

PLAYER STRATEGY CLASSIFICATION USING ARTIFICIAL NEURAL NETWORKS AND LOGISTIC REGRESSION

A Project Report

Submitted To

Amity Institute of Information Technology

Bachelor of Computer Applications

By

SAIF MATHUR

Under the guidance of

Dr. Manoj Devare



Amity Institute of Information Technology

Amity University Mumbai

DECLARATION

I **Saif Mathur** solemnly declare that the project report titled **Player Strategy Classification Using Artificial Neural Network & Logistic Regression**

Is based on my own work carried out during the course of **Bachelor of Computer Applications**, under the guidance of allocated guide **Dr. Manoj Devare**.

I assert the statements made and conclusions drawn are an outcome of my project work.

I further certify that

- I. The work contained in the report is original and has been done by me under the general supervision of my supervisor.
- II. We have followed the guidelines provided by the university
- III. Whenever we have used materials (data, theoretical analysis, and text) from other sources, we have given due credit to them in the text of the report and giving their details in the references.

Saif Mathur

BCA 1608

A71004816004

(Signature)

ACKNOWLEDGEMENT

I would like to express my special thanks of gratitude to my project guide and (Head Of Department) of Amity Institute Of Information Technology,

Dr. Manoj Devare

Who gave me the golden opportunity to do this wonderful project related to Machine Learning and artificial neural networks, which also helped me in doing a lot of research and I came to know about so many new things.

I am really thankful to all the faculty members of AIIT department for supporting my idea of **Player Strategy Classification**

Secondly, I would also like to thank my family and friends who helped in finalizing this project within the limited time frame.

INDEX

Chapter No.	Contents	Page No.
1	INTRODUCTION	
	1.1 Abstract	6
	1.2 Introduction	7
	1.3 Background & Purpose	9
	1.4 Methodology	13
	1.5 Technology Used	20
	1.6 Code	22
2	SYSTEM DETAILS	
	2.1 Proposed System	47
	2.2 Objective Of System	49
	2.3 System Requirements	50
3	ANALYSIS, DESIGN & MODEL GRAPHS	
	3.1 Use-Case Diagram	51
	3.2 Activity Diagram	53
	3.3 Model Graphs <ul style="list-style-type: none"> • Confusion Matrix • Accuracy Graph • Loss Graph • Network Visualization • Model Summary • Layer Visualization 	54
	3.4 User Interface Screens	61
	3.5 Test Case	65

4	DRAWBACKS AND LIMITATIONS	
	4.1 Drawbacks and Limitations	66
5	PROPOSED ENHANCEMENTS	
	5.1 Proposed Enhancements	67
6	CONCLUSION	
	6.1 Conclusion	68
7	BIBLIOGRAPHY	
	7.1 Bibliography	69,70

1.1 ABSTRACT

This project addresses the classification of player strategy based on the feature vector acquired from FIFA`s official gaming website.

The Logistic Regression is used with a sigmoid function for finding the performance of the model. Upon finding over-fitted results, the ANN model is taken into consideration for evaluating the better performance of the model. The **Keras** is used for deploying the ANN model.

1.2 INTRODUCTION

The dataset used here is historical data from FIFA`s official gaming website. The objective of this work is to classify the strategy of a game plan by an individual player i.e. whether the player should be placed in the ‘Attack Position’ or ‘Defence Position’ .

The input is a tuple of different attributes of the player in integer form submitted by the user. For example the user will be asked to provide player specification, the player`s aggression level or player`s accuracy.

As the standard procedure the dataset was cleaned first, replacing missing values with their respective median and manually replaced by checking player statistics from the same website. Also some attributes were removed as they were irrelevant, for example the position of a goal keeper since we are only classifying players into ‘Attack’ or ‘Defence’.

The columns were then rearranged for better understanding.

The dataset was then divided into two sets to train and test the model.

Initially, the algorithm of **Logistic Regression** is applied, since this model is a linear model it could not find a clear boundary and gave poor results.

The second algorithm applied on the same dataset is an **Artificial Neural Network** which gave a better accuracy.

1.3 BACKGROUND & PURPOSE

The Rise Of The Gaming Industry

In the modern day the gaming industry has matured and is maturing, fast. It has opened up new opportunities for ‘gamers’, a gamer is someone who plays video games on a regular basis. The games that are sold in the present day market are quite advanced and require good hand - eye coordination, **strategy, planning**. These games are no longer for children, it requires great skill and practice to master them.

This project was designed to help people like them who make a living out of gaming.

It is based on very popular game amongst gamers called FIFA, it is designed by EA sports

It is a game of soccer or football, it is a strategy based game which requires the player to set up positions carefully according to player statistics. i.e how the player performs on the field.

The project idea is based on the strategy part, where the user has to decide whether a player should be placed in a certain position of play or not.

This model was designed to predict and give near accurate predictions about these strategies.

This model can be redesigned for several strategy based games.

Referred Case Studies:

Control Strategies for Multiplayer Target-Attacker-Defender Differential Games with Double Integrator Dynamics

Mitchell Coon and Dimitra Panagou

Abstract :

The paper was published in December, 2017 to present a method for deriving suitable controls and assigning attacker and defender pairs in a target attacker-defender differential game between an arbitrary numbers of attackers and defenders, all of which are modelled using double integrator dynamics. It is assumed that each player has perfect information about the states and controls of the players within a certain range of themselves, but they are unaware of any players outside of this range. Isochrones are created based on the time-optimal trajectories needed for the players to reach any point in the shortest possible time. The intersections of the players' isochrones are used to determine whether a defender can intercept an attacker before the attacker reaches the target. Sufficient conditions on the detection range of the defenders and the guaranteed capture despite perturbations of the attackers off the nominal trajectories are derived. Then, in simulations with multiple players, attacker-defender pairs are assigned so that the maximum number of attackers are intercepted in the shortest possible time.

Game Player Strategy Pattern Recognition and How UCT Algorithms Apply Pre-knowledge of Player's Strategy to Improve Opponent AI

ISBN: 978-0-7695-3514-2

Publisher: IEEE

[2008 International Conference on Computational Intelligence for Modelling Control & Automation](#)

Abstract:

Player strategy pattern recognition (PSPR) is to apply pattern recognition and its approach to identification of player's strategy during the gameplay. Correctly identified player's strategy, which is called knowledge, could be used to improve game opponent AI which can be implemented by KB-UCT (knowledge-based upper confidence bound for trees). KB-UCT improves adaptability of game AI, the challenge level of the gameplay, and the performance of the opponent AI; as a result the entertainment of game is promoted. In this paper, the prey and predator game genre of dead end game is used as a test-bed. During the PSPR, classification algorithm of KNN (k-nearest neighbour) is chosen to analyse off-line data from the simulated gamers who are choosing different strategies. Based on the information from PSPR, the game AI is promoted through application of KB-UCT, in this case, domain knowledge is used for UCT tree pruning; as a result the performance of the opponent AI is enhanced.

Optimizing NBA Player Selection Strategies Based on Salary and Statistics Analysis

Ramya Nagarajan , Lin Li

2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech)

Abstract:

In National Basketball Association (NBA), how to comprehensively measure a player's efficiency and how to sign talented players with reasonable contracts are two challenging issues. This research explored the key indicators widely used to measure player efficiency and team performance. Through data analysis, two indicators, namely Player Efficiency Rating and Player Defence Rating, were chosen to formulate the prediction of the team winning rate. Besides, different player selection strategies were proposed according to different objectives and constraints. Experimental results show that the developed team winning rate prediction models have high accuracy and the player selection strategies are effective.

1.4 METHODOLOGY

The project was made with the help of **Anaconda IDE**, an open source distribution designed for various data science and machine learning projects:

<https://www.anaconda.com/distribution/>

Under this distribution, the **spyder** environment was used to edit the code

Spyder is a powerful scientific environment written in **Python**, for **Python**, and designed by and for scientists, engineers and data analysts.

Versions:

Anaconda Enterprise 5.3

Spyder 3.3

The code for this project is written using python and the following libraries:

- Numpy 1.16
- Pandas 0.24
- Matplot library 3.02
- Keras 2.2
- Sci-kit learn library 0.2
- Pickle 0.7
- Flask 1.0.2

The code and the usage of these methods and libraries are described below:

1. NumPy

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

- **Numpy library is used to write data to array**
- **Its append function was used frequently in the code**

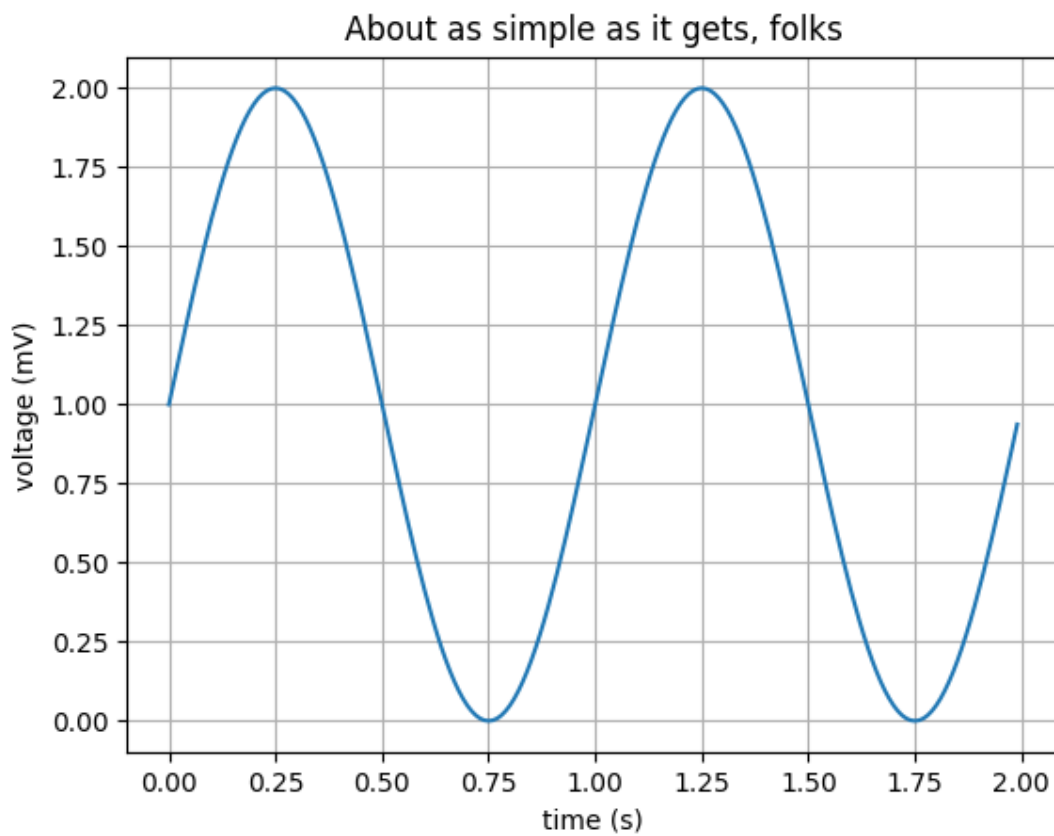
2. Pandas

pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the python programming language.

- Pandas is used to load data into spyder using the **read_csv()** function which takes Microsoft excel's 'comma separated values' file format to read data.
- Pandas is also used frequently to create or change numpy modules and objects to data frames using **pandas.DataFrame()**

3. Matplot Library

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the jupyter notebook, web application servers, and four graphical user interface toolkits.



- Matplotlib library is used to plot accuracy and loss graph of the neural network
- The function `pyplot()` is used

4. Keras

<https://keras.io/>

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow , CNTK, or Theano. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research.

- Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility).
- Supports both convolutional networks and recurrent networks, as well as combinations of the two.
- Runs seamlessly on CPU and GPU.

Methods used from keras:

1. Sequential:

The simplest type of model is the `Sequential` model, a linear stack of layers.

```
from keras.models import Sequential
```

Stacking layers is as easy as `.add()`:

2. Dense

Consists of activation functions and `input_dim`

```
classifier.add
```

3. Compile

Once your model looks good, configure its learning process with `.compile()`

4. Predict

Generate predictions

5. Scikit-Learn

Scikit-learn is a library in Python that provides many unsupervised and supervised learning algorithms.

The functionality that scikit-learn provides include: Regression, including Linear and Logistic Regression. Classification, including K-Nearest Neighbors.

- It is used to divide the data set into training and test split
- It is used to deploy Logistic Regression model
- It is used to import metrics for accuracy calculation and confusion matrix

6. Pickle

It is used for serializing and de-serializing a Python object structure. Any object in python can be pickled so that it can be saved on disk.

What pickle does is that it “serialises” the object first before writing it to file.

Pickling is a way to convert a python object (list, dict, etc.) into a character stream.

The idea is that this character stream contains all the information necessary to reconstruct the object in another python script.

- It is used to save the model named ‘**classifier**’ into a ‘**.pkl**’ file

To be read into a web API

7. Flask

Source: <http://flask.pocoo.org>

Flask is a lightweight WSGI web application framework. It is designed to make getting started quick and easy, with the ability to scale up to complex applications. It began as a simple wrapper around Werkzeug and Jinja and has become one of the most popular Python web application frameworks.

- Flask is used in this project in an attempt to make a web API for this project

1.5 TECHNOLOGY USED

Web services:

- FIFA web service
- Kaggle
- Stack overflow
- Youtube
- Flask web service
- Keras documentation

Software for data separation and cleaning:

- Microsoft Excel
- Anaconda Navigator
- Anaconda Prompt
- Python IDLE
- Spyder
- Web browser (chrome)
- Atom editor

Languages Used for writing code:

- Python
- HTML

System Used:

- Processor - Intel Core i5 8th generation
- Ram – 4GB, 8GB recommended
- Graphics – On board graphics (intel)

1.6 CODE

1. Python code for main file

```
# -*- coding: utf-8 -*-

"""

@author: Saif Mathur

"""


import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import pickle


dataset = pd.read_csv('CompleteDataset.csv')

dataset.head()


#GK pos removed


columns_req = ['Acceleration', 'Aggression', 'Agility', 'Balance', 'Ball control',

               'Composure', 'Crossing', 'Curve', 'Dribbling', 'Finishing',

               'Free kick accuracy', 'Heading accuracy', 'Interceptions',
```

'Jumping', 'Long passing', 'Long shots', 'Marking', 'Penalties',

'Positioning', 'Reactions', 'Short passing', 'Shot power',

'Sliding tackle', 'Sprint speed', 'Stamina', 'Standing tackle',

'Strength', 'Vision', 'Volleys', 'Preferred Positions']

```
columns_rearranged = ['Aggression','Crossing', 'Curve', 'Dribbling', 'Finishing',
```

```
'Free kick accuracy', 'Heading accuracy', 'Long shots','Penalties', 'Shot power', 'Volleys',
```

```
'Short passing', 'Long passing',
```

```
'Interceptions', 'Marking', 'Sliding tackle', 'Standing tackle',
```

```
'Strength', 'Vision', 'Acceleration', 'Agility',
```

```
'Reactions', 'Stamina', 'Balance', 'Ball control','Composure','Jumping',
```

```
'Sprint speed', 'Positioning','Preferred Positions']
```

```
new_dataset = dataset[columns_rearranged]
```

```
new_dataset = pd.DataFrame(new_dataset)
```

```
new_dataset.head()
```

```
#removing GK
```

```
new_dataset['Preferred Positions'] = new_dataset['Preferred Positions'].str.strip()
```

```
new_dataset = new_dataset[new_dataset['Preferred Positions'] != 'GK']
```

```
new_dataset.head()
```

```
new_dataset.isnull().values.any()
```

```
p = new_dataset['Preferred Positions'].str.split().apply(lambda x: x[0]).unique()
```

```
df_new = new_dataset.copy()
```

```
df_new.drop(df_new.index, inplace=True)
```

```
for i in p:
```

```
    df_temp = new_dataset[new_dataset['Preferred Positions'].str.contains(i)]
```

```
    df_temp['Preferred Positions'] = i
```

```
    df_new = df_new.append(df_temp, ignore_index=True)
```

```
df_new.iloc[:500, :]
```

```
cols = [col for col in new_dataset.columns if col not in ['Preferred Positions']]
```

```
for i in cols:
```

```
    df_new[i] = df_new[i].apply(lambda x: eval(x) if isinstance(x,str) else x)
```



```
df_new.iloc[:,500, :]
```

```
mapping = {'ST': 1, 'RW': 1, 'LW': 1, 'RM': 1, 'CM': 1, 'LM': 1, 'CAM': 1, 'CF': 1,  
           'CDM': 0, 'CB': 0, 'LB': 0, 'RB': 0, 'RWB': 0, 'LWB': 0}
```

```
df_new = df_new.replace({'Preferred Positions':mapping})
```

```
from sklearn.preprocessing import StandardScaler
```

```
sc = StandardScaler()
```

```
df_new.iloc[:,0:29] = sc.fit_transform(df_new.iloc[:,0:29])
```

```
#df_new is cleaned at this part
```

```
X = df_new.iloc[:,:-1].values
```

```
y = df_new.iloc[:, -1].values
```

```
from sklearn.model_selection import train_test_split
```

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.2,random_state = 0)
```

```
from sklearn.dummy import DummyClassifier
```

```
dc = DummyClassifier(strategy = 'most_frequent')
```

```
dc.fit(X_train,y_train)
```

```
'''
```

```
from sklearn.linear_model import LogisticRegression
```

```
classifier_log = LogisticRegression().fit(X_train,y_train)
```

```
y_pred = classifier_log.predict(X_test)
```

```
acc_log = classifier_log.score(X_test,y_test)
```

```
'''
```

```
#Deploying ANN
```

```
import keras
```

```
from keras.layers import Dense
```

```
from keras.models import Sequential
```

```
classifier = Sequential()
```

```
classifier.add(Dense(output_dim = 15, init = 'uniform',activation = 'relu',input_dim = 29))
```

```
classifier.add(Dense(output_dim = 15, init = 'uniform',activation = 'relu'))
```

```
classifier.add(Dense(output_dim = 1, init = 'uniform',activation = 'sigmoid'))
```

```
classifier.compile(optimizer = 'adam',loss = 'binary_crossentropy',metrics = ['accuracy'])
```

```
classifier.fit(X_train,y_train,batch_size = 10,epochs = 100)
```

```
y_pred_ann = classifier.predict(X_test)
```

```
pickle.dump(classifier, open('model.pkl','wb'))
```

```
y_pred_ann = y_pred_ann > 0.5
```

```
from sklearn.metrics import confusion_matrix
```

```
cm_for_ann = confusion_matrix(y_test,y_pred_ann)acc_ann = (4654/5451)*100
```

2. Flask File Code:

```
# -*- coding: utf-8 -*-

"""

Created on Sun Apr 21 16:25:11 2019

@author: Saif

"""

import flask

import numpy as np

import tensorflow as tf

#from keras.models import load_model

from flask import request

from flask import render_template

import pickle

app = flask.Flask(__name__)

def init():

    global model,graph

    # load the pre-trained Keras model

    #model = load_model('C:/Users/Saif/desktop/playerPosition.pkl')

    model = pickle.load(open('playerPosition.pkl', 'rb'))
```

```
graph = tf.get_default_graph()
```

```
return model
```

```
@app.route('/project',methods = ['POST'])
```

```
def api():
```

```
    if request.method == 'POST':
```

```
        return render_template('api2.html')
```

```
# API for prediction
```

```
@app.route('/project', methods=['POST'])
```

```
def predict():
```

```
    if request.methods == 'POST':
```

```
        def getParameters():
```

```
            parameters=[]
```

```
            #parameters.append(request.form.get('name'))
```

```
            parameters.append(request.form.get('Aggression') )
```

```
            parameters.append(request.form.get('Crossing'))
```

```
            parameters.append(request.form.get('Curve'))
```

```
            parameters.append(request.form.get('Dribbling'))
```

```
parameters.append(request.form.get('Finishing'))

parameters.append(request.form.get('FreeKickAccuracy'))

parameters.append(request.form.get('HeadingAccuracy'))

parameters.append(request.form.get('LongShots'))

parameters.append(request.form.get('Penalties'))

parameters.append(request.form.get('ShotPower'))

parameters.append(request.form.get('Volleys'))

parameters.append(request.form.get('ShortPassing'))

parameters.append(request.form.get('LongPassing'))

parameters.append(request.form.get('Interceptions'))

parameters.append(request.form.get('Marking'))

parameters.append(request.form.get('SlidingTackle'))

parameters.append(request.form.get('StandingTackle'))

parameters.append(request.form.get('Strength'))

parameters.append(request.form.get('Vision'))

parameters.append(request.form.get('Acceleration'))

parameters.append(request.form.get('Agility'))

parameters.append(request.form.get('Reactions'))

parameters.append(request.form.get('Stamina'))

parameters.append(request.form.get('Balance'))
```

```
parameters.append(request.form.get('BallControl'))

parameters.append(request.form.get('Composure'))

parameters.append(request.form.get('Jumping'))

parameters.append(request.form.get('SprintSpeed'))

parameters.append(request.form.get('Positioning'))


return parameters


parameters = getParameters()

inputFeature = np.asarray(parameters).reshape(1, 29)

with graph.as_default():

    raw_prediction = model.predict(inputFeature)

    if raw_prediction > 0.5:

        prediction = 'Attack'

    else:

        prediction = 'Defense'


return render_template('api2.html',p =prediction)
```

```

# Cross origin support

def sendResponse(responseObj):

    response = flask.jsonify(responseObj)

    response.headers.add('Access-Control-Allow-Origin', '*')

    response.headers.add('Access-Control-Allow-Methods', 'POST','GET ')

    response.headers.add('Access-Control-Allow-Headers', 'accept,content-type,Origin,X-
Requested-With,Content-Type,access_token,Accept,Authorization,source')

    response.headers.add('Access-Control-Allow-Credentials', True)

    return response


if __name__ == "__main__":

    print((" Loading model and Flask starting server..."

"please wait until server has fully started"))

    init()

    #app.debug=True

    app.run(threaded=True)

```

CODE FOR PYTHON APP:

```
import numpy as np

import pickle

model = pickle.load(open('playerPosition.pkl', 'rb'))

import tensorflow as tf

graph = tf.get_default_graph()

name = input('enter player name: ')

if name=="":

    print('Cannot be Empty')

Agression = input('enter player aggression: ')

if Agression>100:

    print('value should be within 0-100 range')

Crossing = input('enter player Crossing: ')

if Crossing>100:

    print('value should be within 0-100 range')

Curve = input('enter player Curve: ')

if Curve>100:

    print('value should be within 0-100 range')

Dribbling = input('enter player Dribbling: ')

if Dribbling>100:

    print('value should be within 0-100 range')
```



```
Finishing = input('enter player Finishing: ')

if Finishing>100:

    print('value should be within 0-100 range')

FreeKickAccuracy = input('enter player Free Kick Accuracy: ')

if FreeKickAccuracy>100:

    print('value should be within 0-100 range')

HeadingAccuracy = input('enter player Heading Accuracy: ')

if HeadingAccuracy>100:

    print('value should be within 0-100 range')

LongShots = input('enter player Long Shots: ')

if LongShots>100:

    print('value should be within 0-100 range')

Penalties = input('enter player Penalties: ')

if Penalties>100:

    print('value should be within 0-100 range')

ShotPower = input('enter player Shot Power: ')

if ShotPower>100:

    print('value should be within 0-100 range')

Volleys = input('enter player Volleys: ')

if Volleys>100:

    print('value should be within 0-100 range')

ShortPassing = input('enter player Short Passing: ')

if ShortPassing>100:
```

```
    print('value should be within 0-100 range')

LongPassing = input('enter player LongPassing: ')

if LongPassing>100:

    print('value should be within 0-100 range')

Interceptions = input('enter player Interceptions: ')

if Interceptions>100:

    print('value should be within 0-100 range')

Marking = input('enter player Marking: ')

if Marking>100:

    print('value should be within 0-100 range')

SlidingTackle = input('enter player Sliding Tackle: ')

if SlidingTackle>100:

    print('value should be within 0-100 range')

StandingTackle = input('enter player Standing Tackle: ')

if StandingTackle>100:

    print('value should be within 0-100 range')

Strength = input('enter player Strength: ')

if Strength>100:

    print('value should be within 0-100 range')

Vision = input('enter player Vision: ')

if Vision>100:

    print('value should be within 0-100 range')

Acceleration = input('enter player Acceleration: ')
```

```
if Acceleration>100:

    print('value should be within 0-100 range')

Agility = input('enter player Agility: ')

if Agility>100:

    print('value should be within 0-100 range')

Reactions = input('enter player Reactions: ')

if Reactions>100:

    print('value should be within 0-100 range')

Stamina = input('enter player Stamina: ')

if Stamina>100:

    print('value should be within 0-100 range')

Balance = input('enter player Balance: ')

if Balance>100:

    print('value should be within 0-100 range')

BallControl = input('enter player Ball Control: ')

if BallControl>100:

    print('value should be within 0-100 range')

Composure = input('enter player Composure: ')

if Composure>100:

    print('value should be within 0-100 range')

Jumping = input('enter player Jumping: ')

if Jumping>100:

    print('value should be within 0-100 range')
```

```
SprintSpeed = input('enter player Sprint Speed: ')
```

```
if SprintSpeed>100:
```

```
    print('value should be within 0-100 range')
```

```
Positioning = input('enter player Positioning: ')
```

```
if Positioning>100:
```

```
    print('value should be within 0-100 range')
```

```
parameters = []
```

```
#parameters.append(name)
```

```
parameters.append(Agression)
```

```
parameters.append(Crossing)
```

```
parameters.append(Curve)
```

```
parameters.append(Dribbling)
```

```
parameters.append(Finishing)
```

```
parameters.append(FreeKickAccuracy)
```

```
parameters.append(HeadingAccuracy)
```

```
parameters.append(LongShots)
```

```
parameters.append(Penalties)
```

```
parameters.append(ShotPower)
```

```
parameters.append(Volleys)
```

```
parameters.append(ShortPassing)
```

```
parameters.append(LongPassing)
```

```
parameters.append(Interceptions)
```

```
parameters.append(Marking)
```

```
parameters.append(SlidingTackle)

parameters.append(StandingTackle)

parameters.append(Strength)

parameters.append(Vision)

parameters.append(Acceleration)

parameters.append(Agility)

parameters.append(Reactions)

parameters.append(Stamina)

parameters.append(Balance)

parameters.append(BallControl)

parameters.append(Composure)

parameters.append(Jumping)

parameters.append(SprintSpeed)

parameters.append(Positioning)

inputFeature = np.asarray(parameters).reshape(1, 29)

with graph.as_default():

    raw_prediction = model.predict(inputFeature)

if raw_prediction > 0.5:

    prediction = 'Attack'

else:

    prediction = 'Defense'

print('Place ' + str(name) + ' In ' + str(prediction))
```

3. HTML code for API

File name: 'api2.html'

```
<!DOCTYPE html>

<html lang="en" dir="ltr">

  <head>

    <meta charset="utf-8">

    <title>Player position prediction</title>

    <style media="screen">

      {

        font-size:25px;

      }

    </style>

  </head>

  <body>

    <br>

    <form class="form1"  action="{{url_for('predict2.html')}}"
method="POST">

      Enter Player Characteristics

      <br>

      name : <input type="text" name="name" >

      <br>

      <br>

      Aggression:  <input type="number" name="Aggression" max="200" >
```

```
<br>

<br>

    Crossing: <input type="number" name="Crossing" max="200" >

<br>

<br>

    Curve: <input type="number" name="Curve" max="200" >

<br>

<br>

    Dribbling: <input type="number" name="Dribbling" max="200" >

<br>

<br>

    Finishing: <input type="number" name="Finishing" max="200" >

<br>

<br>

    Free Kick Accuracy: <input type="number"
name="FreeKickAccuracy" max="200">

<br>

<br>

    Heading accuracy : <input type="number"
name="HeadingAccuracy" max="200" >

<br>

<br>

    Long shots: <input type="number" name="LongShots" max="200" >

<br>

<br>

    Penalties : <input type="number" name="Penalties" max="200" >

<br>
```


Shot power : <input type="number" name="ShotPower" max="200">

Volleys : <input type="number" name="Volleys" max="200" >

Short passing : <input type="number" name="ShortPassing"

max="200" >

Long passing : <input type="number" name="LongPassing"

max="200" >

Interceptions : <input type="number" name="Interceptions"

max="200" >

Marking : <input type="number" name="Marking" max="200" >

Sliding tackle : <input type="number" name="SlidingTackle"
max="200" >

Standing tackle : <input type="number" name="StandingTackle"
max="200" >

Strength : <input type="number" name="Strength" max="200" >

Vision : <input type="number" name="Vision" max="200" >

Acceleration : <input type="number" name="Acceleration"
max="200" >

Agility : <input type="number" name="Agility" max="200" >

Reactions : <input type="number" name="Reactions" max="200" >

Stamina : <input type="number" name="Stamina" max="200" >

Balance : <input type="number" name="Balance" max="200" >

Ball control : <input type="number" name="BallControl"
max="200" >

Composure : <input type="number" name="Composure" max="200" >

Jumping : <input type="number" name="Jumping" max="200" >

Sprint : <input type="number" name="Sprint" max="200" >


```
Positioning : <input type="number" name="Positioning" max="200"
>
<br>
<br>
<button class= "btn" type="submit" name="button"> Predict </button>
<label for="">Player should be placed in {{p}}</label>

</form>

</body>
</html>
```

CODE FOR GRAPHS AND VISUALS

```
# -*- coding: utf-8 -*-
```

```
"""
```

```
Created on Thu Mar 14 07:47:18 2019
```

```
@author: Saif
```

```
"""
```

```
history = classifier.fit(X_train,y_train,batch_size = 10,epochs = 100)
```

```
#confusion matrix for model prediction
```

```
from sklearn.metrics import confusion_matrix
```

```
cm_for_ann = confusion_matrix(y_test,y_pred_ann)
```

```
#actual accuracy
```

```
acc_ann = (4654/5451)*100
```

```
#visualization
```

```
from ann_visualizer.visualize import ann_viz;
```

```
ann_viz(classifier, title="plot for project")
```

```
#summary for classifier
```

```
from keras.utils import plot_model
```

```
plot_model(classifier, to_file='model.png')
```

```
import matplotlib.pyplot as plt
```

```
# Plot training accuracy values
```

```
plt.plot(history.history['acc'])
```

```
#plt.plot(history.history['val_acc'])
```

```
plt.title('Model accuracy')
```

```
plt.ylabel('Accuracy')
```

```
plt.xlabel('Epoch')
```

```
plt.legend(['Train', 'Test'], loc='upper left')
```

```
plt.show()
```

```
# Plot training loss values
```

```
plt.plot(history.history['loss'])
```

```
#plt.plot(history.history['val_loss'])
```

```
plt.title('Model loss')
```

```
plt.ylabel('Loss')
```

```
plt.xlabel('Epoch')
```

```
plt.legend(['Train', 'Test'], loc='upper left')
```

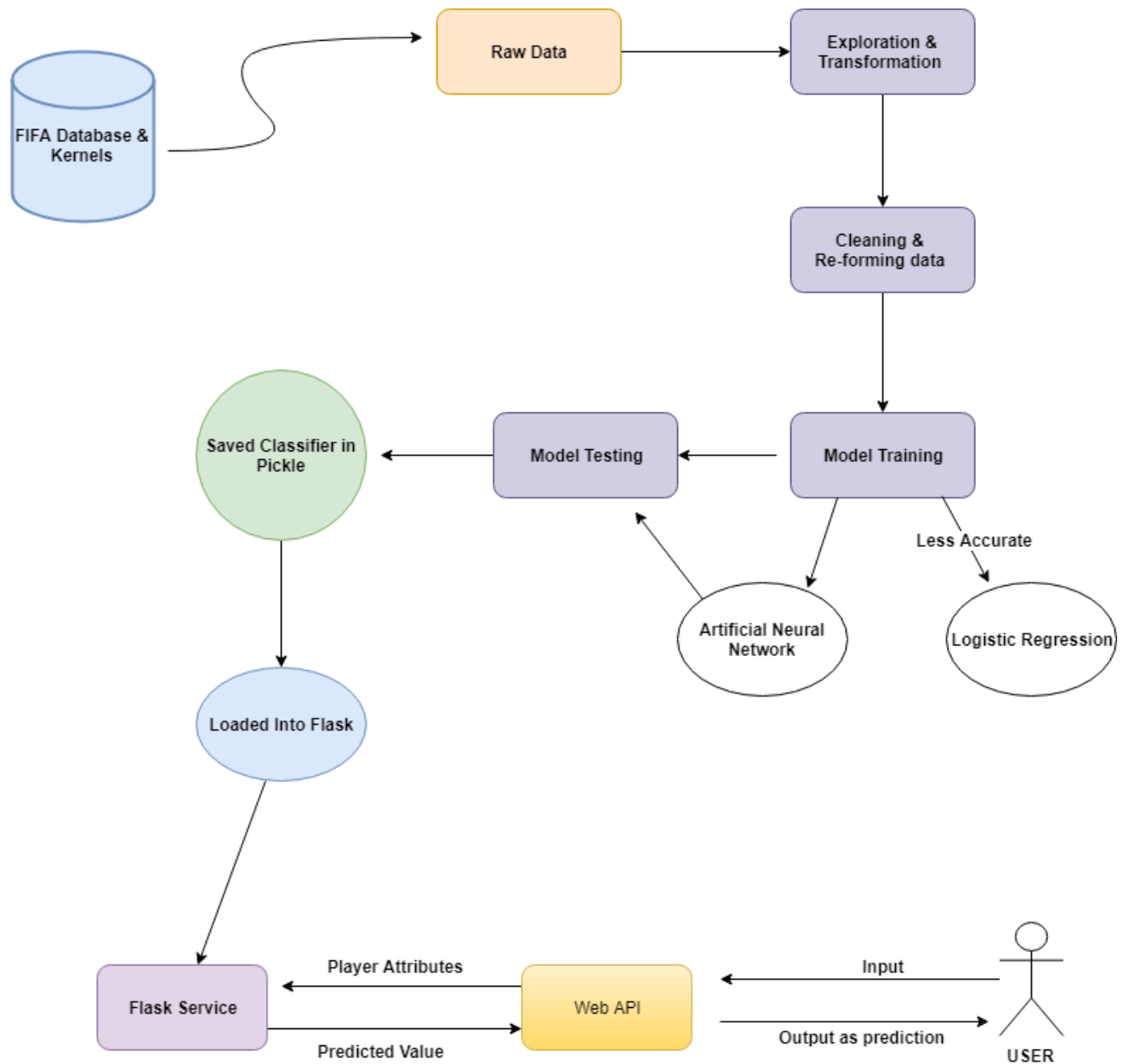
```
plt.show()
```

2.0 SYSTEM DETAILS

2.1 PROPOSED SYSTEM:

The system is fulfilling the following activities:

- Reducing the complexity of placing players during real time gaming
- The system can be applied to any strategy based game, with few changes in the software
- A player can have several attributes, the proposed system is capable of identifying the major attributes for **defence** and **attack** separately
- System will expect user to enter all attributes of a player one by one in order to predict near accurate values.
- The proposed system is only designed for a binary outcome, 1 being attack and 0 being defence. New features will be added in the next versions in order to predict multiple classes.



2.2 OBJECTIVE OF THE SYSTEM

- The objective of the system is to provide the user with the ease of making quick decisions during gameplay
- The system is designed specially for intermediate level players or beginners who have just started playing strategy based games and want to become good at them
- With modifications this system could identify and classify any player position
- The system with enough attributes can promise to give an accurate prediction with an accuracy of 87%
- Make people aware about, how a problem can be fixed easily with historical data and machine learning.

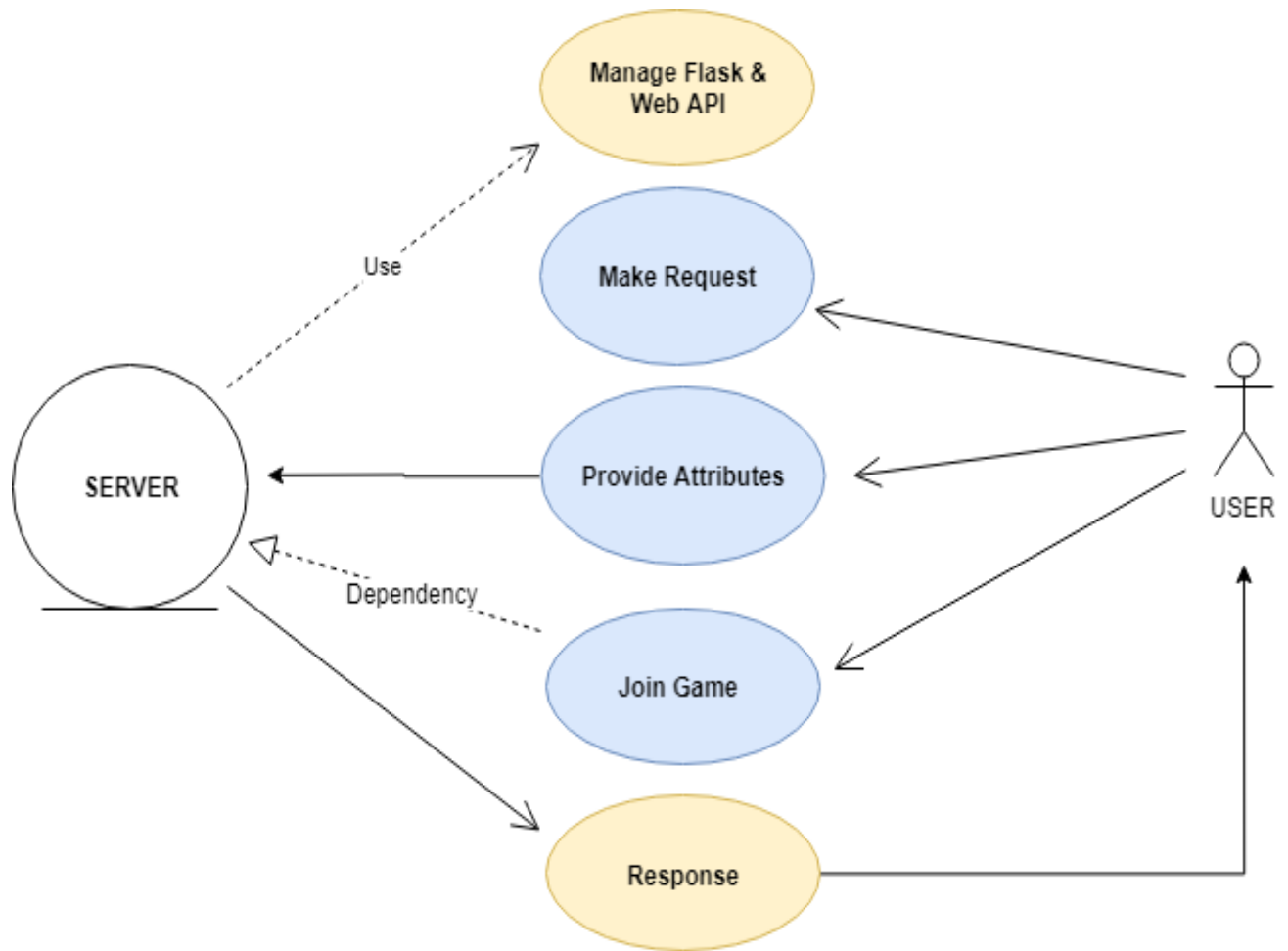
2.3 SYSTEM REQUIREMENTS

Player attributes:

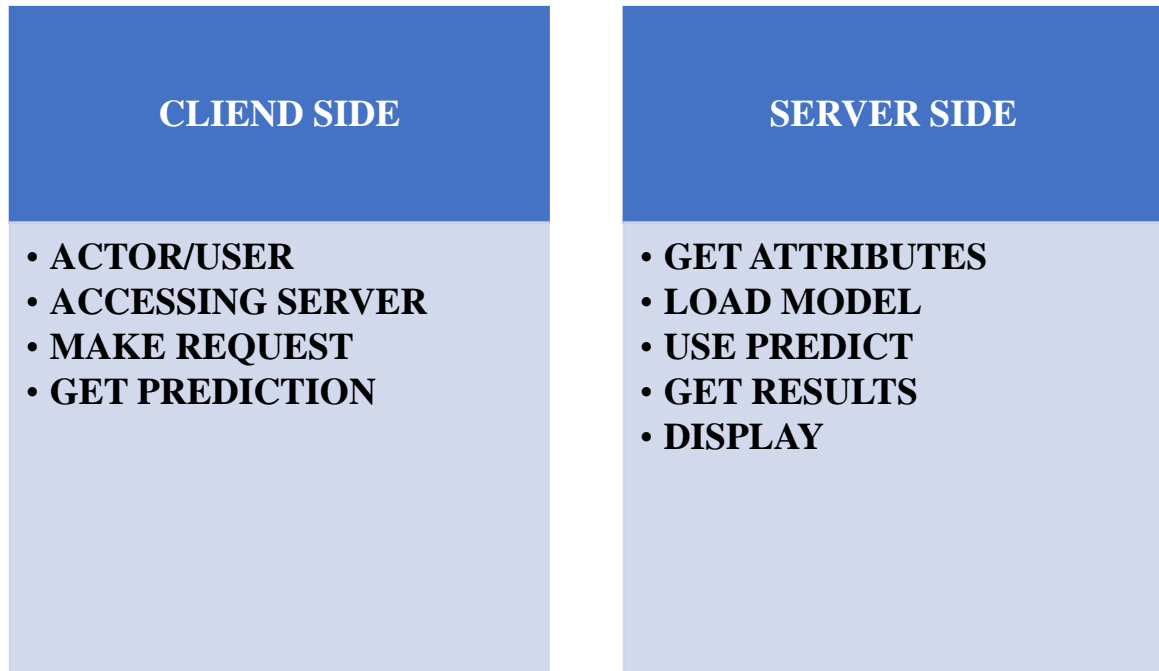
NAME
AGGRESSION
CURVE
DRIBBLING
FINISHING
FREE KICK ACCURACY
HEADING ACCURACY
LONG SHOTS
PENALTIES
SHOT POWER
VOLLEYS
SHORT PASSING
LONG PASSING
INTERCEPTIONS
MARKING
SLIDING TACKLE
STANDING TACKLE
STRENGTH
VISION
ACCELERATION
AGILITY
REACTIONS
STAMINA
BALANCE
BALL CONTROL
COMPOSURE
JUMPING
POSITIONING

3.0 ANALYSIS & DESIGN

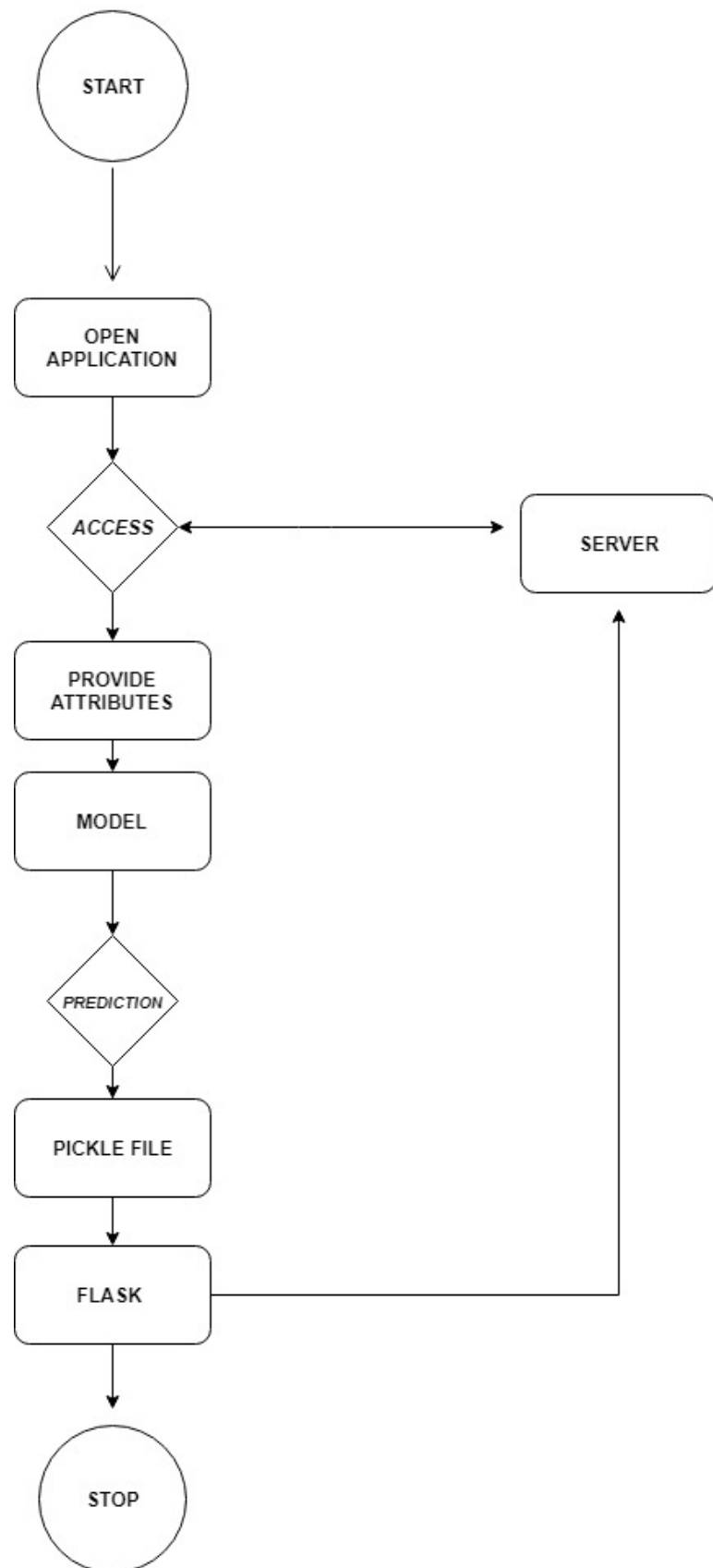
3.1 USE CASE DIAGRAM:



USE CASE DESCRIPTION:



3.2 ACTIVITY DIAGRAM



3.3 MODEL GRAPHS:

RESULTS IN CONFUSION MATRIX FORM FOR ARTIFICIAL NEURAL NETWORK

cm - NumPy array

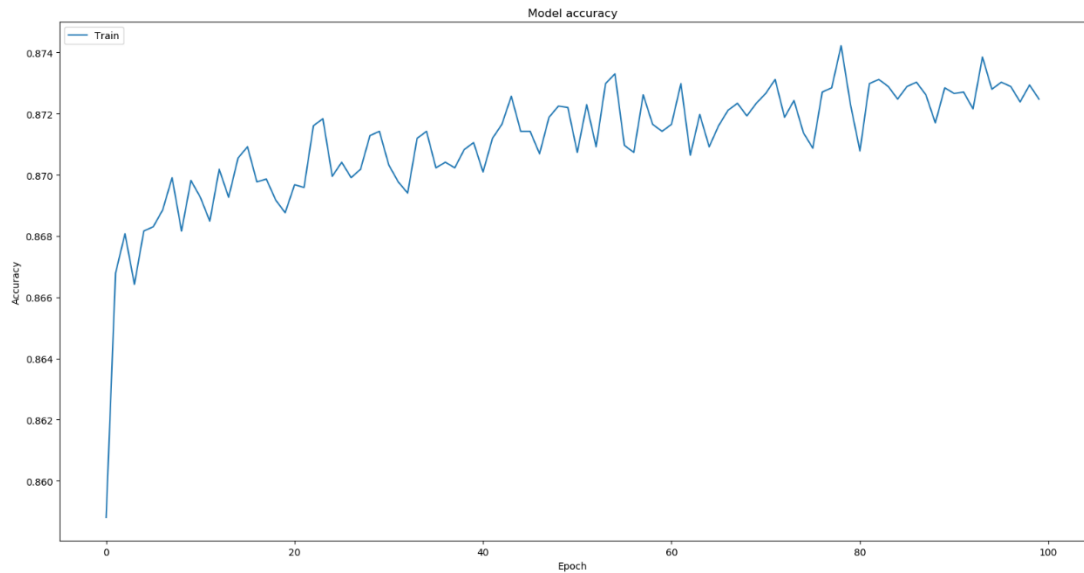
	0	1
0	1824	323
1	411	2893

ACCURACY CALCULATED BY

$$(4717/5451)*100 = 87\%$$

ACCURACY GRAPH FOR MODEL PREDICTION:

Training Set:



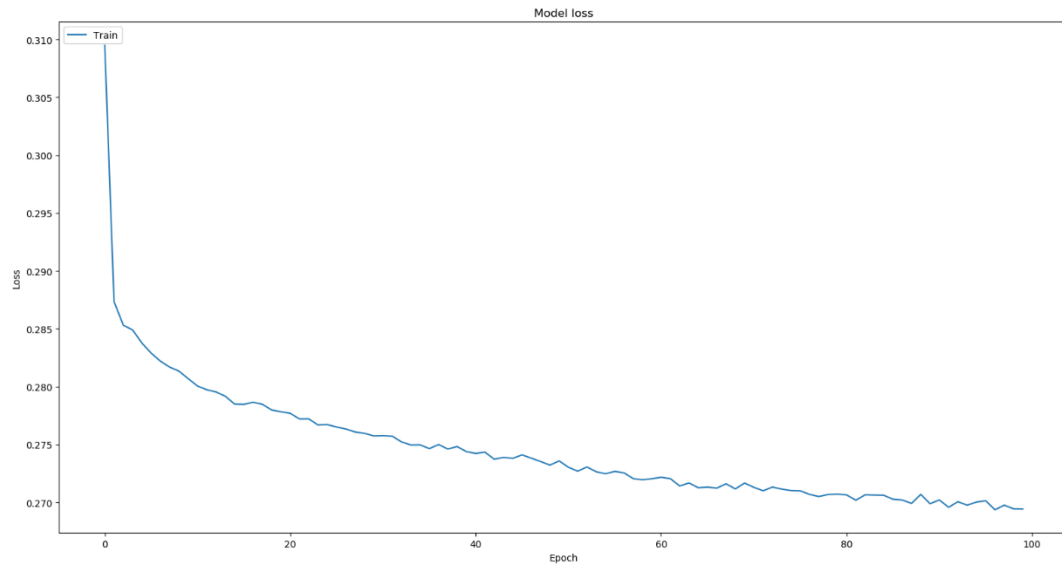
X AXIS – number of epochs used in neural network

Y AXIS – accuracy with each epoch

GRAPH – training set graph

LOSS FUNCTION GRAPH:

Training Set:

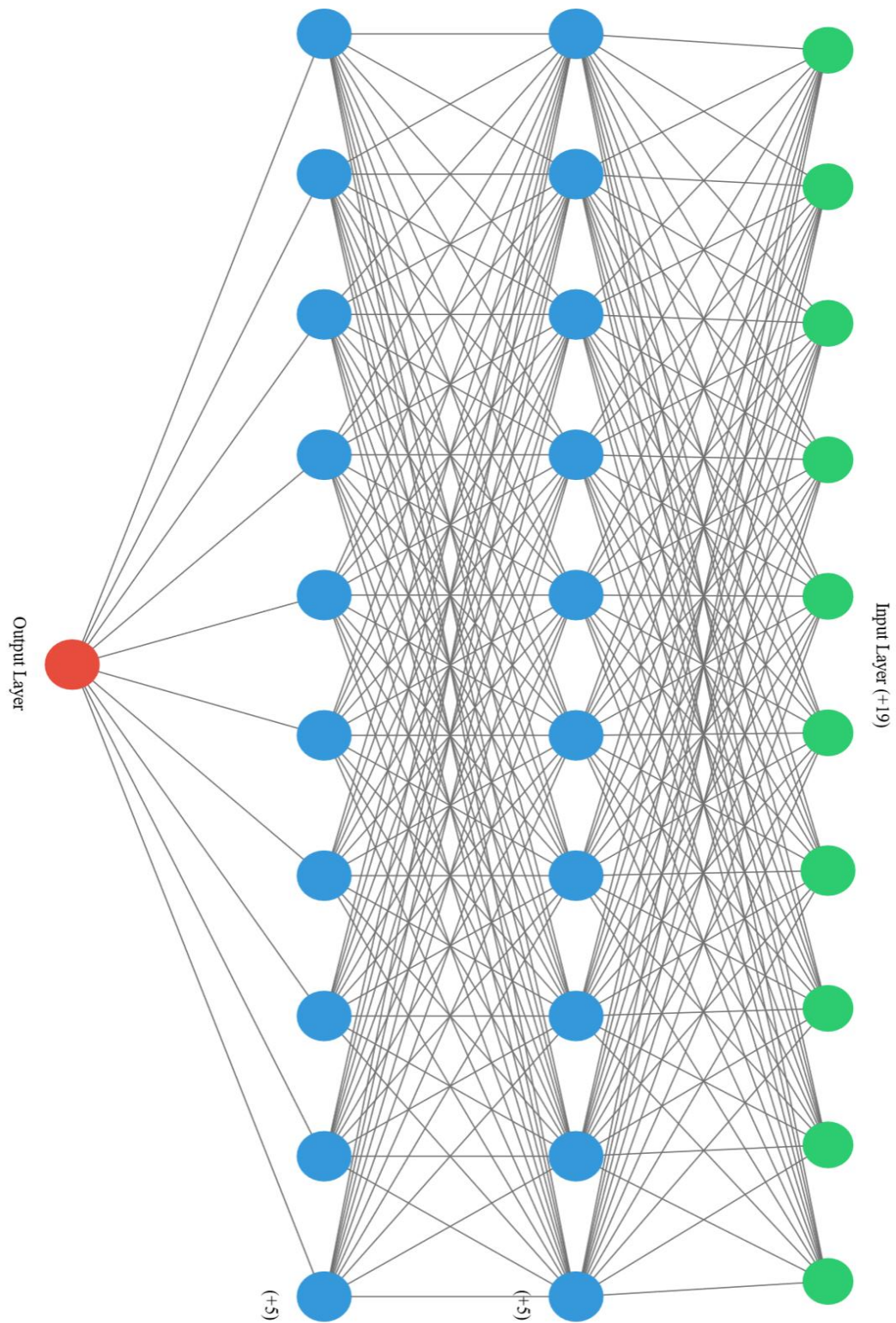


X AXIS – Number of Epochs Used in Neural Network

Y AXIS – Loss with each Epoch

GRAPH – Training Set Graph

VISUALISATION OF NETWORK LAYERS



MODEL SUMMARY:

Keras provides a way to summarize a model.

The summary is textual and includes information about:

- The layers and their order in the model.
- The output shape of each layer.
- The number of parameters (weights) in each layer.
- The total number of parameters (weights) in the model.

The summary can be created by calling the *summary()* function on the model that returns a string that in turn can be printed.

Code:

```
import pickle  
  
model = pickle.load(open('playerPosition.pkl', 'rb'))  
  
model.summary()
```

Output:

Layer (type)	Output Shape	Param #
--------------	--------------	---------

=====

dense_1 (Dense)	(None, 15)	450
-----------------	------------	-----

dense_2 (Dense)	(None, 15)	240
-----------------	------------	-----

dense_3 (Dense)	(None, 1)	16
-----------------	-----------	----

=====

Total params: 706

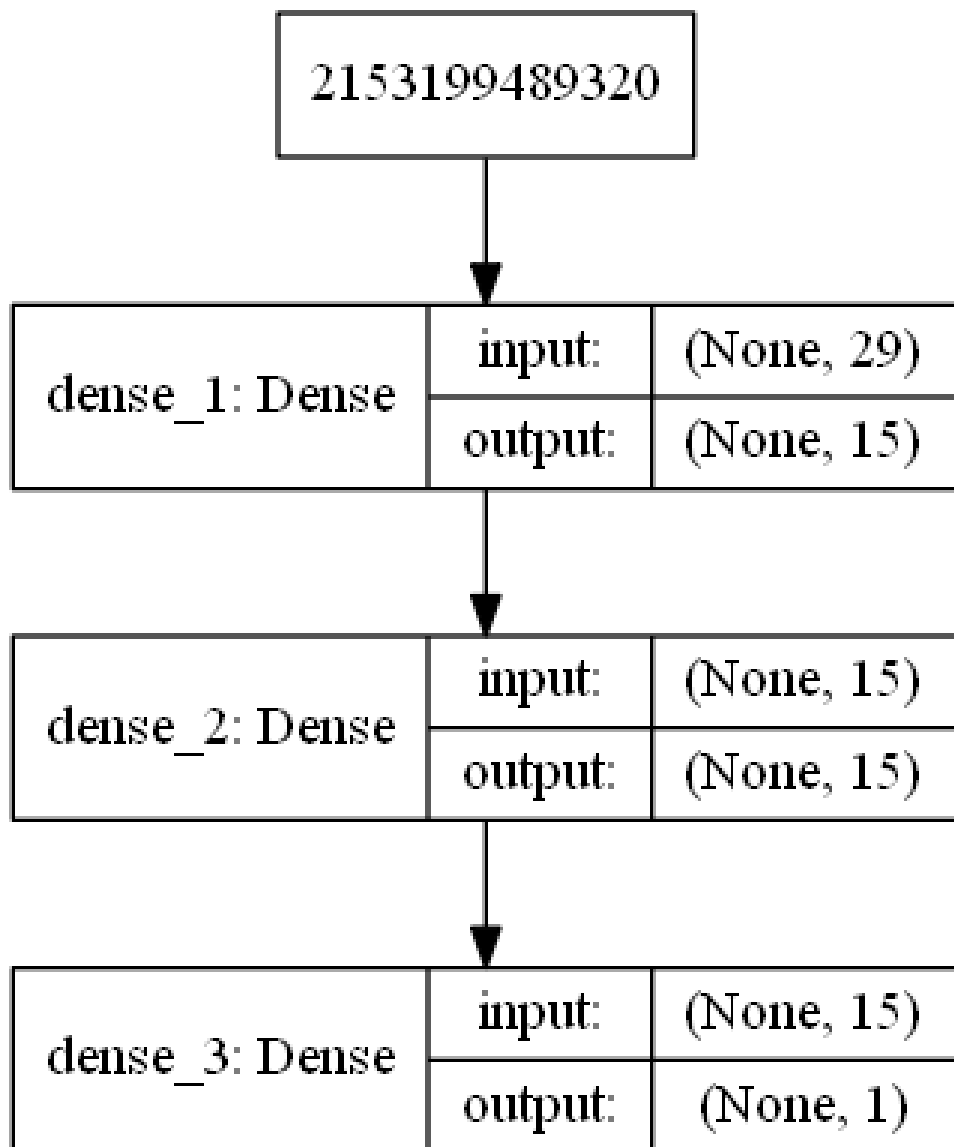
Trainable params: 706

Non-trainable params: 0

Visualize Model

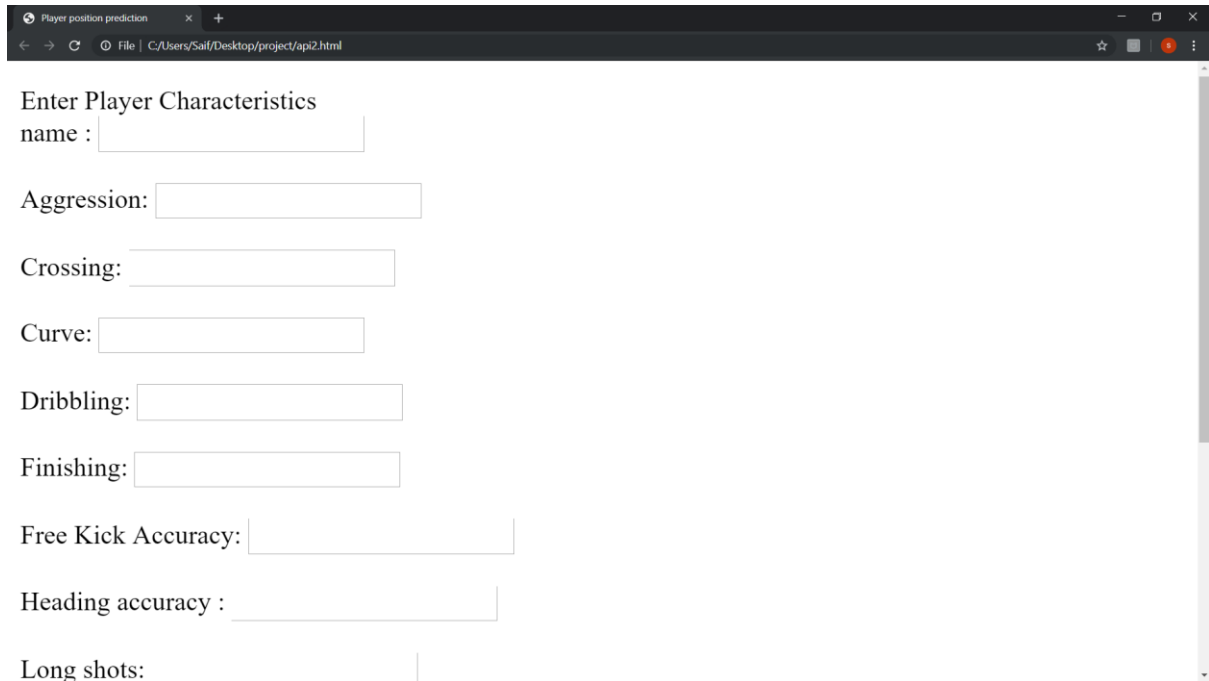
The summary is useful for simple models, but can be confusing for models that have multiple inputs or outputs.

Keras also provides a function to create a plot of the network neural network graph that can make more complex models easier to understand.



3.4 USER INTERFACE SCREENS:

WEB API:



Player position prediction

File | C:/Users/Sail/Desktop/project/api2.html

Enter Player Characteristics

name :

Aggression:

Crossing:

Curve:

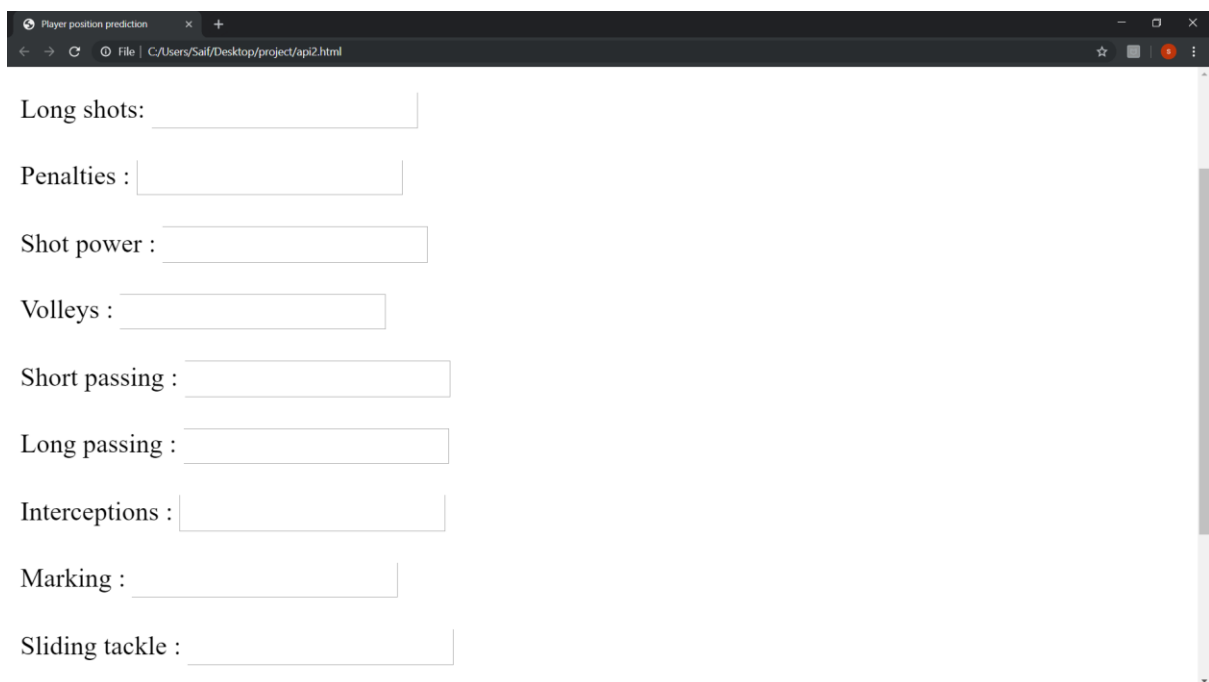
Dribbling:

Finishing:

Free Kick Accuracy:

Heading accuracy :

Long shots:



Player position prediction

File | C:/Users/Sail/Desktop/project/api2.html

Long shots:

Penalties :

Shot power :

Volleys :

Short passing :

Long passing :

Interceptions :

Marking :

Sliding tackle :

Player position prediction

File | C:/Users/Sail/Desktop/project/api2.html

Standing tackle :

Strength :

Vision :

Acceleration :

Agility :

Reactions :

Stamina :

Balance :

Ball control :

Player position prediction

File | C:/Users/Sail/Desktop/project/api2.html

Reactions :

Stamina :

Balance :

Ball control :

Composure :

Jumping :

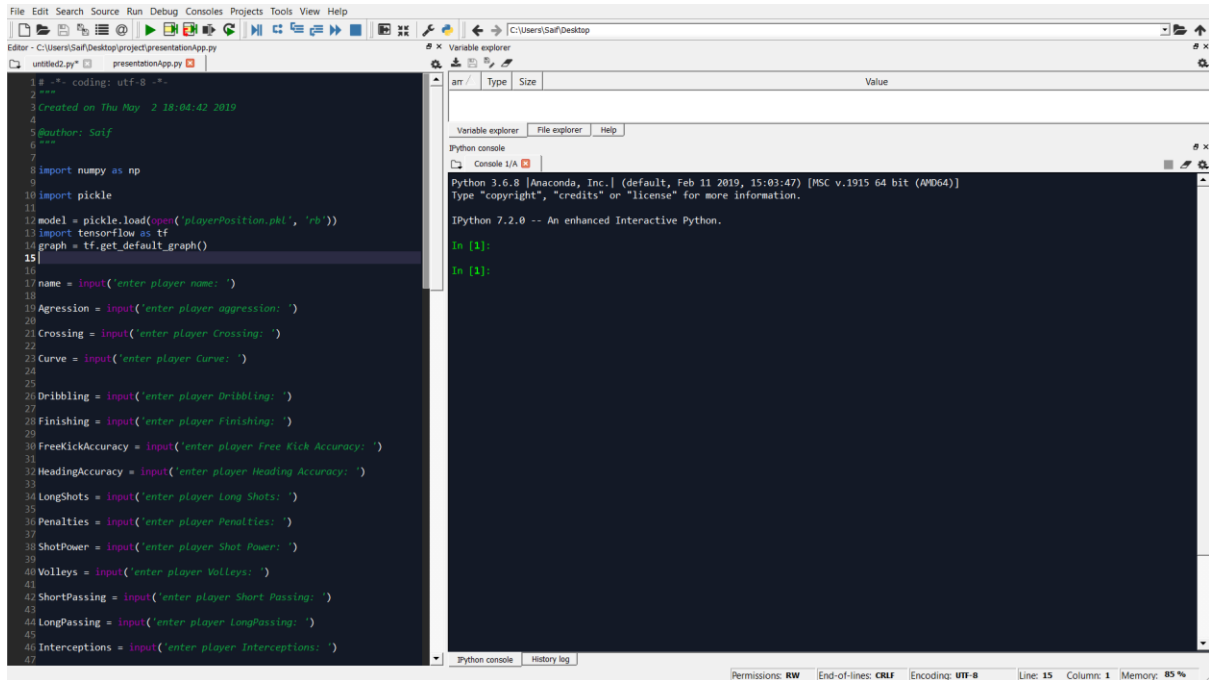
Sprint :

Positioning :

Player should be placed in {{p}}

PROVIDING INPUT TO THE MACHINE:

1. Running script



The screenshot shows an IDE with a Python script on the left and a console on the right. The script is a Jupyter Notebook cell with the following code:

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Thu May 2 18:04:42 2019
4
5 @author: Saif
6 """
7
8 import numpy as np
9
10 import pickle
11
12 model = pickle.load(open('playerPosition.pkl', 'rb'))
13 import tensorflow as tf
14 graph = tf.get_default_graph()
15
16
17 name = input('enter player name: ')
18
19 Aggression = input('enter player aggression: ')
20
21 Crossing = input('enter player Crossing: ')
22
23 Curve = input('enter player Curve: ')
24
25
26 Dribbling = input('enter player Dribbling: ')
27
28 Finishing = input('enter player Finishing: ')
29
30 FreeKickAccuracy = input('enter player Free Kick Accuracy: ')
31
32 HeadingAccuracy = input('enter player Heading Accuracy: ')
33
34 LongShots = input('enter player Long Shots: ')
35
36 Penalties = input('enter player Penalties: ')
37
38 ShotPower = input('enter player Shot Power: ')
39
40 Volleys = input('enter player Volleys: ')
41
42 ShortPassing = input('enter player Short Passing: ')
43
44 LongPassing = input('enter player LongPassing: ')
45
46 Interceptions = input('enter player Interceptions: ')
47
```

The console on the right shows the output of the script, which is a prompt for the user to enter values for each variable. The console also shows the Python version and the IPython version.

2. Entering values



The screenshot shows a Jupyter Notebook console with the following input and output:

```
In [1]:
enter player name: SAIF MATHUR

enter player aggression: 89

enter player Crossing: 60

enter player Curve: 75

enter player Dribbling: 89

enter player Finishing: 87

enter player Free Kick Accuracy: 97

enter player Heading Accuracy: 78

enter player Long Shots: 90

enter player Penalties:
```

3. Entering Parameters

```
enter player Heading Accuracy: 78
enter player Long Shots: 90
enter player Penalties: 78
enter player Shot Power: 78
enter player Volleys: 89
enter player Short Passing: 100
enter player LongPassing: 50
enter player Interceptions: 67
enter player Marking: 56
enter player Sliding Tackle: 556
enter player Standing Tackle: 67
enter player Strength: 99
enter player Vision: 79
enter player Acceleration: 98
enter player Agility: 77
enter player Reactions: 65
```

4. Getting Prediction

```
In [4]: print('Place ' + str(name) + ' In ' + str(prediction))
Place SAIF MATHUR In Attack
```


3.7 TEST CASE:

Test Case ID	1	Test Case Description	Test Input of Attributes of Player		
Created By	Saif	Reviewed By	User	Version	1.0

QA Tester's Log Review comments from Bill incorporate in version 2.1

Tester's Name	Saif	Date Tested	9-5-2019	Test Case (Pass/Fail/Not Executed)	Pass
----------------------	------	--------------------	----------	---	-------------

S #	Prerequisites:
1	Access to Web browser
2	
3	
4	

S #	Test Data
1	Attribute List
2	
3	
4	

Test Scenario Verify on entering values of attributes to be higher than 100 , the user cannot enter

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Enter Value higher than 100	Site should open	As Expected	Pass
2	Run via Web API	The site should load	Not As Expected	Fail
3	Click Predict	User should get prediction	Not As Expected	Fail
4				

4.0 DRAWBACKS AND LIMITATIONS

Drawbacks:

- **The model requires a fairly decent amount of processing power or graphics processing power in order to execute.**
- **The model will become slow if there is any asynchronization with the multiple scripts.**
- **The model is built to classify a binary outcome.**
- **The model has issues with Flask Production and will not allow to load certain methods.**

Limitations:

This research, however, is subject to several limitations

- Limited access to FIFA data, therefore less training data
Affecting model prediction accuracy.
- Due to Time constraints the model is still in raw stages.
- Web API is not loading due to incompatibility.
- Limited computing power, resulting in delayed execution.
- User interface not well defined.
- Noise in data affecting accuracy.

5.0 PROPOSED ENHANCEMENTS

Enhancements & modification ideas:

- Adding data manually to increase the size of training set to increase accuracy of model.
- Designing an interface where all the scripts are auto generated.
- Making an eye catching web API for young gamers.
- To apply regularization on the model.
- Removal of non - relevant attributes for a faster prediction .
- Building an android application and syncing it with FIFA database for ease of access.
- Applying this model to different player strategy games.
- Adding face detection for real time players.
- Adding face recognition via web cam.
- Making predictions as a background service for in game suggestions.

6.1 CONCLUSION:

Classify the strategy of a game plan by an individual player

i.e. whether the player should be placed in the ‘Attack Position’ or ‘Defence Position’.

The classification was done by a Linear Classifier at first, on observing mediocre performance the model was trained on an Artificial Neural Network using Keras libraries with a TensorFlow backed. The data needed cleaning after which I was left with 29 features of a player on which predictions are made about the player`s positioning.

A Flask framework was used to mimic the production usage with the help of an API.

The model is trained to give a binary outcome and will perform better with more training data. If these programs can be implemented, we will surely help a lot of professional gamers and see an increase in activity in this industry.

7.1 BIBLIOGRAPHY:

1. Data Gathering & Cleaning

https://www.fifaindex.com/players/top/fifa18_278/

<https://www.fifaindex.com/players/>

https://www.fifaindex.com/players/fifa18wc_271/

<https://www.kaggle.com/>

2.Cleaning:

<https://stackoverflow.com/>

https://www.tutorialspoint.com/flask/flask_sending_form_data_to_template.htm

3. Error Handling:

<https://github.com/tankala/ai-examples>

<https://stackoverflow.com/questions/10219486/flask-post-request-is-causing-server-to-crash>

<https://stackoverflow.com/questions/33396064/flask-template-not-found>

https://github.com/christabor/flask_jsondash/issues/128

<https://www.quora.com/How-do-I-get-input-values-from-HTML-page-to-Python-in-a-server>

<https://pythonspot.com/flask-with-static-html-files/>

4.Deploying Keras:

<https://www.youtube.com/watch?v=RkmfXz304ck>

<https://keras.io/layers/core/>

<https://github.com/keras-team/keras/issues/10528>

<https://www.kaggle.com/questions-and-answers/54044>

<https://github.com/keras-team/keras/issues/5008>

<https://github.com/keras-team/keras/issues/2603>

5. Research Papers Referred

Source:

<https://ieeexplore.ieee.org/Xplore/home.jsp>

Control strategies for multiplayer target-attacker-defender differential games with double integrator dynamics

<https://ieeexplore.ieee.org/document/8263864>

An Empirical Comparison of Non-adaptive, Adaptive and Self-Adaptive Co-evolution for Evolving Artificial Neural Network Game Players

<https://ieeexplore.ieee.org/document/4017793>

Verifying adaptation of neuro-controlled game opponent by cross validation under supervised and unsupervised player modelling

<https://ieeexplore.ieee.org/document/5542932>

Control Strategies for Multiplayer Target-Attacker-Defender Differential Games with Double Integrator Dynamics

Mitchell Coon and Dimitra Panagou

Game Player Strategy Pattern Recognition and How UCT Algorithms Apply Pre-knowledge of Player's Strategy to Improve Opponent AI

ISBN: 978-0-7695-3514-2

Publisher: IEEE

[2008 International Conference on Computational Intelligence for Modelling Control & Automation](#)

Optimizing NBA Player Selection Strategies Based on Salary and Statistics Analysis

Ramya Nagarajan , Lin Li

2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech)