# PROJECT REPORT

## Prediction of Fc Barcelona Football Matches

By

HARSH DIPDATT PATIL

# INDEX

# 1. INTRODUCTION

Football is most popular sport in the world, predicting football matches outcome is a challenging task due to the dynamic nature of the game. This project focuses on developing a machine learning model to predict match outcome (win, draw or loss) of one specific team “Fc Barcelona”.
Fc Barcelona is one of the most followed clubs in the world. Founded in 1899, the club has won multiple league titles UEFA Champions League, Copa del Rey, Fifa club world cup, La liga Titles and many more.Their matches attracts millions of spectators making them an ideal team for predictive analysis.
The Primary obejctive of this project is to train a machine learning model on custom built dataset and evaluate, that can predict the outcome of Fc Barcelona (win, draw or loss).

## 1.1 Problem Statement

Predicting football match outcome is complex due to numerous variables and due to the dynamic nature of the game. This project addresses the challenges by developing a machine learing model that can predicts Fc Barcelona’s match results based on historical performance metrics and opponents analysis.

## 1.2 Project Scope

The project includes:

- Data collection : Collection Fc Barcelona’s data of past 2 season to built custom dataset.
  Data preprocessing : Cleaning, encoding, and feature engineering to enhance model
- performance.
  Model development : Training an XGBoost and Random forest classifier for outcome prediction.
- Web Appliction : Building on flask based frontend for user interaction.

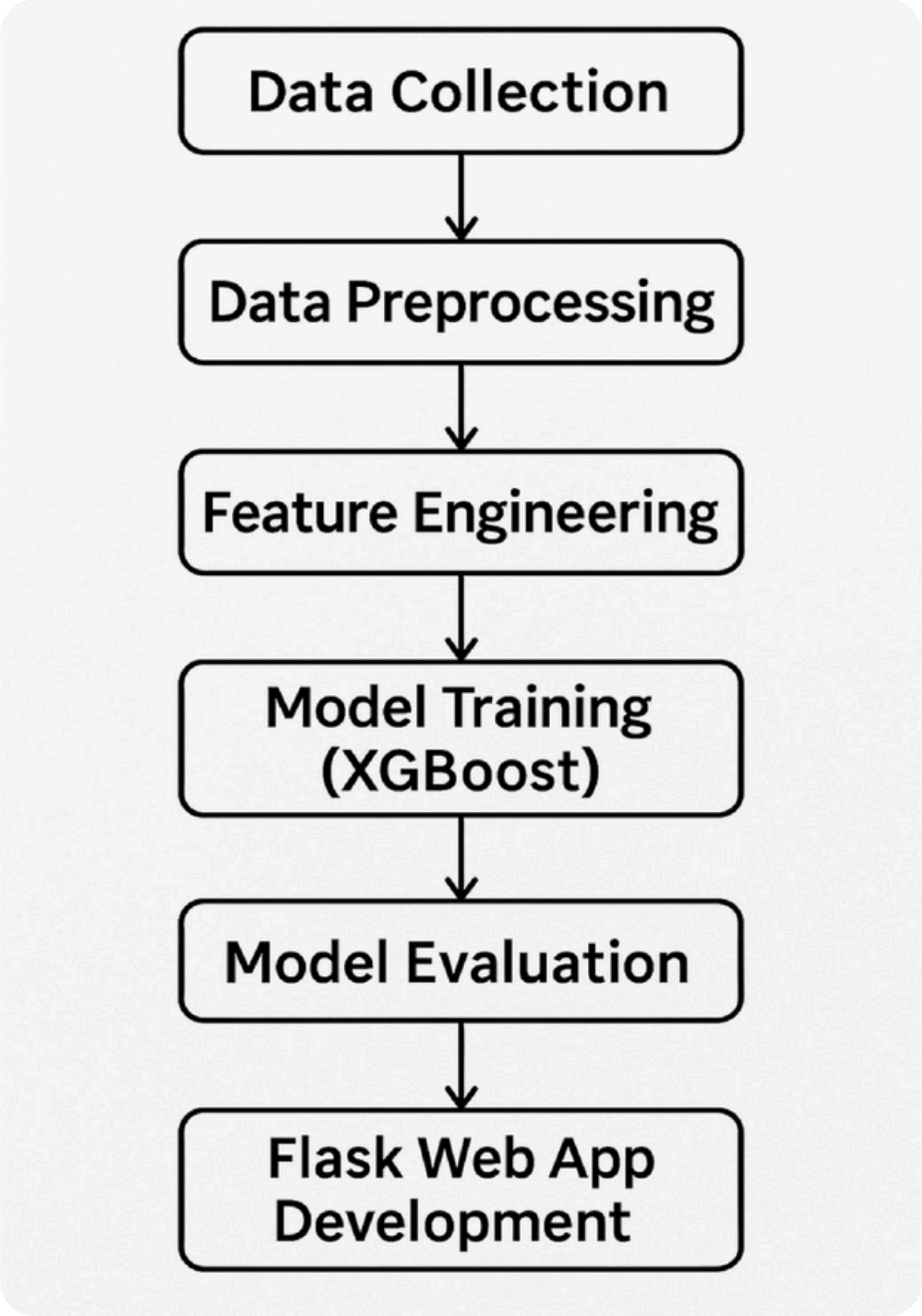
## 1.3 Objectives of the Project

The Primary obejctive of this project is to train a machine learning model on custom built dataset and evaluate, that can predict the outcome of Fc Barcelona (win, draw or loss) and connect the model to web application with the help of flask so that user can have a friendly interface to interact. Aslo to emonstrates the power of machine learning in sports analytics and helps fans and analysts make informed predictions using historical trends.

## 1.4 Tools & Technologies Used

- Software : Microsoft excel, Vscode, Jupyter Nootbook.
- Backend: Python, Flask, XGBoost, Scikit-learn.
- Frontend: HTML, CSS, JavaScript.
- Deployment: Local Flask server.

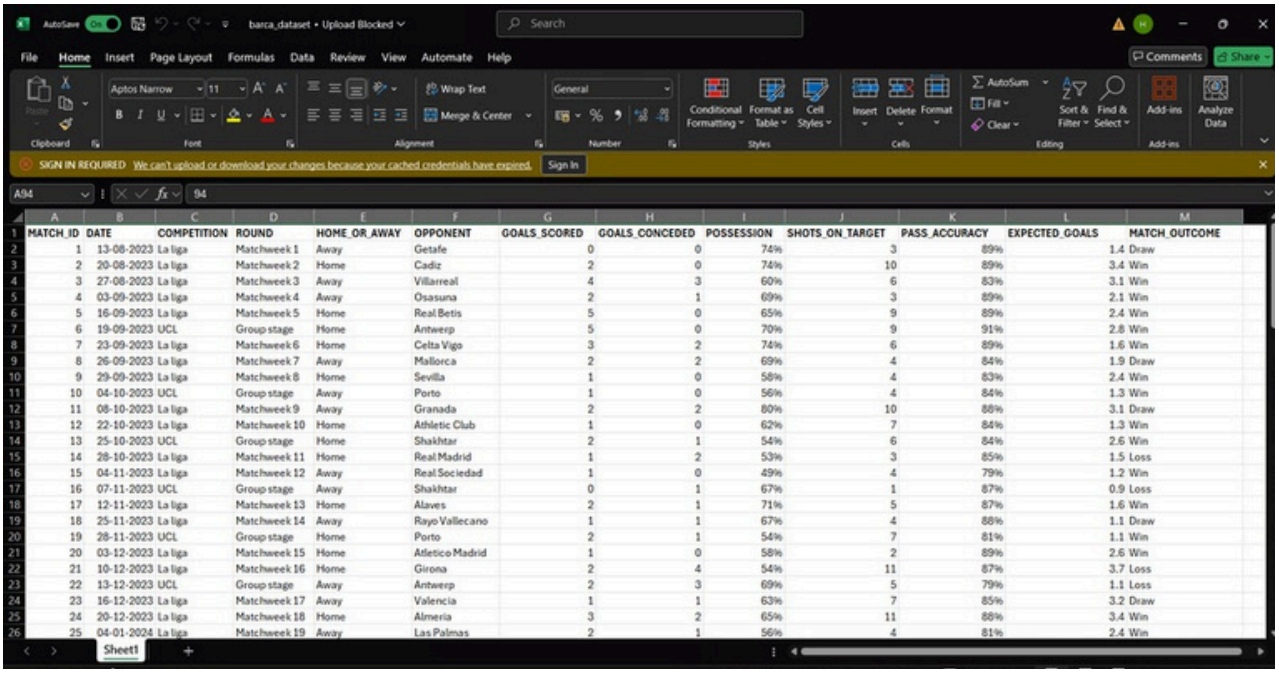
## 2. WORKFLOW DIAGRAM



# 3. PROPOSED SYSTEM

## 3.1 Data Collection

To create a dataset we have taken insights from fbref.com and then created a custom built dataset on Microsoft excel. Our Dataset contain a total of 95 matches, includes all competition FC Barcelona participated in 2023/24 and 2024/25 season, since 2024/25 season currently going on so we only have data till the date of 05-03- 2025.



| MATCH_ID | DATE | COMPETITION | ROUND | HOME_OR_AWAY | OPPONENT | GOALS_SCORED | GOALS_CONCEDED | POSSESSION | SHOTS_ON_TARGET | PASS_ACCURACY | EXPECTED_GOALS | MATCH_OUTCOME |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 13-08-2023 | La liga | Matchweek 1 | Away | Getafe | 0 | 0 | 74% | 3 | 89% | 1.4 | Draw |
| 2 | 20-08-2023 | La liga | Matchweek 2 | Home | Cadiz | 2 | 0 | 74% | 10 | 89% | 3.4 | Win |
| 3 | 27-08-2023 | La liga | Matchweek 3 | Away | Villarreal | 4 | 3 | 60% | 6 | 83% | 3.1 | Win |
| 4 | 03-09-2023 | La liga | Matchweek 4 | Away | Osasuna | 2 | 1 | 69% | 3 | 89% | 2.1 | Win |
| 5 | 16-09-2023 | La liga | Matchweek 5 | Home | Real Betis | 5 | 0 | 65% | 9 | 89% | 2.4 | Win |
| 6 | 19-09-2023 | UCL | Group stage | Home | Antwerp | 5 | 0 | 70% | 9 | 91% | 2.8 | Win |
| 7 | 23-09-2023 | La liga | Matchweek 6 | Home | Celta Vigo | 3 | 2 | 74% | 6 | 89% | 1.6 | Win |
| 8 | 26-09-2023 | La liga | Matchweek 7 | Away | Mallorca | 2 | 2 | 69% | 4 | 84% | 1.9 | Draw |
| 9 | 29-09-2023 | La liga | Matchweek 8 | Home | Sevilla | 1 | 0 | 58% | 4 | 83% | 2.4 | Win |
| 10 | 04-10-2023 | UCL | Group stage | Away | Porto | 1 | 0 | 56% | 4 | 84% | 1.3 | Win |
| 11 | 08-10-2023 | La liga | Matchweek 9 | Away | Granada | 2 | 2 | 80% | 10 | 88% | 3.1 | Draw |
| 12 | 22-10-2023 | La liga | Matchweek 10 | Home | Athletic Club | 1 | 0 | 62% | 7 | 84% | 1.3 | Win |
| 13 | 25-10-2023 | UCL | Group stage | Home | Shakhtar | 2 | 1 | 54% | 6 | 84% | 2.6 | Win |
| 14 | 28-10-2023 | La liga | Matchweek 11 | Home | Real Madrid | 1 | 2 | 53% | 3 | 85% | 1.5 | Loss |
| 15 | 04-11-2023 | La liga | Matchweek 12 | Away | Real Sociedad | 1 | 0 | 49% | 4 | 79% | 1.2 | Win |
| 16 | 07-11-2023 | UCL | Group stage | Away | Shakhtar | 0 | 1 | 67% | 1 | 87% | 0.9 | Loss |
| 17 | 12-11-2023 | La liga | Matchweek 13 | Home | Alaves | 2 | 1 | 71% | 5 | 87% | 1.6 | Win |
| 18 | 25-11-2023 | La liga | Matchweek 14 | Away | Rayo Vallecano | 1 | 1 | 67% | 4 | 88% | 1.1 | Draw |
| 19 | 28-11-2023 | UCL | Group stage | Home | Porto | 2 | 1 | 54% | 7 | 81% | 1.1 | Win |
| 20 | 03-12-2023 | La liga | Matchweek 15 | Home | Atletico Madrid | 1 | 0 | 58% | 2 | 89% | 2.6 | Win |
| 21 | 10-12-2023 | La liga | Matchweek 16 | Home | Girona | 2 | 4 | 54% | 11 | 87% | 3.7 | Loss |
| 22 | 13-12-2023 | UCL | Group stage | Away | Antwerp | 2 | 3 | 69% | 5 | 79% | 1.1 | Loss |
| 23 | 16-12-2023 | La liga | Matchweek 17 | Away | Valencia | 1 | 1 | 63% | 7 | 85% | 3.2 | Draw |
| 24 | 20-12-2023 | La liga | Matchweek 18 | Home | Almeria | 3 | 2 | 65% | 11 | 88% | 3.4 | Win |
| 25 | 04-01-2024 | La liga | Matchweek 19 | Away | Las Palmas | 2 | 1 | 56% | 4 | 81% | 2.4 | Win |

## 3.2 Importing Libraries

Importing needed libraries, Model training and evaluation utilities.

```
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV, KFold, cross_val_score
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import accuracy_score, classification_report, precision_score, recall_score, f1_score
from xgboost import XGBClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_selection import RFE
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

### 3.3 Data preprocessing

Since the dataset was self-created and manually entered in Microsoft Excel half of the preprocessing part was done, the only part was left is to convert “String” datatypes into “integers”, to prepare the data for modelling. Following steps were performed:

[3]:
```
# Check for missing values in the target column
print("Missing values in MATCH_OUTCOME:", data['MATCH_OUTCOME'].isnull().sum())
```

```
Missing values in MATCH_OUTCOME: 0
```

[4]:
```
# Map the outcomes to numerical values
outcome_mapping = {'Win': 0, 'Draw': 1, 'Loss': 2}
data['MATCH_OUTCOME'] = data['MATCH_OUTCOME'].map(outcome_mapping)
```

[5]:
```
# Convert percentage columns to float
percentage_columns = ['POSSESSION', 'PASS_ACCURACY']
for col in percentage_columns:
    data[col] = data[col].str.rstrip('%').astype(float) / 100.0
```

[6]:
```
# Encode home/away and opponent names
data['HOME_OR_AWAY'] = data['HOME_OR_AWAY'].map({'Home': 0, 'Away': 1})
label_encoder = LabelEncoder()
data['OPPONENT'] = label_encoder.fit_transform(data['OPPONENT'])
```

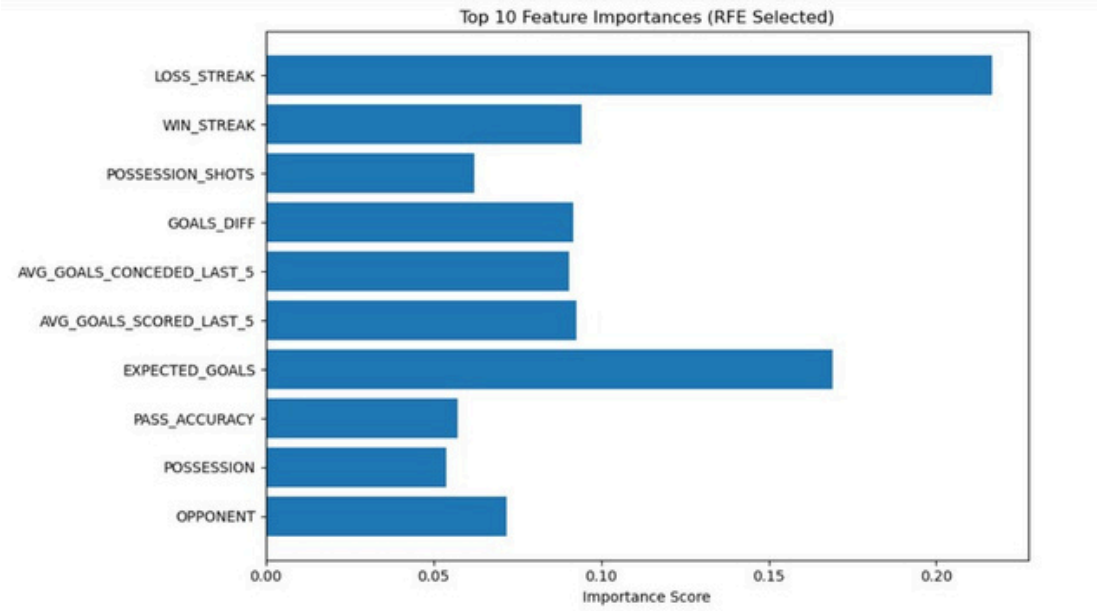
### 3.4 Feature Engineering

We engineered 4 more features to enhance our model and to increase accuracy of our machine learning model, the following features were engineered:

[3]:
```
# Feature Engineering: Add new features
# 1. Average goals scored and conceded in the last 5 matches
data['AVG_GOALS_SCORED_LAST_5'] = data['GOALS_SCORED'].rolling(window=5, min_periods=1).mean()
data['AVG_GOALS_CONCEDED_LAST_5'] = data['GOALS_CONCEDED'].rolling(window=5, min_periods=1).mean()

# 2. Difference between expected goals and actual goals
data['GOALS_DIFF'] = data['GOALS_SCORED'] - data['EXPECTED_GOALS']

# 3. Interaction feature: Possession * Shots on Target
data['POSSESSION_SHOTS'] = data['POSSESSION'] * data['SHOTS_ON_TARGET']

# 4. Win/Loss streak (Last 3 matches)
data['WIN_STREAK'] = data['MATCH_OUTCOME'].rolling(window=3, min_periods=1).apply(lambda x: (x == 0).sum())
data['LOSS_STREAK'] = data['MATCH_OUTCOME'].rolling(window=3, min_periods=1).apply(lambda x: (x == 2).sum())
```

### 3.5 Feature Selection & Scaling

For Feature selections RFE (Recursive Feature Elimination) was used and top 10 most important features to avoid overfitting. Are as follows:




### 3.6 Model Training

For our project Random Forest and XGBoost Classifier was selected, once the model was selected training process was started, in this phase involved data splitting, hyperparameter tuning and final model fitting. **Train Test Split:** The dataset was divided into two distinct subsets: ·Training Set (80%): Used to train the model and perform cross validation. ·Testing Set (20%): Reserved for evaluating the model's generalization ability on unseen data. This stratified split ensured that all outcome classes (Win, Draw, Loss) were proportionally represented in both sets, maintaining class balance **Hyperparameter Tuning:** To optimize model performance, GridSearchCV was utilized for systematic hyperparameter tuning. A predefined range of values was tested for each key parameter:

- For Random Forest: n_estimators, max_depth, min_samples_split, and criterion.
- For XGBoost: learning_rate, max_depth, n_estimators, and subsample

GridSearchCV evaluated each combination using cross-validation and selected the configuration that achieved the best score, primarily using accuracy and F1-score as the selection criteria

[23]:
```
# Hyperparameter Tuning for XGBoost
param_grid = {
    'n_estimators': [50, 100, 200, 300],
    'max_depth': [3, 5, 7],
    'learning_rate': [0.01, 0.1, 0.2],
    'subsample': [0.8, 0.9, 1.0],
    'colsample_bytree': [0.8, 0.9, 1.0],
    'gamma': [0, 0.1, 0.2]
}

xgb_model = XGBClassifier(random_state=42, objective='multi:softprob')
grid_search = GridSearchCV(estimator=xgb_model, param_grid=param_grid, cv=5, scoring='accuracy', n_jobs=-1)
grid_search.fit(X_train_rfe, y_train)

print("Best Parameters for XGBoost:", grid_search.best_params_)
```

```
Best Parameters for XGBoost: {'colsample_bytree': 0.9, 'gamma': 0.1, 'learning_rate': 0.2, 'max_depth': 5, 'n_estimators': 50,
'subsample': 1.0}
```

**Final model fitting** :

[24]:
```
# Train the final model with the best parameters
final_model = grid_search.best_estimator_
final_model.fit(X_train_rfe, y_train)
```

[24]:
```
XGBClassifier
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=0.9, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=0.1, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.2, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=5, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=50, n_jobs=None,
              num_parallel_tree=None, objective='multi:softprob', ...)
```

### 3.7 Flask Web Application Development

To make a our model user friendly we have built a web appliction using HTML, CSS, Javascript and Flask

Backend Workflow:

1.User Input → Flask processes the request.

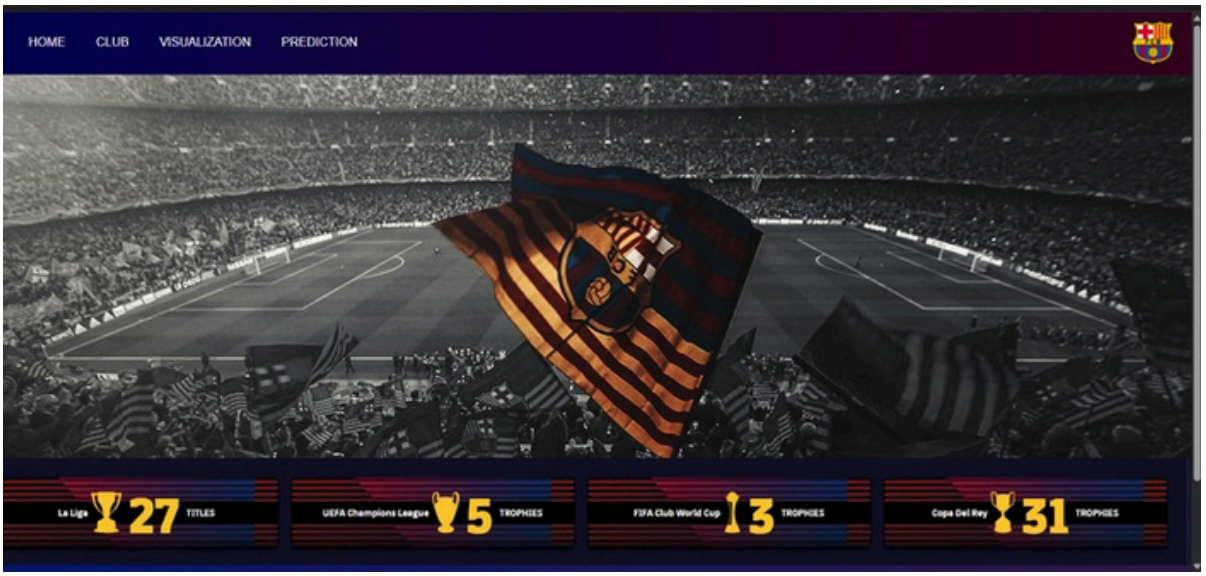
2.Data Encoding → Opponent name converted to numerical ID.

3.Feature Scaling → Input normalized using StandardScaler.

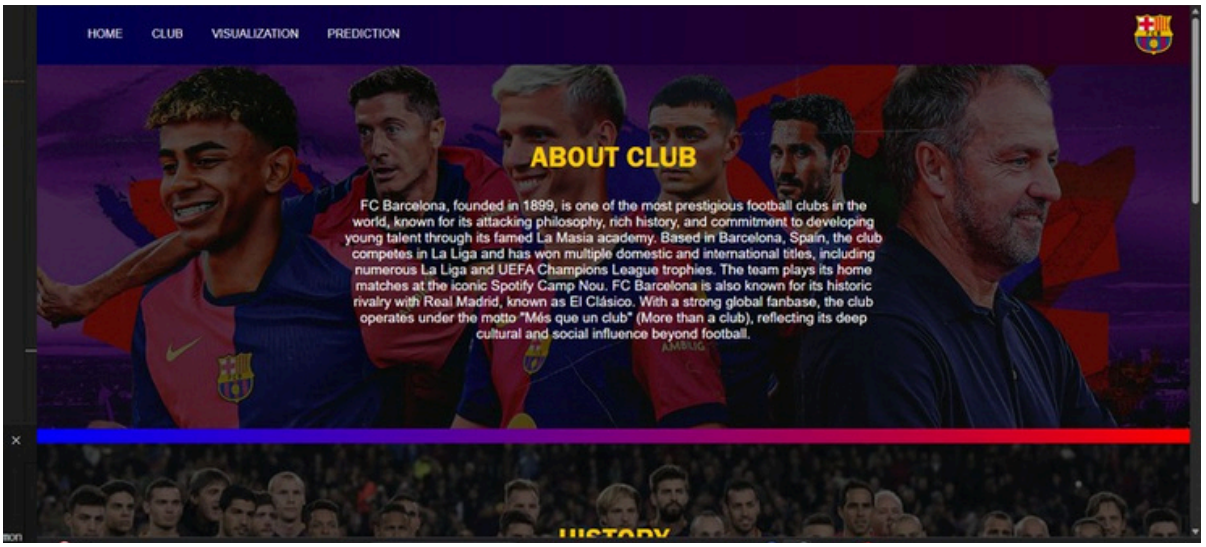
4.Prediction → XGBoost model computes probabilities.

5.Result Display → Probabilities shown to the user.

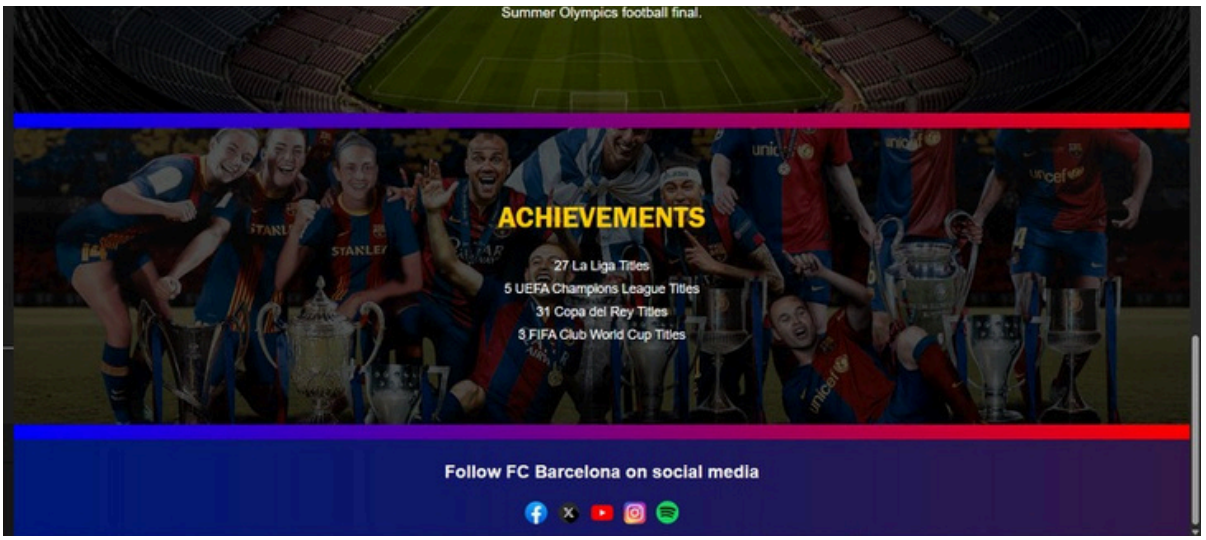
Home Page:



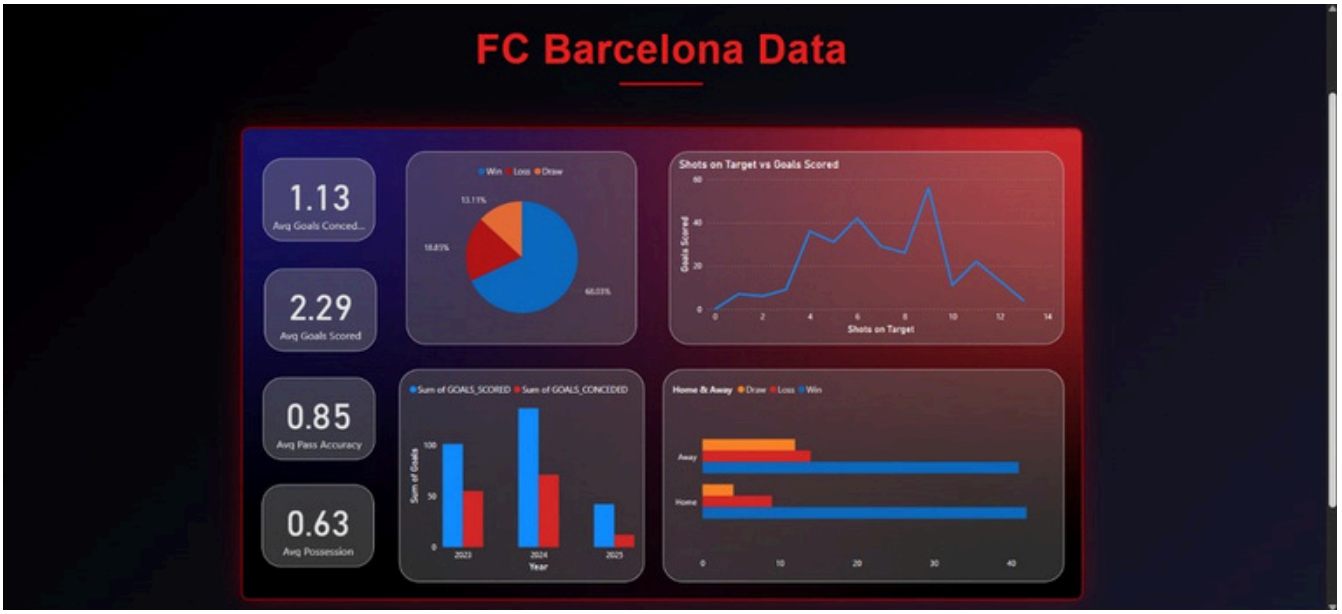

Basic info page of club:



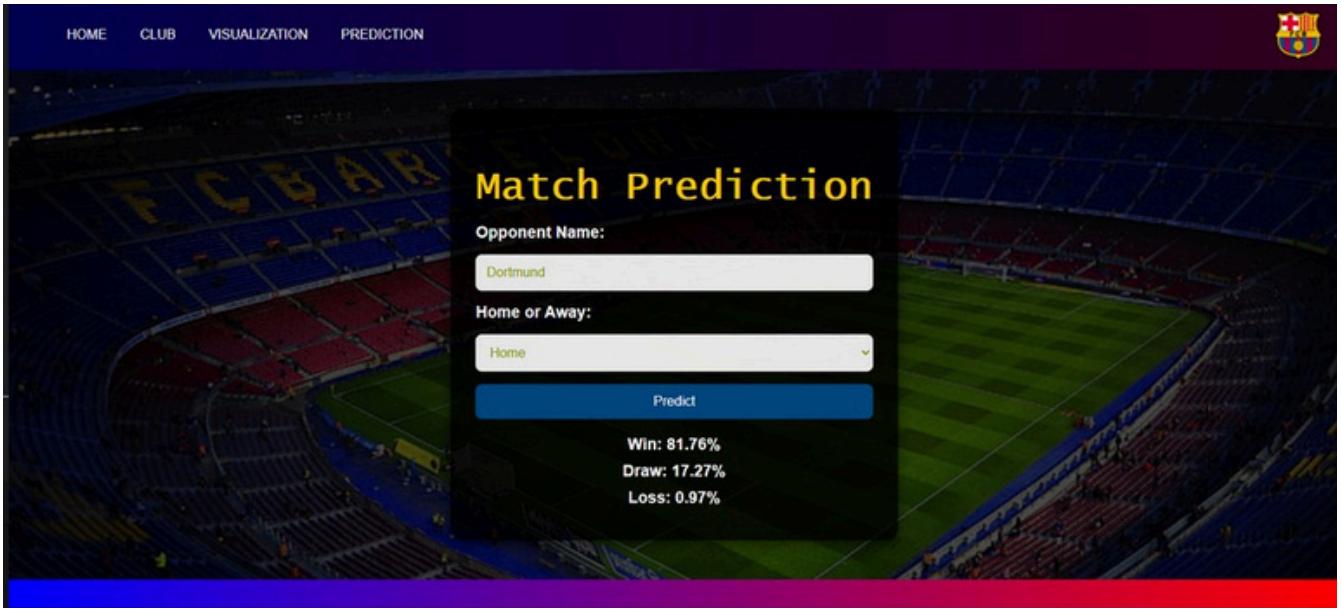

Footer of our web application:



Visulization Page, here we have visulize our dataset using Power Bi by creating a Dashboard:




Prediction page :



# 4. RESULT & OUTCOME

## 4.1 Accuracy :

The XGBoost model achieved an accuracy of 72.22% on the test set, demonstrating its ability to correctly classify match outcomes (Win, Draw, Loss) in approximately 7 out of 10 cases and K-Fold Cross Validation accuracy came out as 73.52%.

```
# Evaluate the model on the test set
y_pred = final_model.predict(X_test_rfe)
accuracy = accuracy_score(y_test, y_pred)
print(f"\nModel Accuracy on Test Set: {accuracy:.2%}")
```

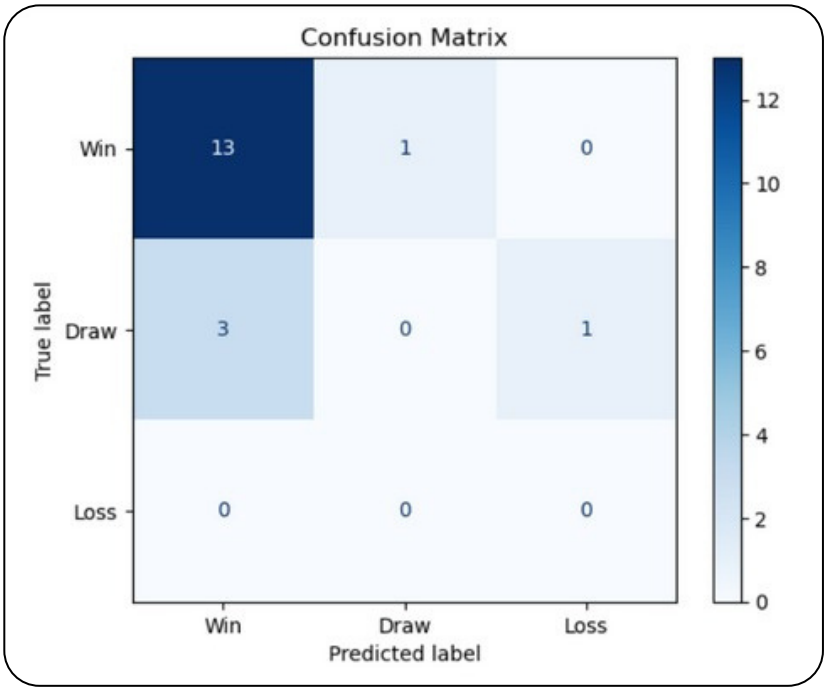
```
Model Accuracy on Test Set: 72.22%
```

```
# K-Fold Cross-Validation
kfold = KFold(n_splits=5, shuffle=True, random_state=42)
cv_scores = cross_val_score(final_model, X_train_rfe, y_train, cv=kfold, scoring='accuracy')
print(f"K-Fold Cross-Validation Accuracy: {cv_scores.mean():.2%} (±{cv_scores.std():.2%})")
```

```
K-Fold Cross-Validation Accuracy: 73.52% (±9.63%)
```

## 4.2 Confusion Matrix :

The purpose of confusion matrix is to show counts of true vs predicted labels to identify misclassifications patterns.



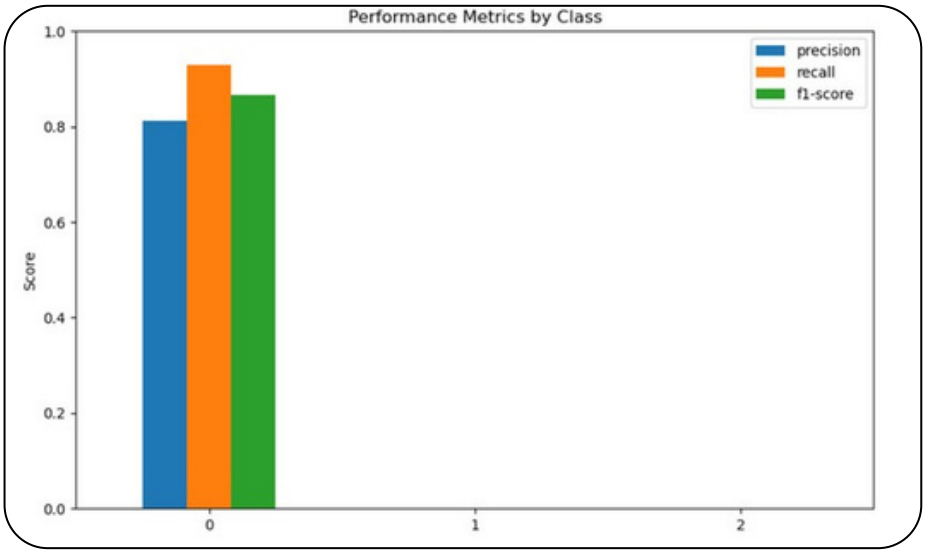
Diagonal cells show correct predictions and Off-diagonal cells show misclassifications (e.g., Draws predicted as Wins).

### 4.3 Performance Metrics by Class :

To highlight precision, recall and F1-score for each class, the following table is the output table after calculating performance metrics.

| Class | Precision | Recall | F1-Score | Supports |
|---|---|---|---|---|
| **Win** | 0.81 | 0.93 | 0.87 | 14 |
| **Draw** | 0.00 | 0.00 | 0.00 | 4 |
| **Loss** | 0.00 | 0.00 | 0.00 | 0 |
| **Macro Avg** | 0.27 | 0.31 | 0.29 | 18 |
| **Weighted Avg** | 0.63 | 0.72 | 0.67 | 18 |

The model performed exceptionally well in predicting "Win" but struggled with "Draw" and "Loss" due to class imbalance and limited samples for these outcomes. Following Graph show Performance Metrics by Class (Precision, Recall and F1-score):

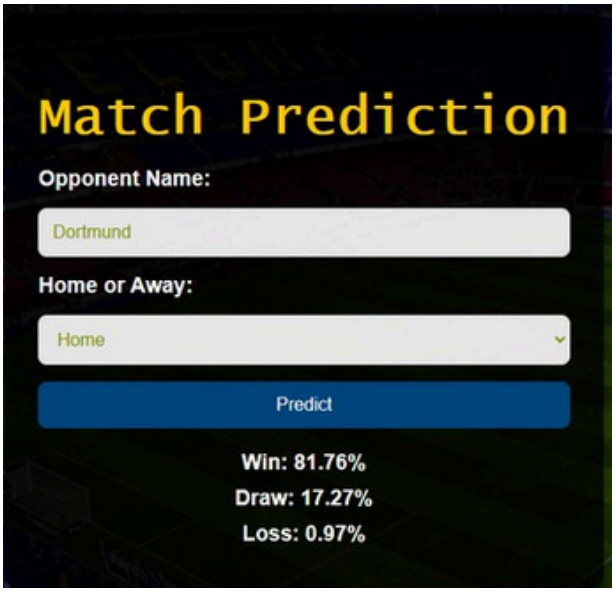


### 4.1 Testing the result on real match

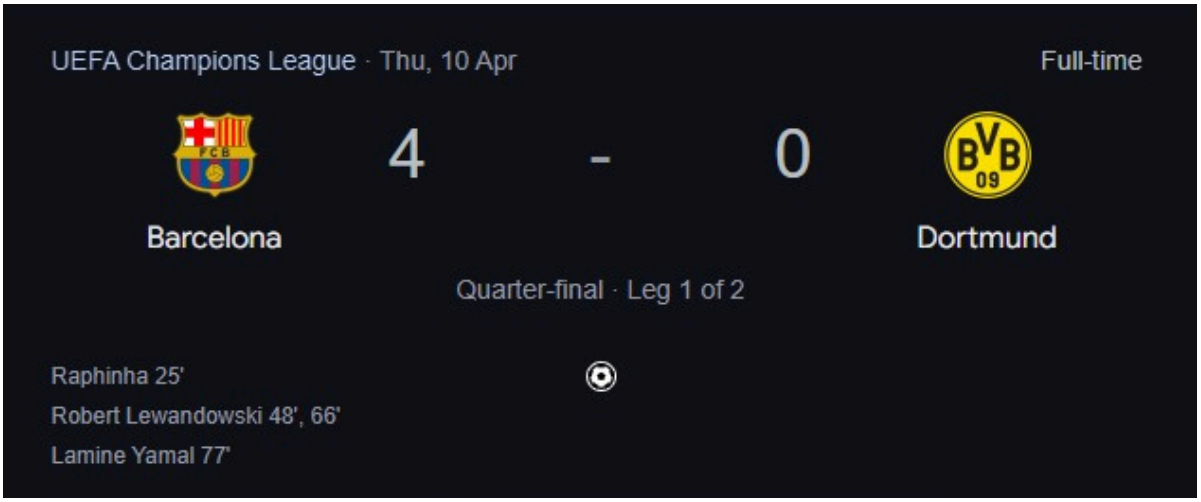
The model was tested interactively to predict the outcome of a upcomming match:
Opponent: Dortmund
Home or Away: Home
Predicted Probabilities:

- Win: 81.76%
- Draw: 17.27%
- Loss: 0.97%

Prediction:




Real time result came out on 10th April 2025 Fc Barcelona won the match by final score of 4 - 0.



# 5. CONCLUSION

Our Project successfully works as the application of machine learning in sports analytics by predicting Fc Barcelona's macth outcomes. The XGBoost model achieves 72% accuracy on our self made dataset also the dashboard provides a good information for our dataset. By combining data science and football analytics, this project highlights the potential of AI in transforming sports predictions.

### 5.1 Future Scope

The future improvements are as follows:

- Expandaing dataset by adding opponent data.
- Integrate live APIs for real time stats.
- Add players features.
- Head to head features.

### 5.2 Limitations of the Project

Current limitations are as follows:

- Limited dataset size.
- Class Imblance bias towards wins.
- Lack of Real-time Player Data.
- Opponent Encoding limitations.
- No Weather/Pitch Conditions.

# 6. REFERENCES

Dateset was manual created from match statistics were gathered from football analytics websites FBref.com of Only team Fc Barcelona, from the 2023/24 and 2024/25 seasons.

Fc Barcelona website has provide information about the club.