PREDECTION OF FOOTBALL PLAYERS PERFORMANCE USING MACHINE LEARNING AND DEEP LEARNING ALGORITHMS

Submitted in partial fulfilment of the requirements for the award of

Bachelor of Engineering in

Computer Science and

Engineering By

DASARI HARSHAVARDHAN (Reg. No. 38110119)
MUDUNURI LEELA SAI RUSHENDRA VARMA (Reg. No. 38110331)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING
SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
JE PPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119, TAMILNADU

MARCH 2022

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC
JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119
www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Professional Training Report is the Bonafide work Of **DASARI.HARSHAVARDHAN (Reg no:38110119) and MUDUNURI LEELA SAI RUSHENDRA VARMA (Reg no: 38110331)** who out the project entitled **"PREDICTION OF FOOTBALL PLAYERS PERFORMANCE USING MACHINE LEARNING ALGORITHMS"** under our supervision from October 2022 to March 2022.

Internal Guide
Dr. J. REFONAA

Head of the Department
DR. VIGNESHWARI S, M.E., Ph. D.,

Submitted for Viva voce Examination held on_____

Internal Examiner **External Examiner**

DECLARATION

we, **DASARI.HARSHAVARDHAN**(38110119) and **MUDUNURI LEELA SAI RUSHENDRA VARMA** (38110331) hereby declare that the Professional Training Report entitled "**PREDECTION OF FOOTBALL PLAYERS PERFORMANCE USING MACHINE LEARNING ALGORITHMS"** done by me under the guidance of Dr. J. REFONAA(Internal), is submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

.

DATE:

PLACE: CHENNAI **SIGNATURE OF THE CANDIDATE**

ACKNOWLEDGEMENT

I am pleased to acknowledge the sincere thanks to Board of Management of SATHYABAMA for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. SASIKALA M.E., Ph.D.,** Dean, School of Computing and **Dr. S. VIGNESHWARI M.E., Ph.D..**, and **Dr. L. Lakshmanan M.E., Ph.D.,** Head of the Department, Dept. of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express sincere and deep sense of gratitude to the 1Project Guide **Mrs. J. Refonaa,** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of the project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

In the game of football (soccer), the evaluation of players for transfer, scouting, squad formation and strategic planning is important. However, due to the vast pool of grassroots level player, short career span, differing performance throughout the individual's career, differing play conditions, positions and varying club budgets, it becomes difficult to identify the individual player's performance value altogether. Our Player Performance Prediction system aims at solving this complex problem analytically and involves learning from various attributes and skills of a football player. It considers the skill set values of the football player and predicts the performance value, which depicts the scope of improvement and the capability of the player. The objective of this system is to help the coaches and team management at the grassroots as well as higher levels to identify the future prospects in the game of football without being biased to subjective conditions like club budget, competitiveness in the league, and importance of the player in the team or region. Our system is based on a data-driven approach and we train our models to generate an appropriate holistic relationship between the players' attributes values, market value and performance value to be predicted. These values are dependent on the position that the football player plays in and the skills they possess.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Football, also known as soccer to the western part of the world, is a team based sport played between two teams, each consisting of eleven players with a spherical ball. This sport is played in over 200 countries and in almost all weather conditions such as snow, rain, summer, etc. Football is governed by FIFA (Fédération Internationale de Football Association) as the highest body and further divides into various other bodies depending on the region and nationality. The competitiveness of the game varies from region to region based on the participation of the people, media coverage, and club budget. This, in turn, brings varying differences in the level of players and also fluctuates the market value and the skill level based on region, hype generated by the media, competitiveness of the league in which they play and their experience. The bigger the role the player plays in his team, the more likely they may be valued in the market like being the finest penalty taker, or spot free kick specialist, or other roles such as being a playmaker, chance creator, having excellent speed, etc. In India, despite the decrease in the youth participation in sports, particularly in the past few decades, the industry is putting in various means and efforts to improve the sports environments in the form of grassroots level programs, facilities, tournaments, coaching, public awareness, scholarships, etc. The problem however lies in the fact that it's difficult to search, analyze and coach the players in every part of the country; especially in rural India which consists of 70 percent of the 1.25 billion people approximately. To overcome this difficulty the clubs recruit scouts of vast experience and regional understanding to identify players. The AIFF is trying to improve the situation by collaborating with various clubs and companies that make it possible to teach the coaches who may be inexperienced by bringing in connecting sessions with the experienced ones, hosting various tournaments at school, city, district, state level, establishing football academies and community initiatives. The

proposed model is aimed specifically at the grassroot level players of India, further scaling to other soccer leagues. The system is trained as per the in-game values of the 2017 version of EA Sports FIFA. The reason for choosing values based on a game is that it seemed to be the only source for a reliable, near accurate and open form of data available for football players spanning across several leagues. Moreover, the very nature of the game being a team based sport makes it difficult to analyze the players due to their dependencies on the skill-set of other team members, varying positions, formations, club budget, competitiveness in the league and injuries across their career span. Our model is designed to estimate the performance value of the player based on the attributes and skill sets that the player possesses. Coaches can then take advantage of this performance value and train the player, reshuffle the team, recruit, and loan or sell the player. Another value added to this process is the market value of the player obtained through the performance value of the player. However, there will be an approximate deviation in that value by a certain amount due to irregularities in the demand for a particular position, club budget, contract period, injuries and current on-field performance.

1.1 PROBLEM DEFINITION

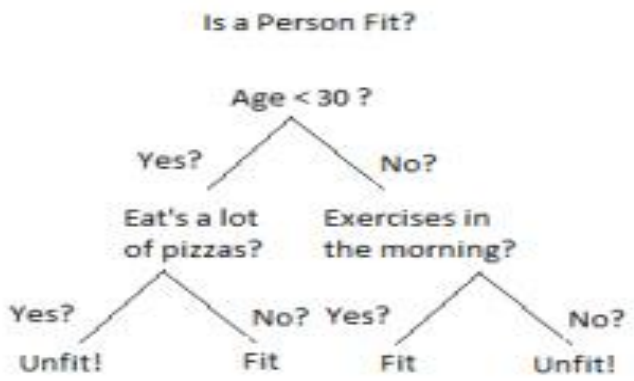
In the game of football (soccer), the evaluation of players for transfer, scouting, squad formation and strategic planning is important. However, due to the vast pool of grassroots level player, short career span, differing performance throughout the individual’s career, differing play conditions, positions and varying club budgets, it becomes difficult to identify the individual player's performance value altogether. Our Player Performance Prediction system aims at solving this complex problem analytically and involves learning from various attributes and skills of a football player. The objective of this system is to help the coaches and team management at the grassroots as well as higher levels to identify the future prospects in the game of football without being biased to

subjective conditions like club budget, competitiveness in the league, and importance of the player in the team or region.

Using machine learning algorithms predicting the results of a football match, we obtained and create set of features, thus developing with high accurate predictive method using machine learning techniques.

1.2. DECISION TREE

Decision Trees are a type of Supervised Machine Learning (that is you explain what the input is and what the corresponding output is in the training data) where the data is continuously split according to a certain parameter. The tree can be explained by two entities, namely decision nodes and leaves. The leaves are the decisions or the final outcomes. And the decision nodes are where the data is split.



An example of a decision tree can be explained using above binary tree. Let's say you want to predict whether a person is fit given their information like age, eating habit, and physical activity, etc. The decision nodes here are questions like 'What's the age?', 'Does he exercise?', 'Does he eat a lot of pizzas'? And the leaves, which are outcomes like either 'fit', or 'unfit'. In this case this was a binary classification problem (a yes no type problem).

There are two main types of Decision Trees:

Classification trees (Yes/No types)

What we've seen above is an example of classification tree, where the outcome was a variable like 'fit' or 'unfit'. Here the decision variable is Categorical.

Regression trees (Continuous data types)

Here the decision or the outcome variable is Continuous, e.g. a number like 123.

Working

Now that we know what a Decision Tree is, we'll see how it works internally. There are many algorithms out there which construct Decision Trees, but one of the best is called as ID3 Algorithm. ID3 Stands for Iterative Dichotomiser 3.

Before discussing the ID3 algorithm, we'll go through few definitions.

1.3.BOOST ALGORITHM

Boosting Algorithms combines each weak learner to create one strong prediction rule. To identify the weak rule, there is a base Learning algorithm (Machine Learning). Whenever the Base algorithm is applied, it creates new prediction rules using the iteration process. After some iteration, it combines all weak rules to create one single prediction rule.

To choose the right distribution follows the below-mentioned steps:

Step 1: The base Learning algorithm combines each distribution and applies equal weight to each distribution.

Step 2: If any prediction occurs during the first base learning algorithm, then we pay high attention to that prediction error.

Step 3: Repeat step 2 until the limit of the Base Learning algorithm has been reached or high accuracy.

Step 4: Finally, it combines all the weak learner to create one strong prediction rule.

CHAPTER-2

LITERATURE SURVEY

Literature survey is the most important step in software development process. Before developing the tool it is necessary to determine the time factor, economy and company strength. Once these things are satisfied, then the next step is to determine which operating system and language can be used for developing the tool. Once the programmers start building the tool the programmers need lot of external support. This support can be obtained from senior programmers, from book or from websites. Before building the system the above consideration are taken into account for developing the proposed system. The major part of the project development sector considers and fully survey all the required needs for developing the project. For every project Literature survey is the most important sector in software development process. Before developing the tools and the associated designing it is necessary to determine and survey the time factor, resource requirement, man power, economy, and company strength. Once these things are satisfied and fully surveyed, then the next step is to determine about the software specifications in the respective system such as what type of operating system the project would require, and what are all the necessary software are needed to proceed with the next step such as developing the tools, and the associated operations.

2.1.Predicting student performance using ID3 and C4.5 classification algorithms

An educational institution needs to have an approximate prior knowledge of enrolled students to predict their performance in future academics. This helps them to identify promising students and also provides them an opportunity to pay attention to and improve those who would probably get lower grades. As a solution, we have developed a system which can predict the performance of students from their previous performances using concepts of data mining techniques under Classification. We have analyzed the data set containing information about students, such as gender, marks scored in the board examinations of classes X and XII, marks and rank in entrance examinations and results in first year of the previous batch of students. By applying the ID3 (Iterative Dichotomiser 3) and C4.5 classification algorithms on this data, we have predicted the general and individual performance of freshly admitted students in future examinations.

2.2.Chess game result prediction system

In this project we train World Chess Federation (FIDE) rating systems using a training dataset of a recent eleven-year period with games from 2000 chess players. We will then use our system to predict the outcome of chess games played by the same players in the following half year. Accuracy between predicted results and actual game results is the primary indicator of whether our approach is a practical chess rating system.

2.3.Predicting sports events from past results: Towards effective betting on football matches

A system for predicting the results of football matches that beats the bookmakers' odds is presented. The predictions for the matches are based on previous results of the teams involved.

2.4.Football result prediction with Bayesian Network in Spanish league Barcelona team

The problem of modeling football data has become increasingly popular in the last few years and many different models have been proposed with the aim of estimating the characteristics that bring a team to lose or win a game, or to predict the score of a particular match. We propose a Bayesian Network (BN) to predict results of football matches. During the last decade, Bayesian networks (and probabilistic graphical models in general) have become very popular in artificial intelligence. In this paper, we look at the performance of a BN in the area of predicting the result of football matches involving Barcelona FC. The period under study was the 2008-2009 season in Spanish football league and we test its performance. We get necessary information about football states from valid websites. This BN is used for prediction of football results in future matches.

CHAPTER 3

SYSTEM REQUIREMENTS

3.1.HARDWARE REQUIREMENTS

System : Pentium i3 Processor

Hard Disk : 500 GB.

Monitor : 15'' LED

Input Devices : Keyboard, Mouse

Ram : 2 GB

3.2.SOFTWARE REQUIREMENTS

Operating system : Windows 10

Coding Language : Python

3.3. LANGUAGE SPECIFICATION

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985-1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). This tutorial gives enough understanding on Python programming language.

3.4. HISTORY OF PYTHON

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

3.5. APPLICATION OF PYTHON

- **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
- **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** − Python provides interfaces to all major commercial databases.

- **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable** − Python provides a better structure and support for large programs than shell scripting.

3.6. FEATURES OF PYTHON

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

3.7 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

The feasibility study investigates the problem and the information needs of the stakeholders. It seeks to determine the resources required to provide an information systems solution, the cost and benefits of such a solution, and the feasibility of such a solution.

The goal of the feasibility study is to consider alternative information systems solutions, evaluate their feasibility, and propose the alternative most

suitable to the organization. The feasibility of a proposed solution is evaluated in terms of its components.

3.7.1 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

3.7.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

3.7.3 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity.

CHAPTER 4
SYSTEM ANALYSIS

4.1 PURPOSE

The purpose of this document is to help the coaches and team management at the grassroots as well as higher levels to identify the future prospects in the game of football without being biased to subjective conditions like club budget, competitiveness in the league, and importance of the player in the team or region. In detail, this document will provide a general description of our project, including user requirements, product perspective, and overview of requirements, general constraints. In addition, it will also provide the specific requirements and functionality needed for this project - such as interface, functional requirements and performance requirements.

4.2 SCOPE

The scope of this SRS document persists for the entire life cycle of the project. This document defines the final state of the software requirements agreed upon by the customers and designers. Finally at the end of the project execution all the functionalities may be traceable from the SRS to the product. The document describes the functionality, performance, constraints, interface and reliability for the entire life cycle of the project.

4.3 EXISTING SYSTEM

We analyze two approaches used in football result prediction in existing systems, these include; statistical and machine learning approaches. For statistical approach, we analyze Hidden Markov Process Model and Ordered Probit Regression model. Also, a detailed analysis of machine learning approach by has been done. Football predictive system is made up of two main components, namely: feature sets/ data sets and implementation techniques. We therefore analyze the data sets and the techniques used in the implementation of existing system.

4.4 DISADVANATGES OF EXISITNG SYSTEM

In this system, it is difficult to identify which system outperforms the others. These variations include: number of goals prediction, win-draw-loss prediction, propensity to score or concede goals, total points earned in a season, etc.

4.5 PROPOSED SYSTEM

Our Player Performance Prediction system aims at solving this complex problem analytically and involves learning from various attributes and skills of a football player. It considers the skill set values of the football player and predicts the performance value, which depicts the scope of improvement and the capability of the player. In the current research the statistical model is proposed to predict the stats of the football player based on previous session data by considering various aspects of the game. Using machine learning algorithms predicting the results of a football match, we obtained and create set of features, thus developing with high accurate predictive method using machine learning techniques. Our system is based on a data-driven approach and we train our models to generate an appropriate holistic relationship between the players' attributes values, market value and performance value to be predicted. These values are dependent on the position that the football player plays in and the skills they possess.

4.6 ADVANTAGES OF PROPOSED SYSTEM

The major impact of this system would be an advantage in identifying the grass-root level talented players who fail to receive exposure as compared to the other renowned football players.

SYSTEM DESIGN

INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.

- Methods for preparing input validations and steps to follow when error occur.

OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

The output form of an information system should accomplish one or more of the following objectives.

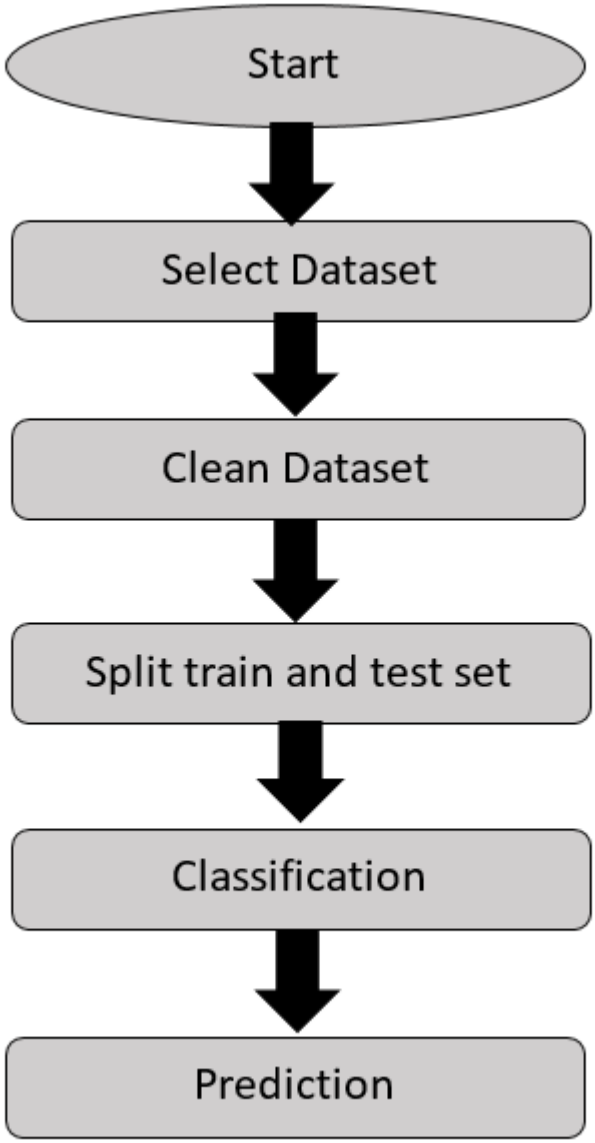
- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action

DATA FLOW DIAGRAM:

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are

the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.

3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.



UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of

two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

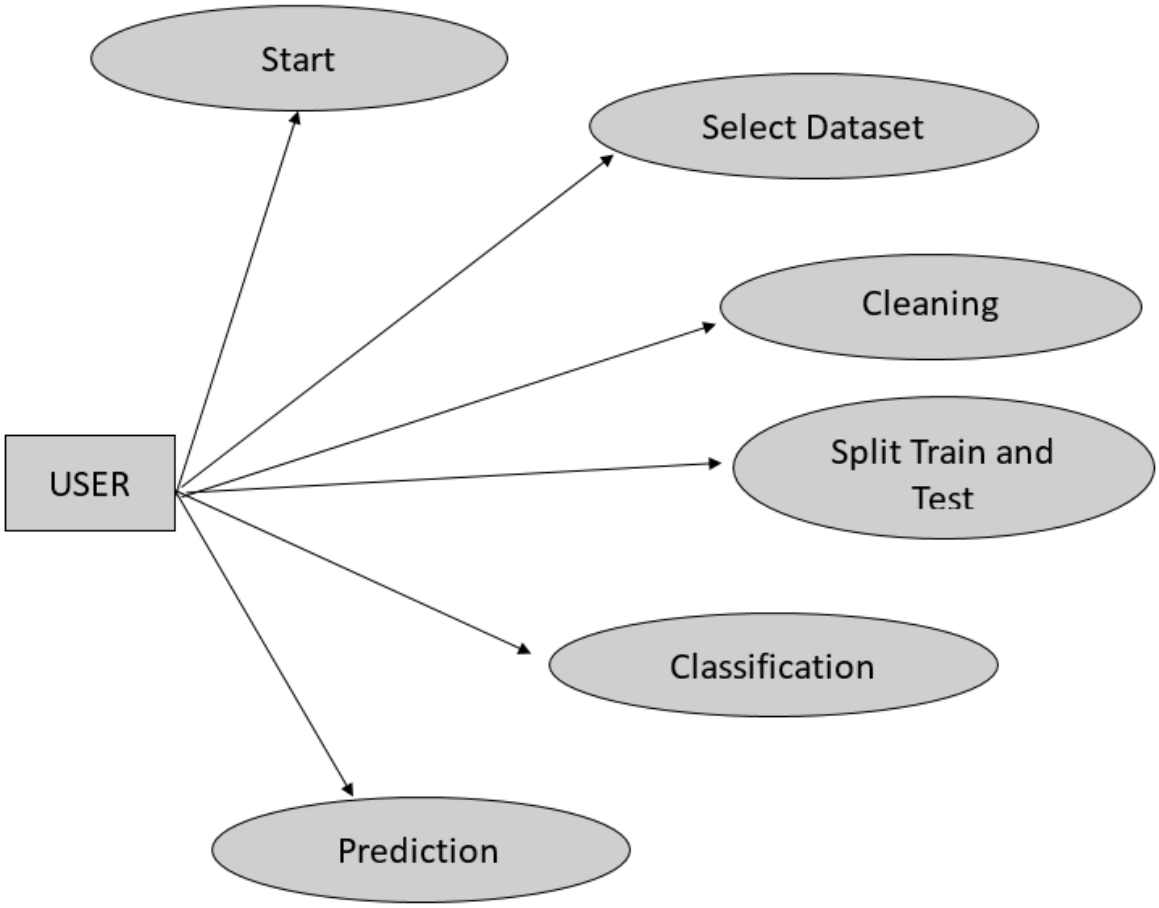
The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type

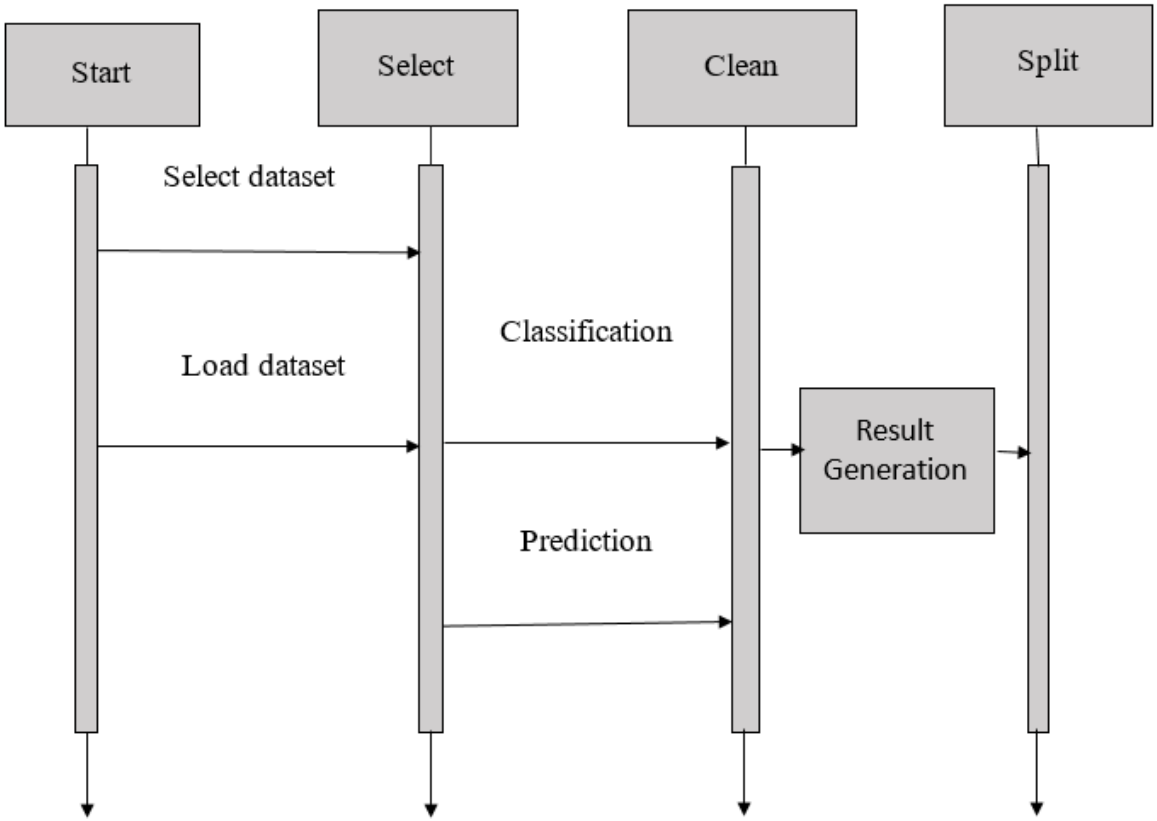
of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



SEQUENCE DIAGRAM:

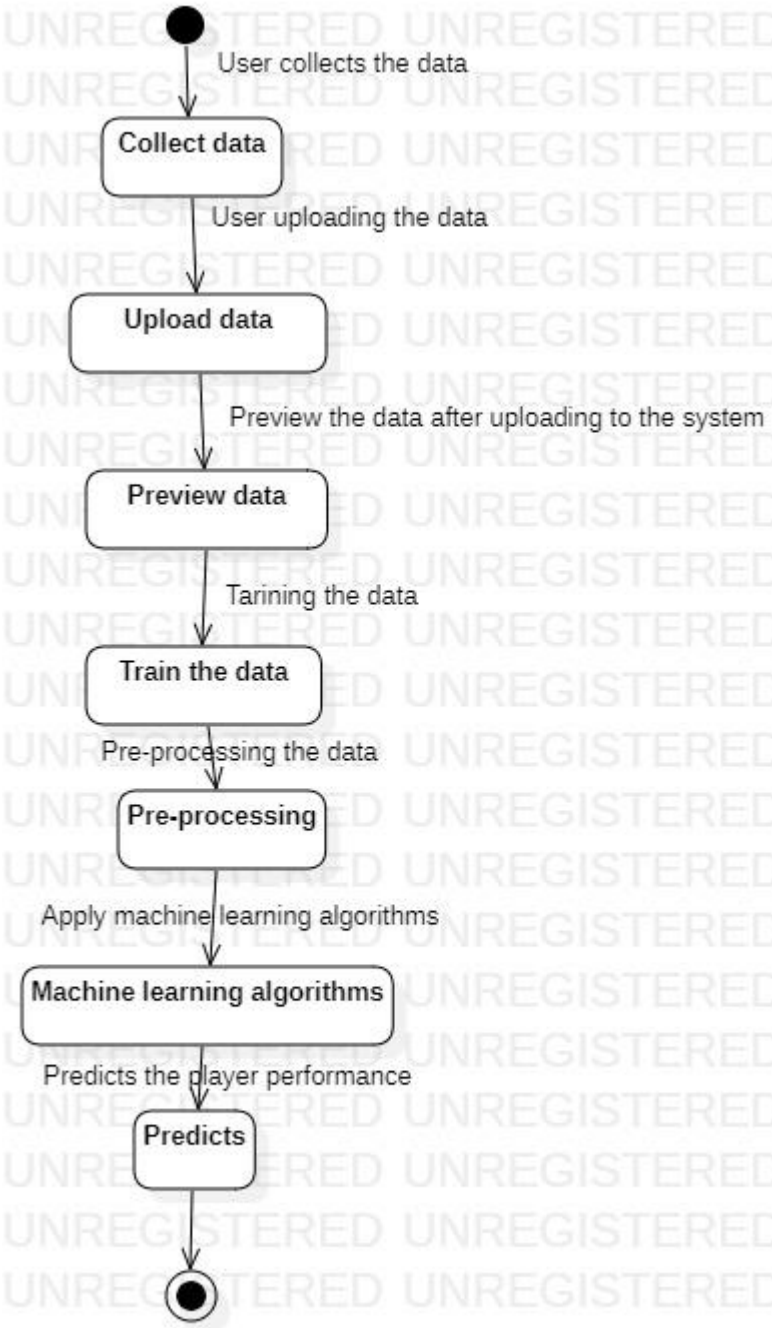
A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in

what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.



CHAPTER 5

MODULES DESCRIPTION

MODULES

- Data Collection
- Pre-Processing
- Apply Machine Learning Techniques
- Prediction Module

MODULE DESCRIPTION

5.1 Data Collection Module

The data used for this work was collected from Kaggle. The following procedures were adopted at this stage of the research: Data Cleaning, Data Selection, Data Transformation and Data Mining.

5.2 Pre-Processing Module

Data preprocessing is a process in which that is actual use for converting the basic data into the clean data set. It is the step in which the data transform or an encode to the state that the machine can be easily parse. The major task of data preprocessing in learning process is to remove the unwanted data and filling the missed value. So that it help to machine can be trained easily.

5.3 Apply Machine Learning Techniques

In this section, we will present an overview of popular supervised Machine Learning techniques, for its subsets of classification and regression. Supervised learning is the task of learning a function that maps input data to output data based on example input-to-output pairs. Classification happens when the output is a category, whereas regression happens when the output is a continuous number. In our case, we want to predict the outcome category (home win/draw/away win) or the number of goals scored by a team (continuous

number), so we are only interested in the supervised learning landscape of Machine Learning.

5.4 Prediction Module

Our Player Performance Prediction system aims at solving this complex problem analytically and involves learning from various attributes and skills of a football player. It considers the skill set values of the football player and predicts the performance value, which depicts the scope of improvement and the capability of the player. In the current research the statistical model is proposed to predict the stats of the football player based on previous session data by considering various aspects of the game. Using machine learning algorithms predicting the results of a football match, we obtained and create set of features, thus developing with high accurate predictive method using machine learning techniques.

CHAPTER 6

SYSTEM IMPLEMENTATION

6.1 SYSTEM ARCHITECTURE

Describing the overall features of the software is concerned with defining the requirements and establishing the high level of the system. During architectural design, the various web pages and their interconnections are identified and designed. The major software components are identified and decomposed into processing modules and conceptual data structures and the interconnections among the modules are identified. The following modules are identified in the proposed system.

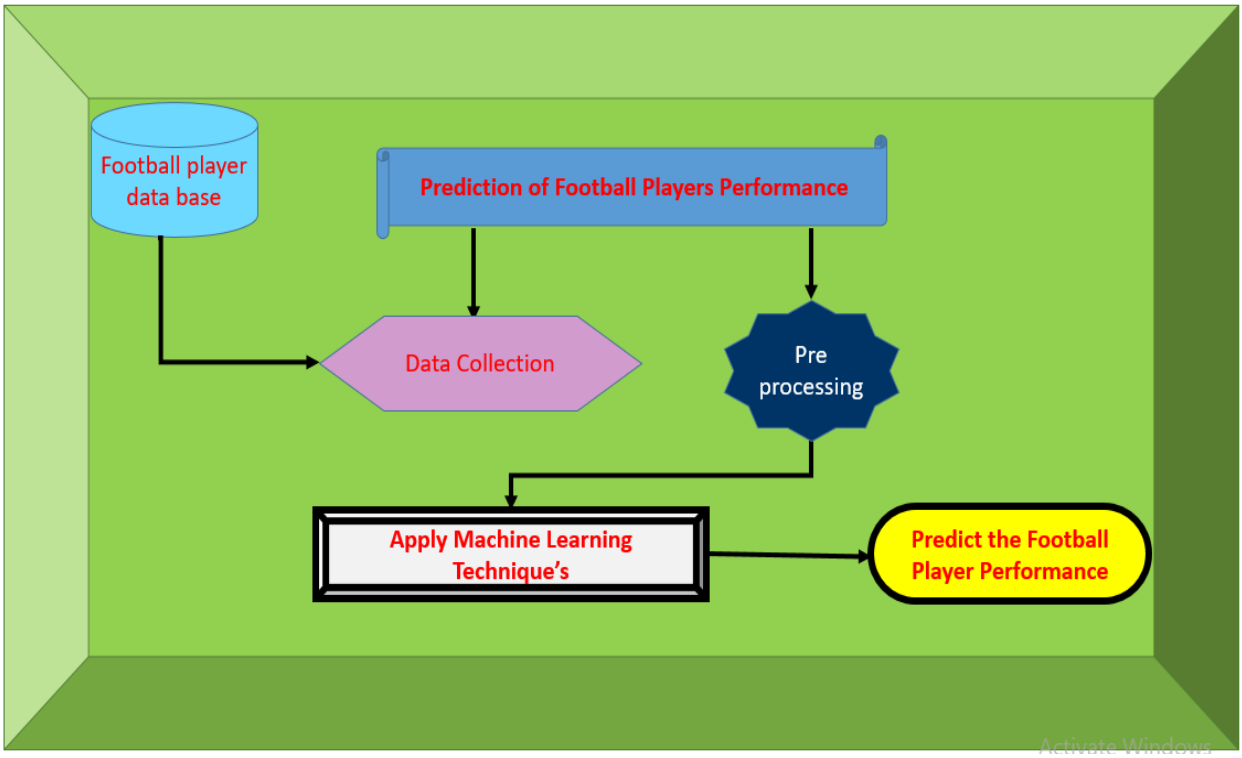


Fig. 6.1 SYSTEM ARCHITECTURE

The above architecture describes the work structure of the system.

- From the football player database and prediction of football players performance the data is collected and pre- processing is applied to the data.

- After completion of pre-processing stage then by applying machine learning algorithms it will predict the football player performance.

6.2 Problem Statement:

In the game of football (soccer), the evaluation of players for transfer, scouting, squad formation and strategic planning is important. However, due to the vast pool of grassroots level player, short career span, differing performance throughout the individual's career, differing play conditions, positions and varying club budgets, it becomes difficult to identify the individual player's performance value altogether. Our Player Performance Prediction system aims at solving this complex problem analytically and involves learning from various attributes and skills of a football player. The objective of this system is to help the coaches and team management at the grassroots as well as higher levels to identify the future prospects in the game of football without being biased to subjective conditions like club budget, competitiveness in the league, and importance of the player in the team or region.Using machine learning algorithms predicting the results of a football match, we obtained and create set of features, thus developing with high accurate predictive method using machine learning techniques.

SYSTEM TESTING

Test plan

Software testing is the process of evaluation a software item to detect differences between given input and expected output. Also to assess the feature of a software item. Testing assesses the quality of the product. Software testing is a process that should be done during the development process. In other words software testing is a verification and validation process.

Verification

Verification is the process to make sure the product satisfies the conditions imposed at the start of the development phase. In other words, to make sure the product behaves the way we want it to.

Validation

Validation is the process to make sure the product satisfies the specified requirements at the end of the development phase. In other words, to make sure the product is built as per customer requirements.

Basics of software testing

There are two basics of software testing: black box testing and white box testing.

Black box Testing

Black box testing is a testing technique that ignores the internal mechanism of the system and focuses on the output generated against any input and execution of the system. It is also called functional testing.

White box Testing

White box testing is a testing technique that takes into account the internal mechanism of a system. It is also called structural testing and glass box testing. Black box testing is often used for validation and white box testing is often used for verification.

Types of testing

There are many types of testing like

- Unit Testing
- Integration Testing
- Functional Testing
- System Testing
- Stress Testing
- Performance Testing
- Usability Testing
- Acceptance Testing
- Regression Testing
- Beta Testing

Unit Testing

Unit testing is the testing of an individual unit or group of related units. It falls under the class of white box testing. It is often done by the programmer to test that the unit he/she has implemented is producing expected output against given input.

Integration Testing

Integration testing is testing in which a group of components are combined to produce output. Also, the interaction between software and hardware is tested in integration testing if software and hardware components have any relation. It may fall under both white box testing and black box testing.

Functional Testing

Functional testing is the testing to ensure that the specified functionality required in the system requirements works. It falls under the class of black box testing.

System Testing

System testing is the testing to ensure that by putting the software in different environments (e.g., Operating Systems) it still works. System testing is done with full system implementation and environment. It falls under the class of black box testing.

Stress Testing

Stress testing is the testing to evaluate how system behaves under unfavorable conditions. Testing is conducted at beyond limits of the specifications. It falls under the class of black box testing.

Performance Testing

Performance testing is the testing to assess the speed and effectiveness of the system and to make sure it is generating results within a specified time as in performance requirements. It falls under the class of black box testing.

Usability Testing

Usability testing is performed to the perspective of the client, to evaluate how the GUI is user-friendly? How easily can the client learn? After learning how to use, how proficiently can the client perform? How pleasing is it to use its design? This falls under the class of black box testing.

Acceptance Testing

Acceptance testing is often done by the customer to ensure that the delivered product meets the requirements and works as the customer expected. It falls under the class of black box testing.

Regression Testing

Regression testing is the testing after modification of a system, component, or a group of related units to ensure that the modification is working correctly and is not damaging or imposing other modules to produce unexpected results. It falls under the class of black box testing

REQUIREMENT ANALYSIS

Requirement analysis, also called requirement engineering, is the process of determining user expectations for a new modified product. It encompasses the tasks that determine the need for analysing, documenting, validating and managing software or system requirements. The requirements should be documentable, actionable, measurable, testable and traceable related to identified business needs or opportunities and define to a level of detail, sufficient for system design.

FUNCTIONAL REQUIREMENTS

It is a technical specification requirement for the software products. It is the first step in the requirement analysis process which lists the requirements of particular software systems including functional, performance and security

requirements. The function of the system depends mainly on the quality hardware used to run the software with given functionality.

Usability

It specifies how easy the system must be use. It is easy to ask queries in any format which is short or long, porter stemming algorithm stimulates the desired response for user.

Robustness

It refers to a program that performs well not only under ordinary conditions but also under unusual conditions. It is the ability of the user to cope with errors for irrelevant queries during execution.

Security

The state of providing protected access to resource is security. The system provides good security and unauthorized users cannot access the system there by providing high security.

Reliability

It is the probability of how often the software fails. The measurement is often expressed in MTBF (Mean Time Between Failures). The requirement is needed in order to ensure that the processes work correctly and completely without being aborted. It can handle any load and survive and survive and even capable of working around any failure.

Compatibility

It is supported by version above all web browsers. Using any web servers like localhost makes the system real-time experience.

Flexibility

The flexibility of the project is provided in such a way that is has the ability to run on different environments being executed by different users.

Safety

Safety is a measure taken to prevent trouble. Every query is processed in a secured manner without letting others to know one's personal information.

NON- FUNCTIONAL REQUIREMENTS

Portability

It is the usability of the same software in different environments. The project can be run in any operating system.

Performance

These requirements determine the resources required, time interval, throughput and everything that deals with the performance of the system.

Accuracy

The result of the requesting query is very accurate and high speed of retrieving information. The degree of security provided by the system is high and effective.

Maintainability

Project is simple as further updates can be easily done without affecting its stability. Maintainability basically defines that how easy it is to maintain the system. It means that how easy it is to maintain the system, analyse, change and test the application. Maintainability of this project is simple as further updates can be easily done without affecting its stability.

CHAPTER 7

CONCLUSION

CONCLUSION

In this paper, we have discussed that how our proposed system predicts the performance of the football player. The proposed system is also scalable for predicting the performance of the players for the individual person by data processing. The system is not having complex process to predict the performance of player like the existing system. Proposed system gives genuine and fast result than existing system.

REFERENCES

[1]. Saritha, M. and Milton, R.S., 2020. A probabilistic logic approach to outcome prediction in team games using historical data and domain knowledge. JOURNAL OF AMBIENT INTELLIGENCE AND HUMANIZED COMPUTING.

[2]. Stübinger, J., Mangold, B. and Knoll, J., 2020. Machine Learning in Football Betting: Prediction of Match Results Based on Player Characteristics. Applied Sciences, 10(1), p.46

[3]. Constantinou, A.C., 2019. Dolores: a model that predicts football match outcomes from all over the world. Machine Learning, 108(1), pp.49-75.

[4]. Baboota, R. and Kaur, H., 2019. Predictive analysis and modelling football results using machine learning approach for English Premier League. International Journal of Forecasting, 35(2), pp.741-755.

[5]. Rathan, M., Deepthi, R.N., Anupriya, S. and Vishnu, V., 2018. Football Match Outcome Prediction Using Sentiment Analysis of Twitter Data. International Journal of Advanced Research in Computer Science, 9(Special Issue 3), p.78.

[6]. Danisik, N., Lacko, P. and Farkas, M., 2018, August. Football match prediction using players attributes. In 2018 World Symposium on Digital Intelligence for Systems and Machines (DISA) (pp. 201-206). IEEE.

CHAPTER -8

SourceCode:

```
# basic operations
import numpy as np
# for dataframe manipulations
import pandas as pd
# for data visualizations
import matplotlib.pyplot as plt
import seaborn as sns
# for missing values
import missingno as mno
# for date time manipulation
import datetime
# for interactivity
import ipywidgets as widgets
from ipywidgets import interact
from ipywidgets import interact_manual
# setting up the background style for the plots
plt.style.use('fivethirtyeight')
```

Reading the Data

```
# reading the data and also checking the computation time
%time data = pd.read_csv('data.csv')

# lets also check the shape of the dataset
print(data.shape)
```

```
data.columns
```

Cleaning Data

```
# checking if the data contains any NULL value
# Visualize missing values as a matrix
mno.bar(data.iloc[:, :40],
        color = 'orange',
        sort = 'ascending')
plt.title('Checking Missing Values Heat Map for first half of the data', fontsize = 15)
plt.show()
# Visualize missing values as a matrix
mno.bar(data.iloc[:, 40:])
plt.title('Checking Missing Values Heat Map for second half of the data')
plt.show()
```

Missing Values Imputation

```
# filling the missing value for the continous variables for proper data visualization


data['ShortPassing'].fillna(data['ShortPassing'].mean(), inplace = True)
data['Volleys'].fillna(data['Volleys'].mean(), inplace = True)
data['Dribbling'].fillna(data['Dribbling'].mean(), inplace = True)
data['Curve'].fillna(data['Curve'].mean(), inplace = True)
data['FKAccuracy'].fillna(data['FKAccuracy'], inplace = True)
data['LongPassing'].fillna(data['LongPassing'].mean(), inplace = True)
data['BallControl'].fillna(data['BallControl'].mean(), inplace = True)
data['HeadingAccuracy'].fillna(data['HeadingAccuracy'].mean(), inplace = True)
data['Finishing'].fillna(data['Finishing'].mean(), inplace = True)
```

```
data['Crossing'].fillna(data['Crossing'].mean(), inplace = True)


data['Weight'].fillna('200lbs', inplace = True)
data['Contract Valid Until'].fillna(2019, inplace = True)
data['Height'].fillna("5'11", inplace = True)
data['Loaned From'].fillna('None', inplace = True)
data['Joined'].fillna('Jul 1, 2018', inplace = True)
data['Jersey Number'].fillna(8, inplace = True)
data['Body Type'].fillna('Normal', inplace = True)
data['Position'].fillna('ST', inplace = True)
data['Club'].fillna('No Club', inplace = True)
data['Work Rate'].fillna('Medium/ Medium', inplace = True)
data['Skill Moves'].fillna(data['Skill Moves'].median(), inplace = True)
data['Weak Foot'].fillna(3, inplace = True)
data['Preferred Foot'].fillna('Right', inplace = True)
data['International Reputation'].fillna(1, inplace = True)
data['Wage'].fillna('€200K', inplace = True)
pd.set_option('max_rows', 100)
data.isnull().sum()
# impute with 0 for rest of the columns
data.fillna(0, inplace = True)


# lets check whether the data still has any missing values
data.isnull().sum().sum()
```

Feature Engineering

```
# creating new features by aggregating the features


def defending(data):
```

```
    return int(round((data[['Marking', 'StandingTackle',
                            'SlidingTackle']].mean()).mean()))

def general(data):
    return int(round((data[['HeadingAccuracy', 'Dribbling', 'Curve',
                            'BallControl']].mean()).mean()))

def mental(data):
    return int(round((data[['Aggression', 'Interceptions', 'Positioning',
                            'Vision','Composure']].mean()).mean()))

def passing(data):
    return int(round((data[['Crossing', 'ShortPassing',
                            'LongPassing']].mean()).mean()))

def mobility(data):
    return int(round((data[['Acceleration', 'SprintSpeed',
                            'Agility','Reactions']].mean()).mean()))
def power(data):
    return int(round((data[['Balance', 'Jumping', 'Stamina',
                            'Strength']].mean()).mean()))

def rating(data):
    return int(round((data[['Potential', 'Overall']].mean()).mean()))

def shooting(data):
    return int(round((data[['Finishing', 'Volleys', 'FKAccuracy',
                            'ShotPower','LongShots', 'Penalties']].mean()).mean()))
```

adding these categories to the data

data['Defending'] = data.apply(defending, axis = 1)

data['General'] = data.apply(general, axis = 1)

data['Mental'] = data.apply(mental, axis = 1)

data['Passing'] = data.apply(passing, axis = 1)

data['Mobility'] = data.apply(mobility, axis = 1)

data['Power'] = data.apply(power, axis = 1)

data['Rating'] = data.apply(rating, axis = 1)

data['Shooting'] = data.apply(shooting, axis = 1)

lets check the column names in the data after adding new features

data.columns

Data Visualization

lets check the Distribution of Scores of Different Skills

plt.rcParams['figure.figsize'] = (18, 12)

plt.subplot(2, 4, 1)

sns.distplot(data['Defending'], color = 'red')

plt.grid()

plt.subplot(2, 4, 2)

sns.distplot(data['General'], color = 'black')

plt.grid()

plt.subplot(2, 4, 3)

sns.distplot(data['Mental'], color = 'red')

plt.grid()

```
plt.subplot(2, 4, 4)
sns.distplot(data['Passing'], color = 'black')
plt.grid()

plt.subplot(2, 4, 5)
sns.distplot(data['Mobility'], color = 'red')
plt.grid()

plt.subplot(2, 4, 6)
sns.distplot(data['Power'], color = 'black')
plt.grid()

plt.subplot(2, 4, 7)
sns.distplot(data['Shooting'], color = 'red')
plt.grid()

plt.subplot(2, 4, 8)
sns.distplot(data['Rating'], color = 'black')
plt.grid()

plt.suptitle('Score Distributions for Different Abilities')
plt.show()
# comparison of preferred foot over the different players

plt.rcParams['figure.figsize'] = (8, 3)
sns.countplot(data['Preferred Foot'], palette = 'pink')
plt.title('Most Preferred Foot of the Players', fontsize = 20)
```

```
plt.show()
labels = ['1', '2', '3', '4', '5']  #data['International Reputation'].index
sizes = data['International Reputation'].value_counts()
colors = plt.cm.copper(np.linspace(0, 1, 5))
explode = [0.1, 0.1, 0.2, 0.5, 0.9]

plt.rcParams['figure.figsize'] = (9, 9)
plt.pie(sizes, labels = labels, colors = colors, explode = explode, shadow = True,)
plt.title('International Repuatation for the Football Players', fontsize = 20)
plt.legend()
plt.show()
data[data['International Reputation'] == 5][['Name','Nationality',
                    'Overall']].sort_values(by = 'Overall',
                            ascending = False).style.background_gradient(cmap = 'magma')
# plotting a pie chart to represent the share of week foot players

labels = ['5', '4', '3', '2', '1']
size = data['Weak Foot'].value_counts()
colors = plt.cm.Wistia(np.linspace(0, 1, 5))
explode = [0, 0, 0, 0, 0.1]

plt.pie(size, labels = labels, colors = colors, explode = explode, shadow = True, startangle = 90)
plt.title('Distribution of Week Foot among Players', fontsize = 25)
plt.legend()
plt.show()
```

```
# different positions acquired by the players

plt.figure(figsize = (13, 15))
plt.style.use('fivethirtyeight')
ax = sns.countplot(y = 'Position', data = data, palette = 'bone')
ax.set_xlabel(xlabel = 'Different Positions in Football', fontsize = 16)
ax.set_ylabel(ylabel = 'Count of Players', fontsize = 16)
ax.set_title(label = 'Comparison of Positions and Players', fontsize = 20)
plt.show()
# defining a function for cleaning the Weight data

def extract_value_from(value):
  out = value.replace('lbs', '')
  return float(out)

# applying the function to weight column
#data['value'] = data['value'].apply(lambda x: extract_value_from(x))
data['Weight'] = data['Weight'].apply(lambda x : extract_value_from(x))

# plotting the distribution of weight of the players
sns.distplot(data['Weight'], color = 'black')
plt.title("Distribution of Players Weight", fontsize = 15)
plt.show()
# defining a function for cleaning the wage column

def extract_value_from(column):
    out = column.replace('€', '')
    if 'M' in out:
```

```
        out = float(out.replace('M', ''))*1000000
    elif 'K' in column:
        out = float(out.replace('K', ''))*1000
    return float(out)
data['Value'] = data['Value'].apply(lambda x: extract_value_from(x))
data['Wage'] = data['Wage'].apply(lambda x: extract_value_from(x))

# visualizing the data
plt.rcParams['figure.figsize'] = (16, 5)
plt.subplot(1, 2, 1)
sns.distplot(data['Value'], color = 'violet')
plt.title('Distribution of Value of the Players', fontsize = 15)

plt.subplot(1, 2, 2)
sns.distplot(data['Wage'], color = 'purple')
plt.title('Distribution of Wages of the Players', fontsize = 15)
plt.show()
# Skill Moves of Players

plt.figure(figsize = (10, 6))
ax = sns.countplot(x = 'Skill Moves', data = data, palette = 'pastel')
ax.set_title(label = 'Count of players on Basis of their skill moves', fontsize = 20)
ax.set_xlabel(xlabel = 'Number of Skill Moves', fontsize = 16)
ax.set_ylabel(ylabel = 'Count', fontsize = 16)
plt.show()
data[(data['Skill Moves'] == 5.0) & (data['Age'] < 20)][['Name','Age']]
# To show Different Work rate of the players participating in the FIFA 2019
```

```
plt.figure(figsize = (15, 5))
plt.style.use('fivethirtyeight')

sns.countplot(x = 'Work Rate', data = data, palette = 'hls')
plt.title('Different work rates of the Players Participating in the FIFA 2019', fontsize = 20)
plt.xlabel('Work rates associated with the players', fontsize = 16)
plt.ylabel('count of Players', fontsize = 16)
plt.xticks(rotation = 90)
plt.show()
# To show Different potential scores of the players participating in the FIFA 2019

plt.figure(figsize=(16, 4))
plt.style.use('seaborn-paper')

plt.subplot(1, 2, 1)
x = data.Potential
ax = sns.distplot(x, bins = 58, kde = False, color = 'y')
ax.set_xlabel(xlabel = "Player's Potential Scores", fontsize = 10)
ax.set_ylabel(ylabel = 'Number of players', fontsize = 10)
ax.set_title(label = 'Histogram of players Potential Scores', fontsize = 15)

plt.subplot(1, 2, 2)
y = data.Overall
ax = sns.distplot(y, bins = 58, kde = False, color = 'y')
ax.set_xlabel(xlabel = "Player's Overall Scores", fontsize = 10)
```

```
ax.set_ylabel(ylabel = 'Number of players', fontsize = 10)
ax.set_title(label = 'Histogram of players Overall Scores', fontsize = 15)
plt.show()
# violin plot

plt.rcParams['figure.figsize'] = (20, 7)
plt.style.use('seaborn-dark-palette')

sns.boxplot(data['Overall'], data['Age'], hue = data['Preferred Foot'], palette = 'Greys')
plt.title('Comparison of Overall Scores and age wrt Preferred foot', fontsize = 20)
plt.show()
# picking up the countries with highest number of players to compare their overall scores

data['Nationality'].value_counts().head(10).plot(kind = 'pie', cmap = 'inferno',
                                    startangle = 90, explode = [0, 0, 0, 0, 0, 0, 0, 0, 0.1, 0])
plt.title('Countries having Highest Number of players', fontsize = 15)
plt.axis('off')
plt.show()
# Every Nations' Player and their Weights

some_countries = ('England', 'Germany', 'Spain', 'Argentina', 'France', 'Brazil', 'Italy', 'Columbia')
data_countries = data.loc[data['Nationality'].isin(some_countries) & data['Weight']]
```

```
plt.rcParams['figure.figsize'] = (15, 7)
ax = sns.violinplot(x = data_countries['Nationality'], y = data_countries['Weight'], palette = 'Reds')
ax.set_xlabel(xlabel = 'Countries', fontsize = 9)
ax.set_ylabel(ylabel = 'Weight in lbs', fontsize = 9)
ax.set_title(label = 'Distribution of Weight of players from different countries', fontsize = 20)
plt.show()
# Every Nations' Player and their overall scores

some_countries = ('England', 'Germany', 'Spain', 'Argentina', 'France', 'Brazil', 'Italy', 'Columbia')
data_countries = data.loc[data['Nationality'].isin(some_countries) & data['Overall']]

plt.rcParams['figure.figsize'] = (15, 7)
ax = sns.barplot(x = data_countries['Nationality'], y = data_countries['Overall'], palette = 'spring')
ax.set_xlabel(xlabel = 'Countries', fontsize = 9)
ax.set_ylabel(ylabel = 'Overall Scores', fontsize = 9)
ax.set_title(label = 'Distribution of overall scores of players from different countries', fontsize = 20)
plt.show()
# Every Nations' Player and their wages

some_countries = ('England', 'Germany', 'Spain', 'Argentina', 'France', 'Brazil', 'Italy', 'Columbia')
data_countries = data.loc[data['Nationality'].isin(some_countries) & data['Wage']]
```

```
plt.rcParams['figure.figsize'] = (15, 7)
ax = sns.barplot(x = data_countries['Nationality'], y = data_countries['Wage'], palette = 'Purples')
ax.set_xlabel(xlabel = 'Countries', fontsize = 9)
ax.set_ylabel(ylabel = 'Wage', fontsize = 9)
ax.set_title(label = 'Distribution of Wages of players from different countries', fontsize = 15)
plt.grid()
plt.show()
# Every Nations' Player and their International Reputation

some_countries = ('England', 'Germany', 'Spain', 'Argentina', 'France', 'Brazil', 'Italy', 'Columbia')
data_countries = data.loc[data['Nationality'].isin(some_countries) & data['International Reputation']]

plt.rcParams['figure.figsize'] = (15, 7)
ax = sns.boxenplot(x = data_countries['Nationality'], y = data_countries['International Reputation'], palette = 'autumn')
ax.set_xlabel(xlabel = 'Countries', fontsize = 9)
ax.set_ylabel(ylabel = 'Distribution of reputation', fontsize = 9)
ax.set_title(label = 'Distribution of International Repuatation of players from different countries', fontsize = 15)
plt.grid()
plt.show()
some_clubs = ('CD Leganés', 'Southampton', 'RC Celta', 'Empoli', 'Fortuna Düsseldorf', 'Manchestar City',
```

```
'Tottenham Hotspur', 'FC Barcelona', 'Valencia CF', 'Chelsea', 'Real Madrid')

data_clubs = data.loc[data['Club'].isin(some_clubs) & data['Overall']]

plt.rcParams['figure.figsize'] = (15, 8)
ax = sns.boxplot(x = data_clubs['Club'], y = data_clubs['Overall'], palette = 'inferno')
ax.set_xlabel(xlabel = 'Some Popular Clubs', fontsize = 9)
ax.set_ylabel(ylabel = 'Overall Score', fontsize = 9)
ax.set_title(label = 'Distribution of Overall Score in Different popular Clubs', fontsize = 20)
plt.xticks(rotation = 90)
plt.grid()
plt.show()
# Distribution of Ages in some Popular clubs

some_clubs = ('CD Leganés', 'Southampton', 'RC Celta', 'Empoli', 'Fortuna Düsseldorf', 'Manchestar City',
    'Tottenham Hotspur', 'FC Barcelona', 'Valencia CF', 'Chelsea', 'Real Madrid')

data_club = data.loc[data['Club'].isin(some_clubs) & data['Wage']]

plt.rcParams['figure.figsize'] = (15, 8)
ax = sns.boxenplot(x = 'Club', y = 'Age', data = data_club, palette = 'magma')
ax.set_xlabel(xlabel = 'Names of some popular Clubs', fontsize = 10)
ax.set_ylabel(ylabel = 'Distribution', fontsize = 10)
```

```
ax.set_title(label = 'Disstribution of Ages in some Popular Clubs', fontsize = 20)
plt.xticks(rotation = 90)
plt.grid()
plt.show()
# Distribution of Wages in some Popular clubs

some_clubs = ('CD Leganés', 'Southampton', 'RC Celta', 'Empoli', 'Fortuna Düsseldorf', 'Manchestar City',
        'Tottenham Hotspur', 'FC Barcelona', 'Valencia CF', 'Chelsea', 'Real Madrid')

data_club = data.loc[data['Club'].isin(some_clubs) & data['Wage']]

plt.rcParams['figure.figsize'] = (16, 8)
ax = sns.boxplot(x = 'Club', y = 'Wage', data = data_club, palette = 'magma')
ax.set_xlabel(xlabel = 'Names of some popular Clubs', fontsize = 10)
ax.set_ylabel(ylabel = 'Distribution', fontsize = 10)
ax.set_title(label = 'Disstribution of Wages in some Popular Clubs', fontsize = 20)
plt.xticks(rotation = 90)
plt.show()
# Distribution of Wages in some Popular clubs

some_clubs = ('CD Leganés', 'Southampton', 'RC Celta', 'Empoli', 'Fortuna Düsseldorf', 'Manchestar City',
        'Tottenham Hotspur', 'FC Barcelona', 'Valencia CF', 'Chelsea', 'Real Madrid')
```

```
data_club = data.loc[data['Club'].isin(some_clubs) & data['International Reputation']]

plt.rcParams['figure.figsize'] = (16, 8)
ax = sns.boxenplot(x = 'Club', y = 'International Reputation', data = data_club, palette = 'copper')
ax.set_xlabel(xlabel = 'Names of some popular Clubs', fontsize = 10)
ax.set_ylabel(ylabel = 'Distribution of Reputation', fontsize = 10)
ax.set_title(label = 'Distribution of International Reputation in some Popular Clubs', fontsize = 20)
plt.xticks(rotation = 90)
plt.grid()
plt.show()
```

Query Analysis

Best Players per each position with their age, club, and nationality based on their Overall Scores

```
# best players per each position with their age, club, and nationality based on their overall scores

data.iloc[data.groupby(data['Position'])['Overall'].idxmax()][['Position', 'Name', 'Age', 'Club',
                    'Nationality','Overall']].sort_values(by = 'Overall',
                        ascending = False).style.background_gradient(cmap = 'pink')
## Let's Analyze the Skills of Players

@interact
def skill(skills = ['Defending', 'General', 'Mental', 'Passing',
                    'Mobility', 'Power', 'Rating','Shooting'], score = 75):
```

```
    return data[data[skills] > score][['Name', 'Nationality', 'Club', 'Overall', skills]].sort_values(by = skills,
                                ascending = False).head(20).style.background_gradient(cmap = 'Blues')
# lets make an interactive function for getting a report of the players country wise

# lets make a function to see the list of top 15 players from each country
@interact
def country(country = list(data['Nationality'].value_counts().index)):
    return data[data['Nationality'] == country][['Name','Position','Overall',
                'Potential']].sort_values(by = 'Overall',
                        ascending = False).head(15).style.background_gradient(cmap = 'magma')
# lets make an interactive function to get the list of top 15 players from each of the club

# lets define a function
@interact
def club(club = list(data['Club'].value_counts().index[1:])):
    return data[data['Club'] == club][['Name','Jersey Number','Position','Overall','Nationality','Age','Wage',
                'Value','Contract Valid Until']].sort_values(by = 'Overall',
                            ascending = False).head(15).style.background_gradient(cmap = 'inferno')
# finding 5 youngest Players from the dataset

youngest = data[data['Age'] == 16][['Name', 'Age', 'Club', 'Nationality', 'Overall']]
```

```
youngest.sort_values(by = 'Overall', ascending =
False).head().style.background_gradient(cmap = 'magma')
# finding 15 eldest players from the dataset

data.sort_values('Age', ascending = False)[['Name', 'Age', 'Club',
                    'Nationality',
'Overall']].head(15).style.background_gradient(cmap = 'Wistia')
# The longest membership in the club

now = datetime.datetime.now()
data['Join_year'] = data.Joined.dropna().map(lambda x: x.split(',')[1].split(' ')[1])
data['Years_of_member'] = (data.Join_year.dropna().map(lambda x: now.year -
int(x))).astype('int')
membership = data[['Name', 'Club', 'Years_of_member']].sort_values(by =
'Years_of_member', ascending = False).head(10)
membership.set_index('Name', inplace=True)
membership.style.background_gradient(cmap = 'Reds')
import ipywidgets as widgets
from ipywidgets import interact
@interact
def check(column = 'Years_of_member',
        club = ['FC Barcelona', 'Real Madrid', 'Chelsea'], x = 4):
   return data[(data[column] > x) & (data['Club'] == club)][['Name', 'Club',
                                'Years_of_member']].sort_values(by =
'Years_of_member',
                                                ascending =
False).style.background_gradient(cmap = 'magma')
# defining the features of players
```

```
player_features = ('Acceleration', 'Aggression', 'Agility',
                   'Balance', 'BallControl', 'Composure',
                   'Crossing', 'Dribbling', 'FKAccuracy',
                   'Finishing', 'GKDiving', 'GKHandling',
                   'GKKicking', 'GKPositioning', 'GKReflexes',
                   'HeadingAccuracy', 'Interceptions', 'Jumping',
                   'LongPassing', 'LongShots', 'Marking', 'Penalties')

# Top four features for every position in football

for i, val in data.groupby(data['Position'])[player_features].mean().iterrows():
    print('Position {}: {}, {}, {}'.format(i, *tuple(val.nlargest(4).index)))
# Top 10 left footed footballers

data[data['Preferred Foot'] == 'Left'][['Name', 'Age', 'Club',
         'Nationality', 'Overall']].sort_values(by = 'Overall',
            ascending = False).head(10).style.background_gradient(cmap = 'bone')
# Top 10 Right footed footballers

data[data['Preferred Foot'] == 'Right'][['Name', 'Age', 'Club',
                  'Nationality', 'Overall']].sort_values(by = 'Overall',
                        ascending  =  False).head(10).style.background_gradient(cmap
= 'copper')
# comparing the performance of left-footed and right-footed footballers
# ballcontrol vs dribbing

sns.lmplot(x = 'BallControl', y = 'Dribbling', data = data, col = 'Preferred Foot')
plt.show()
```

Predictions:

```
print(os.getcwd())
# inorder to change directory for direct upload
#os.chdir("C:\Users\harsh\OneDrive\Documents\final project\code")
pdf = pd.read_csv("all_players.csv")
pdf.head(10)
pdf.describe()
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.tree import export_graphviz
from sklearn import tree

from IPython.display import Image
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
from sklearn import datasets
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error
pdf=pdf.drop('Club',axis=1)

pdf=pdf.drop('PKG/A',axis=1)
pdf["SOG%"]=pdf["SOG%"].fillna(0)
pdf.dtypes
pdf.isnull().sum()
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
```

```
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
import sklearn.linear_model as lm
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
from sklearn import datasets, linear_model, metrics
from sklearn.metrics import mean_squared_error
from sklearn import svm
from sklearn.svm import SVC
from sklearn.model_selection import LeaveOneOut
from sklearn.model_selection import cross_val_score
from numpy import mean
from numpy import std
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.naive_bayes import GaussianNB
from xgboost import XGBRFClassifier
from sklearn.datasets import make_classification
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.metrics import classification_report
from mpl_toolkits.mplot3d import Axes3D
from sklearn.preprocessing import StandardScaler
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
```

```
import matplotlib.pyplot as mtp
from sklearn.cluster import DBSCAN
label_encoder=LabelEncoder()
pdf['Player']= label_encoder.fit_transform(pdf['Player'])
pdf['POS']= label_encoder.fit_transform(pdf['POS'])


pdf['Season']= label_encoder.fit_transform(pdf['Season'])
y=pdf.GWG
x=pdf.drop('GWG',axis=1)
x, y = make_classification(n_samples=1000, n_features=20, n_informative=15, n_redundant=5, random_state=7)
model = XGBRFClassifier(n_estimators=100, subsample=0.9, colsample_bynode=0.2)
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
n_scores = cross_val_score(model, x, y, scoring='accuracy', cv=cv, n_jobs=-1)
print('Mean Accuracy: %.3f (%.3f)' % (mean(n_scores), std(n_scores)))
```

```
Mean Accuracy: 0.894 (0.032)
```

Reduced features¶

```
print('Original feature number:', x.shape[1])
 cv = LeaveOneOut()
```

```
Original feature number: 20
```

create model

```
model = RandomForestClassifier(random_state=1)
```

evaluate model

```
scores = cross_val_score(model, x, y, scoring='accuracy', cv=cv, n_jobs=-1)
```

report performance

```
print('Accuracy of LOOCV: %.3f (%.3f)' % (mean(scores), std(scores)))
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.4)
clf = svm.SVC(kernel='linear') # Linear Kernel
print('Accuracy of LOOCV: %.3f (%.3f)' % (mean(scores), std(scores)))
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.4)
```

clf = svm.SVC(kernel='linear') # Linear Kernel

```
Accuracy of LOOCV: 0.920 (0.271)
```

x_train

y_train

x_test

y_test

#Train the model using the training sets

clf.fit(x_train, y_train)

```
SVC(kernel='linear')
```

#Predict the response for test dataset

y_pred = clf.predict(x_test)

print("SVM")

print("------Classification Report------")

print(classification_report(y_pred,y_test))

print()

print("------Accuracy------")

print("Accuracy in SVM:",metrics.accuracy_score(y_test, y_pred))

```
SVM
------Classification Report------
              precision    recall  f1-score   support

           0       0.82      0.90      0.86       193
           1       0.90      0.82      0.86       207

    accuracy                           0.86       400
   macro avg       0.86      0.86      0.86       400
weighted avg       0.86      0.86      0.86       400


------Accuracy------
Accuracy in SVM: 0.86
```

pdf= DBSCAN(eps=0.3, min_samples=10).fit(x)

core_samples_mask = np.zeros_like(pdf.labels_, dtype=bool)

core_samples_mask[pdf.core_sample_indices_] = True

labels =pdf.labels_

Number of clusters in labels, ignoring noise if present.

n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)

```
print(labels)
```

Plot result

```
unique_labels = set(labels)
colors = ['y', 'b', 'g', 'r']
print(colors)
for k, col in zip(unique_labels, colors):
    if k == -1:
        # Black used for noise.
        col = 'k'

    class_member_mask = (labels == k)

    xy = x[class_member_mask & core_samples_mask]
    plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=col,
                                    markeredgecolor='g',
                                    markersize=6)

    xy = x[class_member_mask & ~core_samples_mask]
    plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=col,
                                        markeredgecolor='r',
                                        markersize=6)

plt.title('number of clusters: %d' %n_clusters_)
plt.show()
```