Dokumentacja Projektu: Przewidywanie Wyników Mistrzostw Świata 2022

# Przegląd Projektu

Projekt ten ma na celu przewidywanie wyników meczów podczas Mistrzostw Świata 2022 w piłce nożnej za pomocą technik uczenia maszynowego. Obejmuje przetwarzanie danych, trening modelu oraz interfejs użytkownika do wizualizacji przewidywań.

# Przegląd Plików

1. main\_final.py  
 - Cel: Ten skrypt jest odpowiedzialny za przetwarzanie danych, trening modelu oraz zapisywanie wytrenowanego modelu i enkoderów.  
 - Główne Biblioteki: numpy, tensorflow, sklearn, joblib  
 - Kluczowe Funkcjonalności:  
 - Ładowanie danych: Wczytywanie wyników meczów z pliku tekstowego results-v2.txt.  
 - Przetwarzanie danych: Przygotowanie danych wejściowych i wyjściowych, kodowanie nazw drużyn za pomocą OneHotEncoder, skalowanie cech.  
 - Trening modelu: Trenowanie modelu sieci neuronowej przy użyciu tensorflow i keras.  
 - Zapisywanie modelu: Zapisywanie wytrenowanego modelu i encodera do plików model.h5 i encoder.pkl.

2. prediction\_final.py  
 - Cel: Ten skrypt tworzy aplikację GUI do przewidywania wyników meczów i wyświetlania drabinki turniejowej.  
 - Główne Biblioteki: tkinter, numpy, tensorflow, joblib  
 - Kluczowe Funkcjonalności:  
 - Interfejs użytkownika: Tworzenie GUI za pomocą tkinter do wprowadzania nazw drużyn i wyświetlania przewidywanych wyników.  
 - Ładowanie modelu: Ładowanie wytrenowanego modelu i encodera.  
 - Przewidywanie wyników: Wykonywanie przewidywań na podstawie wprowadzonych drużyn.  
 - Wyświetlanie drabinki: Wyświetlanie wyników rund eliminacyjnych i finału w interfejsie użytkownika.

# Szczegółowy Opis Skryptów

## Main\_final.py

Ten skrypt jest kluczowy dla procesu przygotowania i treningu modelu. Składa się z kilku kroków:

### Importowanie bibliotek

Importowane są niezbędne biblioteki: numpy do operacji na tablicach, tensorflow i keras do budowy i trenowania modelu sieci neuronowej, sklearn do przetwarzania danych i joblib do zapisywania modeli.

import numpy as np  
import tensorflow as tf  
from tensorflow import keras  
from sklearn.model\_selection import train\_test\_split  
from sklearn.preprocessing import StandardScaler, OneHotEncoder  
import joblib

### Ładowanie danych

Dane są wczytywane z pliku results-v2.txt. Każda linia pliku reprezentuje wynik jednego meczu.

with open('results-v2.txt', encoding='utf-8-sig') as file:  
 matches\_X = []  
 matches\_Y = []  
 for line in file:  
 matches\_X.append([])  
 matches\_Y.append([])  
 elements = [element.strip() for element in line.split(',')]  
 for j, element in enumerate(elements):  
 if j < 3:  
 if j == 2:  
 matches\_X[-1].append(int(element))  
 else:  
 matches\_X[-1].append(element)  
 else:  
 matches\_Y[-1].append(int(element))

### Przygotowanie danych

Dane wejściowe matches\_X i wyjściowe matches\_Y są konwertowane na tablice numpy. Następnie dane są rozszerzane, aby zawierały także mecze z zamienionymi miejscami drużyn (co podwaja zestaw danych).

data = np.array(matches\_X)  
y = np.array(matches\_Y)  
  
data\_flipped = np.hstack((data[:, 1].reshape(-1, 1), data[:, 0].reshape(-1, 1), data[:, 2].reshape(-1, 1)))  
data\_combined = np.vstack((data, data\_flipped))  
y\_combined = np.vstack((y, y))

### Kodowanie i skalowanie danych

Nazwy drużyn są kodowane przy użyciu OneHotEncoder, a cechy są skalowane przy użyciu StandardScaler.

encoder = OneHotEncoder(handle\_unknown='ignore', sparse\_output=False)  
encoded\_teams = encoder.fit\_transform(data[:, :2])  
joblib.dump(encoder, 'encoder.pkl')  
  
X = np.hstack((encoded\_teams, data[:, 2].reshape(-1, 1)))  
  
scaler = StandardScaler()  
X\_train = scaler.fit\_transform(X\_train)  
X\_val = scaler.transform(X\_val)  
X\_test = scaler.transform(X\_test)

Dane testowe:

### Trening modelu

Model sieci neuronowej jest definiowany i trenowany. Składa się z dwóch warstw ukrytych z aktywacją ReLU oraz warstwy wyjściowej.

model = keras.Sequential([  
 keras.layers.Dense(64, activation='relu', input\_shape=(X\_train.shape[1],)),  
 keras.layers.Dense(64, activation='relu'),  
 keras.layers.Dense(2)  
])  
  
model.compile(optimizer='adam', loss='mean\_squared\_error', metrics=['mae'])  
model.fit(X\_train, y\_train, epochs=1000, validation\_data=(X\_val, y\_val))

### Zapisywanie modelu

Po zakończeniu treningu, model jest zapisywany do pliku model.h5.

model.save('model.h5')

## predictions\_final.py

Ten skrypt tworzy aplikację GUI, która pozwala użytkownikowi wprowadzać drużyny i wyświetlać przewidywane wyniki meczów. Składa się z następujących kroków:

### Importowanie bibliotek

Importowane są niezbędne biblioteki: tkinter do tworzenia GUI, numpy do operacji na tablicach, tensorflow do ładowania modelu i joblib do ładowania enkodera.

import tkinter as tk  
import numpy as np  
from tensorflow import keras  
import joblib

### Definiowanie klasy aplikacji

Klasa SoccerPredictionApp tworzy główne okno aplikacji i definiuje wszystkie niezbędne widgety oraz funkcje.

class SoccerPredictionApp:  
 def \_\_init\_\_(self, master):  
 self.master = master  
 self.master.title('World Cup 2022 Predictor')  
   
 self.model = keras.models.load\_model('model.h5')  
 self.encoder = joblib.load('encoder.pkl')  
   
 self.create\_widgets()

### Tworzenie widgetów GUI

Widgety GUI są tworzone do wprowadzania nazw drużyn, przycisku przewidywania i etykiety wyników.

def create\_widgets(self):  
 self.team1\_label = tk.Label(self.master, text='Team 1:')  
 self.team1\_label.grid(row=0, column=0)  
 self.team1\_entry = tk.Entry(self.master)  
 self.team1\_entry.grid(row=0, column=1)  
   
 self.team2\_label = tk.Label(self.master, text='Team 2:')  
 self.team2\_label.grid(row=1, column=0)  
 self.team2\_entry = tk.Entry(self.master)  
 self.team2\_entry.grid(row=1, column=1)  
   
 self.predict\_button = tk.Button(self.master, text='Predict', command=self.predict)  
 self.predict\_button.grid(row=2, column=0, columnspan=2)  
   
 self.result\_label = tk.Label(self.master, text='')  
 self.result\_label.grid(row=3, column=0, columnspan=2)

### Funkcja przewidywania

Ta funkcja pobiera dane z pól wejściowych, przetwarza je, wykonuje przewidywanie za pomocą modelu i wyświetla wyniki w GUI.

def predict(self):  
 team1 = self.team1\_entry.get()  
 team2 = self.team2\_entry.get()  
   
 input\_data = np.array([[team1, team2, 2022]])  
 encoded\_input = self.encoder.transform(input\_data[:, :2])  
 final\_input = np.hstack((encoded\_input, input\_data[:, 2].reshape(-1, 1)))  
   
 prediction = self.model.predict(final\_input)  
 self.result\_label.config(text=f'Predicted Score: {prediction[0][0]:.1f} - {prediction[0][1]:.1f}')

### Uruchomienie aplikacji

Tworzenie instancji klasy SoccerPredictionApp i uruchomienie głównej pętli tkinter.

if \_\_name\_\_ == '\_\_main\_\_':  
 root = tk.Tk()  
 app = SoccerPredictionApp(root)  
 root.mainloop()

# Użycie

1. Trening modelu:  
 - Uruchom main\_final.py, aby przetworzyć dane, wytrenować model i zapisać go do pliku model.h5.  
  
 python main\_final.py  
  
2. Interfejs użytkownika:  
 - Uruchom prediction\_final.py, aby otworzyć aplikację GUI. Wprowadź nazwy drużyn i kliknij 'Predict', aby zobaczyć przewidywane wyniki meczów.  
  
 python prediction\_finsl.py

# Szczegółowy Opis Treningu Modelu

1. Importowanie danych i bibliotek:  
 - Dane są wczytywane z pliku results-v2.txt. Każda linia pliku zawiera informacje o meczu w formacie: drużyna1, drużyna2, rok, wynik\_drużyna1, wynik\_drużyna2.  
 - Importowane są biblioteki numpy, tensorflow, keras, sklearn i joblib.  
  
2. Przygotowanie danych:  
 - Dane są wczytywane i przetwarzane do tablic numpy.  
 - Dane są rozszerzane poprzez dodanie wersji, gdzie drużyny są zamienione miejscami (np. jeśli oryginalny mecz to A vs B, to dodajemy także B vs A).  
 - Nazwy drużyn są kodowane do formatu jednowymiarowego wektorów binarnych (OneHotEncoder).  
 - Cechy są skalowane do standardowego zakresu (StandardScaler).  
  
3. Podział danych:  
 - Dane są dzielone na zestawy treningowe, testowe i walidacyjne (train\_test\_split).  
  
4. Definiowanie modelu:  
 - Model sieci neuronowej jest zdefiniowany z dwoma ukrytymi warstwami z aktywacją ReLU i jedną warstwą wyjściową.  
 - Model jest kompilowany przy użyciu optymalizatora Adam i funkcji straty mean\_squared\_error.  
  
5. Trening modelu:  
 - Model jest trenowany na danych treningowych przez 1000 epok z walidacją na danych walidacyjnych.  
 - Podczas treningu, model dostosowuje swoje wagi w celu minimalizacji funkcji straty.  
  
6. Zapisywanie modelu:  
 - Po zakończeniu treningu, model i encoder są zapisywane do plików model.h5 i encoder.pkl, co umożliwia ich późniejsze użycie bez potrzeby ponownego treningu.

Dane testowe:

W ramach naszego projektu przewidującego wyniki Mundialu 2022, przeprowadziliśmy trening modelu wykorzystując dane z ostatnich 15 meczów każdej z drużyn. Nasz model został zoptymalizowany pod kątem tej liczby meczów, ponieważ uzyskane wyniki były najbardziej obiecujące.

Dla celów porównawczych, przeprowadziliśmy również trening modelu przy użyciu danych z ostatnich 5 oraz 30 meczów każdej z drużyn. Wyniki tych eksperymentów okazały się jednak mniej satysfakcjonujące. W przypadku modelu trenowanego na podstawie 5 ostatnich meczów, dane były zbyt ograniczone, co prowadziło do niedostatecznego odwzorowania formy drużyn. Z kolei model trenowany na podstawie 30 meczów zawierał zbyt wiele zmiennych danych, co skutkowało mniejszą precyzją przewidywań.

Podsumowując, optymalna liczba 15 ostatnich meczów pozwoliła na osiągnięcie najlepszych wyników w przewidywaniu wyników meczów na Mundialu 2022.

W ramach naszego projektu przewidywania wyników Mundialu 2022 inspirowaliśmy się różnymi podejściami i technikami z istniejących projektów i artykułów naukowych. Oto kilka źródeł, które mogłyby być użyteczne:

1. **GitHub - Football Match Predictor**: Projekt ten przedstawia szczegółowe podejście do przewidywania wyników meczów piłkarskich, w tym czyszczenie danych, wybór cech i modelowanie. Wykorzystuje różne cechy, takie jak liczba strzałów na bramkę i kartki czerwone oraz żółte, które zostały analizowane pod kątem ich wpływu na wynik meczu. [GitHub - Football Match Predictor](https://github.com/aziztitu/football-match-predictor)​ ([GitHub](https://github.com/aziztitu/football-match-predictor))​
2. **How to Use Python and Machine Learning to Predict Football Match Winners**: Artykuł ten szczegółowo opisuje proces przetwarzania danych, obliczania średnich wyników z poprzednich meczów oraz implementacji różnych modeli predykcyjnych, takich jak rozkład Poissona i maszyny wektorów nośnych (SVM). KDnuggets - How to Use Python and Machine Learning to Predict Football Match Winners​ ([KDnuggets](https://www.kdnuggets.com/2023/01/python-machine-learning-predict-football-match-winners.html" \t "_blank))​
3. **Machine Learning for Soccer Match Result Prediction**: Ten artykuł naukowy omawia różne algorytmy uczenia maszynowego stosowane do przewidywania wyników meczów piłkarskich, analizując dostępne zestawy danych oraz metody oceny wydajności modeli. [arXiv - Machine Learning for Soccer Match Result Prediction](https://arxiv.org/abs/2403.07669)​ ([ar5iv](https://ar5iv.org/abs/2403.07669))​

Te źródła dostarczają wszechstronnego wglądu w metody przetwarzania danych, wybór cech oraz różne podejścia do modelowania, które były pomocne w naszym projekcie.