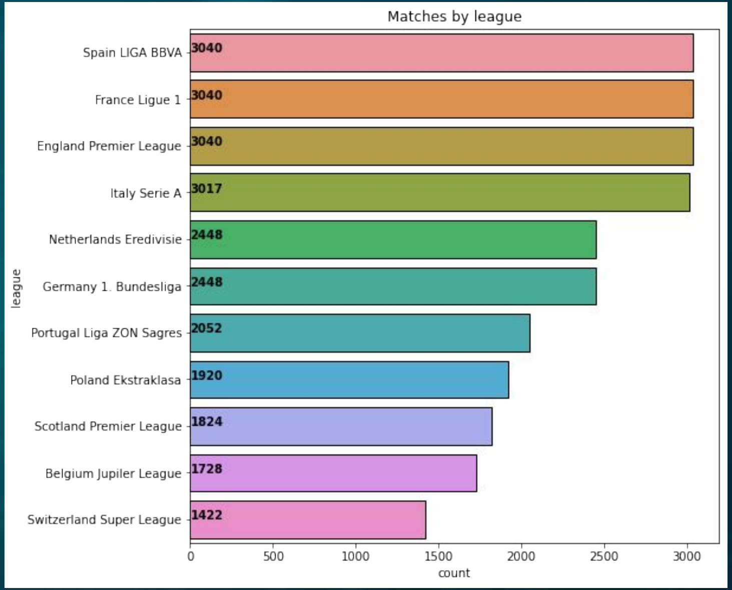
- Home win: **46%**
- Away win: **29%**
- Draw: **25%**



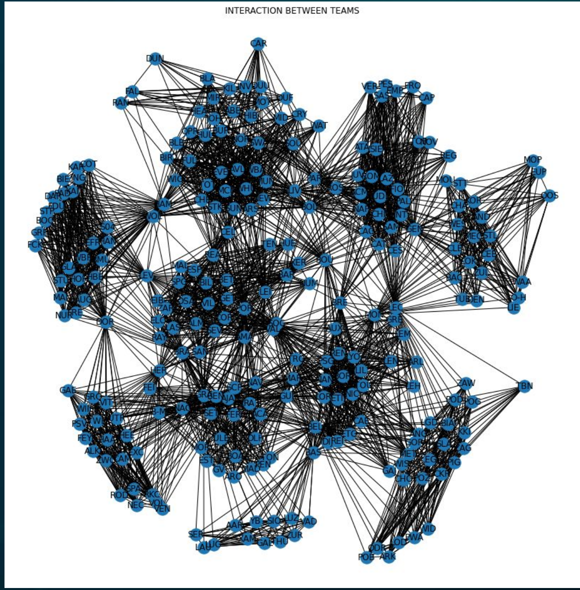


07 Dataset Presentation

Matches by league:



Interaction between teams:



08 Thoughts and methods

Set a base classifier

By predicting HOME_WIN every time you get **46%** success rate. This is works as base classifier and our reference.

Data modification

- Removed some columns not needed for classification e.g. Betting odds for corners.
- Generated a label, based on the goals scored:
  - HOME_WIN
  - DRAW
  - AWAY_WIN

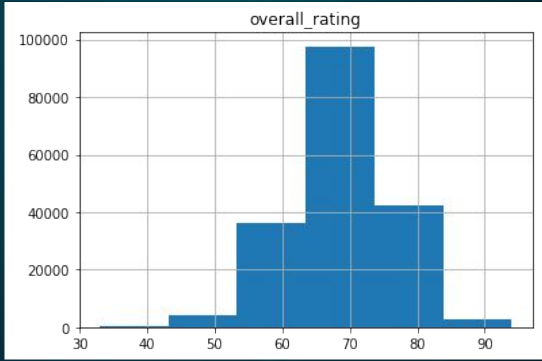
Stats that might help

- Home team's track record playing at home
- Away team's track record playing away
- Away team's track record playing at the location the current match is at

08 Thoughts and methods

Star players // Bottom players

Sometimes a single star player can win a match for their team and that is our intuition behind these features.
Also a very bad player can "help" the enemy team win by mistake.



Bad data

- Define IMPORTANT values to us, drop the rest.
- Missing values: **pandas.dropna()**

```
Number of samples before removing cases of no data: 25979
Number of samples after removing cases of no data: 21374
```

- Not a big loss of information

Player classification

Goalkeeper // Defense // Midfield // Attack

We validate our method against the famous players.
The reason we wish to make this distinction is to better divide a team's rating based on players into defense, attack and midfield; rather than an overall average rating.

08 Thoughts and methods

Team features

- Home team
  - All time home record
  - Record this season thus far
- Away team
  - All time home record
  - Record this season thus far
  - Record at this ground
- Teams head to head
- Team form guide: Last 5 matches:
  - Define it as a string
  - Encode to categorical value
  - Categorical labels

Top 1% of the players

Based on the histogram above, we will define a top player to be a player with overall_rating > 80.

Bottom 1% of the players

Analyze player distribution with histogram. Based on the histogram above, we will define a bottom player to be a player with overall_rating < 50.

Order:

The last match is at the front of the string; therefore if in this season, a team lost its last 2 matches and the won the three before those, form guide will be **LLWWW**.

08 Thoughts and methods

Filling NaN Values

What if we are missing match history **for this season** between teams?

- Previously: filled this values with NaN
- Now: Imputer

Instead of Imputer (example)

- For a team's *home record this season* take the mean of the other seasons.
- Replace empty value with most common of the above values.

NO HISTORY EVER:

- Replace with custom values
- Head_2_Head_Wins/Loss/Draw = 0.33

We could drop more data but we preferred to keep it with transformation.
It was sad to drop data. 😔

09 | Classifiers Implemented

Decision:

- QuadraticDiscriminantAnalysis
- MLPClassifier
- AdaBoostClassifier

Training:

- 1% training data
- 10% training data
- 100% training data

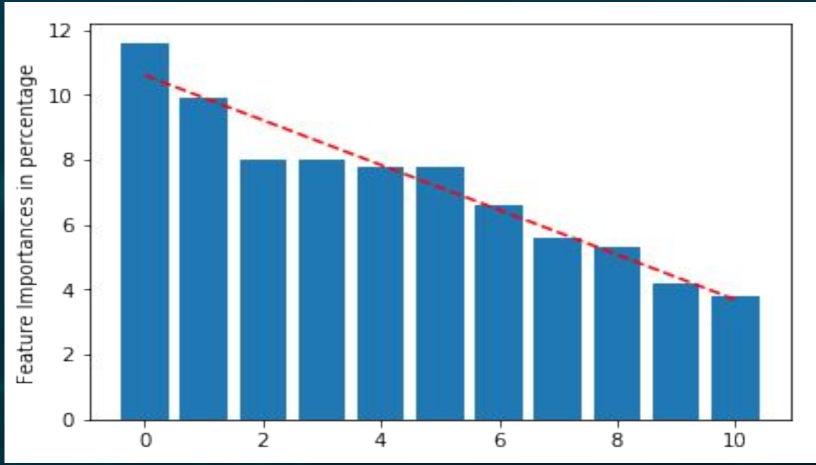
Best Training Score:

AdaBoostClassifier f-score: 54.34%

09 | Classifiers Implemented

Feature Importance

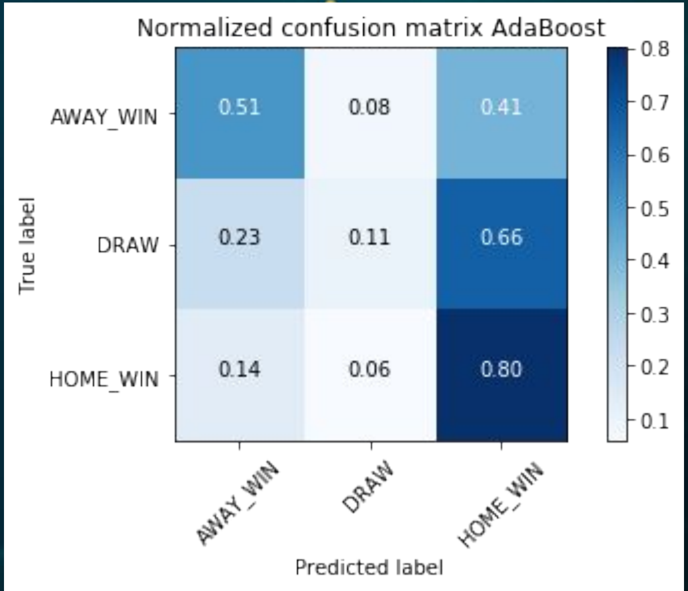
- Interesting that the away win rate at this ground has the highest importance.
- Good to see form guides having some importance!
- Seems that the features number of top players home/away, num of bottom players home/away are not useful at all.



10 | Improvements & Results

Confusion Matrix (AdaBoostClassifier):

- Very good at predicting **home wins**.
  - Home wins generally occur for 46% of the time
  - Predicting 80% is really good.
- Medium performance of **away wins**.
  - 51% is a slightly good result based on other work with the same dataset.
- Very bad performance on **draws**.
  - We did not generate features for draws. Only W/L.
  - One more try without draws.



10 | Improvements & Results

Decision:

- QuadraticDiscriminantAnalysis
- MLPClassifier
- AdaBoostClassifier

Training:

- 1% training data
- 10% training data
- 100% training data

Best Training Score:

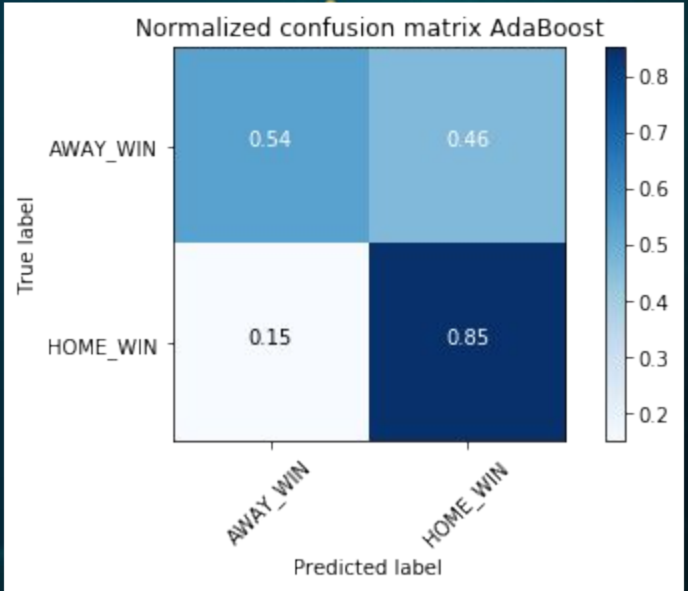
AdaBoostClassifier f-score: ~90%

10 | Improvements & Results

Confusion Matrix No Draws (AdaBoostClassifier):

- Even better at predicting **home wins**.
  - Home wins generally occur for 46% of the time
  - Predicting 85% is really good.
- Again Medium performance of **away wins**.
  - 54% is a slightly good result based on other work with the same dataset.

In general: Better results, but not what expected.



Unoptimized model

Accuracy score on testing data: 0.5421

F-score on testing data: 0.4635

Optimized Model / No draws

Final accuracy score on the testing data: 0.5434

Final F-score on the testing data: 0.7321473314958657

10 | Improvements & Results

Accuracy based on country:

Germany: 0.5105
England: 0.5194
Belgium: 0.4691
Switzerland: 0.4025
Poland: 0.5269
Scotland: 0.4790
Italy: 0.4909
Spain: 0.5185
France: 0.4852
Netherlands: 0.5061
Portugal: 0.5117

Thank you !