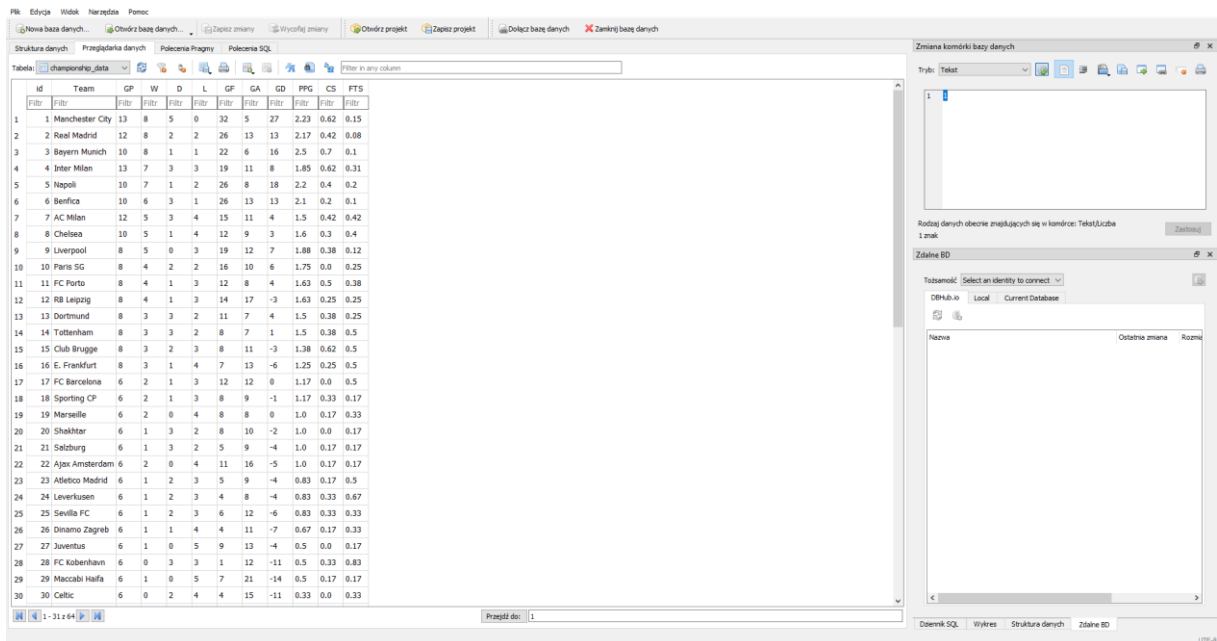


Dokumentacja - baza danych

W naszym projekcie zdecydowaliśmy się na użycie SQLite jako naszej bazy danych. SQLite to lekka biblioteka programistyczna, która dostarcza silnik bazy danych relacyjnej. To, co czyni SQLite wyjątkowym, to fakt, że jest to system zarządzania bazami danych, który nie wymaga osobnego procesu serwera. W zamian, SQLite działa wprost w aplikacji, co sprawia, że jest wyjątkowo przenośne i łatwe w użyciu.

„DB Browser dla SQLite” umożliwia tworzenie i zarządzanie tabelami, indeksami, wyzwalaczami itp. Jest łatwy w obsłudze, co pozwala na szybkie i wydajne manipulowanie danymi. Jego zdolność do wykonywania zapytań SQL w czasie rzeczywistym i wyświetlania wyników, czyni go szczególnie użytecznym dla osób uczących się SQL lub testujących zapytania.

Polecamy podczas sprawdzania projektu do posiłkowania się tym programem, ponieważ ułatwia wprowadzanie danych drużyn. Bazę należy wczytać z pliku `\instance\predit.db`



The screenshot shows the DB Browser for SQLite interface. The main window displays a table named 'championship_data' with the following columns: id, Team, GP, W, D, L, GF, GA, GD, PPG, CS, FTS. The table contains 30 rows of data, representing various football teams and their performance statistics. The right sidebar shows the 'Zmiana komendki bazy danych' (Change database command) window, which is currently empty.

| id | Team | GP | W | D | L | GF | GA | GD | PPG | CS | FTS |
|----|-----------------|----|---|---|---|----|----|-----|------|------|------|
| 1 | Manchester City | 13 | 8 | 5 | 0 | 32 | 5 | 27 | 2.23 | 0.62 | 0.15 |
| 2 | Real Madrid | 12 | 8 | 2 | 2 | 26 | 13 | 13 | 2.17 | 0.42 | 0.08 |
| 3 | Bayern Munich | 10 | 8 | 1 | 1 | 22 | 6 | 16 | 2.5 | 0.7 | 0.1 |
| 4 | Inter Milan | 13 | 7 | 3 | 3 | 19 | 11 | 8 | 1.85 | 0.62 | 0.31 |
| 5 | Napoli | 10 | 7 | 1 | 2 | 26 | 8 | 18 | 2.2 | 0.4 | 0.2 |
| 6 | Benfica | 10 | 6 | 3 | 1 | 26 | 13 | 13 | 2.1 | 0.2 | 0.1 |
| 7 | AC Milan | 12 | 5 | 3 | 4 | 15 | 11 | 4 | 1.5 | 0.42 | 0.42 |
| 8 | Chelsea | 10 | 5 | 1 | 4 | 12 | 9 | 3 | 1.6 | 0.3 | 0.4 |
| 9 | Liverpool | 8 | 5 | 0 | 3 | 19 | 12 | 7 | 1.88 | 0.38 | 0.12 |
| 10 | Paris SG | 8 | 4 | 2 | 2 | 16 | 10 | 6 | 1.75 | 0.0 | 0.25 |
| 11 | FC Porto | 8 | 4 | 1 | 3 | 12 | 8 | 4 | 1.63 | 0.5 | 0.38 |
| 12 | RB Leipzig | 8 | 4 | 1 | 3 | 14 | 17 | -3 | 1.63 | 0.25 | 0.25 |
| 13 | Dortmund | 8 | 3 | 3 | 2 | 11 | 7 | 4 | 1.5 | 0.38 | 0.25 |
| 14 | Tottenham | 8 | 3 | 3 | 2 | 8 | 7 | 1 | 1.5 | 0.38 | 0.5 |
| 15 | Club Brugge | 8 | 3 | 2 | 3 | 8 | 11 | -3 | 1.38 | 0.62 | 0.5 |
| 16 | E. Frankfurt | 8 | 3 | 1 | 4 | 7 | 13 | -6 | 1.25 | 0.25 | 0.5 |
| 17 | FC Barcelona | 6 | 2 | 1 | 3 | 12 | 12 | 0 | 1.17 | 0.0 | 0.5 |
| 18 | Sporting CP | 6 | 2 | 1 | 3 | 8 | 9 | -1 | 1.17 | 0.33 | 0.17 |
| 19 | Marseille | 6 | 2 | 0 | 4 | 8 | 8 | 0 | 1.0 | 0.17 | 0.33 |
| 20 | Shakhtar | 6 | 1 | 3 | 2 | 8 | 10 | -2 | 1.0 | 0.0 | 0.17 |
| 21 | Salzburg | 6 | 1 | 3 | 2 | 5 | 9 | -4 | 1.0 | 0.17 | 0.17 |
| 22 | Ajax Amsterdam | 6 | 2 | 0 | 4 | 11 | 16 | -5 | 1.0 | 0.17 | 0.17 |
| 23 | Adelico Madrid | 6 | 1 | 2 | 3 | 5 | 9 | -4 | 0.83 | 0.17 | 0.5 |
| 24 | Leverkusen | 6 | 1 | 2 | 3 | 4 | 8 | -4 | 0.83 | 0.33 | 0.67 |
| 25 | Sevilla FC | 6 | 1 | 2 | 3 | 6 | 12 | -6 | 0.83 | 0.33 | 0.33 |
| 26 | Dinamo Zagreb | 6 | 1 | 1 | 4 | 4 | 11 | -7 | 0.67 | 0.17 | 0.33 |
| 27 | Juventus | 6 | 1 | 0 | 5 | 9 | 13 | -4 | 0.5 | 0.0 | 0.17 |
| 28 | FC Kobenhavn | 6 | 0 | 3 | 3 | 1 | 12 | -11 | 0.5 | 0.33 | 0.83 |
| 29 | Maccabi Haifa | 6 | 1 | 0 | 5 | 7 | 21 | -14 | 0.5 | 0.17 | 0.17 |
| 30 | Celtic | 6 | 0 | 2 | 4 | 4 | 15 | -11 | 0.33 | 0.0 | 0.33 |

Tabela team_data

W tabeli znajdują się dane które zostały wpisane przez użytkownika:

- Team1 – to pierwsza nazwa drużyny wprowadzonej przez użytkownika
- Team2 – to druga nazwa drużyny wprowadzonej przez użytkownika
- Score1, Score2 – wynik obliczeń z danych tabeli „championship_data”, pokazuje prawdopodobieństwo wyświetlone użytkownikowi na stronie.

Tabela championship_data

Jest to tabela, z której dane są wykorzystywane do przeprowadzenia obliczeń stosowanych, do określenia prawdopodobieństwa wygranej drużyny (wykorzystanie skryptu: „mlearning_sklearn.py”).

Tabela game_data

Rozegrane spotkania pomiędzy dwoma drużynami oraz końcowy wynik.

(Warto posłkować się tą tabelą, żeby wiedzieć jakie dane można wprowadzać na stronie.)

Dane były importowane z plików .csv które znajdują się w głównym katalogu projektu.

Dokumentacja – app.py

To jest dokumentacja Python. Opisuje on tworzenie aplikacji Flask, obsługę żądań HTTP, zarządzanie bazą danych, importowanie danych z plików CSV i renderowanie szablonów HTML.

Python

```
import numpy as np
import pandas as pd
from flask import Flask, render_template, url_for, redirect, request
from flask_sqlalchemy import SQLAlchemy
from mlearning_sklearn import calculate_likelihood
from data_reader import does_exist, check_values
from pathlib import Path
import re
```

Importuje wymagane moduły i biblioteki do wykonania aplikacji. **Numpy**, **Pandas**, **Flask**, **SQLAlchemy**, **mlearning_sklearn**, **data_reader** i **pathlib** są używane do różnych operacji, takich jak manipulacja danymi, obsługa bazy danych, szablonów renderowania i przetwarzanie danych.

Python

```
app = Flask(__name__, static_url_path='/static',
            static_folder='static')
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///predit.db'
db = SQLAlchemy(app)
```

Tworzy instancję aplikacji **Flask** i konfiguruje bazę danych **SQLAlchemy**. Ustala adres URL dla plików statycznych (**CSS**, **JavaScript**) i innych zasobów w folderze "static". Ustala URI bazy danych SQLite.

Python

```
class TeamData(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    team1 = db.Column(db.String(200))
```

```

team2 = db.Column(db.String(200))

score1 = db.Column(db.String(200))

score2 = db.Column(db.String(200))

```

Definiuje model tabeli TeamData w bazie danych. Tabela zawiera kolumny *id*, *team1*, *team2*, *score1* i *score2*.

```

Python
class ChampionshipData(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    Team = db.Column(db.String(200))
    GP = db.Column(db.String(200))
    W = db.Column(db.String(200))
    D = db.Column(db.String(200))
    L = db.Column(db.String(200))
    GF = db.Column(db.String(200))
    GA = db.Column(db.String(200))
    GD = db.Column(db.String(200))
    PPG = db.Column(db.String(200))
    CS = db.Column(db.String(200))
    FTS = db.Column(db.String(200))

```

Definiuje model tabeli ChampionshipData w bazie danych. Tabela zawiera kolumny *id*, *Team*, *GP*, *W*, *D*, *L*, *GF*, *GA*, *GD*, *PPG*, *CS* i *FTS*.

```

Python
class GameData(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    Team1 = db.Column(db.String(200))
    Score = db.Column(db.String(200))
    Team2 = db.Column(db.String(200))

```

Definiuje model tabeli GameData w bazie danych. Tabela zawiera kolumny *id*, *Team1*, *Score* i *Team2*.

Python

```
@app.route("/")
def display_csv():
    return render_template('index.html')
```

Ustala ścieżkę "/" jako stronę główną. Funkcja renderuje szablon "index.html" i zwraca go jako odpowiedź.

Python

```
@app.route('/wynik' methods=['POST'])
def redirect_page():
    team1 = request.form['team1']
    team2 = request.form['team2']
    value1, value2 = calculate_likelihood(team1, team2)

    new_game_pred = TeamData(team1=team1, team2=team2, score1=value1,
                              score2=value2)

    db.session.add(new_game_pred)
    db.session.commit()
    print(TeamData.query.order_by(TeamData.id).all())

    return redirect(url_for('another_page', team1=f'{team1} {value1:
.2f}%' team2=f'{team2} {value2: .2f}%'))
```

Obsługuje żądanie POST na ścieżce *"/wynik"*. Pobiera wartości z formularza, oblicza wynik i zapisuje go w bazie danych. Przekierowuje na inną stronę z przekazanymi danymi.

Python

```
@app.route('/wynik')
def another_page():
    team1 = request.args.get('team1')
    team2 = request.args.get('team2')
    return render_template('/wynik_mecz.html', team1=team1,
team2=team2)
```

Obsługuje żądanie GET na ścieżce **"/wynik"**. Pobiera przekazane parametry z adresu URL i renderuje szablon **"wynik_mecz.html"** z przekazanymi danymi.

Python

```
@app.route("/import-championship-data")
def import_championship_data():
    pattern = r'football_data\d+.csv'
    dir_path = Path.cwd()
    direcories = [direcory for direcory in dir_path.iterdir() if
re.search(pattern, str(direcory))]
    merger = pd.concat([pd.read_csv(dir_name) for dir_name in
direcories])
    df_championship = merger

    db.session.query(ChampionshipData).delete()
    db.session.commit()

    df_championship.to_sql('championship_data', con=db.engine,
if_exists='append', index=False)

    return 'Championship data imported successfully!'
```

Obsługuje żądanie **GET** na ścieżce **"/import-championship-data"**. Odczytuje plik CSV i tworzy **DataFrame** w **Pandas**. Usuwa istniejące dane w tabeli **ChampionshipData** w bazie danych. Importuje **DataFrame** do tabeli **ChampionshipData**.

Python

```
@app.route("/import-game-data")
def import_game_data():
    df_game = pd.read_csv('table_data0.csv')

    db.session.query(GameData).delete()
    db.session.commit()

    df_game.to_sql('game_data', con=db.engine, if_exists='replace',
index=False)

    return 'Game data imported successfully!'
```

Obsługuje żądanie **GET** na ścieżce **"/import-game-data"**. Odczytuje plik CSV i tworzy **DataFrame** w **Pandas**. Usuwa istniejące dane w tabeli **GameData** w bazie danych. Importuje **DataFrame** do tabeli **GameData**.

Python

```
if __name__ == '__main__':
    does_exist()

    app.run(debug=True)
```

Sprawdza, czy plik istnieje za pomocą funkcji **does_exist()** i uruchamia aplikację w trybie debugowania.

Dokumentacja – web_scraping.py

Przedstawiony kod wykonuje web scraping statystyk piłkarskich ze strony internetowej 'soccerstats.com' i zapisuje pobrane dane do plików CSV. Wykorzystuje następujące biblioteki:

- **csv**: Zapewnia funkcje do odczytu i zapisu plików CSV.
- **requests**: Pozwala na wysyłanie żądań HTTP pod wskazany adres URL.
- **BeautifulSoup**: Biblioteka do analizy składni HTML i XML.
- **pathlib**: Oferuje klasy reprezentujące ścieżki systemu plików.
- **pandas**: Potężna biblioteka do manipulacji danymi w celu analizy danych.
- **lxml.etree**: Biblioteka do przetwarzania dokumentów XML i HTML.

Przejdźmy przez kod i opiszmy każdą funkcję oraz jej cel:

```
Python
import csv
import requests
from bs4 import BeautifulSoup
from pathlib import Path
import pandas as pd
import lxml.etree as etree
```

Powyższe linie importują wymagane biblioteki.

```
Python
def clean_team_data_saver(what_to_write, file_path):
    # file_path = Path('football_data.csv')
    with open(file_path, 'w', newline='') as file:
        what_to_write.to_csv(file)
```

Funkcja **clean_team_data_saver** zapisuje przekazane dane (**what_to_write**) do pliku CSV określonego przez **file_path**. Wykorzystuje metodę **to_csv** z biblioteki **pandas**.

```
Python
def make_first_team_data(what_to_write):
    file_path = Path('data.csv')
    with open(file_path, 'w', newline='') as file:
        what_to_write.to_csv(file, index=False, header=False)
```

Funkcja **make_first_team_data** jest podobna do **clean_team_data_saver**. Zapisuje przekazane dane do pliku CSV (**data.csv**) bez dołączania indeksu lub nagłówka.

Python

```
def read_dataframe(path):  
    return pd.read_csv(path, index_col=0)
```

Funkcja `read_dataframe` odczytuje plik CSV o wskazanej ścieżce (*path*) przy użyciu biblioteki pandas i zwraca wynikowy obiekt DataFrame.

Python

```
def clear_the_data(path):  
    data = read_dataframe(path)  
    data = data.drop(['Form', 'Pts'], axis=1)  
    data.index = data.index.astype(int)  
    data = data.rename(columns={'Unnamed: 1': 'Team'})  
    data['CS'] = data['CS'].str.rstrip('%').astype(float) / 100  
    data['FTS'] = data['FTS'].str.rstrip('%').astype(float) / 100  
    return data
```

Funkcja `clear_the_data` przyjmuje ścieżkę do pliku CSV, czyści dane w obiekcie DataFrame poprzez usunięcie niepotrzebnych kolumn, przekształcenie indeksu na wartości całkowite, zmianę nazwy kolumny i dokonanie kilku transformacji danych. Zwraca oczyszczony obiekt DataFrame.

Python

```
def retrieve_data(url):  
    response = requests.get(url)  
    response.raise_for_status()  
    soup = BeautifulSoup(response.content, 'html.parser')  
    return soup
```

Funkcja `retrieve_data` wysyła żądanie HTTP GET pod wskazany adres URL, pobiera zawartość HTML i zwraca ją jako obiekt BeautifulSoup.

Python

```
def get_the_table_of_championship(soup):  
    statistic_table = soup.find('h2', string="Statistical table")  
    table_body = statistic_table.find_next_sibling('table')  
    return table_body
```

Funkcja `get_the_table_of_championship` przyjmuje obiekt BeautifulSoup (**soup**), znajduje element HTML ze stringiem "Statistical table" w tagu `h2` i zwraca następny element rodzeństwa - tabelę `table`.

Python

```
def convert_html_to_pandas(url):  
    table_body = get_the_table_of_championship(retrieve_data(url))  
    pandas_table = pd.read_html(str(table_body))  
    make_first_team_data(pandas_table[0])
```

Funkcja `convert_html_to_pandas` konwertuje tabelę HTML ze wskazanego adresu URL na obiekt DataFrame przy użyciu metody `read_html`. Następnie wywołuje `make_first_team_data`, aby zapisać wynikowy obiekt DataFrame.

Python

```
def data_of_teams(links):  
    all_links = [link['href'] for link in links if 'href' in link.attrs]  
    links_to_team = []  
    for link in all_links[:2]:  
        links_to_team.append('https://www.soccerstats.com/' + link)  
    for x in links_to_team:  
        team_data = retrieve_data(x)  
        all_tables = team_data.find_all('table')  
        all_tables = [all_tables[x] for x in [3]]  
        all_tables = pd.read_html(str(all_tables))  
        all_tables[0] = all_tables[0].drop(0, axis=1)
```

```

all_tables[0] = all_tables[0].drop(4, axis=1)
for index_of_x, x in enumerate(all_tables[0][2]):
    all_tables[0][2][index_of_x] = x[0:5]
for index, table in enumerate(all_tables):
    if index == 0:
        with open(f'table_data{index}.csv', 'a', newline='')
as team_data_file:
            try:
                table.to_csv(team_data_file, index=False,
header=False)
            except UnicodeEncodeError:
                pass

print("complete")

```

Funkcja *data_of_teams* przetwarza linki do stron poszczególnych drużyn. Pobiera zawartość HTML dla każdej drużyny, wybiera żadaną tabelę ze strony, oczyszcza ją i zapisuje jako plik CSV. Nazwy plików CSV są ustalane na podstawie indeksu tabel.

Python

```

if __name__ == '__main__':
    # Wykonuje się, gdy plik jest uruchomiony jako skrypt
    # ...

```

Kod w bloku *if __name__ == '__main__':* stanowi punkt wejścia programu. Inicjalizuje niezbędne zmienne, wywołuje funkcje do wykonania web scrapingu, oczyszczania danych i zapisu, a następnie wypisuje "complete" po zakończeniu.

Ogólnie rzecz biorąc, kod wykonuje web scraping statystyk piłkarskich z kilku adresów URL, konwertuje pobrane dane na obiekty DataFrame w bibliotece pandas, czyści i przetwarza dane, a następnie zapisuje je do osobnych plików CSV.

Dokumentacja – data_reader.py

Ta dokumentacja zawiera przegląd i wyjaśnienie funkcjonalności podanego kodu, który opisuje funkcje do sprawdzania istnienia plików, odczytu i zapisu danych, a także odczytu danych mistrzostw z plików CSV.

Python

```
import pandas as pd
import subprocess
from pathlib import Path
```

Importuje wymagane moduły i biblioteki do wykonania kodu. Pandas, subprocess i pathlib są używane do różnych operacji, takich jak manipulacja danymi, uruchamianie procesów i obsługa ścieżek plików.

python

```
def does_exist():
    files_for_data = ["/table_data0.csv", "/football_data0.csv", "/football_data1.csv"]
    for file in files_for_data:
        file = Path(file)
        if not Path.exists(file):
            subprocess.run(['python', 'web_scraping.py'])
        else
```

Definiuje funkcję **does_exist()**. Sprawdza istnienie plików danych. Jeśli któryś z plików nie istnieje, uruchamia skrypt **web_scraping.py** za pomocą **subprocess.run()**. W przeciwnym razie wyświetla komunikat potwierdzający, że plik istnieje.

Python

```
def game_played(read_write='r'):  
    path_to_data = "table_data0.csv"  
    if read_write != 'r':  
        with open(path_to_data, 'w') as data:  
            pass  
        print(f"Usunięto dane z {path_to_data} generuje nowe")  
        subprocess.run(['python', 'web_scraping.py'])  
    else  
        with open(path_to_data, read_write) as data:  
            columns = ['team1', 'score', 'team2']  
            data_of_games = pd.read_csv(data, names=columns)  
        return data_of_games
```

Definiuje funkcję **game_played()** z parametrem opcjonalnym **read_write**, który domyślnie jest ustawiony na 'r'. Jeśli **read_write** jest różne od 'r', czyli inny tryb odczytu, funkcja usuwa dane z pliku "table_data0.csv" i wywołuje skrypt **web_scraping.py** za pomocą **subprocess.run()**, aby wygenerować nowe dane. W przeciwnym razie odczytuje dane z pliku i zwraca DataFrame.

Python

```
def championship_scores(path):  
    with open(path) as data:  
        data_of_games = pd.read_csv(data)  
    return data_of_games
```

Definiuje funkcję **championship_scores()** z parametrem **path**. Otwiera plik o podanej ścieżce i zwraca DataFrame zawierający dane z pliku CSV.

Python

```
if __name__ == '__main__':  
    does_exist()  
  
    path_to_championship_data = ["football_data0.csv",  
    "football_data1.csv"]  
  
    """if len(game_played()) != 500:  
        game_played('w')  
    else:  
        print('not empty')"""
```

Sprawdza, czy skrypt jest uruchamiany jako plik główny. Wywołuje funkcję ***does_exist()***. Zdefiniowane są ścieżki do plików danych mistrzostw w zmiennej ***path_to_championship_data***, ale zakomentowane jest wywołanie funkcji ***game_played()*** z warunkiem sprawdzającym ilość odczytanych gier.

Dokumentacja – mlearning_skilearn.py

Ta dokumentacja zawiera przegląd i wyjaśnienie funkcjonalności podanego kodu, który służy do analizy danych i uczenia maszynowego.

Python

```
from data_reader import game_played, championship_scores
import numpy as np
import pandas as pd
```

Importuje funkcje *game_played* i *championship_scores* z modułu *data_reader*. Importuje również bibliotekę *numpy* jako *np* oraz *pandas* jako *pd*.

python

```
def open_game_data():
    data_frame = game_played()
    return data_frame
```

Definiuje funkcję *open_game_data()*. Wywołuje funkcję *game_played()* z modułu *data_reader* i zwraca **DataFrame** zawierający dane z pliku "table_data0.csv".

Python

```
def open_championship_data(path):
    data_frame = championship_scores(path)
    return data_frame
```

Definiuje funkcję *open_championship_data(path)*. Wywołuje funkcję *championship_scores(path)* z modułu *data_reader* i zwraca **DataFrame** zawierający dane z pliku CSV o podanej ścieżce.

Python

```
def calculate_likelihood(team1, team2):  
    path_to_championship_data = ["football_data0.csv",  
    "football_data1.csv"]  
  
    merger =  
    pd.concat([open_championship_data(path_to_championship_data[0]),  
  
    open_championship_data(path_to_championship_data[1])])  
  
    team_stats_df = merger    game_scores_df = open_game_data()  
    team1 = team1.strip()  
    team2 = team2.strip()  
  
    team1_stats = team_stats_df.loc[team_stats_df['Team'] == team1]  
    team2_stats = team_stats_df.loc[team_stats_df['Team'] == team2]
```

Definiuje funkcję ***calculate_likelihood(team1, team2)***. Pobiera dane z dwóch plików CSV zawierających statystyki drużyn (mistrzostwa) i łączy je w jedną ramkę danych za pomocą funkcji ***pd.concat()***. Przypisuje połączoną ramkę danych do zmiennej ***team_stats_df***. Pobiera również dane meczowe z funkcji ***open_game_data()*** i przypisuje je do zmiennej ***game_scores_df***. Następnie oczyszcza nazwy drużyn z ewentualnych białych znaków.

Python

```
try:  
    team1_goals_scored = game_scores_df[(game_scores_df['team1']  
    == team1) &  
  
    (game_scores_df['team2']  
    == team2)][ 'score'].apply(lambda x: int(x.split('-')[0].strip()))  
  
    team1_goals_conceded =  
    game_scores_df[(game_scores_df['team1'] == team2) &  
  
    (game_scores_df['team2'] == team1)][ 'score'].apply(lambda x:  
    int(x.split('-')[1].strip()))  
  
    team2_goals_scored = game_scores_df[(game_scores_df['team1']  
    == team2) &
```



```

                                (game_scores_df['team2']
== team1)][ 'score'].apply(lambda x: int(x.split('-')[0].strip()))

    team2_goals_conceded =
game_scores_df[(game_scores_df['team1'] == team1) &

(game_scores_df['team2'] == team2)][ 'score'].apply(lambda x:
int(x.split('-')[1].strip()))

```

Oblicza liczbę bramek strzelonych i straconych przez obie drużyny w meczach między nimi. Korzysta z DataFrame'u **game_scores_df** do znalezienia wyników meczów i ekstrakcji liczby bramek dla poszczególnych drużyn.

Python

```

denominator = (team1_stats['W'].values[0] +
team1_stats['GF'].values[0] + team2_stats['L'].values[0] +
                team2_stats['GA'].values[0] + team1_goals_scored.mean()
+ team2_goals_conceded.mean() +
                team2_stats['W'].values[0] + team2_stats['GF'].values[0]
+ team1_stats['L'].values[0] +
                team1_stats['GA'].values[0] + team2_goals_scored.mean()
+ team1_goals_conceded.mean())

    if denominator == 0:
        return np.nan, np.nan

```

Oblicza mianownik dla obliczenia prawdopodobieństwa. Jeśli mianownik wynosi 0, zwraca NaN (Not a Number) dla obu drużyn.

Python

```

team1_likelihood = (team1_stats['W'].values[0] +
team1_stats['GF'].values[0] + team2_stats['L'].values[0] +
                    team2_stats['GA'].values[0] +
team1_goals_scored.mean() + team2_goals_conceded.mean()) /
denominator

    team2_likelihood = 1 - team1_likelihood

```

```
    return team1_likelihood * 100, team2_likelihood * 100

except IndexError:
    return np.nan, np.nan
```

Oblicza prawdopodobieństwo dla drużyn **team1** i **team2**. Korzysta z danych statystycznych drużyn i wyników meczów, aby obliczyć prawdopodobieństwo na podstawie pewnych formuł. Zapewnia, że sumaryczne prawdopodobieństwo wynosi 100%. Zwraca wynik jako krotkę z prawdopodobieństwem dla **team1** i **team2** pomnożonym przez 100. Jeśli wystąpi błąd indeksu (np. brak danych dla drużyn), zwraca NaN dla obu drużyn.