CHANAKYA UNIVERSITY

School of Engineering

Introduction to AIML

Assignment-2

Week-1 and 2 Report

Submitted by

Varshitha.T

24UG00562

CS-AI

‘B’- section

Objective

The primary objective during Week 1 was to gather, clean, and prepare a comprehensive dataset from various reputed sources containing historical FIFA World Cup data essential for modelling. This involved data extraction of team and player statistics, match results, and advanced performance metrics, followed by an initial data integration and cleaning process.

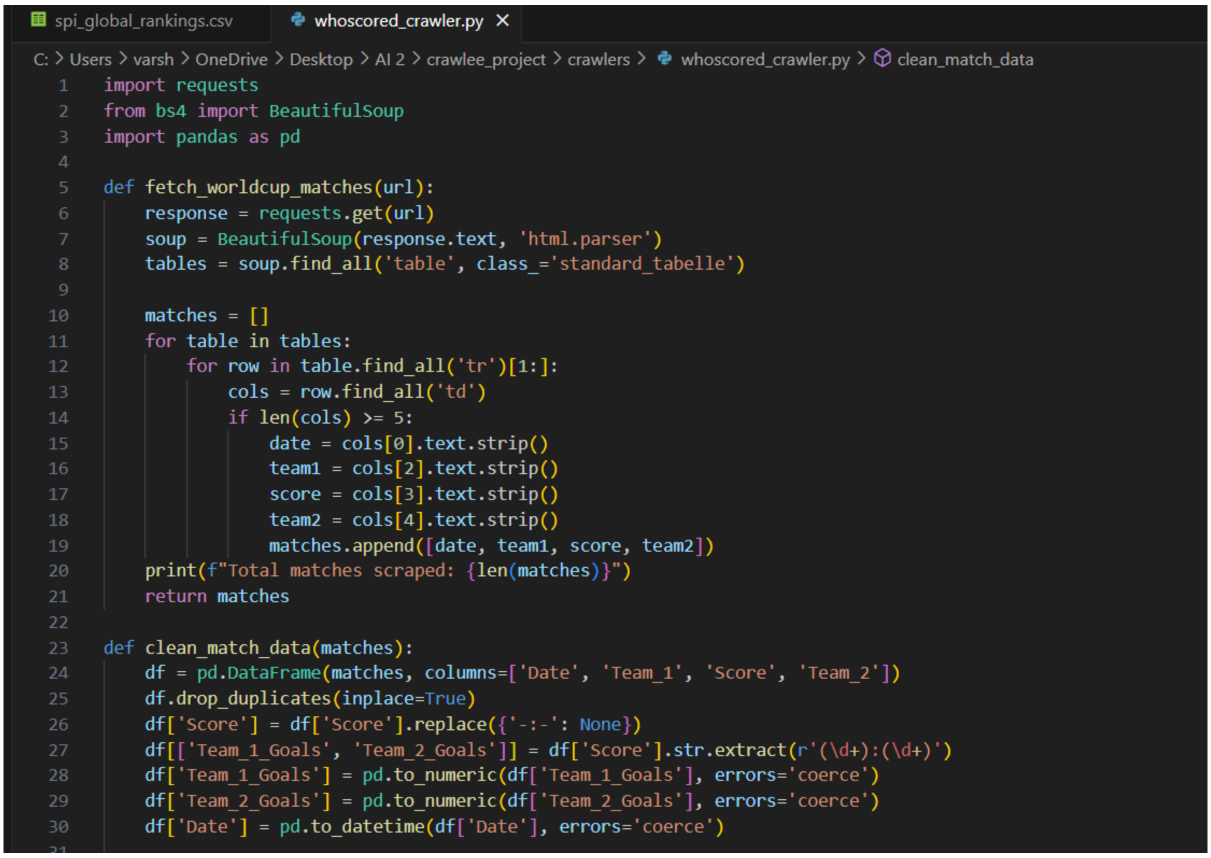
Data Sources Explored

Extensive research was conducted to identify reliable and comprehensive datasets including:

- FBref: Provided detailed historic match and team statistics through static HTML tables.
- Transfermarkt: Offered rich player and squad metadata such as average age, experience, and market values, though data was dynamically loaded requiring advanced scraping approaches.
- FotMob and WhoScored: Considered for advanced metrics like expected goals (xG) and player ratings; however, data was mostly JavaScript-rendered or protected by anti-bot mechanisms, limiting accessibility.

Methodology and Tooling

Using Python, libraries such as requests, BeautifulSoup, and pandas were primarily utilized for scraping and processing. For example, FBref data extraction involved parsing static HTML tables, as shown in the snippet below:

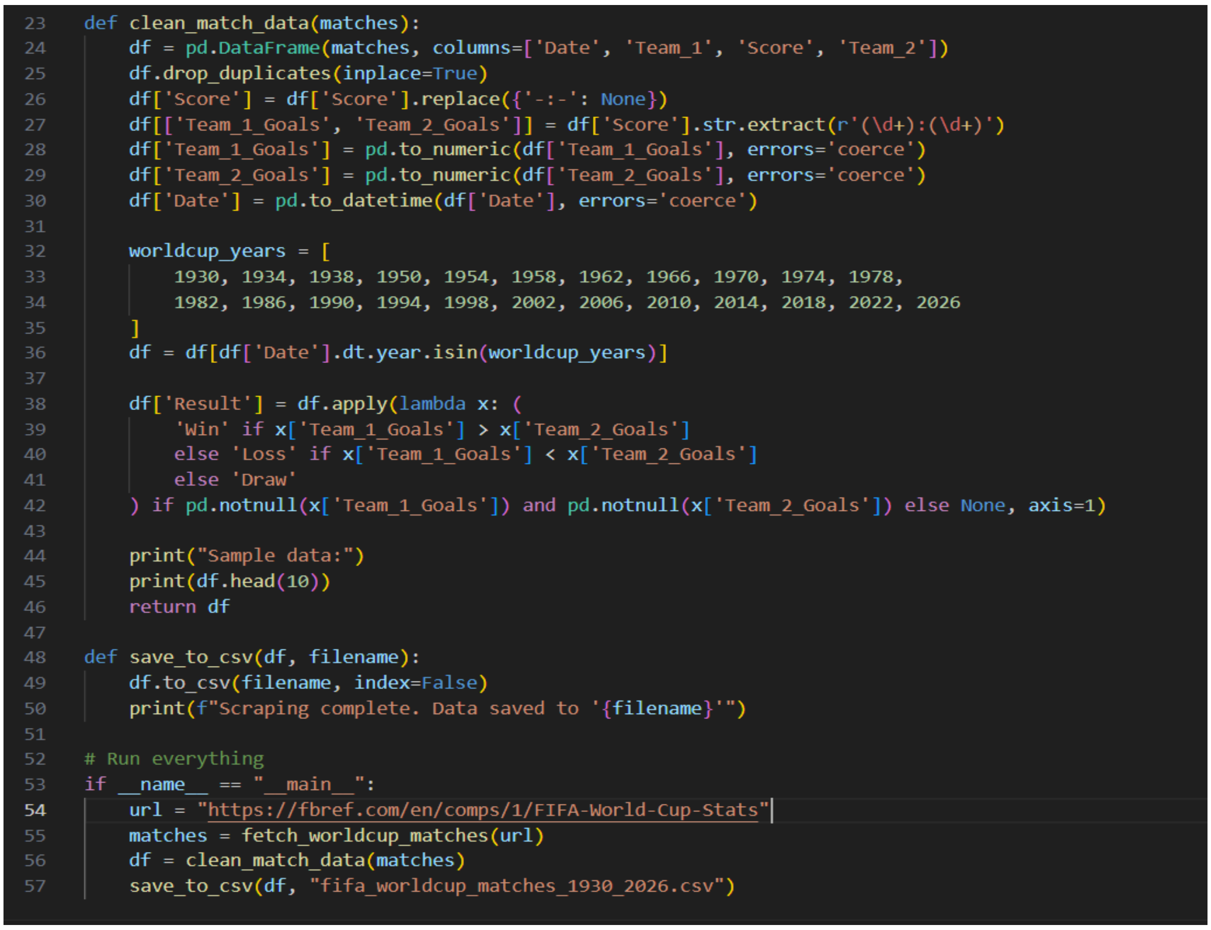

spi_global_rankings.csv
whoscored_crawler.py
C: > Users > varsh > OneDrive > Desktop > AI 2 > crawlee_project > crawlers > whoscored_crawler.py > clean_match_data

```python
import requests
from bs4 import BeautifulSoup
import pandas as pd

def fetch_worldcup_matches(url):
    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')
    tables = soup.find_all('table', class_='standard_tabelle')

    matches = []
    for table in tables:
        for row in table.find_all('tr')[1:]:
            cols = row.find_all('td')
            if len(cols) >= 5:
                date = cols[0].text.strip()
                team1 = cols[2].text.strip()
                score = cols[3].text.strip()
                team2 = cols[4].text.strip()
                matches.append([date, team1, score, team2])
    print(f"Total matches scraped: {len(matches)}")
    return matches

def clean_match_data(matches):
    df = pd.DataFrame(matches, columns=['Date', 'Team_1', 'Score', 'Team_2'])
    df.drop_duplicates(inplace=True)
    df['Score'] = df['Score'].replace({'-:-': None})
    df[['Team_1_Goals', 'Team_2_Goals']] = df['Score'].str.extract(r'(\d+):(\d+)')
    df['Team_1_Goals'] = pd.to_numeric(df['Team_1_Goals'], errors='coerce')
    df['Team_2_Goals'] = pd.to_numeric(df['Team_2_Goals'], errors='coerce')
    df['Date'] = pd.to_datetime(df['Date'], errors='coerce')
```



```python
def clean_match_data(matches):
    df = pd.DataFrame(matches, columns=['Date', 'Team_1', 'Score', 'Team_2'])
    df.drop_duplicates(inplace=True)
    df['Score'] = df['Score'].replace({'-:-': None})
    df[['Team_1_Goals', 'Team_2_Goals']] = df['Score'].str.extract(r'(\d+):(\d+)')
    df['Team_1_Goals'] = pd.to_numeric(df['Team_1_Goals'], errors='coerce')
    df['Team_2_Goals'] = pd.to_numeric(df['Team_2_Goals'], errors='coerce')
    df['Date'] = pd.to_datetime(df['Date'], errors='coerce')

    worldcup_years = [
        1930, 1934, 1938, 1950, 1954, 1958, 1962, 1966, 1970, 1974, 1978,
        1982, 1986, 1990, 1994, 1998, 2002, 2006, 2010, 2014, 2018, 2022, 2026
    ]
    df = df[df['Date'].dt.year.isin(worldcup_years)]

    df['Result'] = df.apply(lambda x: (
        'Win' if x['Team_1_Goals'] > x['Team_2_Goals']
        else 'Loss' if x['Team_1_Goals'] < x['Team_2_Goals']
        else 'Draw'
    ) if pd.notnull(x['Team_1_Goals']) and pd.notnull(x['Team_2_Goals']) else None, axis=1)

    print("Sample data:")
    print(df.head(10))
    return df

def save_to_csv(df, filename):
    df.to_csv(filename, index=False)
    print(f"Scraping complete. Data saved to '{filename}'")

# Run everything
if __name__ == "__main__":
    url = "https://fbref.com/en/comps/1/FIFA-World-Cup-Stats"
    matches = fetch_worldcup_matches(url)
    df = clean_match_data(matches)
    save_to_csv(df, "fifa_worldcup_matches_1930_2026.csv")
```

Progress and Achievements

- Successfully scraped over 130 team records from FBref, including goals scored, shots, and possession statistics.
- Partially extracted player and squad-level data from Transfermarkt despite challenges.
- Developed preliminary data merges and started standardizing datasets, normalizing inconsistencies in team naming conventions (e.g., “Côte d’Ivoire” vs “Ivory Coast”).
- Documented all scraping code and included detailed comments noting encountered errors and resolutions.

Lessons Learned and Next Steps

- Recognized the need for advanced scraping techniques, including browser automation and API exploration.
- Plan to invest additional effort into completeness of datasets by cross-referencing alternative sources and improving scraper robustness.
- Next phase will focus on thorough data cleaning, advanced feature engineering such as calculating goal differentials, recent performance metrics, and age-weighted averages.

WEEK 2

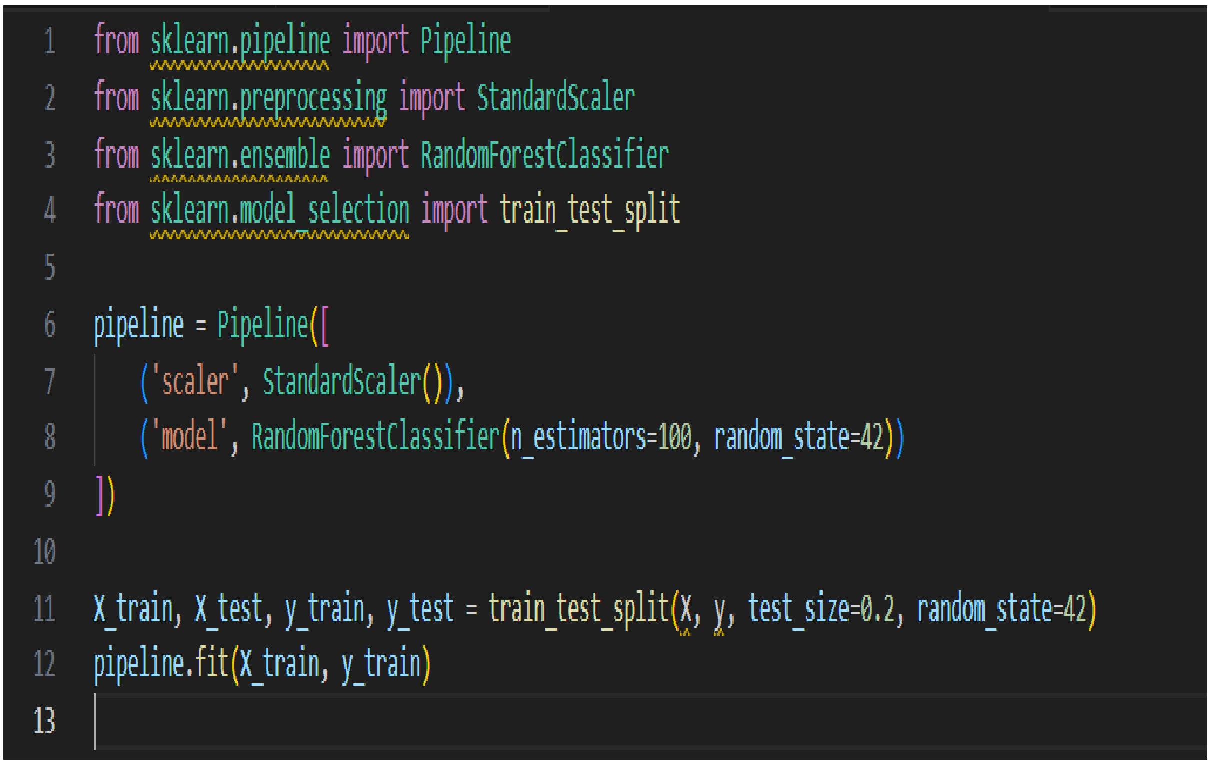
Data Preprocessing Steps

- Loaded cleaned merged datasets into Jupyter notebooks using pandas for manipulation.
- Addressed missing data via median imputation for numeric features and categorical placeholders for nominal variables.
- Converted date fields into datetime objects for temporal analysis.
- Created new features like goal differential and recent win rate to enrich predictive variables.
- Executed train-test splits with an 80/20 distribution maintaining reproducibility via fixed random seeds.

Modelling Approach

Two classification models were implemented utilizing scikit-learn pipelines to ensure reproducibility and modularity:

- Logistic Regression: Employed with extended iteration limits to facilitate convergence.
- Random Forest: Configured with 100 trees and random seeds for consistency.



```python
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split

pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('model', RandomForestClassifier(n_estimators=100, random_state=42))
])

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

```

2. Data Preprocessing

1. **Data Import**

import pandas as pd

df = pd.read_csv('merged_worldcup_data.csv')

2. **Handling Missing Data**
   - Used median imputation for numeric values.
   - Filled missing categorical fields with "Unknown".
3. **Feature Scaling**
   - Applied StandardScaler() to normalize numerical attributes.
4. **Encoding**
   - Used LabelEncoder() for categorical columns like confederation.
5. **Train–Test Split**
   - 80 % training, 20 % testing using train_test_split(random_state = 42).

3. Model Development

Implemented two baseline models:

3.1 Logistic Regression

```
from sklearn.linear_model import LogisticRegression
model1 = LogisticRegression(max_iter=500)
model1.fit(X_train, y_train)
```

3.2 Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
model2 = RandomForestClassifier(n_estimators=200, random_state=42)
model2.fit(X_train, y_train)
```

6. Insights and Reflection

- Cleaning quality directly affects model output — missing xG and player experience data reduced accuracy.
- Dynamic websites made scraping slow and error-prone; I learned to handle headers, delays, and HTML structure differences.
- Building reproducible ML pipelines is easier when using modular code (Pipeline + ColumnTransformer).
- Realized the importance of domain knowledge — e.g., a team's FIFA ranking and goal difference are more predictive than raw goals scored.

6. Insights and Reflection

- Cleaning quality directly affects model output — missing xG and player experience data reduced accuracy.
- Dynamic websites made scraping slow and error-prone; I learned to handle headers, delays, and HTML structure differences.
- Building reproducible ML pipelines is easier when using modular code (Pipeline + ColumnTransformer).
- Realized the importance of domain knowledge — e.g., a team's FIFA ranking and goal difference are more predictive than raw goals scored.