DIC Project Phase 3 Report

Name	UBIT No
Manish Bikumalla	50560699
Hariharan Sriram	50560515
Krithiman Manikonda	50560514

Introduction:

In the modern world, data is everywhere. By analyzing this data, one can obtain a multitude of advantages, including the capacity to make well-informed decisions, minimize risk, seize new possibilities, improve competition within their respective areas, spur innovation, and more. It is also applicable in a number of departments. Data analysis is therefore crucial in today's environment.

The information used in the project is an assortment of different football-related data points and attributes, including player names, nationalities, overall ratings, salaries, stats, birthdates, positions played, football teams played for, and so forth. They offer an understanding of the game of football and its participants.

For this phase of the project, we used the XGBoost algorithm to analyze and predict the market value of a proposed player with respect to his in game player statistics. XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework.

Problem Statement:

With this project we aim to eliminate the problems that arise during the decision making process of football player recruitment into the team. This may help the team management make more informed and predictable decisions which may affect the future of the team and the players. We will be using the XGBoost machine learning algorithm to analyze the data and to make the prediction regarding the expected worth

of the individual to the team and their market value based on their physical, mental and playmaking stats.

We have decided to use XGBoost for this phase out of the other machine learning algorithms because we understood that this algorithm works the best in this scenario and is the right fit for our predictive model. That is because it is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. That can be used to give the best prediction with the data that was collected thus far.
With the data we have collected and the XGBoost algorithm, we will try to successfully visualize the information that has been collected and predict the market value of the football player.

Inferences from Phase-1:

The following steps were taken during phase-1:

- The data set is taken from Kaggle and is 3.8 MB. The dataset is in the format of a csv file. It consists of 20860 rows and 51 columns (Link for the dataset).
- The raw dataset that is found in kaggle has a lot of inconsistencies and missing values. Additionally, certain irregularities were added to the dataset to perform more cleaning and processing operations.
- The various steps taken while cleaning the dataset were:
    - Character encoding
    - Finding duplicates
    - Dropping columns with very high missing values
    - Filling ‘release clause value’ column with zero
    - Filling the columns ‘value_euro’ and ‘wage_euro’ with median
    - Formatting ‘Nationality’ column
    - Formatting the ‘full_name’ column
    - Formatting the date column
    - Formatting the body_type column
    - Adding a new variable Body Mass Index
    - Creating a ‘primary_position’ column
    - Performing feature engineering to extract more features and dropping other columns
- Exploratory Data Analysis (EDA)
    - Five number summary
    - Analysis of outliers
    - Analysis of key features
    - Scatterplot wages vs setpieces, attacking, defending, skills
    - Boxplot of wages, market value and primary position
    - Market value and Primary Position

- Analysis of data with respect to the preferred foot of players
- Comparison of Set Piece stats by Preferred Foot
- Correlation Matrix
- Joint plot to see how market values and wages vary with other variables
- Analysis of data with nationality of the players
- Mean Potential by Country
- Analysis of body mass index with age, nationality, primary position
- BMI vs nationality of the players
- BMI vs primary position of the players

With the help of the above steps we were able to conclude with the below points:

- It is observed that as the set piece stats increases, the wages of the players increases.
- From the graph, it can be said that players who play in primary positions such as ST, CM have more wages.
- From the graph, it can be said that players who play in primary positions such as ST, CM,CAM have more Market value.
- It is observed that players who play in LWB as their primary position have the highest BMI with a value over 25 and the value decreases as we go down.
- It is observed that the BMI of the players increase as their age increases and then decrease as they get older. It is observed that the BMI is the highest for players whose age is 42.
- It is observed that the graph is skewed in the center for this as the ages increase, the wages also increase.
- From the correlation graph we can find the strong relationships between the values. We can detect multicollinearity issues among independent variables. The height in cm and BMI has the lowest correlation and release clause euro and value_euro has the highest correlation.
- It can be concluded that when compared to the mean of set piece stats, the players who prefer left foot are more when compared to right foot.

Inferences from Phase-2:

The following steps were taken during the second phase of the project:

- We used various machine learning algorithms to test the compatibility of the dataset so that we get the most accuracy and quick decisions with an algorithm.
- This phase was used to test the machine learning algorithm accuracy of predicting the “Actual value euro and Predicted value Euro” when tested with the data which was acquired from phase 1.

- The machine learning algorithms that we used during this phase are:
    - Linear Regression
    - Support Vector Regressor
    - Decision Tree
    - XGBoost
    - Random Forest
    - Gradient Boosting
    - Multi Layer Perceptron

The following results were observed with respect to the accuracy of the algorithms:

Name of the Algorithm	**R2 for Value_euro**	**R2 for Wage_euro**
Linear Regression	0.59	0.53
Support Vector Regressor	-0.09	-0.075
Decision Tree	0.89	0.67
XGBoost	0.97	0.74
Random Forest	0.96	0.76
Gradient Boosting	0.97	0.79
Multi Layer Perceptron	28916496738267	477854503

From the above we can observe that the XGBoost, Random Forest, and Gradient Boosting have the highest accuracy.

Algorithms Shortlisted for Phase-3:

For this phase we had shortlisted three machine learning algorithms which we felt were suitable for the predictive analysis of the data that has been acquired so far and have shown the highest accuracy with the given dataset. The algorithms that were shortlisted are:

Random-Forest Regressor

Random Forest Regressor is a machine learning algorithm that belongs to the ensemble learning family, specifically to the bagging method. It is used for regression tasks, where the goal is to predict a continuous outcome variable. It builds multiple decision trees during training and outputs the average prediction (in regression tasks) or the mode of predictions (in classification tasks) of the individual trees.

Lasso Regression and Gradient Booster

We use gradient booster to optimize the model. By learning from the mistakes of the previous models and combining them we get a more accurate answer. Gradient descent is used to optimize a differentiable loss function. Every new model is trained to reduce the ensemble's residual errors, or the discrepancy between the values that were predicted and those that actually occurred.

XGBoost

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost belongs to the family of ensemble learning methods, specifically gradient boosting algorithms. It builds a strong predictive model by combining multiple weak learners sequentially.

After careful deliberation and testing with the data we have on the above 3 algorithms, we have concluded that XGBoost would be the most suitable for the current scenario.

Why XGBoost?

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost belongs to the family of ensemble learning methods, specifically gradient boosting algorithms. It builds a strong predictive model by combining multiple weak learners sequentially. It is designed with optimization techniques that make it faster and more efficient compared to traditional gradient boosting implementations. These techniques include parallelization, tree-pruning, and regularization and primarily consists of a sequential amalgamation of decision trees where each subsequent tree corrects the errors made by the previous ones.

To prevent overfitting, XGBoost incorporates regularization techniques which penalizes complex models, encouraging simpler and more generalizable solutions. XGBoost is very flexible and scalable. It can handle a variety of data types, including numerical and categorical features. It also supports different loss functions and evaluation metrics. It can also efficiently handle large datasets.

XGBoost (Extreme Gradient Boosting) algorithm offers several advantages over Gradient Boosting and Random Forest algorithms:

1. Improved Performance: XGBoost is optimized for both speed and performance. It implements parallelized tree boosting, which makes it significantly faster than traditional Gradient Boosting implementations.

2. Regularization: XGBoost incorporates regularization techniques, such as L1 (Lasso) and L2 (Ridge) regularization, which help prevent overfitting and improve generalization performance. Regularization is not as straightforward in traditional Gradient Boosting.

3. Handling Missing Values: XGBoost has built-in capabilities to handle missing values in the data, which means you don't need to pre-process missing values as extensively as you might with other algorithms. This can save time and effort in data preparation.

4. Flexibility: XGBoost supports a variety of objective functions and evaluation metrics, allowing you to customize the algorithm based on your specific problem and data characteristics. It also supports user-defined objective functions.

5. Feature Importance: XGBoost provides feature importance scores, which can help in feature selection and interpretation of the model. While Random Forest also provides feature importance, XGBoost's feature importance tends to be more accurate.

6. Tree Pruning: XGBoost uses a technique called tree pruning, which removes splits beyond which there is no positive gain. This helps in reducing the complexity of the model and avoiding overfitting.

7. Scalability: XGBoost is highly scalable and can handle large datasets efficiently. It supports distributed computing frameworks such as Spark and Hadoop, making it suitable for big data applications.

8. Wide Adoption: XGBoost has gained widespread popularity and is widely used in data science competitions (e.g., Kaggle) and real-world applications. This popularity has led to extensive community support, documentation, and resources for users.

While Gradient Boosting and Random Forest algorithms have their own strengths and are suitable for certain tasks, XGBoost often outperforms them in terms of speed, performance, and flexibility, making it a popular choice in many machine learning applications.

Define Parametres:

In this regression model, we use the following attributes as independent variables:

features = ['age', 'overall_rating', 'potential', 'international_reputation(1-5)', 'weak_foot(1-5)', 'skill_moves(1-5)', 'ind_performance', 'mean_skill_ind']

The values to be predicted are called dependent variables. They are:

'value_euro'

Evaluation:

On calculating the R2 we can see the values for

```
Results for value_euro:
 XGBoost Regressor: R² : 0.9681551761296813
```

Prediction:

We use this algorithm to predict the dependent variables ('value_euro') which is the market value of the player. Through this algorithm we calculate R2(a metric that determines that our machine learning algorithm is behaving as expected) and it predicts the goodness of fit for the model. The value of R2_score lies in between 0 to 1. The Higher the R2, the more accurate are the dependent variables in the algorithm.

Deployment of the Code for the UI:

We used the streamlit library of python to create a webapp and further used libraries like matplotlib, seaborn to visualize the data.

The following is the code that is used to run the UI.

It has 2 code files.
1) app.py
2) Visualization.py

1) app.py:

```
import streamlit as st

import pandas as pd

from sklearn.preprocessing import StandardScaler

from sklearn.model_selection import train_test_split
```

```
from sklearn.metrics import r2_score

from xgboost import XGBRegressor

import matplotlib.pyplot as plt

import seaborn as sns

import datetime

from Visualizations import draw_scatter_plots, draw_box_plots, draw_joint_plots, draw_bar_plots

import numpy as np


# Function to upload CSV file

def upload_file():

    st.title("Please Upload the csv file to train the model")


    uploaded_file = st.file_uploader("Choose a file")

    if uploaded_file is not None:

        df = pd.read_csv(uploaded_file, encoding='utf-8')

        st.write(df)

        return df

    return None


# Function to collect player attributes

def collect_player_attributes():
```

```
    st.title('Enter Player Details')

    input_values = {}

    input_values['name'] = st.text_input("Enter Name", value='')

    input_values['full_name'] = st.text_input("Enter Full Name", value='')

      input_values['age'] = st.number_input("Enter Age", min_value=0, max_value=150, value=0)

      input_values['height_cm'] = st.number_input("Enter Height (in cm)", min_value=0, value=0)

     input_values['weight_kgs'] = st.number_input("Enter Weight (in kg)", min_value=0.0, value=0.0, step=0.1)

          input_values['positions'] = st.text_input("Enter Positions (comma-separated)", value='')

    input_values['nationality'] = st.text_input("Enter Nationality", value='')

     input_values['overall_rating'] = st.number_input("Enter Overall Rating", min_value=0, max_value=100, value=0)

           input_values['potential'] = st.number_input("Enter Potential", min_value=0, max_value=100, value=0)

    input_values['wages_euro'] = st.number_input("Enter Wages", min_value=0, value=0)

     input_values['preferred_foot'] = st.selectbox("Select Preferred Foot", options=['Left', 'Right'])

         input_values['international_reputation(1-5)'] = st.selectbox("Select International Reputation", options=[1, 2, 3, 4, 5])

    input_values['weak_foot(1-5)'] = st.selectbox("Select Weak Foot", options=[1, 2, 3, 4, 5])

     input_values['skill_moves(1-5)'] = st.selectbox("Select Skill Moves", options=[1, 2, 3, 4, 5])
```

```
    input_values['body_type'] = st.selectbox("Select Body Type", options=['Normal', 'Lean', 'Stocky'])

    input_values['release_clause_euro'] = st.number_input("Enter Release Clause", min_value=0, value=0)



  # Add input fields for player attributes

    attributes = ['crossing', 'finishing', 'heading_accuracy', 'short_passing', 'volleys', 'dribbling', 'curve',

                  'freekick_accuracy', 'long_passing', 'ball_control', 'acceleration', 'sprint_speed', 'agility',

                 'reactions', 'balance', 'shot_power', 'jumping', 'stamina', 'strength', 'long_shots', 'aggression',

                'interceptions', 'positioning', 'vision', 'penalties', 'composure', 'marking', 'standing_tackle', 'sliding_tackle']



  for attribute in attributes:

        input_values[attribute] = st.number_input(f"Enter {attribute}", min_value=0, max_value=100, value=0)



  df = pd.DataFrame(input_values, index=[0])

  return df



# Function for feature engineering

def feature_engineering(df):
```

```
    # Create primary position

    df['primary_position'] = df['positions'].fillna('').str.split(',').str[0]

    df.insert(8,'primary_position',df.pop('primary_position'))



    # Setpiece

    set_piece_columns = ['crossing', 'long_passing', 'curve', 'penalties']

    df['setpiece_stats'] =df[set_piece_columns].mean(axis = 1).round(2)



    # Attacking

        attacking_columns = ['crossing', 'finishing', 'heading_accuracy', 'short_passing',
'volleys']

    df['attacking_stats'] = df[attacking_columns].mean(axis = 1).round(2)



    # Skill

    skill_columns = ['dribbling', 'curve', 'freekick_accuracy', 'long_passing', 'ball_control']

    df['skill_stats'] = df[skill_columns].mean(axis = 1).round(2)



    # Movement

    movement_columns = ['acceleration','sprint_speed', 'agility', 'reactions', 'balance']

    df['movement_stats'] = df[movement_columns].mean(axis = 1).round(2)



    # Power
```

```
    power_columns = ['shot_power', 'jumping', 'stamina', 'strength', 'long_shots']

    df['power_stats'] = df[power_columns].mean(axis = 1).round(2)



    # Mentality

      mentality_columns = ['aggression', 'interceptions', 'positioning', 'vision', 'penalties',
'composure']

    df['mentality_stats'] = df[mentality_columns].mean(axis = 1).round(2)



    # Defending

    defending_columns = ['marking', 'standing_tackle', 'sliding_tackle']

    df['defending_stats'] = df[defending_columns].mean(axis = 1).round(2)



    # Further feature engineering

    df['ind_performance']=((df['overall_rating']+df['potential'])/2).round(2)


skill_ind=['attacking_stats','skill_stats','movement_stats','power_stats','mentality_stats','d
efending_stats']

    df['mean_skill_ind']= df[skill_ind].mean(axis=1).round(2)

    return df



# Function to train the model

def train_model(X_train, y_train):

    model = XGBRegressor()
```

```
    model.fit(X_train, y_train)

    return model



# Function to test the model

def test_model(model, X_test, y_test):

    y_pred = model.predict(X
```

Explanation and Functioning of the code:

This Python script is a Streamlit web application for predicting football player market values based on various attributes. Here's a brief explanation of its functionality:

1. Upload File: Users can upload a CSV file containing football player data to train the prediction model.

2. Collect Player Attributes: Users can input attributes of a football player, such as age, skills, and reputation.

3. Feature Engineering: The script performs feature engineering on the uploaded data and user inputs to create new features relevant for model training.

The following are the features that were engineered to create new features for training.

- Set Piece Statistics: Average attributes related to set pieces.
- Attacking Statistics: Average attributes for attacking skills.
- Skill Statistics: Average attributes for dribbling and ball control.
- Movement Statistics: Average attributes for player movement.
- Mentality Statistics: Average attributes for player mentality.
- Defending Statistics: Average attributes for defensive skills.
- Individual Performance: Combine overall rating and potential attributes.
- Mean Skill Index: Average all skill-related statistics.
- Primary Position: Extract the main playing position of players.
- Power Statistics: Average attributes for player strength and shot power.

4. Visualization: Users can select visualization types (graphs) to explore relationships between features in the dataset.

Graph types:
Scatter plots
Box plots
Joint plots
Bar graph plots

5. Model Training and Testing: The script splits the data into training and testing sets, trains an XGBoost regression model, and evaluates its performance using the R2 score.

6. Prediction: Users can input player attributes, and the trained model predicts the player's market value based on those attributes.

7. Button Trigger: Users can click a button to trigger the prediction based on their input.

Overall, this application provides a user-friendly interface for exploring football player data, visualizing relationships between attributes, training a prediction model, and making predictions based on user input.

2) Visualization.py

Code:

```
import matplotlib.pyplot as plt

import seaborn as sns

import pandas as pd

import numpy as np

import streamlit as st



st.set_option('deprecation.showPyplotGlobalUse', False)



def draw_scatter_plots(data, x_feature):
```

```
    st.subheader(f'Scatter Plot: Market Value vs {x_feature}')
    fig, ax = plt.subplots(figsize=(6, 6))
    ax.scatter(data[x_feature], data['value_euro'])
    ax.set_xlabel(x_feature)
    ax.set_ylabel('value_euro')
    ax.set_title(f'{x_feature} vs Market Value')
    st.pyplot(fig)


def draw_box_plots(data, x_feature):
    st.subheader(f'Box Plot: Market Value vs {x_feature}')
    plt.figure(figsize=(8, 6))
    sns.boxplot(x=x_feature, y='value_euro', data=data)
    plt.xlabel(x_feature)
    plt.ylabel('value_euro')
    plt.title(f'{x_feature} vs Market Value')
    st.pyplot()


def draw_joint_plots(data, x_feature):
    st.subheader(f"Joint Plot: {x_feature.capitalize()} vs Value Euro")
    plt.figure(figsize=(8, 6))
        sns.jointplot(x=x_feature, y="value_euro", data=data, kind="reg", xlim=(0, data[x_feature].max()), ylim=(0, data['value_euro'].max()))
```

```
    st.pyplot()


def draw_bar_plots(data, feature):
    st.subheader(f'Bar Plot: Mean {feature.capitalize()} by Country (Top 20)')
    mean_feature_by_country = data.groupby('nationality')[feature].mean()
    top_countries = mean_feature_by_country.nlargest(20)


    color = (0.1, 0.3, 0.3, 0.5)


    plt.figure(figsize=(12, 8))
    plt.bar(top_countries.index, top_countries.values, color=color)
    plt.xlabel('Country')
    plt.ylabel(f'Mean {feature.capitalize()}')
    plt.title(f'Mean {feature.capitalize()} by Country (Top 20)')
    plt.xticks(rotation=45, ha='right')
    plt.grid(axis='y')
    st.pyplot()
```

Explanation and Functional Description of the Code:

This code contains all the methods that are required to plot the different kinds of graphs. The graphs are as follows:

Drawing Scatter Plots:
The function creates a scatter plot to visualize the relationship between a specified feature (x_feature) and the target variable (value_euro).

Drawing Box Plots:
The function is used to generate a box plot to illustrate the distribution of value_euro across different categories of a specified feature (x_feature).

Drawing Joint Plots:
It is used to create a joint plot, showcasing the relationship between a specified feature (x_feature) and value_euro.

Drawing Bar Plots:
The function generates a bar graph plot to visualize the mean of a specified feature (feature) by country (top 20 countries).

This code is called in app.py using:

from Visualizations import draw_scatter_plots, draw_box_plots, draw_joint_plots, draw_bar_plots

draw_scatter_plots, draw_box_plots, draw_joint_plots, draw_bar_plots are the method names declared in Visualization.py to plot respective graphs.

Instructions to Run WebApp:

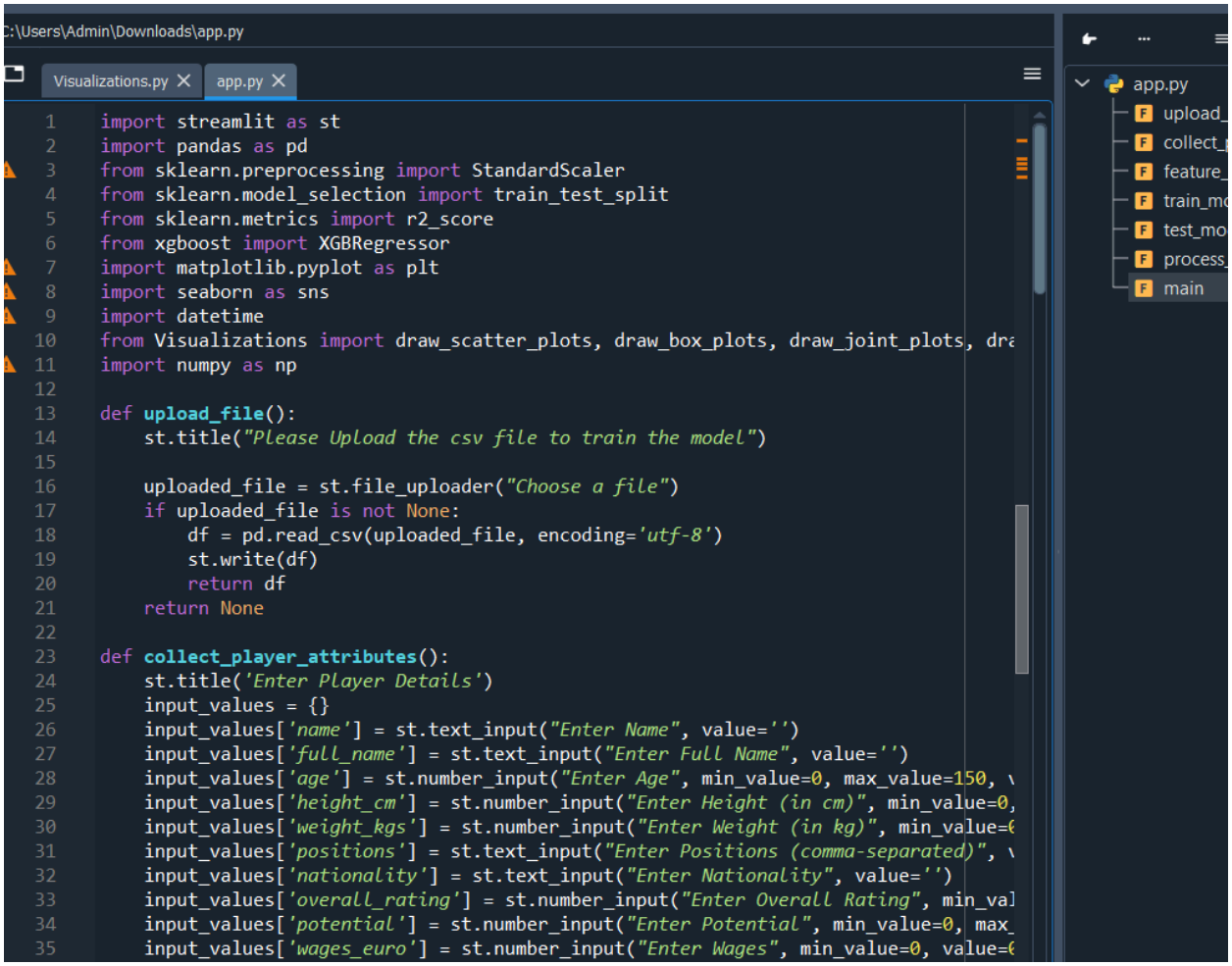
Instructions to use the web app

- Open the terminal window
- Run **pip install streamlit** to install streamlit
- Use the code streamlit run app.py
- Upon running this code, the web app is opened in the default browser
- Using the upload csv button, user can upload a clean csv file which contains the following fields and also should adhere to the data types
  ['name', 'full_name', 'birth_date', 'age', 'height_cm', 'weight_kgs', 'positions', 'nationality', 'overall_rating', 'potential', 'value_euro', 'wage_euro', 'preferred_foot', 'international_reputation(1-5)', 'weak_foot(1-5)', 'skill_moves(1-5)', 'body_type', 'release_clause_euro', 'crossing', 'finishing', 'heading_accuracy', 'short_passing', 'volleys', 'dribbling', 'curve', 'freekick_accuracy', 'long_passing', 'ball_control', 'acceleration', 'sprint_speed', 'agility', 'reactions', 'balance', 'shot_power', 'jumping', 'stamina', 'strength', 'long_shots', 'aggression', 'interceptions', 'positioning', 'vision', 'penalties', 'composure', 'marking', 'standing_tackle',

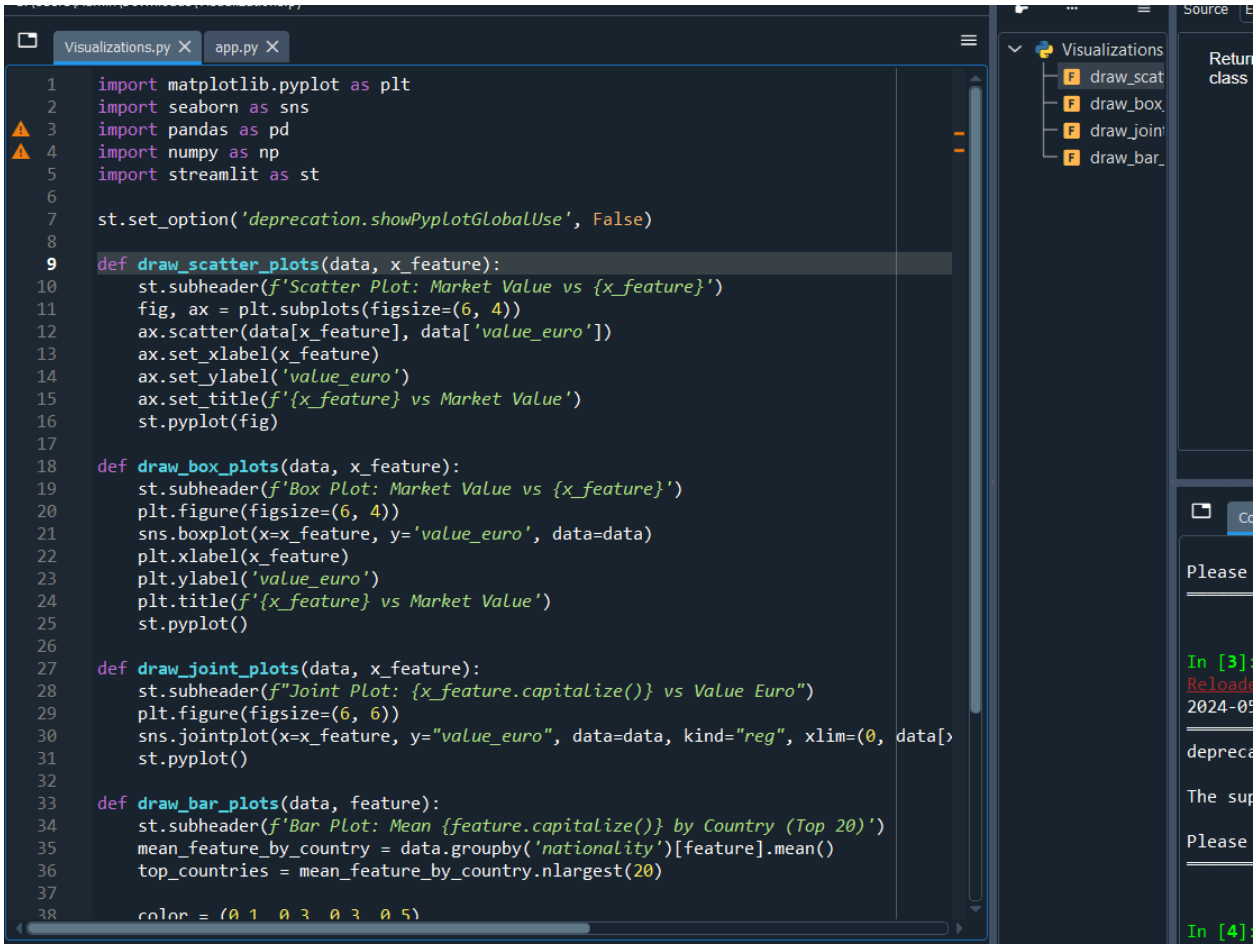
'sliding_tackle']

- The code is written in such a way that it takes the input dataset, performs feature engineering, splits the dataset, trains and tests the dataset after fitting the model.
- It also provides certain visualizations.
- The user can input his own data fields and predict based on the previously trained model.

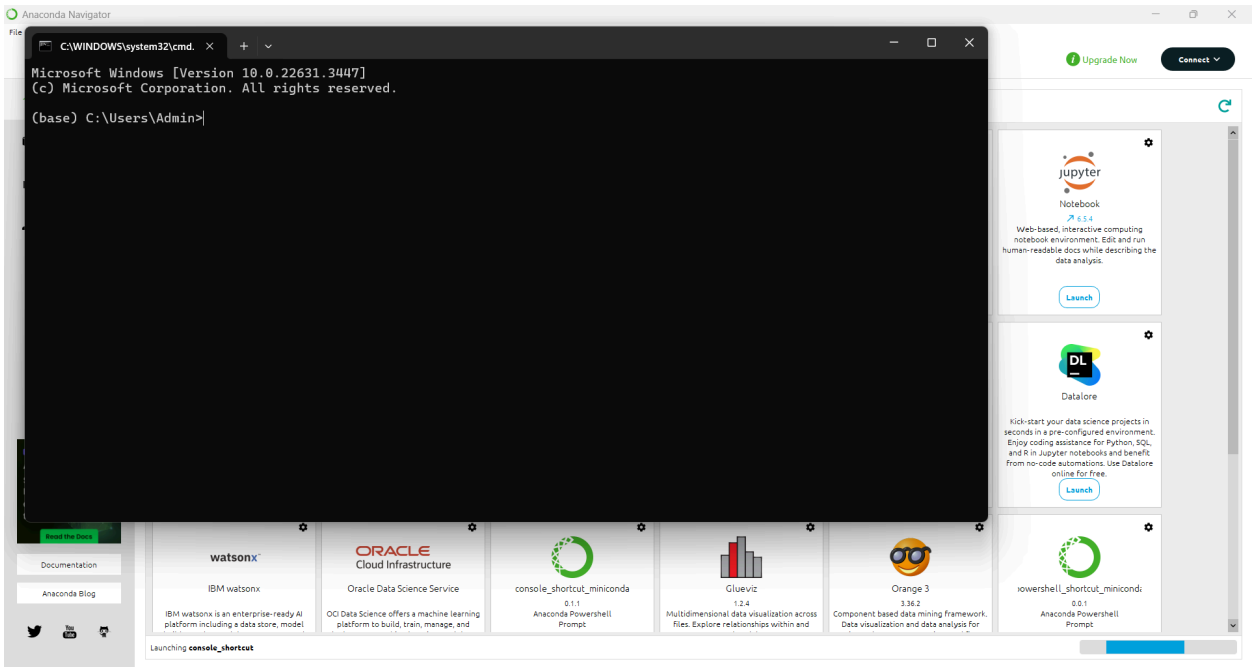
Complete Flow of the web page and UI:



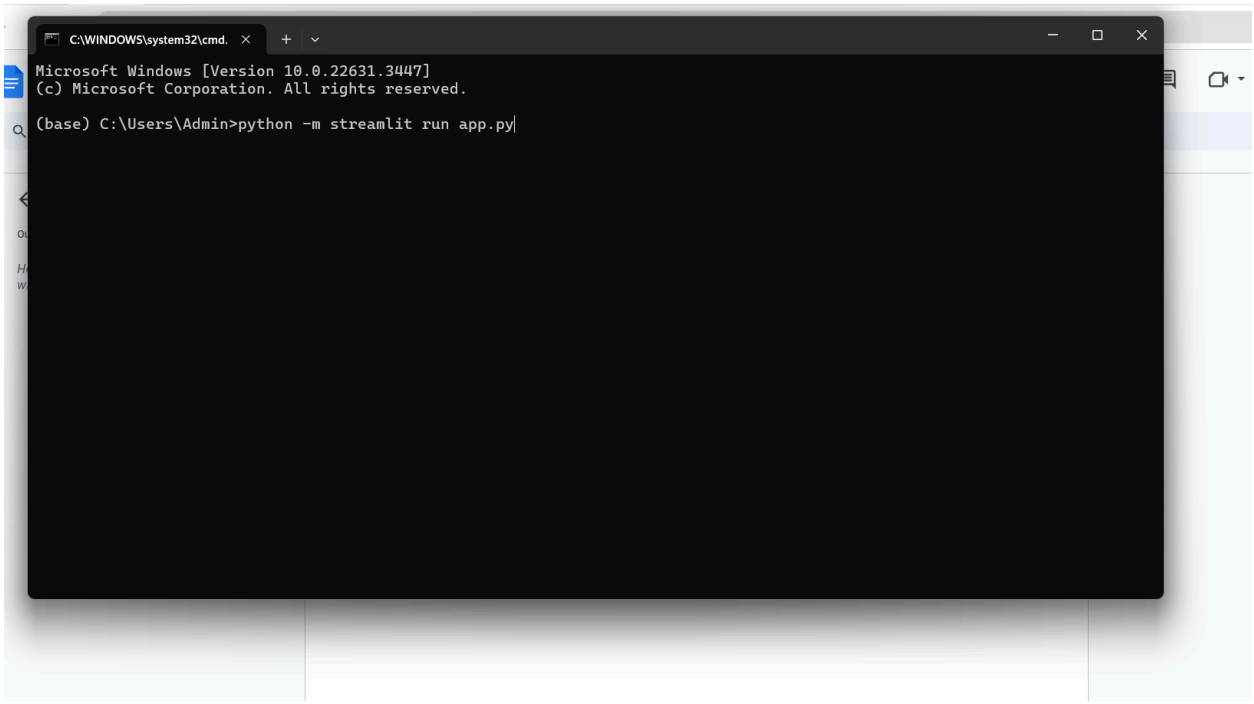

```python
import streamlit as st
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
from xgboost import XGBRegressor
import matplotlib.pyplot as plt
import seaborn as sns
import datetime
from Visualizations import draw_scatter_plots, draw_box_plots, draw_joint_plots, dra
import numpy as np

def upload_file():
    st.title("Please Upload the csv file to train the model")

    uploaded_file = st.file_uploader("Choose a file")
    if uploaded_file is not None:
        df = pd.read_csv(uploaded_file, encoding='utf-8')
        st.write(df)
        return df
    return None

def collect_player_attributes():
    st.title('Enter Player Details')
    input_values = {}
    input_values['name'] = st.text_input("Enter Name", value='')
    input_values['full_name'] = st.text_input("Enter Full Name", value='')
    input_values['age'] = st.number_input("Enter Age", min_value=0, max_value=150, \
    input_values['height_cm'] = st.number_input("Enter Height (in cm)", min_value=0,
    input_values['weight_kgs'] = st.number_input("Enter Weight (in kg)", min_value=(
    input_values['positions'] = st.text_input("Enter Positions (comma-separated)", \
    input_values['nationality'] = st.text_input("Enter Nationality", value='')
    input_values['overall_rating'] = st.number_input("Enter Overall Rating", min_va]
    input_values['potential'] = st.number_input("Enter Potential", min_value=0, max_
    input_values['wages_euro'] = st.number_input("Enter Wages", min_value=0, value=(
```



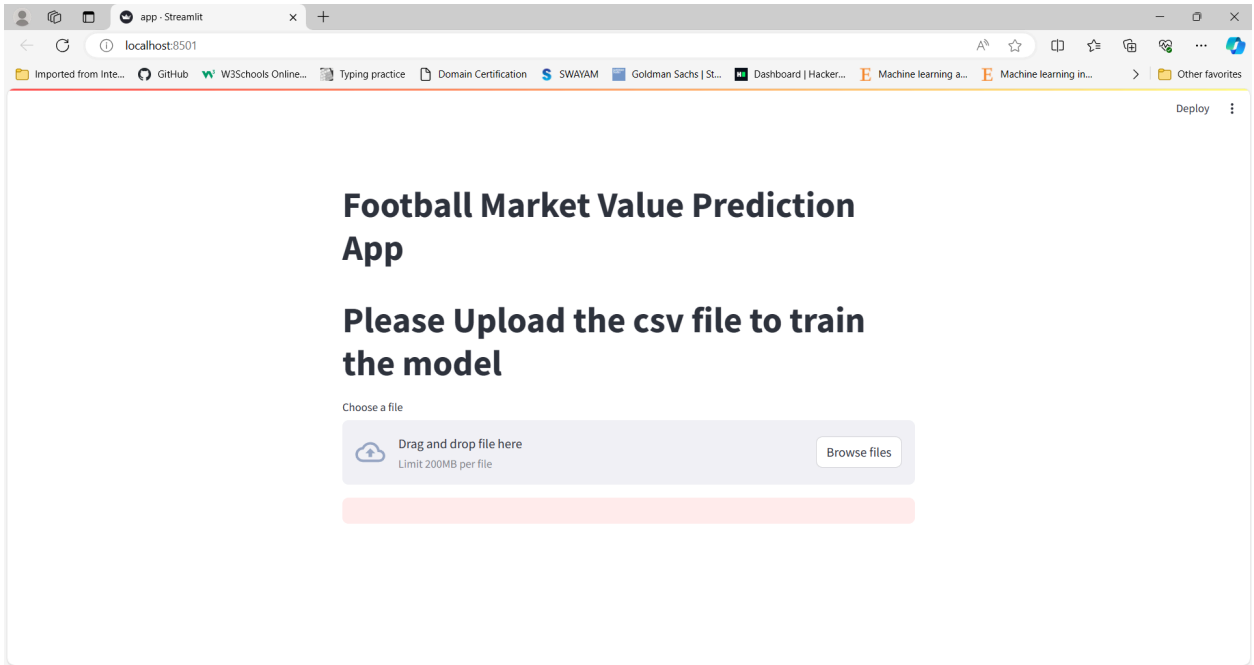
Running the code in cmd



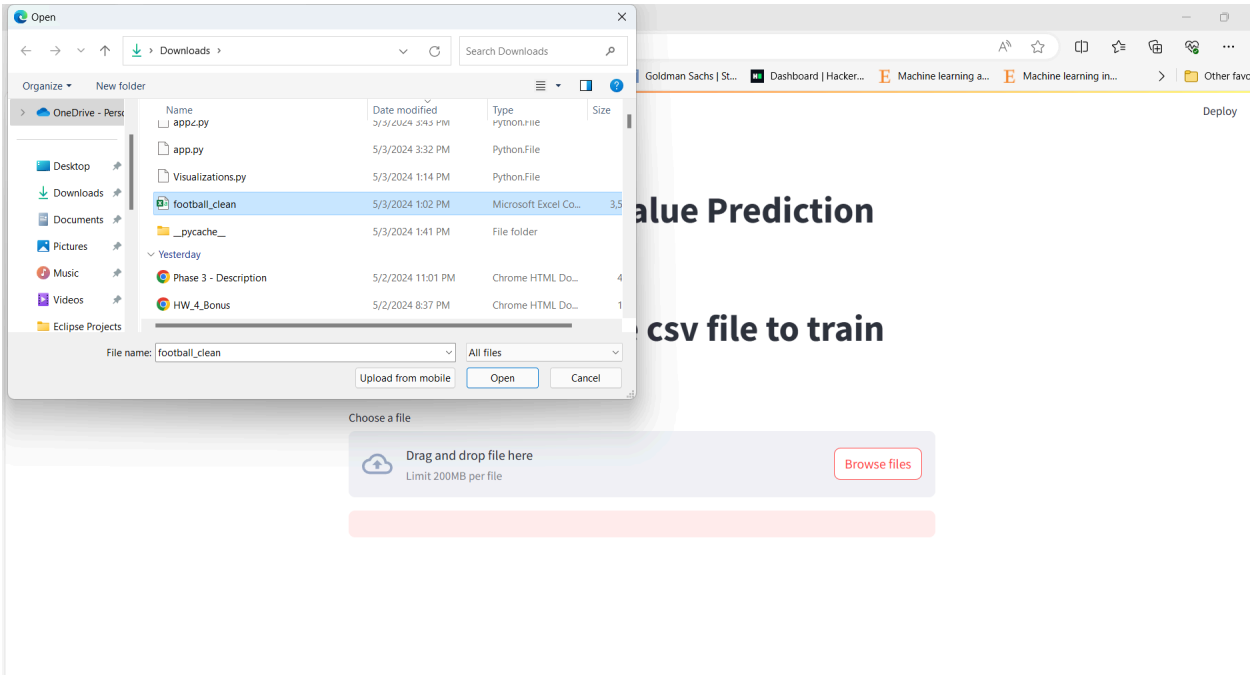
Command ran in prompt to activate streamlit
python -m streamlit run app.py



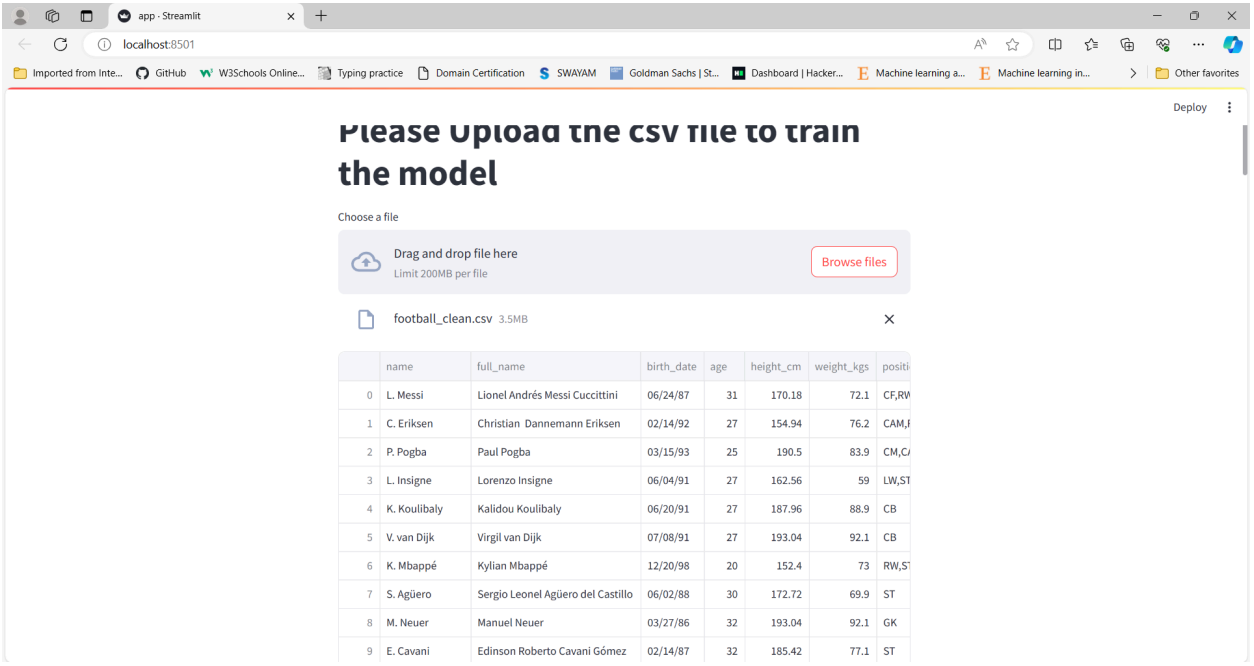
Opening the webpage



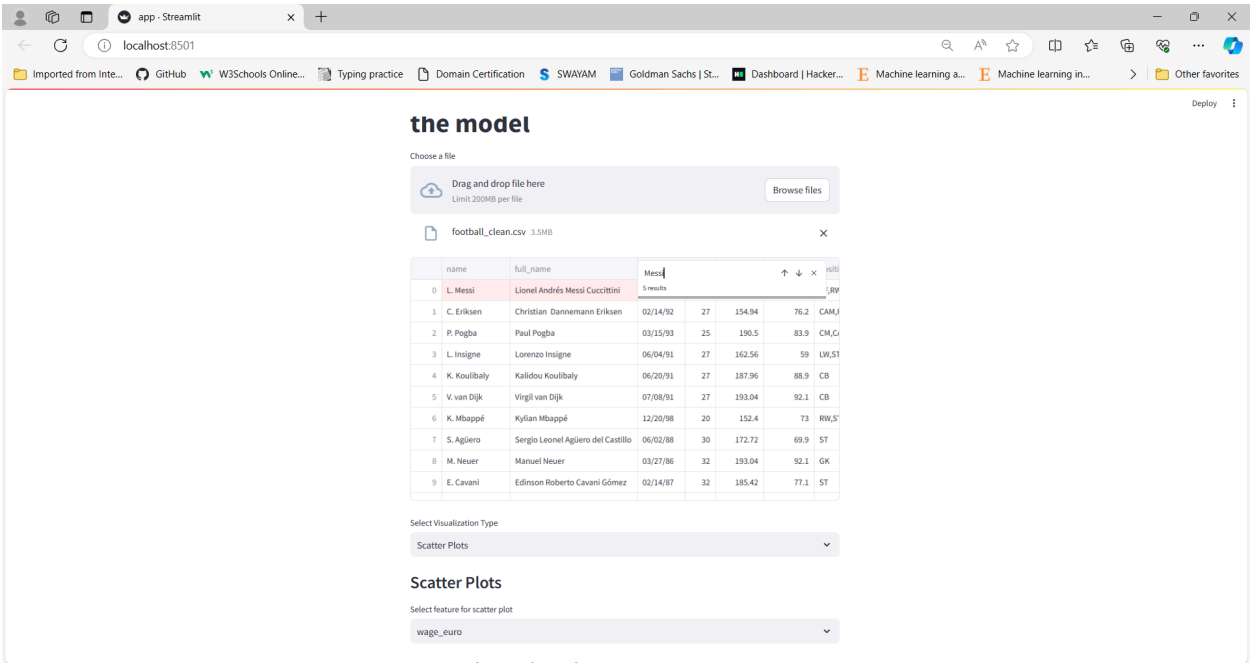
Uploading the csv file

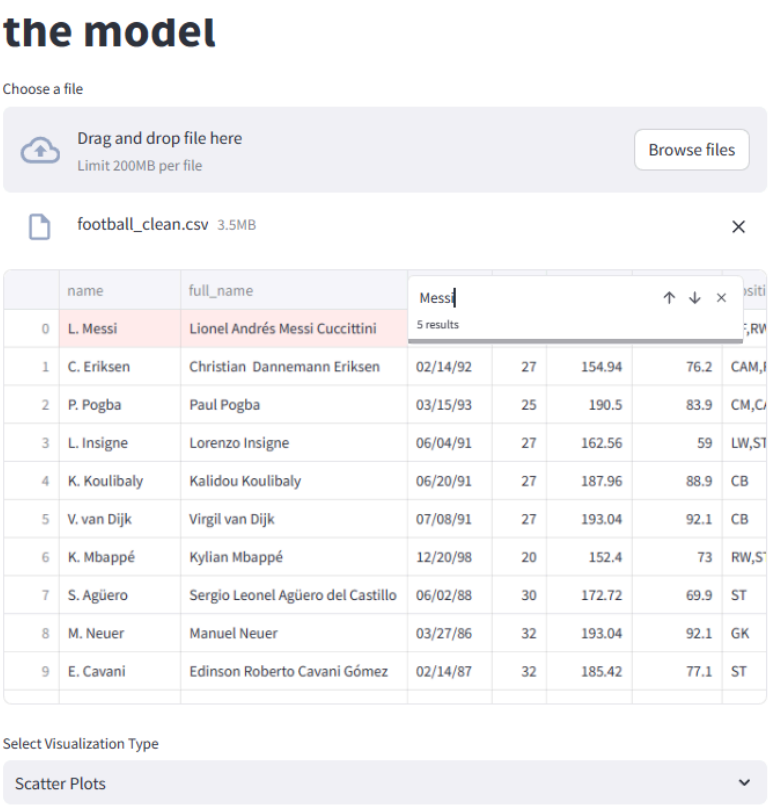


Once the csv file is uploaded, the viewer will be able to see the entire csv file.



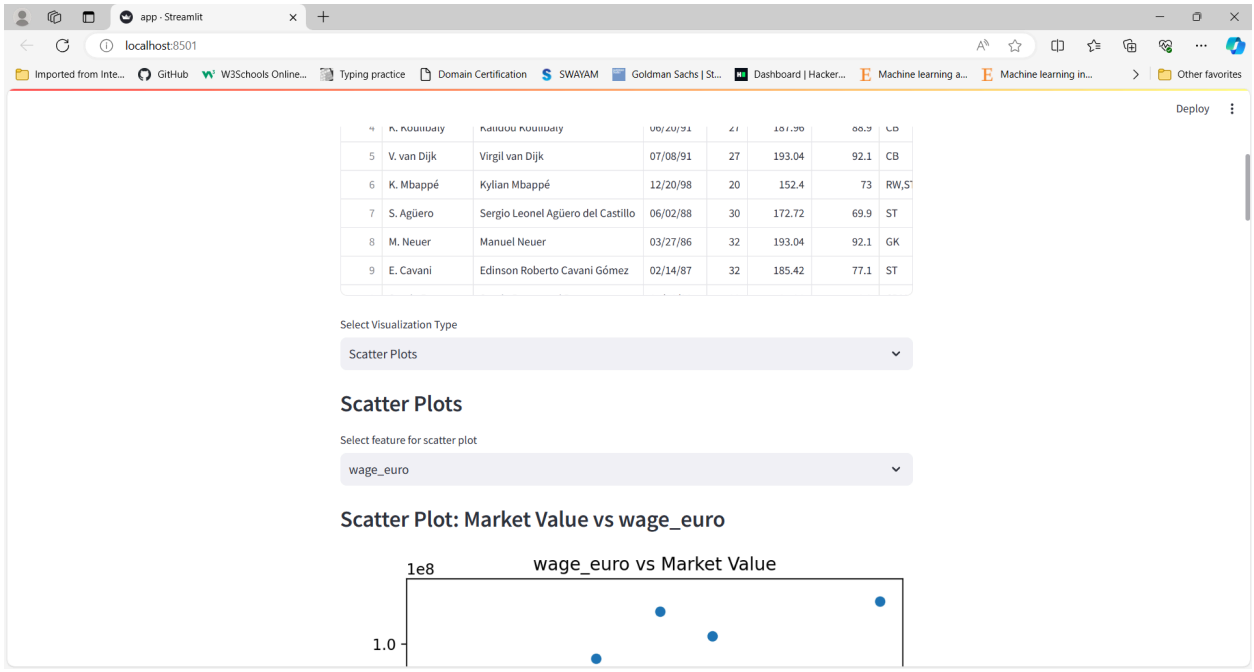
We get to search the list and find the element required.



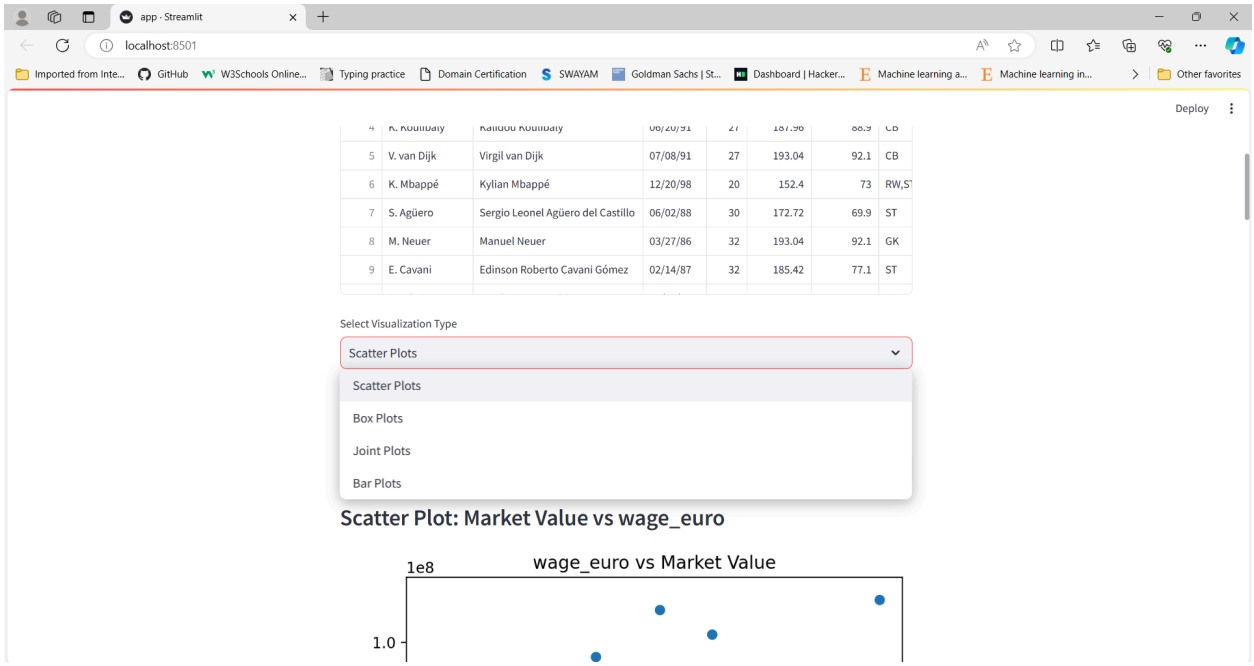


	name	full_name					
0	L. Messi	Lionel Andrés Messi Cuccittini					
1	C. Eriksen	Christian Dannemann Eriksen	02/14/92	27	154.94	76.2	CAM,I
2	P. Pogba	Paul Pogba	03/15/93	25	190.5	83.9	CM,CA
3	L. Insigne	Lorenzo Insigne	06/04/91	27	162.56	59	LW,ST
4	K. Koulibaly	Kalidou Koulibaly	06/20/91	27	187.96	88.9	CB
5	V. van Dijk	Virgil van Dijk	07/08/91	27	193.04	92.1	CB
6	K. Mbappé	Kylian Mbappé	12/20/98	20	152.4	73	RW,ST
7	S. Agüero	Sergio Leonel Agüero del Castillo	06/02/88	30	172.72	69.9	ST
8	M. Neuer	Manuel Neuer	03/27/86	32	193.04	92.1	GK
9	E. Cavani	Edinson Roberto Cavani Gómez	02/14/87	32	185.42	77.1	ST

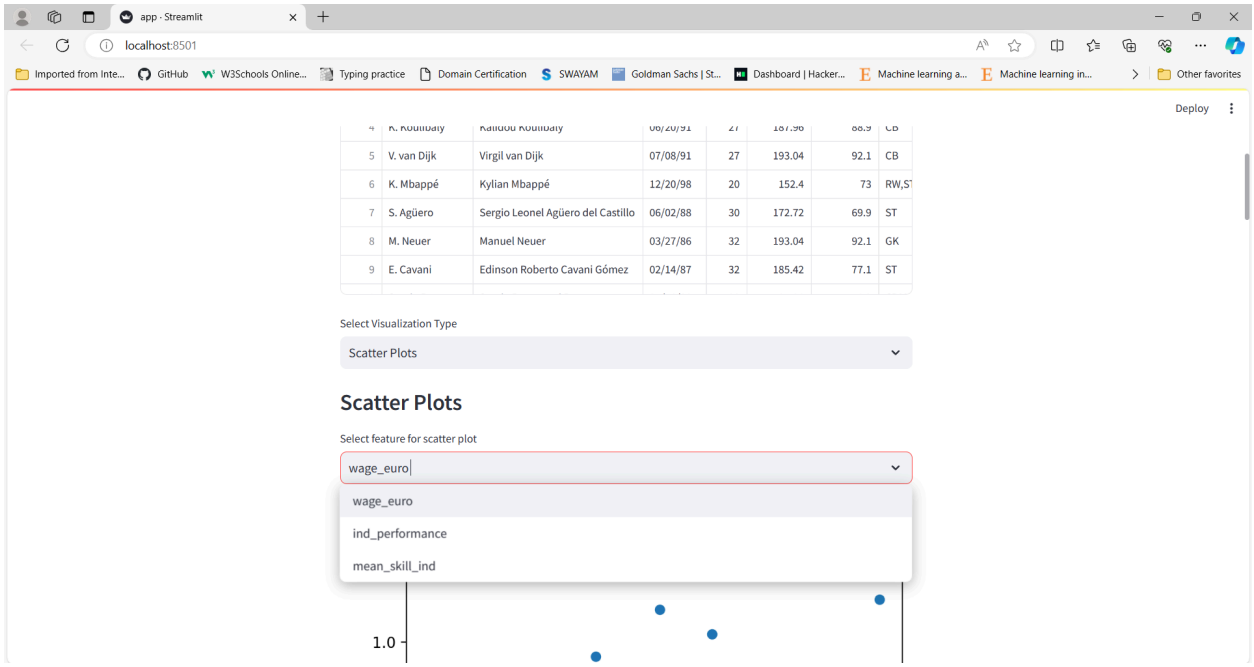
There are 2 drop down boxes
1) Visualization to select which kind of plots
2) The feature to which we need to compare with

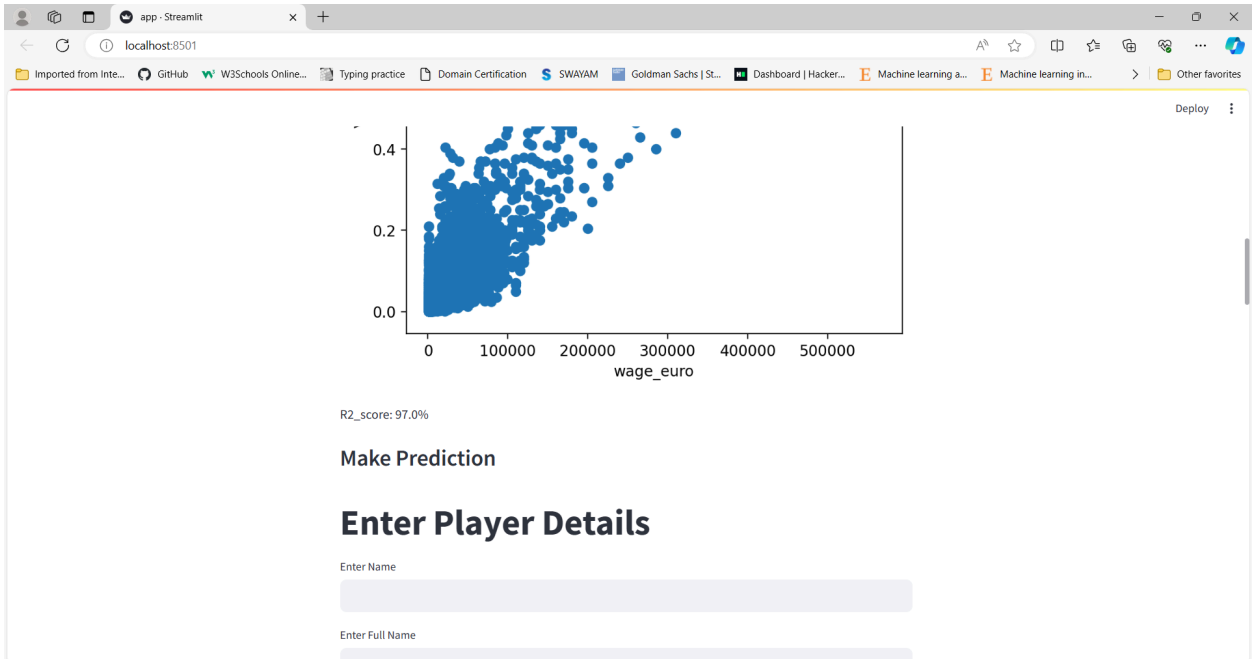


When the drop down is selected,we get to see the different types of graphs we can plot.



For every graph, there are selected features we're going to use to compare.
On selecting the feature, we get to see the graph being plotted.





To test the accuracy of the model,we are finding the R2_Score value.

R2_Score value:

wage_euro

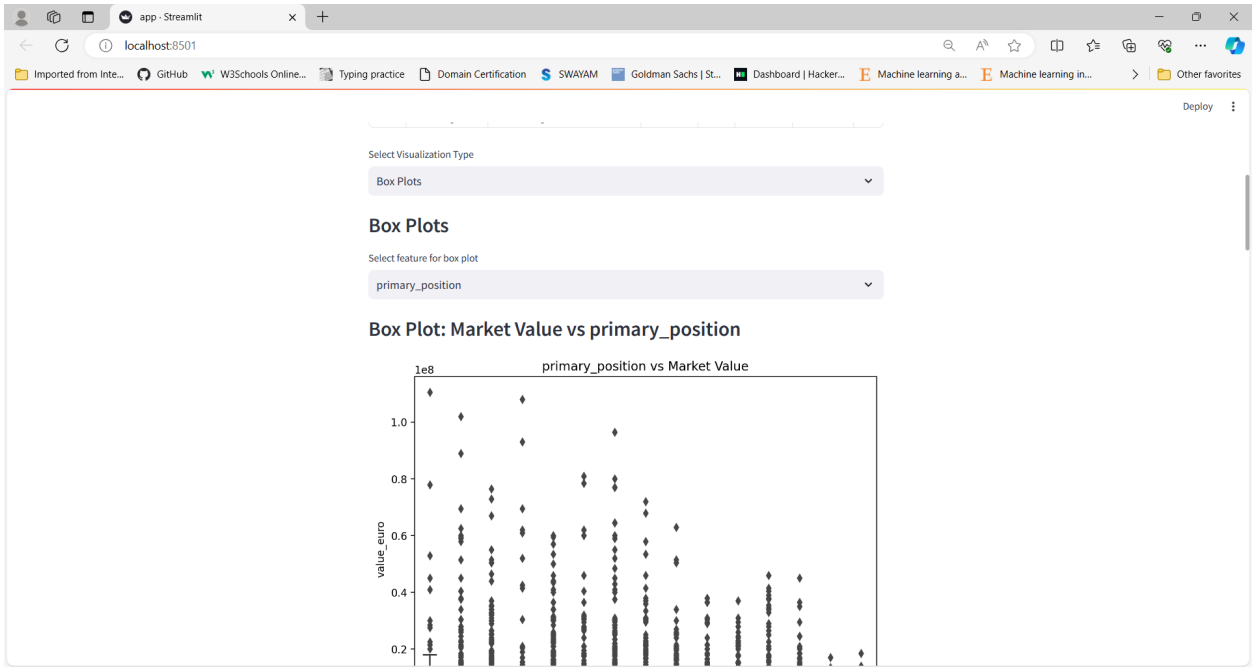
R2_score: 97.0%

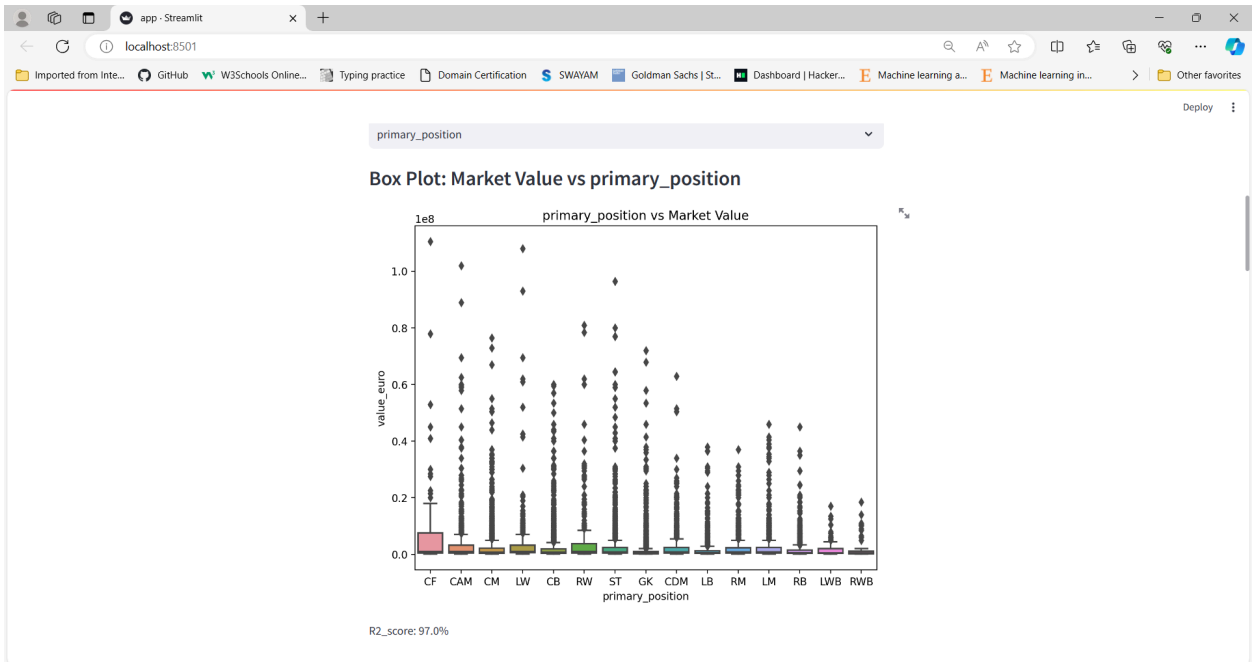
Make Prediction

Enter Player Details

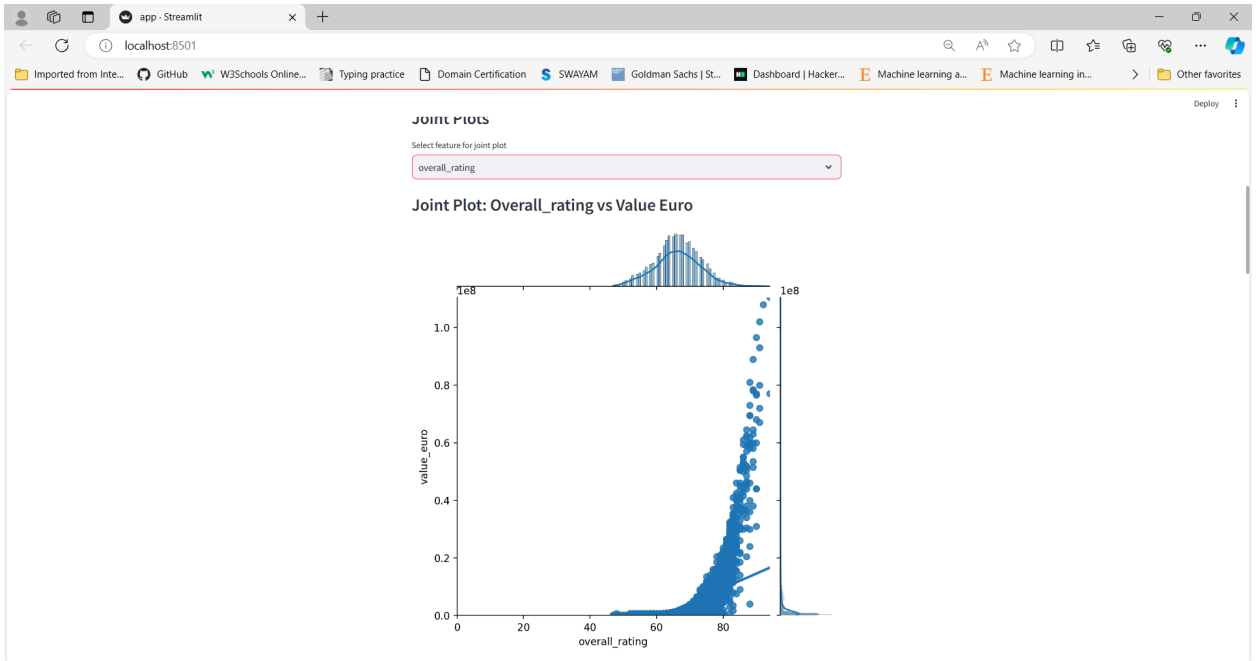
A visual representation of all the graphs being plotted.

Box Plots:

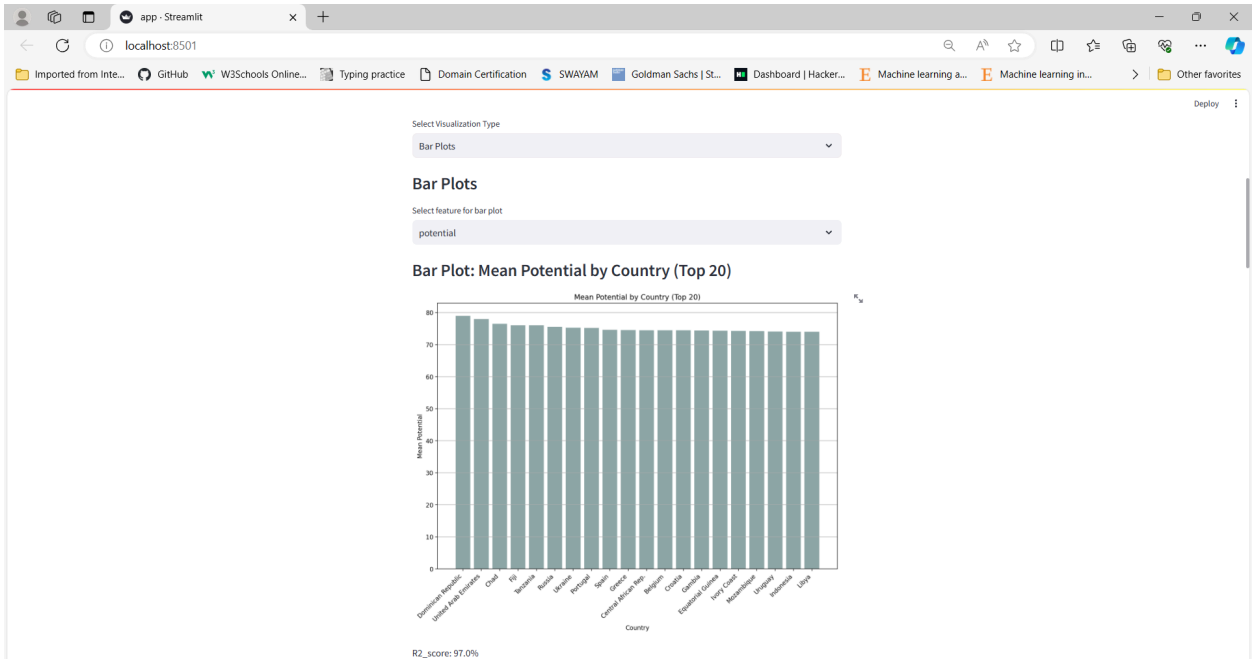



app · Streamlit
localhost:8501
Deploy
primary_position
Box Plot: Market Value vs primary_position
primary_position vs Market Value
1e8
value_euro
primary_position
CF CAM CM LW CB RW ST GK CDM LB RM LM RB LWB RWB
R2_score: 97.0%

Joint Plot:

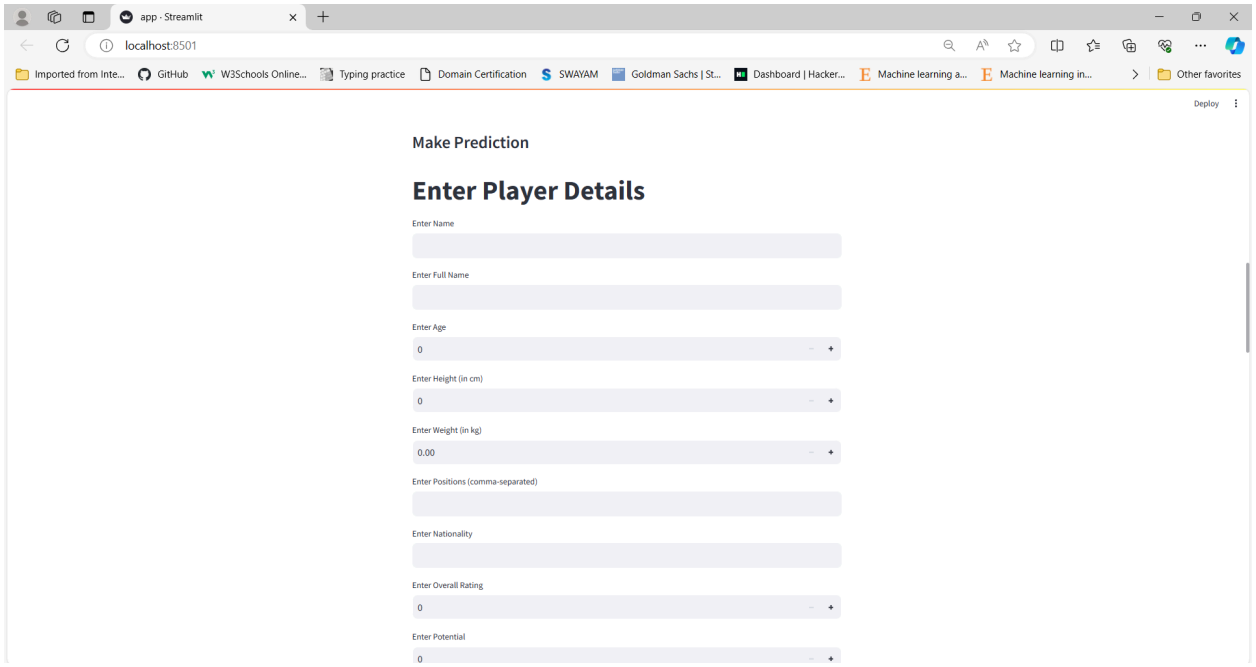

app · Streamlit
localhost:8501
Deploy
Joint Plots
Select feature for joint plot
overall_rating
Joint Plot: Overall_rating vs Value Euro
1e8
1e8
value_euro
overall_rating

Bar Plots:

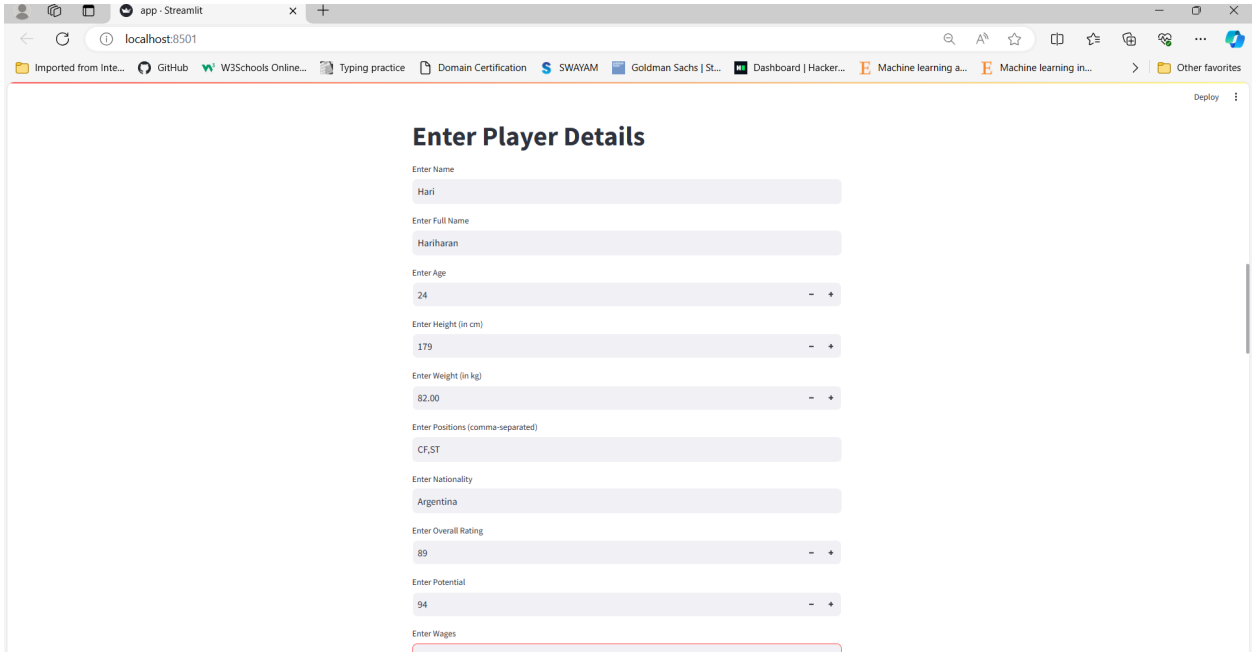


Make Prediction Area:

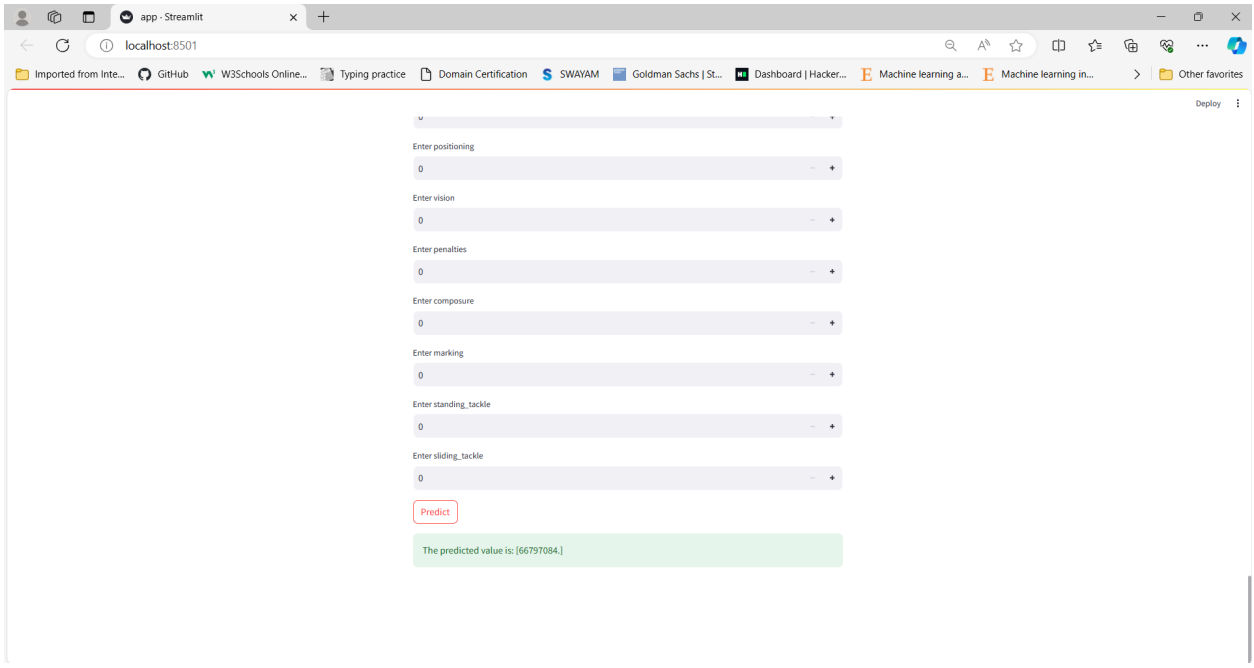
In this section, we need to enter the player details along with the positions they play, the rating, wage etc, and on doing so, we get to PREDICT the Market value of the player.



On Entering random Values:



Predicted Market Value:



CONCLUSION

We can conclude that the analysis of football player data using machine learning models has provided us with some valuable insights into various aspects of player performance, injury risk, market value, team success, and player development. Through the utilization of sophisticated algorithms on the datasets, we have been able to obtain useful information that can significantly impact decision-making processes within the football teams.

Transfer Market Value Prediction:

Our analysis has also shed light on the predictive capabilities of machine learning models in estimating the market value of football players. By leveraging attributes such as age, position, performance statistics, and market trends, these models can provide accurate valuations that inform clubs' decisions in player transfers, contract negotiations, and overall squad management. This predictive insight empowers clubs to make informed investment decisions, optimize resource allocation, and maximize returns and profit on player assets in the highly competitive football transfer market.

Player Development Analysis:

Lastly, our analysis has provided valuable insights into player development trajectories within the football industry. By identifying patterns and trends in player data, machine learning models can help clubs identify talent early, optimize youth development programs, and nurture the next generation of football stars. This analysis facilitates informed decision-making in talent scouting, recruitment, and academy management, thus fostering a sustainable pipeline of talent and ensuring long-term success for football clubs.

In summary, the analysis of football player data using machine learning models holds immense promise for revolutionizing decision-making processes within the football industry. By harnessing the power of data-driven insights, clubs, coaches, and analysts can gain a competitive edge, optimize performance, and drive success both on and off the field. As the field of sports analytics continues to evolve, leveraging advanced technologies and innovative methodologies, the potential for transformative impact within the football industry is limitless.

Future Enhancements

Team Success Prediction:

Furthermore, our analysis has demonstrated the potential of machine learning models to predict team success based on player data. By analyzing player attributes, performance metrics, and team dynamics, these models can forecast outcomes of matches and entire seasons with remarkable accuracy. This predictive capability equips clubs with valuable foresight, enabling them to devise effective strategies, allocate resources efficiently, and enhance their prospects of achieving success on the field.

Player Performance Prediction:

The machine learning models used in our analysis have shown us the capability to predict player performance with a high degree of accuracy. By considering a multitude of factors such as player attributes, historical performance data, and contextual variables, these models offer coaches and managers the opportunity to make informed decisions regarding team selection, tactical strategies, and training regimes. This predictive capability can contribute to optimizing team performance and achieving competitive advantages on the field.

References:

1. https://streamlit.io/cloud - streamlit library and documentation
2. https://docs.streamlit.io/get-started/installation
3. https://pandas.pydata.org/docs/- pandas documentation
4. https://matplotlib.org/stable/index.html - matplotlib documentation
5. https://seaborn.pydata.org/ - seaborn documentation
6. https://xgboost.readthedocs.io/en/stable/ - xgboost documentation
7. https://numpy.org/doc/ - numpy documentation
8. https://scikit-learn.org/stable/ - scikit learn documentation
9. Jupyter notebook and vs code for running the code

Peer Evaluation Form for Final Group Work CSE 487/587B

- Please write the names of your group members.

Group member 1 : Manish Bikumalla

Group member 2 : Hariharan Sriram

Group member 3 : Krithiman Manikonda

- Rate each groupmate on a scale of 5 on the following points, with 5 being HIGHEST and 1 being LOWEST.

Evaluation Criteria	Group member 1	Group member 2	Group member 3
How effectively did your group mate work with you?	5	5	5
Contribution in writing the report	5	5	5
Demonstrates a cooperative and supportive attitude.	5	5	5
Contributes significantly to the success of the project .	5	5	5
TOTAL	20	20	20

Also please state the overall contribution of your teammate in percentage below, with total of all the three members accounting for 100% (33.33+33.33+33.33 ~ 100%):

Group member 1: 33.33%
Group member 2: 33.33%
Group member 3: 33.33%