Preliminary Work and Plan: Prediction of European Football Player Performance

Brian Pascente, Carson Holscher, Justin O'Donnell, Jacob Richardson

Link to shared github: https://github.com/rhit-pascenby/CSSE415-EurpoeanSoccer

Dataset Overview:

- **Dataset 1 Football Players' Transfer Fee Prediction Dataset (csv)**: The first dataset contains statistics for European football players. This includes the following columns: player, team, name, position, height, age, appearances, goals, assists, yellow cards.
    - **Number of examples**: 10754 players with 22 columns
- **Dataset 2 European Soccer Database (SQLite)**: The second dataset is an SQLite database containing player attributes such as potential, overall rating, preferred foot, attacking work rate, defensive work rate, and crossing. We are focusing on the Player and the Player_Attribute tables. We are going to do this by doing an inner join between the two tables based on the ID.
    - **Number of examples**: 11060 players with 49 columns
    - Note: The SQLite file is not in the github repo since the file is too big to commit/push. The zip file of it is in the resources file, but to use it the user would need to unzip it themselves.

What We Are Predicting:

- **Target Variable**: Player performance (e.g., goals scored or assists). We will use performance metrics from the first dataset (goals, assists) to predict future performance based on attributes from the second dataset.
- **Problem Type**: This is a regression problem, as the goal is to predict continuous values (goals or assists).

Initial Analysis and Proof of Concept (Preliminary Work)

Link: https://github.com/rhit-pascenby/CSSE415-EurpoeanSoccer/blob/main/Project.ipynb

In [ ]:
```
import pandas as pd
from sklearn.preprocessing import StandardScaler
def preprocess(df, split_column, drop_first=True):
    '''
    Does the usual preprocessing steps on a pandas dataframe:
    1. Does one-hot encoding
    2. Standardizes the data
    3. Drop N/A
    4. Splits the data into x & y along split_column
    5. Returns (x, y)
    '''
    stds = StandardScaler().set_output(transform='pandas')
    df = stds.fit_transform(pd.get_dummies(df, drop_first=drop_first)).dropna()
    x = df.drop(split_column, axis=1)
    return (x, df[split_column][x.index])
```

In [ ]:
```
import sqlite3
import pandas as pd
```

In [ ]:
```
conn = sqlite3.connect('resources/database.sqlite')
```

In [ ]:
```
df2 = pd.read_sql_query("SELECT * FROM player_attributes INNER JOIN Player ON Player.id = player_attributes.id", conn)
df2.shape
```

In [ ]:
```
df1 = pd.read_csv('resources/final_data.csv')
df1.rename(columns={'name': 'player_name'}, inplace=True)
```

In [ ]:
```
merged_df = pd.merge(df1, df2, on='player_name')
merged_df = merged_df.dropna()  # Drop rows with missing values
merged_df = merged_df.drop_duplicates(subset='player_name')

merged_df.shape
```

In [ ]:
```
merged_df['goal_per_appearance'] = merged_df['goals'] / merged_df['appearance']
merged_df['assist_per_appearance'] = merged_df['assists'] / merged_df['appearance']
```

In [ ]:
```
merged_df.to_csv("resources/test.csv", index=False)
```

In [ ]:
```

```

In [9]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.dummy import DummyRegressor


(X, y) = preprocess(merged_df, 'goals')

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')

#Baseline
dummy = DummyRegressor(strategy='mean')
dummy.fit(X_train, y_train)
y_dummy_pred = dummy.predict(X_test)
baseline_mse = mean_squared_error(y_test, y_dummy_pred)
print(f'Baseline MSE: {baseline_mse}')
```

```
Mean Squared Error: 0.0008381504680398298
Baseline MSE: 0.03418535776085412
```

In [ ]:
```
print(y.describe())
```

In [ ]:
```
rmse = np.sqrt(0.0008381504680398298)
print("RMSE:", rmse)
```

(does data exploration, Data Cleaning, Feature Engineering, merge the two datasets into one, and creating a baseline and simple regression)

2. Iterative Enhancement Plan

Goal	**Description**	**Due date**
Minimum	1) Create a working model that predicts player goals with reasonable accuracy using basic features (age, height, appearance, etc.) from the merged dataset. 2) Finish up project for Final Presentation	End of week 8, End of week 10

Reasonable	1) Find the “most important position” by using coefficient factors and other models to find the position that makes the most impact for wins per the price of the players in that position. 2) Improve model performance using more complex models (e.g., Random Forest or Gradient Boosting) and implementing feature engineering	End of week 9
Stretch	1) Search the current player market and data and try to find the “best players” on the transfer market right now. In terms of individual statistics/wins per cost of the player.	End of week 10