A Project report

on

Parkinson Disease Detection

Submitted in partial fulfilment of the requirements

for the award of the degree of

BACHELOR OF TECHNOLOGY

in

Computer Science & Engineering

by

K. SHIRISHA	**(184G1A0586)**
V. THRIPURA	**(184G1A05A5)**
M. SAHITHA	**(184G1A0570)**
B. VENKATA PRASAD REDDY	**(194G5A0511)**

Under the Guidance of

Mr. M. Narasimhulu M.Tech., (Ph.D)

Assistant Professor

Department of Computer Science & Engineering

SRINIVASA RAMANUJAN INSTITUTE OF TECHNOLOGY

(Affiliated to JNTUA & Approved by AICTE)

(Accredited by NAAC with 'A' Grade & Accredited by NBA(EEE, ECE & CSE))

Rotarypuram Village, B K Samudram Mandal, Ananthapuramu-515701.

2021-2022

SRINIVASA RAMANUJAN INSTITUTE OF TECHNOLOGY
(Affiliated to JNTUA & Approved by AICTE)
(Accredited by NAAC with 'A' Grade & Accredited by NBA (EEE, ECE & CSE)
Rotarypuram Village, B K Samudram Mandal, Ananthapuramu-515701.

Certificate

This is to certify that the project report entitled Parkinson Disease Detection is the bonafide work carried out by **K. Shirisha** bearing Roll Number **184G1A0586, V. Thripura** bearing Roll Number **184G1A05A5, M. Sahitha** bearing Roll Number **184G1A0570, B. Venkata Prasad Reddy** bearing Roll Number **194G5A0511** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering** during the academic year 2021-2022.

Guide	**Head of the Department**
Mr. M. Narasimhulu, M. Tech.,(Ph.D).	Mr. P. Veera Prakash, M. Tech.,(Ph.D).
Assistant Professor	Assistant Professor & HOD

Date: **EXTERNAL EXAMINER**

Place: Ananthapuramu

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of people who made it possible, whose constant guidance and encouragement crowned our efforts with success. It is a pleasant aspect that I have now the opportunity to express my gratitude for all of them.

It is with immense pleasure that I would like to express my indebted gratitude to my Guide **Mr. M. Narasimhulu, Assistant Professor, and Computer Science & Engineering**, who has guided me a lot and encouraged me in every step of the project work. We thank him for the stimulating guidance, constant encouragement and constructive criticism, which have made possible to bring out this project work.

We are very much thankful to **Mr. P. Veera Prakash, Assistant Professor & Head of the Department, Computer Science & Engineering**, for his kind support and for providing necessary facilities to carry out the project work.

We wish to convey my special thanks to **Dr. G. Bala Krishna, Ph.D., Principal** of **Srinivasa Ramanujan Institute of Technology** for giving the required information in doing our project work. Not to forget, we thank all other faculty and non-teaching staff, and my friends who had directly or indirectly helped and supported us in completing our project work in time.

We also express our sincere thanks to the Management for providing excellent facilities**.**

Finally, we wish to convey my gratitude to our family who fostered all the requirements and facilities that we need.

Project Associates

Declaration

We K. Shirisha bearing reg no: 184G1A0586, V. Thripura bearing reg no:184G1A05, M. Sahitha bearing reg no: 184G1A0570, B. Venkata Prasad Reddy bearing reg no: 194G5A0511, students of SRINIVASA RAMANUJAN INSTITUTE OF TECHNOLOGY, Rotarypuram, hereby declare that the dissertation entitled " PARKINSON DISEASE DETECTION" embodies the report of our project work carried out by us during IV year Bachelor of Technology under the guidance of Mr. M. Narasimhulu, M. Tech., (Ph.D) Department of CSE, SRINIVASA RAMANUJAN INSTITUTE OF TECHNOLOGY, and this work has been submitted for the partial fulfilment of the requirements for the award of the Bachelor of Technology degree.

The results embodied in this project have not been submitted to any other University of Institute for the award of any Degree or Diploma.

K. Shirisha	Reg no: 184G1A0586
V. Thripura	Reg no: 184G1A05A5
M. Sahitha	Reg no: 184G1A0570
B. Venkata Prasad Reddy	Reg no: 194G5A0511

CONTENTS

List of Figures

List of Abbreviations

PD	Parkinson Disease
DFA	Detrended Fluctuations analysis
CSV	Comma Separated Values
SVM	Support Vector Machine
KNN	K-Nearest Neighbour
XGB	Extreme Gradient Boosting
UML	Unified Modeling Language

ABSTRACT

Parkinson’s disease is a progressive neurodegenerative disorder that affects a lot only people significantly affecting their quality of life. These diseases mostly affect on the organic functions of the human. The main organic symptoms called “parkinsonism” or “parkinsonian syndrome”. The symptoms of Parkinson’s disease will occur slowly, the symptoms include shaking, rigidity, slowness of movement and difficulty with walking Thinking and behavior change, Depression and anxiety are common.

In the year 2015, there was an existing model for detecting Parkinson using voice. This existing model confirm the symptoms of Parkinson’s disease based on deflections measured in a voice. This existing model proved an efficiency of 73.8%. The Propose model will focus on a huge amount of collected data from the normal person and previously affected person by Parkinson’s disease, these data is trained using machine learning algorithms. From the whole data, 60% used for training and 40% to be for testing. The data of any person to be entered in the database to check whether the person is to be affected by Parkinson’s disease or not. There are 24 columns in the data set each column the symptom values of a patient except the status column. The status column has 0’s and 1’s; those values will decide the affected person with Parkinson’s disease. One indicates person effected, zero indicates person unaffected. Finally, the accuracy of model would measure with the existing model.

CHAPTER – 1
INTRODUCTION

Parkinson's disease is a disorder of the central nervous system affecting movement and inducing tremors and stiffness a neurodegenerative disorder affecting dopamine neurons in brain. Parkinson's disease is difficult to diagnose. Common diagnostic criteria require the medication before. In this model, the huge data is collected from previously affected person and then by using machine learning algorithm.

Parkinson's disease (PD) manifests as the death of dopaminergic neurons in the substantia nigra pars compacta within the mid brain [4]. This neurodegeneration leads to a range of symptoms including coordination issues, bradykinesia, vocal changes, and rigidity [5], [6]. Dysarthria is also observed in PD patients; it is characterized by weakness, paralysis, and lack of coordination in the motor-speech system: affecting respiration, phonation, articulation, and prosody [7]. Since symptoms and the disease course vary, PD is often not diagnosed for many years. Therefore, there is a need for more sensitive diagnostic tools for PD detection because, as the disease progresses, more symptoms arise that make PD harder to treat. The main deficits of PD speech are loss of intensity, monotony of pitch and loudness, reduced stress, inappropriate silences, short rushes of speech, variable rate, imprecise consonant articulation, and harsh and breathy voice (dysphonia). The range of voice related symptoms is promising for a potential detection tool because recording voice data is non-invasive and can be done easily with mobile devices.

PD is difficult to detect early due to the subtle initial symptoms. There is a significant burden to patients and the health care system due to delays in diagnosis [8]. The difficulty in early PD diagnosis has inspired researchers to develop screening tools relying on automated algorithms to differentiate healthy controls from people with PD. This binary diagnosis focuses on the first step of validating digital biomarkers in distinguishing disease from control; it does not offer a form of differential diagnosis where the model may distinguish PD among a variety of disorders that present PD-like symptoms (e.g. Lewy-Body Dementia, Essential Tremor). The current research is a promising first step toward a long-term goal of providing a decision support algorithm for physicians in screening patients for PD [9]. In this paper, we apply several different machine learning models to classify PD from controls using the mPower Voice dataset.

The data used for this analysis were collected through mPower, a clinical observational study conducted by Sage Bionetworks using an iPhone app to collect digital biomarkers and health data on participants both with and without PD [10]. To maintain user confidentiality and enable linking across datasets, each participant was uniquely identified with a distinct healthcode. The method used for collecting the audio data was a smartphone voice activity that recorded participants articulating the /aa/ phoneme for 10 seconds. The mPower study aimed to allow researchers to understand and analyze PD severity and features of patients to create more personalized treatment. The mPower dataset is broken down into several smaller datasets that were used in this study to characterize PD features. Typically, the symptoms of PD are attenuated by the use of dopaminergic medications such as levodopa. During data collection, patients were asked to give information regarding when, relative to taking medication, they provided their data. The options included: Just after Parkinson medication (at your best), Another time, Immediately before Parkinson medication, I don't take Parkinson medications, and no value. These medication time points were interpreted to mean: time of best symptom control, on medication but not immediately before or after, time of worst symptoms, not on medications, and not applicable, respectively.

1.1 Problem Definition

Now a days in Health Industry there are various problems related to machines or devices which will give wrong or unaccepted results, so to avoid those results and get the correct and desired results we are building a program or project which will give the accurate predictions based on information provided by the user and based on the datasets that are available in that machine. The health industry in information yet and knowledge poor and this industry is very vast industry which has lot of work to be done. So, with the help of all those algorithms, techniques, and methodologies we have done this project which will help the peoples who are in the need. So the problem here is that many people goes to hospitals or clinic to know how is their health and how much they are improving in the given days, but they have to travel to get to know their answers and sometimes the patients may or may not get the results based on various factors such as doctor might be on leave or some whether problem so he might not have come to the hospital and more reasons will be there so to avoid all those reasons and confusion we are making a project which will help all those person's and all the patients who are in need to know the condition of their health, and at sometimes if the person has been observing few symptoms and he/she is not sure about the diabetes in future. So, to avoid that and get to know the Parkinsons in early stages of the symptoms this Parkinoson’s disease detection will help a lot to the various people's ranging from children to teenagers to adults and the senior citizens.

- Early detection of Parkinson’s disease (PD) is particularly relevant, as people at early stages of the disease are more likely to benefit from neuroprotective treatments.
- The problem definition of Parkinson disease detection is previously detected by voice that is very difficult to find the disease.
- And also it is taking more time to give result.

1.2 Recognizing early Symptoms of Parkinson's Disease

- The identification of pre-motor symptoms is key to early detection of PD. Early symptoms of PD may include:
- Gastrointestinal difficulties, like constipation and slowed movement of food from the stomach into the intestines (gastroparesis).
- Reduced sense of smell, also known as hyposmia.
- Sleep problems, including insomnia, REM sleep behavior disorder (acting out dreams while asleep), restless legs syndrome and excessive daytime sleepiness.
- Sexual dysfunction, like erectile dysfunction in men, poor lubrication in women, or difficulty achieving orgasm in both men and women.
- Mood disorders, like depression or anxiety.

1.3 Objective

The main objective of the project is to develop a model by applying different classification techniques on the dataset which can perform early prediction of Parkinsons for the patient with the higher accuracy by combining the results of different machine learning techniques.

CHAPTER – 2
LITERATURE SURVEY

2.1 Study of Journals

2.1.1 Title 1: Detection of Parkinson Disease Using Clinical Voice Data Mining:

For all 195 phonations, features are extracted. For feature extraction only first half of the recordings are considered, because the second half of the recording is influenced by reduced lung pressure. Various traditional measures and nonstandard measures are extracted. A set of 22 features is prepared. Feature set consists of Fo(Hz), Fhi(Hz), Flo(Hz), Jitter(%), Jitter(Abs), RAP, PPQ, Jitter:DDP, Shimmer, Shimmer(dB), Shimmer:APQ3, Shimmer:APQ5, APQ , Shimmer:DDA, NHR, HNR, RPDE, DFA, spread1, spread2, D2, PPE. The vocal fold vibration frequency is known as fundamental frequency. The perturbation in the frequency and amplitude in successive vocal fold cycles is termed as jitter and shimmer respectively. The noise-to-harmonic and harmonic-to-noise measures are measured using estimates of signal to noise by calculation of autocorrelation of each cycle. D2 is the correlation dimension between the signal and its first time delay embedded signal whereas the RPDE (recurrence period density) is the measure of periodicity of the reconstructed signal after embedding time delay. DFA (Detrended fluctuation analysis) is the log-log plot of the time scales L and amplitude variation F (L). Non linear measure of fundamental frequency variation is defined in terms of spread 1 and spread 2. The logarithmic scale of pitch sequence is explained as semitone pitch p(t) where t is the time. The entropy of relative semitone variation is known as pitch period entropy (PPE). All these parameters show variation for the healthy and parkinson's case. Next, features are selected among these to get best classification among the two groups.

IV. FEATURE SELECTION

Feature are selected which have more separable values than others and a new feature data subset is prepared which contains 15 features as shown in Table 1 . Support vector machine classifier is used and their performance is evaluated. Classifier used is supervised classifiers and therefore dataset is divided into training and testing datasets.75% of the data is used for training purpose and rest 25% is for testing. Out of 195 observations, 146 are used for training (110 parkinson +36 healthy) and 49 are used for testing.

Various classifiers are present which provide good results. ANN (artificial neural network), SVM (support vector machines), GMM (Gaussian Mixture Model) and HMM (Hidden Markov Model are mostly used in speech processing applications [17, 18, 19]. SVMs are set of related supervised learning methods used for classification and regression. Support vector machines builds a model using set of training examples, each marked to its category and then used for classification. SVM simultaneously minimize the empirical classification error and maximize the geometric margin. SVM map input vector to a higher dimensional space where a maximal separating hyperplane is constructed. In all cases, a maximum margin hyperplane is selected which have the largest separation between the two classes. Maximum margin hyper plane is a plane from which distance to the nearest data point on both sides is maximized.

2.1.2 Title 2: Chirag Mittal, Amritanshu Sharma, "Parkinson's Disease Detection Using Different Machine Learning Algorithms'':

Parkinson's Disease is a widespread degenerative syndrome that affects the nervous system. Its early symptoms include tremor, rigidity, and vocal impairment (dysphonia). This paper proposes performance of Machine Learning Methods in Diagnosing Parkinson's Disease. Several machine-learning techniques were considered and trained with the same data set to classify healthy and Parkinson's Disease patients. Methods:-We have used various machine learning-based techniques for Parkinson's disease (PD) diagnosis. These ma[1]chine learning techniques includes K Nearest Neighbors (k[1]NN), Naïve Bayes (NB), Decision Tree (DT), Support Vector Machine (SVM), Stochastic Gradient Descent (SGD), Random Forest and XGBoost. We have used these algorithms to check the best algorithm for detection of Parkinson's Dis- ease. Result:-After applying these algorithms we found our accuracies of these algorithms are as follows: - Naive Bayes(71.79%), SGD(84.61%), Decision Tree(84.61%), KNN(97.43%), Random Forest(89.74%), SVM(87.18%) and XGBoost(94.87%). After considering all algorithms and analyzing their accuracies we found out that KNN is the best of all the algorithms used by us for detection of Parkinson Disease with accuracy of 97.43 percent.

2.2 Existing System

In existing system, Parkinson disease is detected at the secondary stage only (Dopamine deficiency) which leads to medical challenges. Also doctor has to manually examine and suggest medical diagnosis in which the symptoms might vary from person to person so suggesting medicine is also a challenge. Thus the mental disorders are been poorly characterized and have many health complications. PD is generally diagnosed with the following clinical methods as,

- MRI or CT scan - Conventional MRI cannot detect early signs of Parkinson's disease
- PET scan - is used to assess activity and function of brain regions involved in movement
- SPECT scan - can reveal changes in brain chemistry, such as a decrease in dopamine

This results in a high misdiagnosis rate (up to 25% by non-specialists) and many years before diagnosis, people can have the disease. Thus existing system is not effective in early prediction and accurate medicinal diagnosis to the affected people.

2.3 Proposed System

The proposed system of Parkinson Disease Detection using machine learning is that we have used many techniques and algorithms and all other various tools to build a system which detects the Parkinson Disease of the patient using the symptoms and by taking those symptoms we are comparing with the system's dataset that is previously available. By taking those datasets and comparing with the patient's symptoms we will detect the accurate percentage Parkinson Disease of the patient. The dataset and symptoms go to the prediction model of the system where the data is pre-processed for the future references and then the user will enter the various symptoms. Then the classification of those data is done with the help of various algorithms and techniques such as Decision Tree, KNN, Random Forest and SVM. This system takes symptoms from the user and detects the disease accordingly. Based on the symptoms that it takes and also from the dataset. And then it calculates the accuracy that the user may have disease or not in the future and shows the result.

CHAPTER - 3

REQUIREMENTS

It is essential to understand the requirements of the project which plays a crucial role in developing and achieving or meeting the desired goal. The Hardware Requirements and the Software Requirements of the project for developing and reproducing the results of the projects are discussed in detail.

3.1 Hardware Requirements

This section provides knowledge about the implementation environment and throws light on the actual steps for the implementation of dataset to get better accuracy to predict diabetes by using different classifiers combination. These are the essential hardware requirements that are necessary to build the classification model and to develop a web application:

Processor	:	i5
RAM	:	Minimum of 8GB
Hard Disk	:	512GB
Input Device	:	Keyboard and Mouse
Output Device	:	Monitor or PC

3.2 Software Requirements

Software Requirement Specification is a document created by a system analyst after the requirements are collected. SRS defines how the intended software will interact with hardware, external interfaces, speed of operation, response time of system, portability of software across various platforms, maintainability, speed of recovery after crashing, Security, Quality, Limitations etc.

The requirements received from clients are written in natural language. It is the responsibility of system analysts to document the requirements in technical language so that they can be comprehended and useful by the software development team.

Operating system: Windows 7 0r Windows 10

Tools: PyCharm

Dataset: CSV file

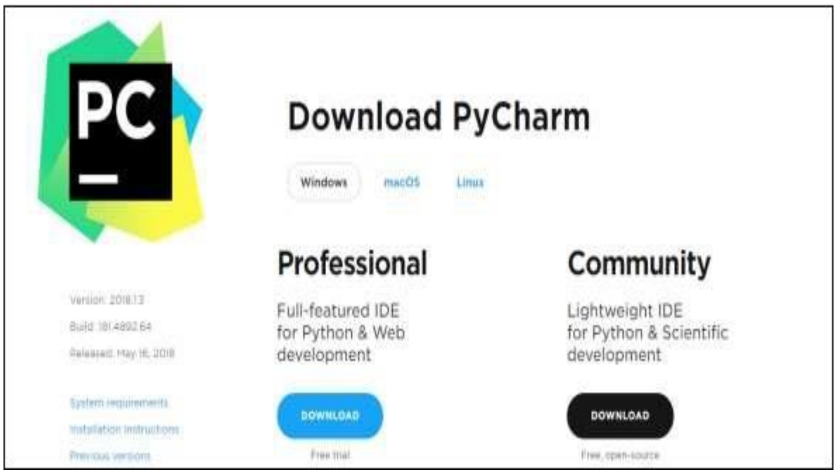
Languages Used: Python

Steps Involved to download and Install PyCharm

You will have to follow the steps given below to install PyCharm on your system. These steps show the installation procedure starting from downloading the PyCharm package from its official website to creating a new project.

Step 1

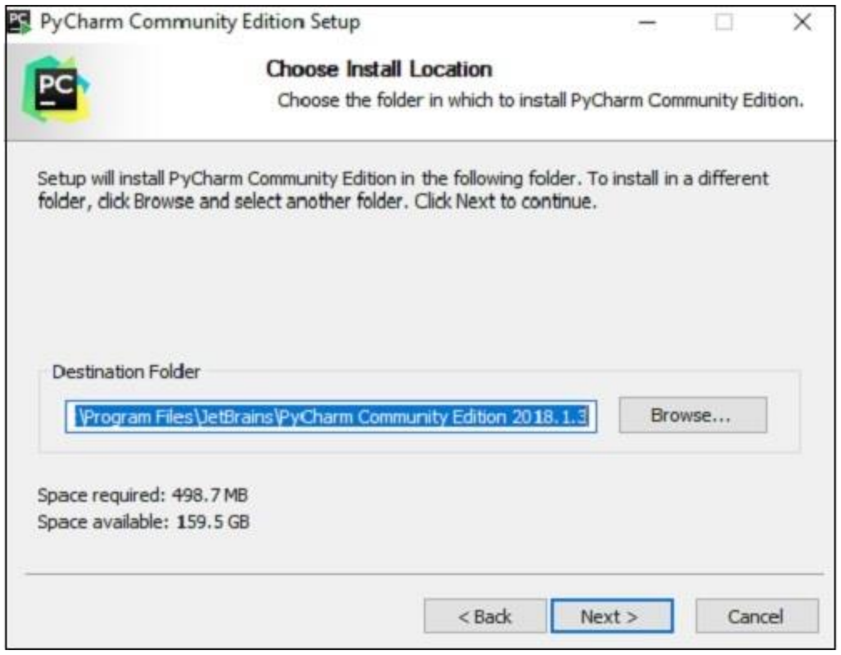
- Download the required package or executable from the official website PyCharm https://www.jetbrains.com/pycharm/download/#section=windows Here you will observe two versions of package for Windows as shown in the screenshot given below –



Note that the professional package involves all the advanced features and comes with free trial for few days and the user has to buy a licensed key for activation beyond the trial period. Community package is for free and can be downloaded and installed as and when required. It includes all the basic features needed for installation. Note that we will continue with community package throughout this tutorial.

Step 2

Download the community package (executable file) onto your system and mention a destination folder as shown below −



PyCharm Community Edition Setup

Choose Install Location

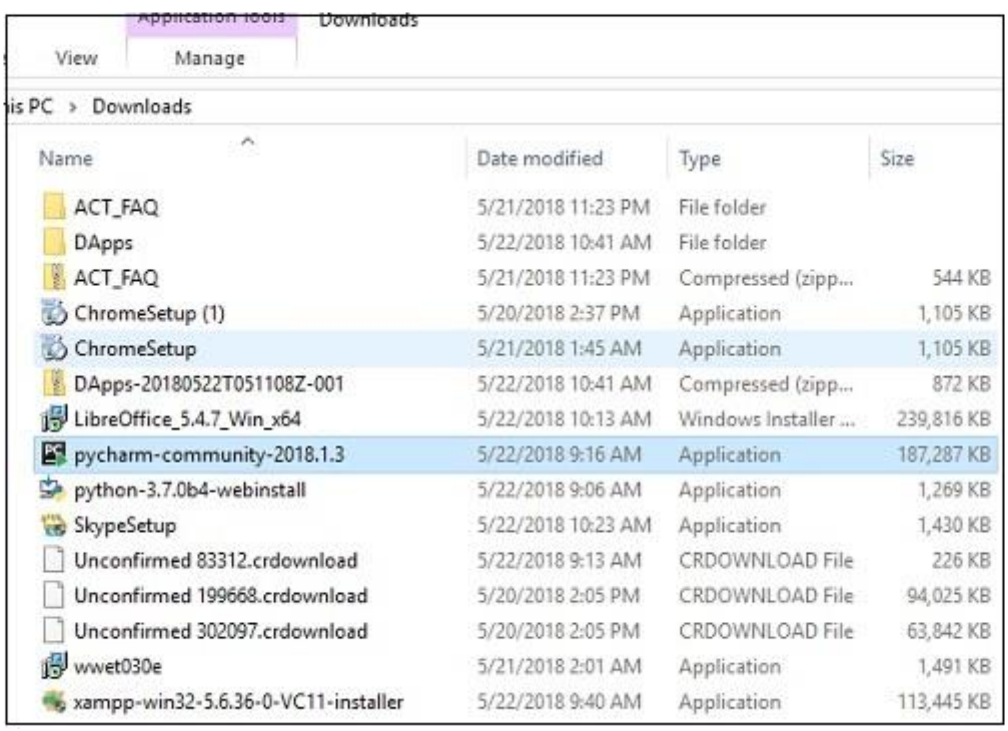
Choose the folder in which to install PyCharm Community Edition.

Setup will install PyCharm Community Edition in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.

Destination Folder

\Program Files\JetBrains\PyCharm Community Edition 2018.1.3

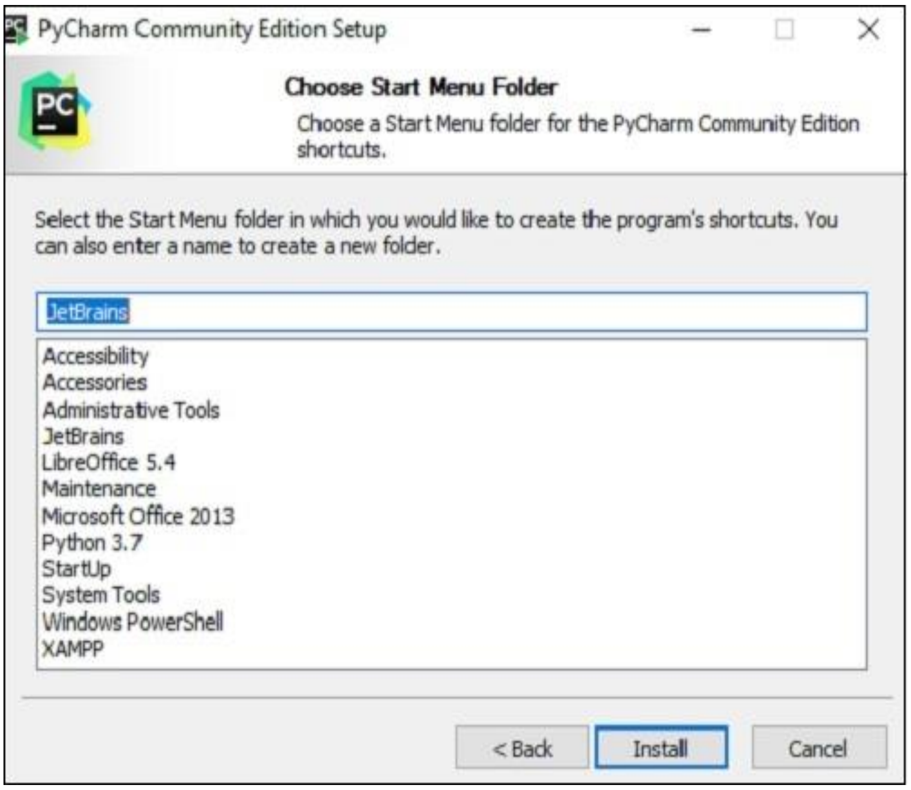
Space required: 498.7 MB
Space available: 159.5 GB



Name	Date modified	Type	Size
ACT_FAQ	5/21/2018 11:23 PM	File folder	
DApps	5/22/2018 10:41 AM	File folder	
ACT_FAQ	5/21/2018 11:23 PM	Compressed (zipp...	544 KB
ChromeSetup (1)	5/20/2018 2:37 PM	Application	1,105 KB
ChromeSetup	5/21/2018 1:45 AM	Application	1,105 KB
DApps-20180522T051108Z-001	5/22/2018 10:41 AM	Compressed (zipp...	872 KB
LibreOffice_5.4.7_Win_x64	5/22/2018 10:13 AM	Windows Installer ...	239,816 KB
pycharm-community-2018.1.3	5/22/2018 9:16 AM	Application	187,287 KB
python-3.7.0b4-webinstall	5/22/2018 9:06 AM	Application	1,269 KB
SkypeSetup	5/22/2018 10:23 AM	Application	1,430 KB
Unconfirmed 83312.crdownload	5/22/2018 9:13 AM	CRDOWNLOAD File	226 KB
Unconfirmed 199668.crdownload	5/20/2018 2:05 PM	CRDOWNLOAD File	94,025 KB
Unconfirmed 302097.crdownload	5/20/2018 2:05 PM	CRDOWNLOAD File	63,842 KB
wwet030e	5/21/2018 2:01 AM	Application	1,491 KB
xampp-win32-5.6.36-0-VC11-installer	5/22/2018 9:40 AM	Application	113,445 KB

Step 3

Now, begin the installation procedure similar to any other software package.


PyCharm Community Edition Setup

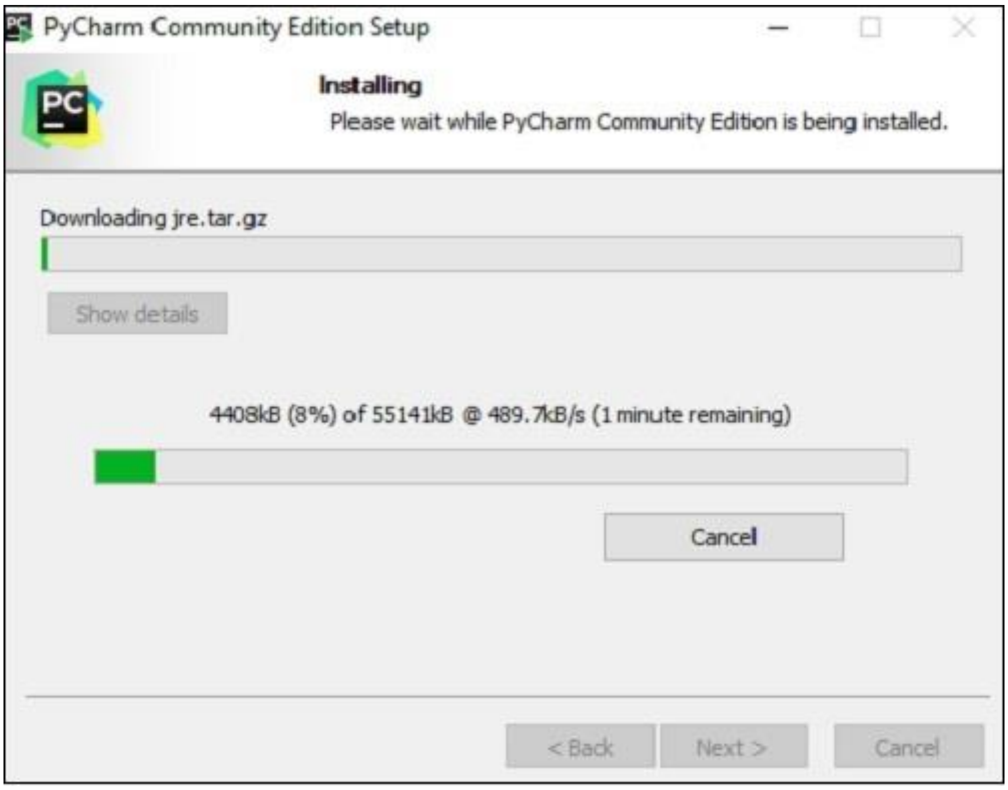
Choose Start Menu Folder

Choose a Start Menu folder for the PyCharm Community Edition shortcuts.

Select the Start Menu folder in which you would like to create the program's shortcuts. You can also enter a name to create a new folder.

JetBrains

Accessibility
Accessories
Administrative Tools
JetBrains
LibreOffice 5.4
Maintenance
Microsoft Office 2013
Python 3.7
StartUp
System Tools
Windows PowerShell
XAMPP


PyCharm Community Edition Setup

Installing

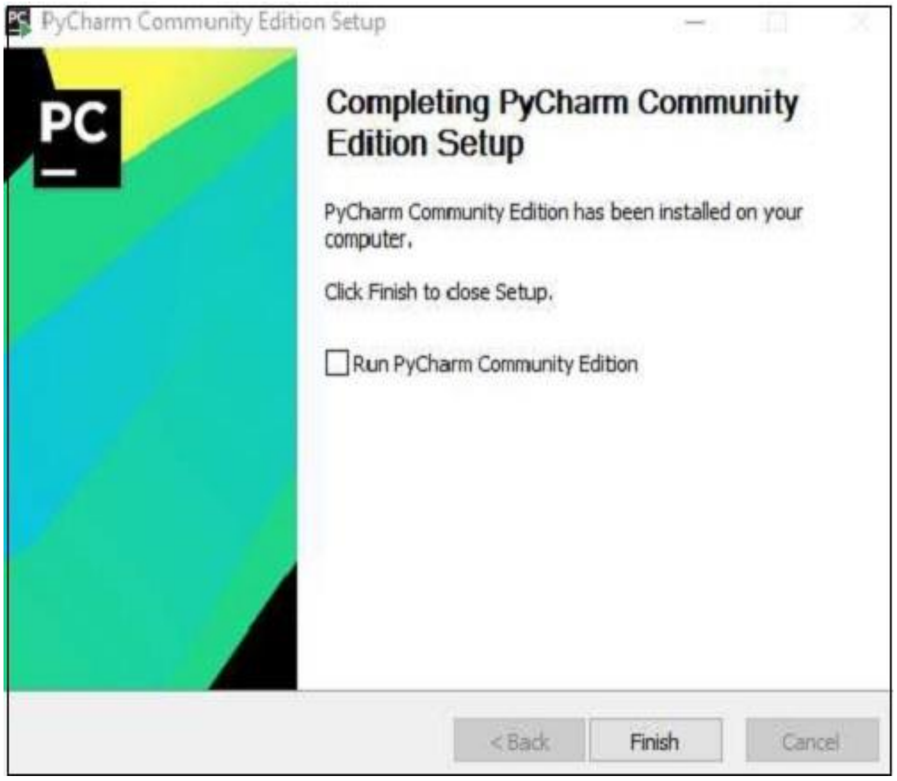
Please wait while PyCharm Community Edition is being installed.

Downloading jre.tar.gz

4408kB (8%) of 55141kB @ 489.7kB/s (1 minute remaining)

Step 4

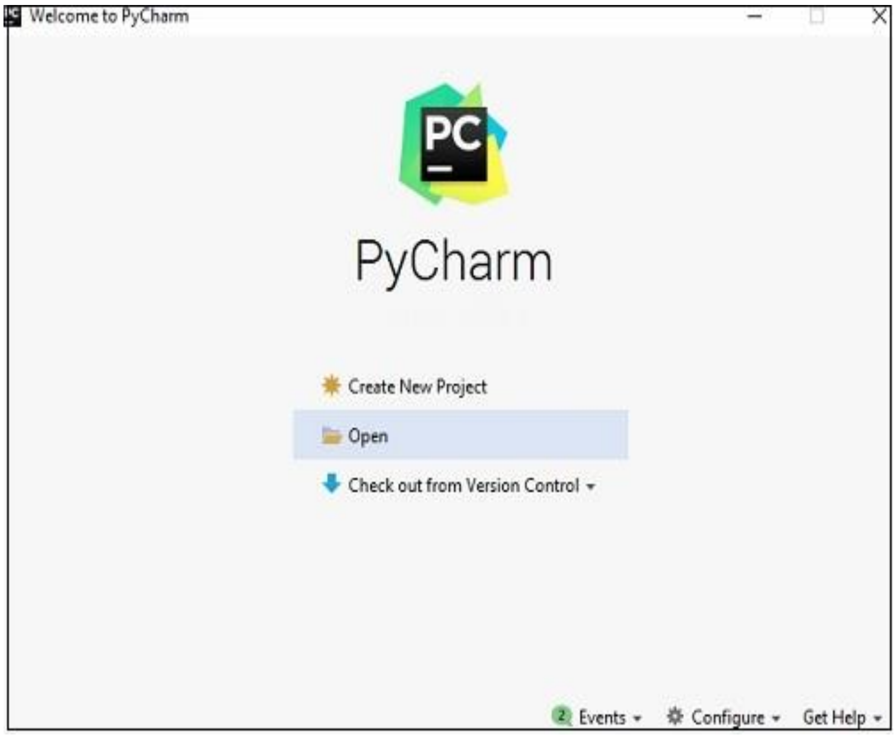
Once the installation is successful, PyCharm asks you to import settings of the existing package if any.

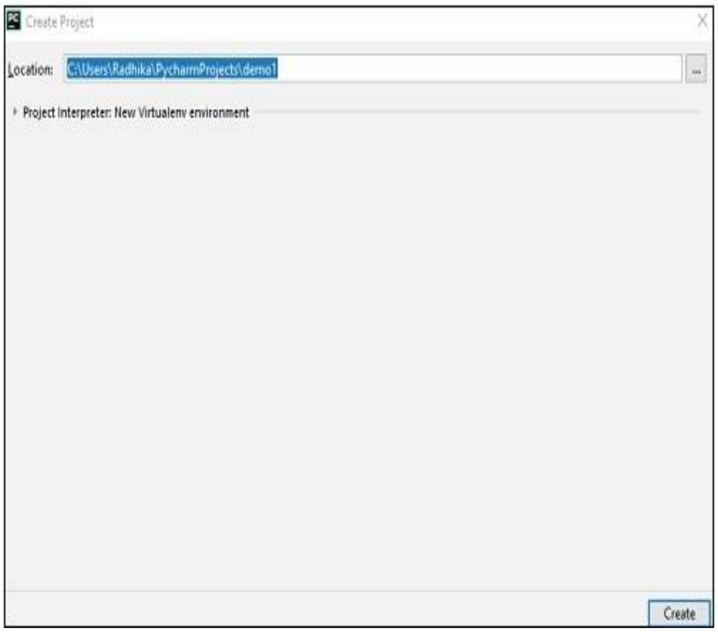

PyCharm Community Edition Setup

Completing PyCharm Community Edition Setup

PyCharm Community Edition has been installed on your computer.

Click Finish to close Setup.

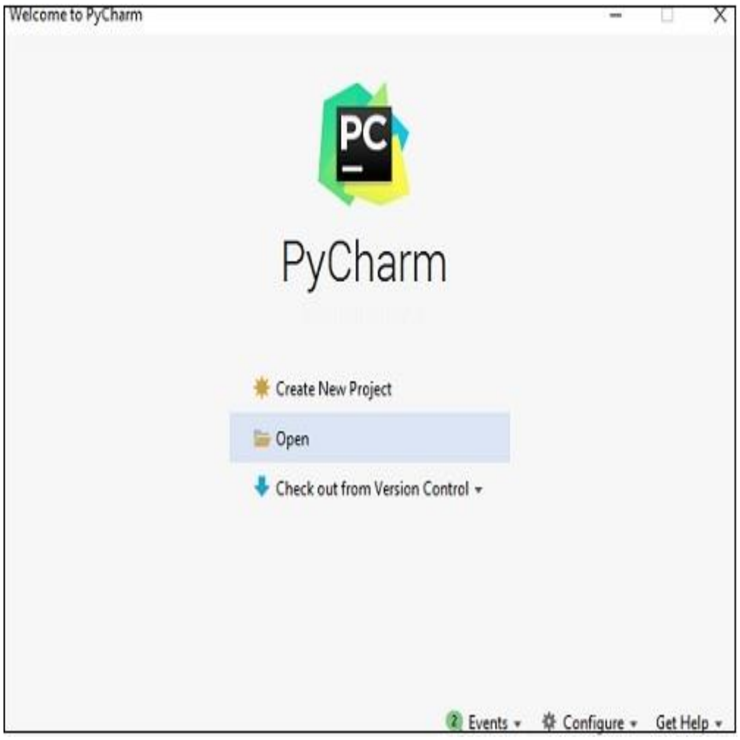

Welcome to PyCharm


Create Project
Location: C:\Users\Radhika\PycharmProjects\demo1
Project Interpreter: New Virtualenv environment
Create

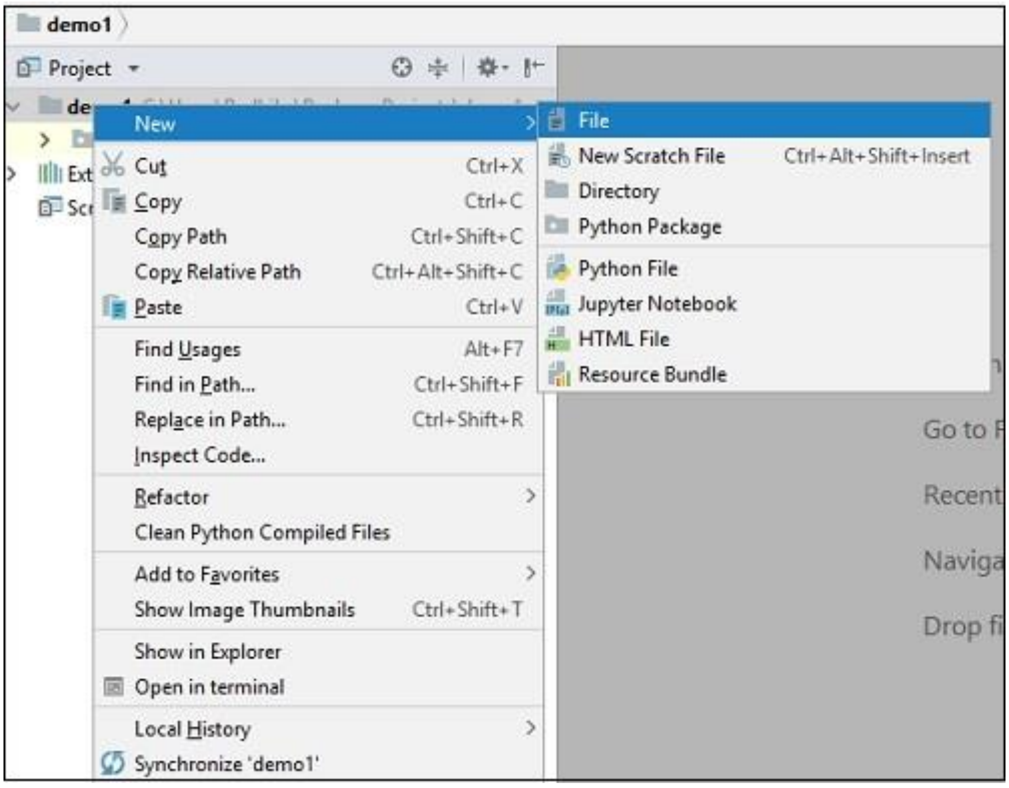
This helps in creating a new project of Python where you can work from the scratch. Note that unlike other IDEs, PyCharm only focusses on working with projects of Python scripting language.

When you launch PyCharm for the first time, you can see a welcome screen with entry points to IDE such as −

- Creating or opening the project
- Checking out the project from version control
- Viewing the documentation
- Configuring the IDE



Recall that in the last chapter, we created a project named demo1 and we will be referring to the same project throughout this tutorial. Now we will start creating new files in the same project to understand the basics of PyCharm Editor.

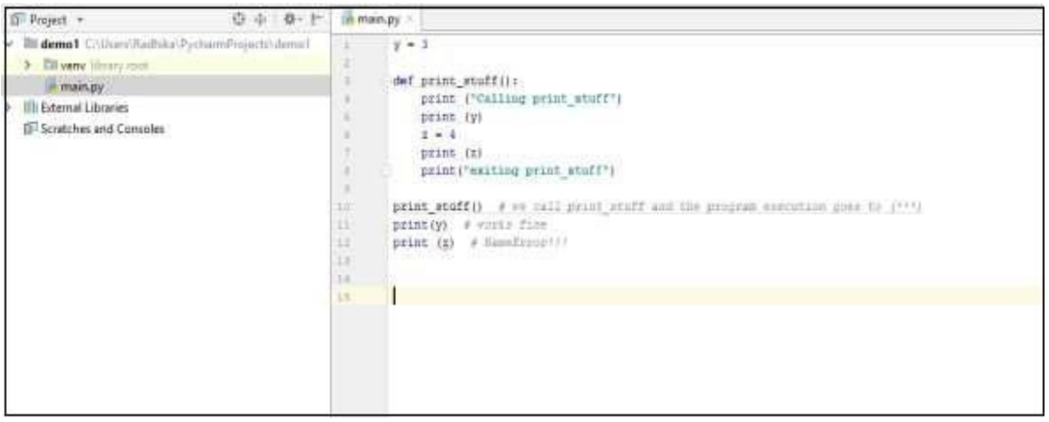


The above snapshot describes the project overview of demo1 and the options to create a new file. Let us create a new file called main.py.

The code included in main.py is as follows −

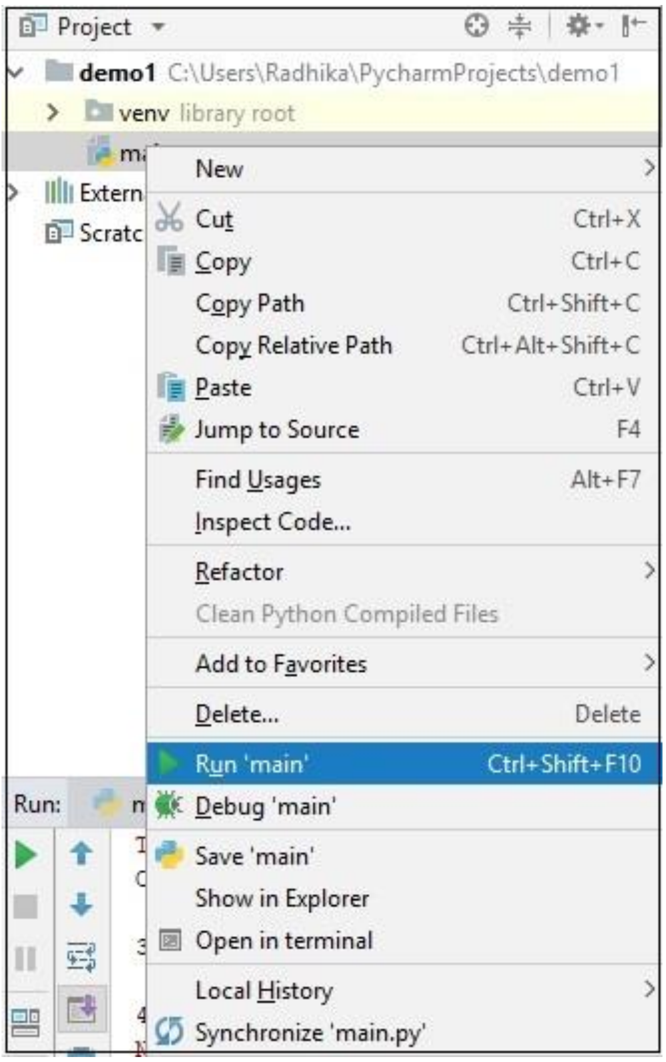
```
y = 3

def print_stuff():
   print ("Calling print_stuff")
   print (y)
   z = 4
   print (z)
   print("exiting print_stuff")

print_stuff() # we call print_stuff and the program execution goes to (***)
print(y) # works fine
print (z) # NameError!!!
```

The code created in the file main.py using PyCharm Editor is displayed as shown below −

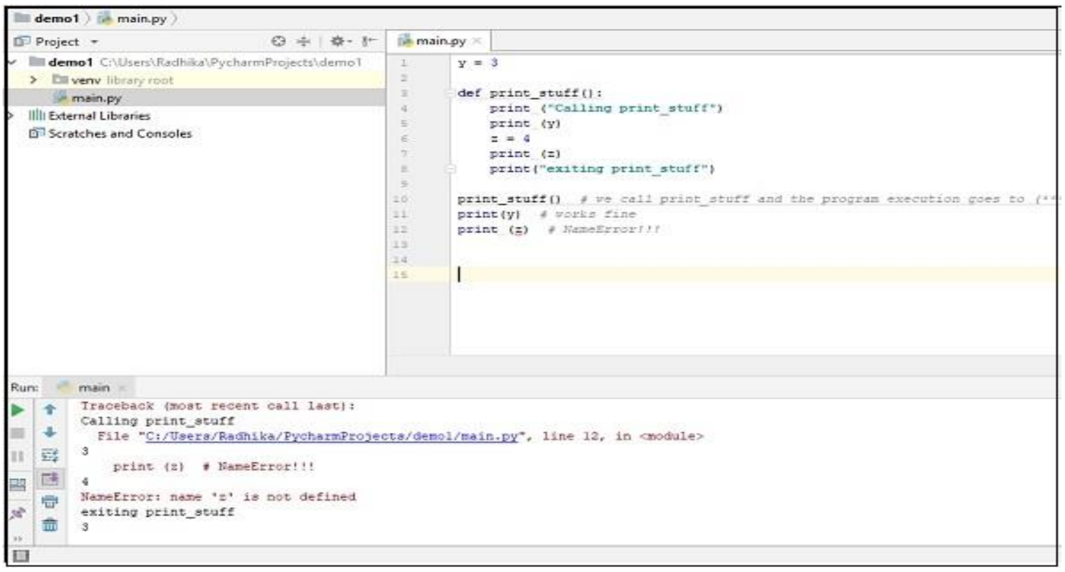


```
y = 3

def print_stuff():
    print ("Calling print_stuff")
    print (y)
    z = 4
    print (z)
    print("exiting print_stuff")

print_stuff()  # we call print_stuff and the program execution goes to (***)
print(y)  # works fine
print (z)  # NameError!!!
```

This code can be run within IDE environment. The basic demonstration of running a program is discussed below −



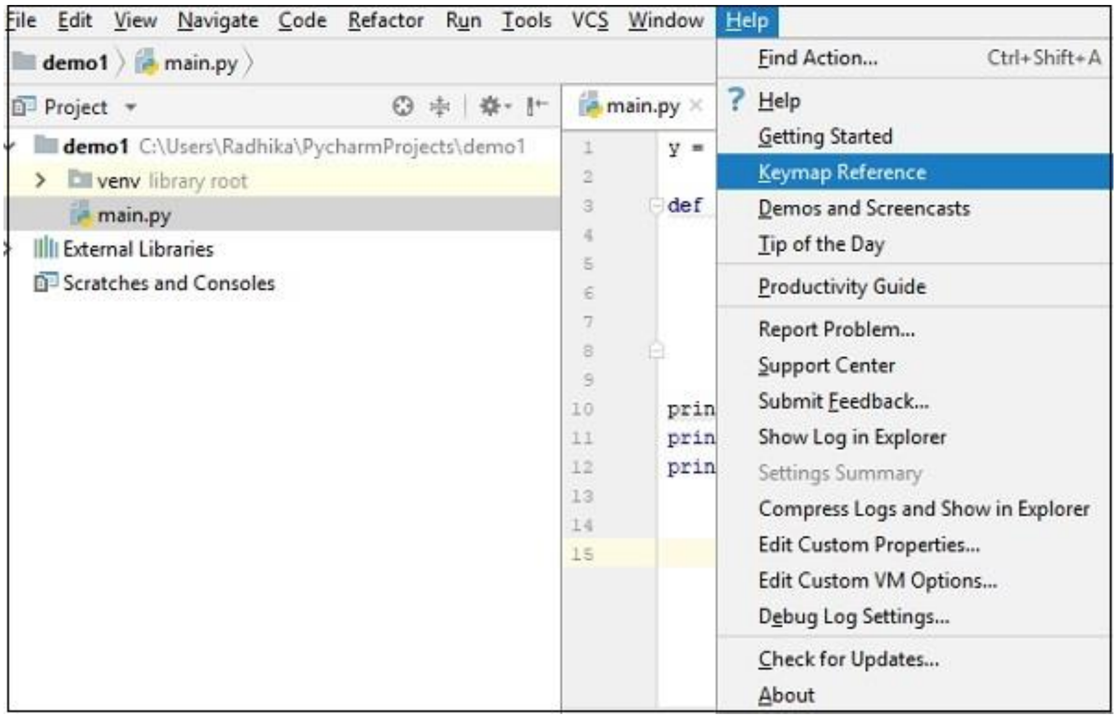
Note that we have included some errors within the specified code such that console can execute the code and display output as the way it is intended to.



```
y = 3

def print_stuff():
    print ("Calling print_stuff")
    print (y)
    z = 4
    print (z)
    print("exiting print_stuff")

print_stuff()  # we call print_stuff and the program execution goes to (**
print(y)  # works fine
print (z)  # NameError!!!
```

```
Traceback (most recent call last):
Calling print_stuff
  File "C:/Users/Radhika/PycharmProjects/demo1/main.py", line 12, in <module>
3
    print (z)  # NameError!!!
4
NameError: name 'z' is not defined
exiting print_stuff
3
```

You can find the list of Keymaps available in the file menu Help -> Keymap Reference as shown in the screenshot given below –

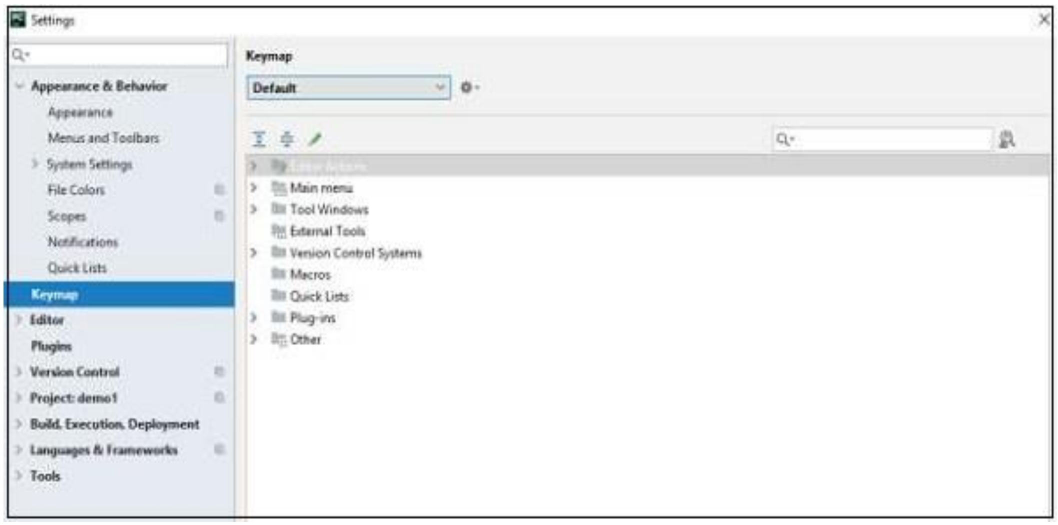


You can find the list of Keymaps and the available shortcuts in PDF format as shown below –

PyCharm
DEFAULT KEYMAP
Editing
Running
Debugging
Navigation
Search/Replace
Usage Search
Refactoring
VCS/Local History
Live Templates
General

Note – The default Keymap for Windows and Linux operating systems is default, while in Mac OS the default Keymap is OSX 10.5.

You can also view the list of Keymaps available using the Settings option in Windows and Linux Operating system (Preferences in Mac OS) as shown in the screenshot given below –



The default Keymap includes various sections for Editor Actions, Main Menu, Tool Windows, External tools, Version Control System, Macros, Quick Lists, Plug-ins and Other options as well.

CHAPTER - 4

DESIGN

The UML diagrams are the way of visualizing a software program using a collection of diagrams. The notation has evolved from the work of Grady Booch, James Rumbaugh, Ivar Jacobson, and the Rational Software Corporation to be used for object-oriented design, but it has since been extended to cover a wider variety of software engineering projects. The Design consist of various design which we have implemented in our system diabetes prediction using machine learning. This system has built with various designs such as data flow diagram, sequence diagram, class diagram, use case diagram, component diagram, activity diagram, state chart diagram, deployment diagram. After doing these various diagrams and based on these diagrams we have done our project.

4.1 Purpose of UML Diagrams

With the use of UML, an appropriate UML development tool, and an application process or methodology, the design and refining of the application is shifted from the development phase to an analysis and design phase. This reduces risk and provides a vehicle for testing the architecture of the system before coding begins. The analysis and design overhead will eventually pay dividends as the system has been user driven, documented and when it's time to start developing, many UML tools will generate skeleton code that will be efficient, object oriented and promote re-use.

Furthermore, the use of UML will help:

- The communication of the desired structure and behavior of a system between analysts, architects, developers, stakeholders, and users.
- The visualization and control of system architecture.
- Promote a deeper understanding of the system, exposing opportunities for simplification and re-use.
- Manage risk.

4.2 Diagrams

There are eight most widely used UML diagrams. They are:

- Class Diagram
- Use Case Diagram
- Sequence Diagram
- Activity Diagram
- Collaboration Diagram
- Deployment Diagram
- State Chart Diagram
- Component Diagram

1. Class Diagram

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modelling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

2. Use Case Diagram

Use case diagrams consists of actors, use cases and their relationships. The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system.

It is defined and created from 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. Hence to model the entire system, a number of use case diagrams are used.

3. Sequence Diagram

A sequence diagram simply depicts interaction between objects in a sequential order i.e., the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems.

4. Activity Diagram

Activity Diagrams are generally used to illustrate the flow of control in a system. We can also use an activity diagram to refer to the steps involved in the execution of a use case. We model sequential and concurrent activities using activity diagrams. So, we basically depict workflows visually using an activity diagram. It focuses on condition of flow and the sequence in which it happens. We describe or depict what causes a particular event using an activity diagram.

5. Collaboration Diagram

Collaboration diagrams which are also known as Communication Diagrams are used to show how objects interact to perform the behavior of a particular use case, or a part of a use case. Along with sequence diagrams, collaboration is used by designers to define and clarify the roles of the objects that perform a particular flow of events of a use case. They are the primary source of information used to determining class responsibilities and interfaces.

6. Deployment Diagram

A deployment diagram is a UML diagram type that shows the execution architecture of a system, including nodes such as hardware or software execution environments, and the middleware connecting them.

Deployment diagrams can be used to model the hardware topology of a system, model the embedded system, model the hardware details for a client/server system, model the hardware details of a distributed application and even in forward, reverse engineering.

7. State Chart Diagram

A State Chart diagram describes a state machine. State machine can be defined as a machine which defines different states of an object, and these states are controlled by external or internal events. State Chart diagrams are useful to model the reactive systems. Reactive systems can be defined as a system that responds to external or internal events

State Chart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists, and it changes when some event is triggered. The most important purpose of State Chart diagram is to model lifetime of an object from creation to termination.

8. Component Diagram

In the Unified Modelling Language, a component diagram depicts how components are wired together to form larger components and or software systems, They are used to illustrate the structure of arbitrarily complex systems. It does not describe the functionality of the system, but it describes the components used to make those functionalities.

Component diagrams are very important from implementation perspective. It can be mostly used in modelling the components of a system, modelling the database schema, executables of an application and even system's source code.

4.3 Sequence Diagram

The Sequence diagram of the project diabetes prediction using machine learning consist of all the various aspects a normal sequence diagram requires. This sequence diagram shows how from starting the model flows from one step to another, like he enters into the system then enters all the symptoms that goes into the system, and here the system by using the algorithms and dataset predicts the result. And the result is sent by the system to the User Interface then the user can see the result through the User Interface. Here the sequence of all the entities are linked to each other where the user gets started with the system.

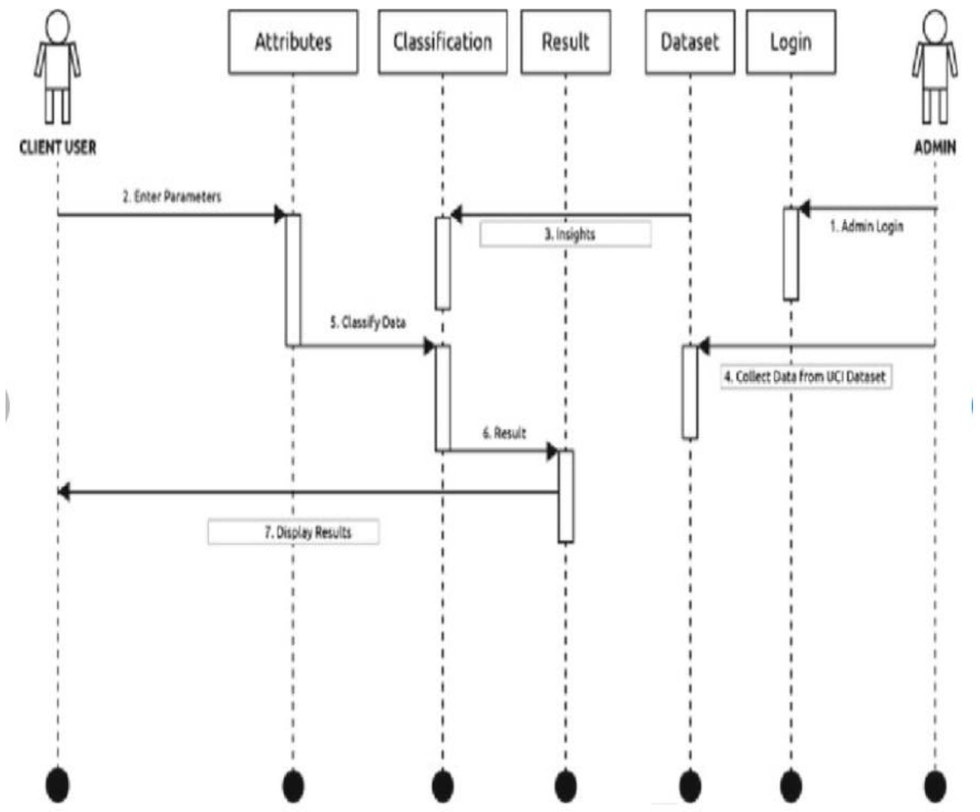


Fig 4.3: Sequence Diagram

4.4 Use case Diagram

The Use case diagram of the project diabetes prediction using machine learning consists of all various aspects a normal sequence diagram requires. This sequence diagram shows how from starting the model flows from one step to another, like he enters into the system then enters all the symptoms that goes into the system, and here the system by using the algorithms and dataset predicts the result. And the result is sent by the system to the User Interface then the user can see the result through the User Interface.

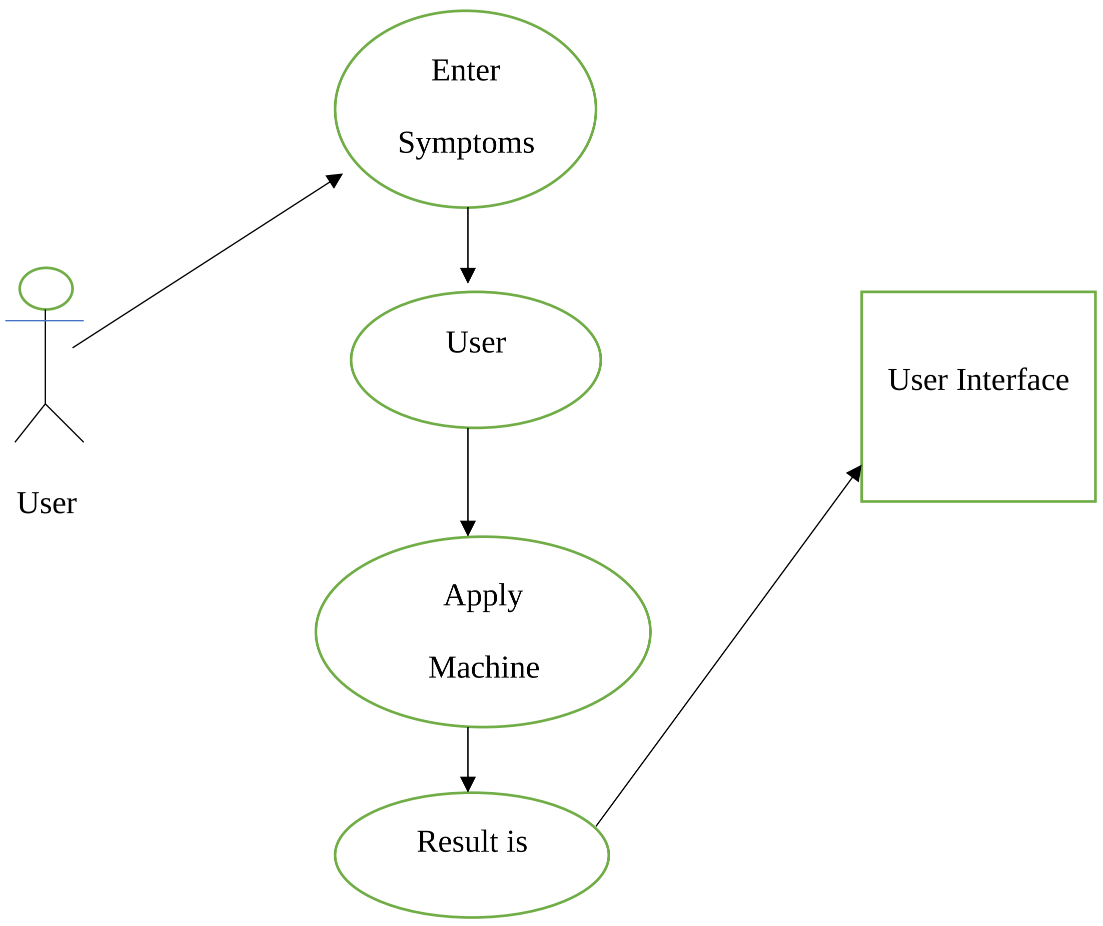


Fig 4.4: Use Case Diagram

4.5 System Architecture

Parkinson disease detection using machine learning predicts the presence of the diabetes for the user based on various symptoms and the information the user gives such as sugar level, skin thickness, age and many more such general information through the symptoms.

The architecture of the system Parkinson disease using machine learning consist of dataset through which we will compare the symptoms of the user and predicts it, then the datasets are transformed into the smaller sets and from there it gets classified based on the classification algorithms later on the classified data is then processed into the machine learning technologies through which the data gets processed and goes in to the Parkinsons disease detection model using all the inputs from the user.

Then after user entering the above information and overall processed data combines and compares in the prediction model of the system and finally predicts whether the user have Parkinson disease or not. An architecture diagram is a graphical representation of a set of concepts, that are part of an architecture, including their principles, elements, and components. The diagram explains about the system software in perception of overview of the system.

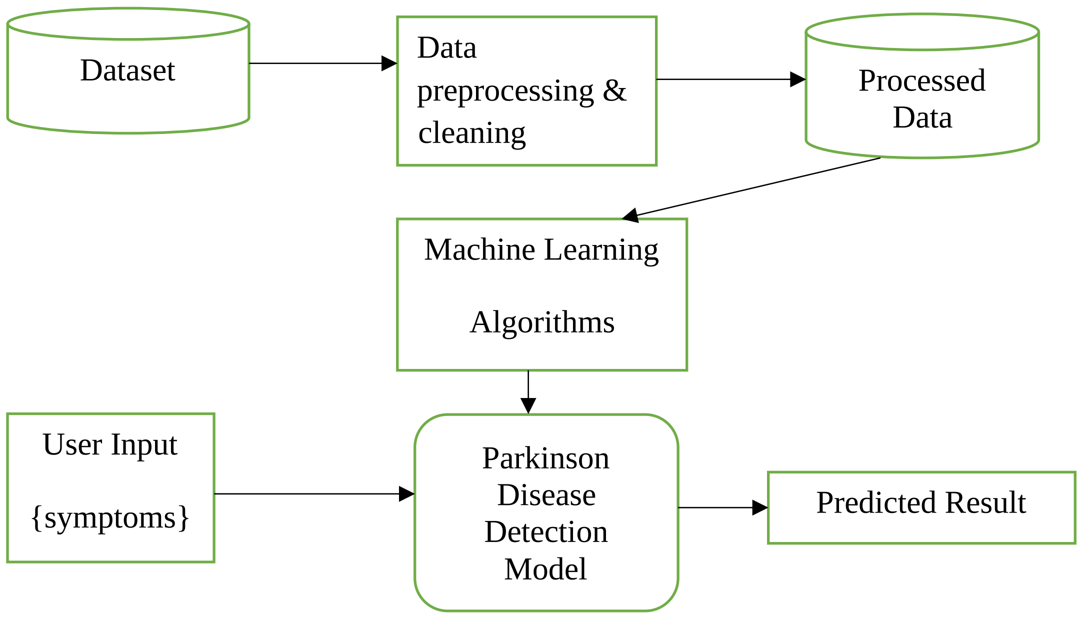

Dataset
Data preprocessing & cleaning
Processed Data
Machine Learning Algorithms
User Input {symptoms}
Parkinson Disease Detection Model
Predicted Result

Fig 4.5: System Architecture

CHAPTER – 5

IMPLEMENTATION

5.1 Overview

The project Parkinson Disease Detection using Machine Learning is developed to overcome diabetes in earlier stages as we all know in competitive environment of economic development the mankind has involved so much that he/she is not concerned about health according to research there are 40% peoples how ignores about general disease which leads to harmful disease later. The Project “Parkinson Disease Detection using Machine Learning” is implemented using python completely. Even the interface of this project is done using python's library interface. Here first the user needs to give the symptoms that are asked they are skin thickness, blood pressure, voice etc. For more accurate result the user needs to enter all the given symptoms, then the system will provide the accurate result. This prediction is basically done with the help of 4 algorithms of machine learning such as Decision Tree, Random Forest, KNN, SVM. From these four algorithms we are calculating the accuracy after calculating we get to know that logistic algorithm gives the most accuracy comparing to the other algorithms. Based on the symptoms that user have entered this model will calculate the accuracy and displays the result by a user interface.

The project is designed user friendly and also secure to use ever user requires a authentication to enter into the system after which it provides the result based on the user input let me explain the complete implementation and working of project step wise below:

Parkinson Disease Detection Using Machining If the user wants to know whether he/she have any Parkinson Disease or the future prediction of that they need to open and give details. System user has to provide the symptoms which he/she is going through based on which we have several algorithms which predict the disease and also displays the percentage of accuracy.

The user needs to enter all the columns of symptoms to get the accurate result, Data collection and dataset preparation This will involve collection of medical information from various sources like hospitals, then pre-processing is applied on dataset which will remove all the unnecessary data and extract important features from data.

Training and experimentation on datasets The Parkinson Disease Detection model will be trained on the dataset of diabetes to do the prediction accurately and produce Confusion matrix. In this project 4 different algorithms were used they are SVM, KNN algorithm, Decision tree algorithm, Random forest algorithm.

Deployment and analysis on real life scenario the trained and tested prediction model will be deployed in a real-life scenario made by the human experts & will be leveraged for further improvement in the methodology.The working and basic explanation of those 4 algorithms Random Forest, Decision Tree, KNN and SVM is given below.

5.2 Algorithms

5.2.1 SVM

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:

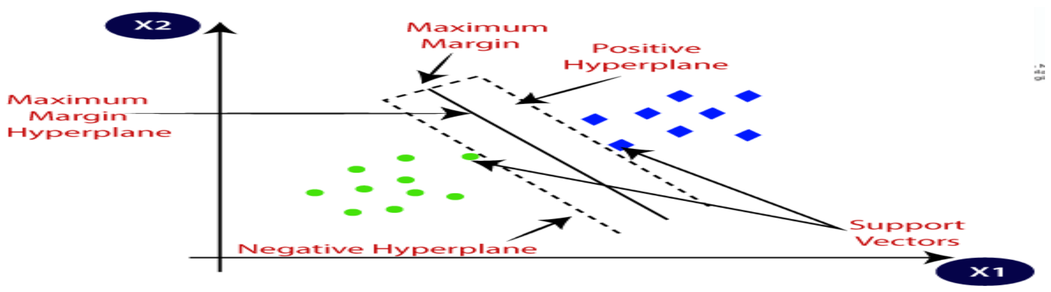


Fig 5.1: SVM

Example: SVM can be understood with the example that we have used in the KNN classifier. Suppose we see a strange cat that also has some features of dogs, so if we want a model that can accurately identify whether it is a cat or dog, so such a model can be created by using the SVM algorithm. We will first train our model with lots of images of cats and dogs so that it can learn about different features of cats and dogs, and then we test it with this strange creature. So as support vector creates a decision boundary between these two data (cat and dog) and choose extreme cases (support vectors), it will see the extreme case of cat and dog.

SVM algorithm can be used for **Face detection, image classification, text categorization,** etc.

Types of SVM

SVM can be of two types

- **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
- **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

Hyperplane and Support Vectors in the SVM algorithm:

Hyperplane: There can be multiple lines/decision boundaries to segregate the classes in n-dimensional space, but we need to find out the best decision boundary that helps to classify the data points. This best boundary is known as the hyperplane of SVM.

The dimensions of the hyperplane depend on the features present in the dataset, which means if there are 2 features (as shown in image), then hyperplane will be a straight line. And if there are 3 features, then hyperplane will be a 2-dimension plane.

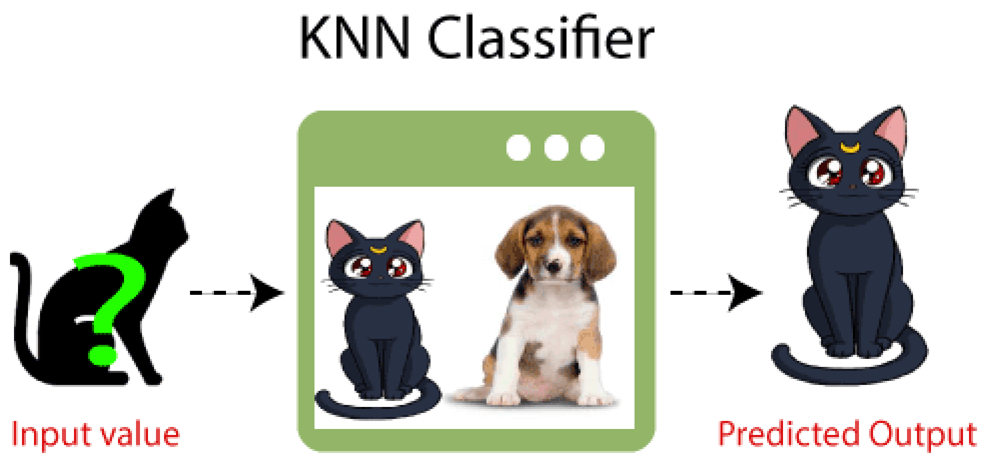
We always create a hyperplane that has a maximum margin, which means the maximum distance between the data points.

Support Vectors: The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector.

5.2.2 KNN

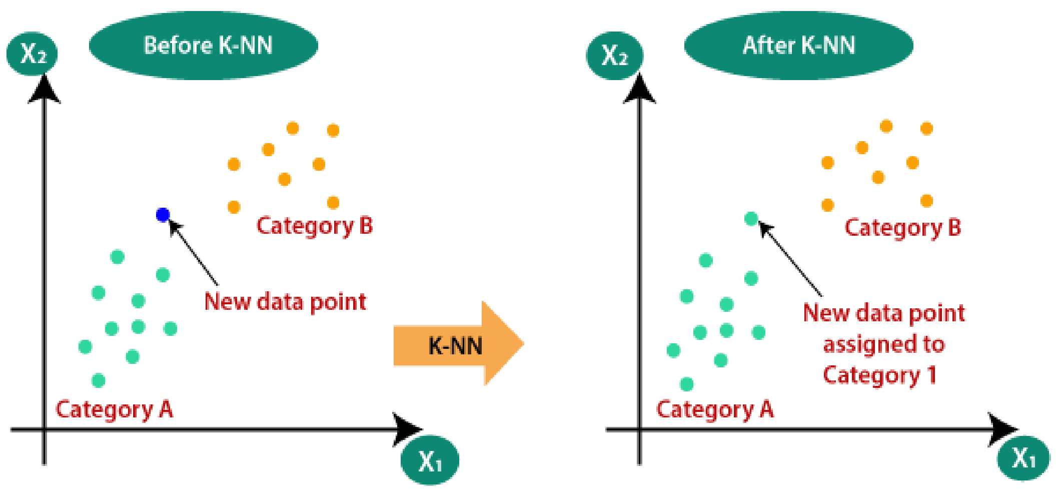
- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
- K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.
- It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

Example: Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.



Why do we need a K-NN Algorithm?

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:



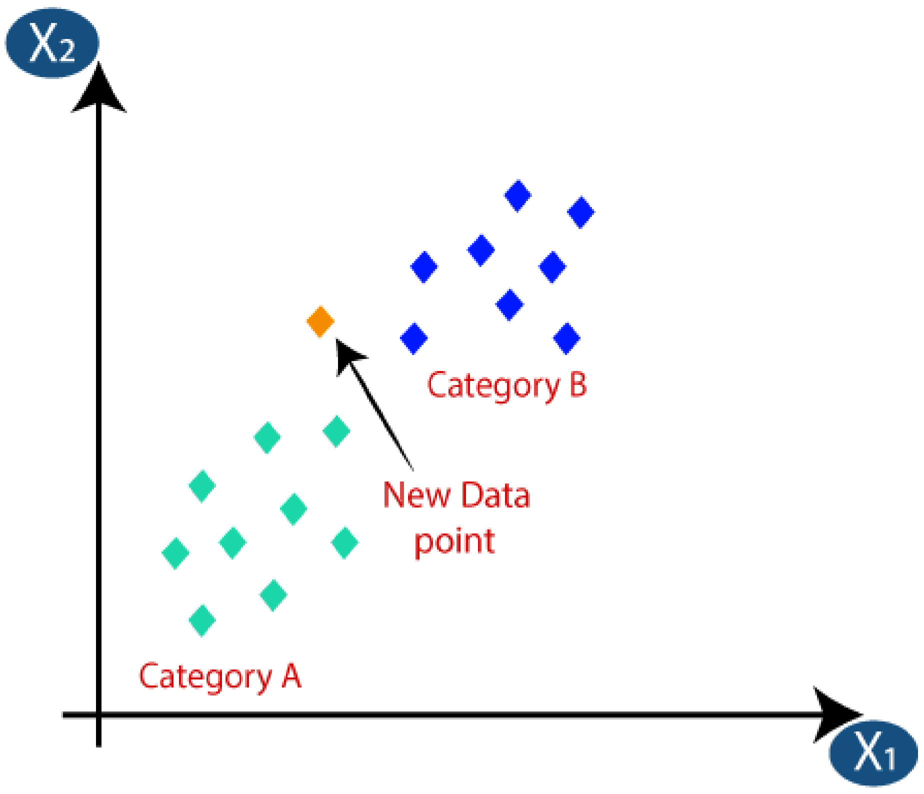
How does K-NN work?

The K-NN working can be explained on the basis of the below algorithm:

- **Step-1:** Select the number K of the neighbors
- **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
- **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.

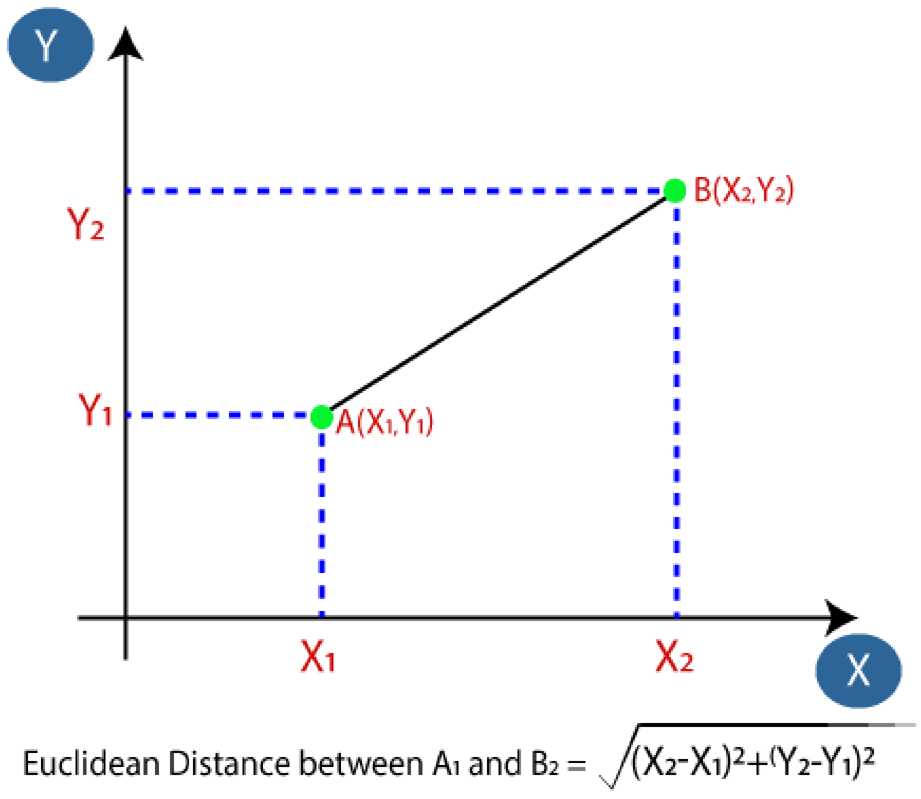
- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
- **Step-6:** Our model is ready.

Suppose we have a new data point and we need to put it in the required category. Consider the below image:



- Firstly, we will choose the number of neighbors, so we will choose the k=5.
- Next, we will calculate the **Euclidean distance** between the data points. The Euclidean distance is the distance between two points, which we have already

studied in geometry. It can be calculated as:


Y
$B(X_2,Y_2)$
Y_2
Y_1
$A(X_1,Y_1)$
X_1
X_2
X
Euclidean Distance between A_1 and B_2 = $\sqrt{(X_2-X_1)^2+(Y_2-Y_1)^2}$

- By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:

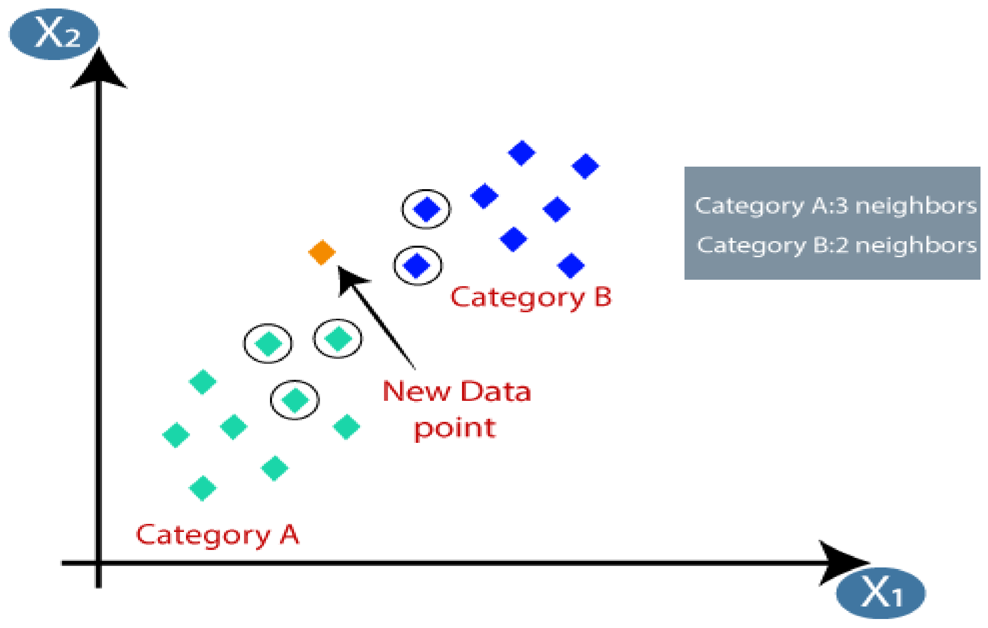


Fig 5.2: KNN

5.2.3 XGBOOST

XGBoost or the Extreme Gradient boost is a machine learning algorithm that is used for the implementation of gradient boosting decision trees. Why decision trees? When we talk about unstructured data like the images, unstructured text data, etc., the ANN models (Artificial neural network) seems to reside at the top when we try to predict. While when we talk about structured/semi-structured data, decision trees are currently the best. XGBoost was basically designed for improving the speed and performance of machine learning models greatly, and it served the purpose very well.

Working of XGBoost Algorithm

The XGBoost is having a tree learning algorithm as well as linear model learning, and because of that, it is able to do parallel computation on a single machine. This makes it 10 times faster than any of the existing gradient boosting algorithms. The XGBoost and the GBMs (i.e. Gradient Boosting Machines) uses tree methods by using the gradient descent architecture. The area where XGBoost leaves the other GBMs behind is the area of system optimization and enhancements over the algorithms. Let us see those in detail:

1. System Optimization

- **Tree Pruning –** The XGBoost algorithm uses the depth-first approach, unlike the stopping criterion for tree splitting used by GBMS, which is greedy in nature and it also depends upon the negative loss criterion. The XGBoost instead uses the max depth feature/parameter, and hence it prunes the tree in a backward direction.
- **Parallelization –** The process of sequential tree building is done using the parallelized implementation in the XGBoost algorithm. This is made possible due to the outer and inner loops that are interchangeable. The outer loop lists the leaf nodes of a tree, while the inner loop will calculate the features. Also, in order for the outer loop to start, the inner loop must get completed. This process of switching improves the performance of the algorithm.
- **Hardware Optimization –** Hardware optimization was also considered during the design of the XGBoost algorithm. Internal buffers are allocated for each of the threads to store the gradient statistics.

2. Algorithmic Enhancements

- **Awareness of Sparsity –** XGBoost is known to handle all different types of sparsity patterns very efficiently. This algorithm learns the nest missing value by seeing the training loss.
- **Regularization –** In order to prevent overfitting, it corrects more complex models by implementing both the LASSO (also called L1) and Ridge regularization (also called L2).
- **Cross-Validation –**It is having built-in cross-validation features that are being implemented at each iteration in the model creation. This prevents the need to calculate the number of boosting iterations needed.
- **Distributed Weighted Quantile Sketch –** It uses the distributed weighted quantile sketch to get the optimal number of split points among the weighted datasets.

Features of XGBoost

Although XGBoost was designed for improving the speed and performance of machine learning models greatly, it does offer a good number of advanced features as well.

Model Features:

The features such as that of a sci-kit learn regularization, and R language implementation is supported by XGBoost.

5.2.4 Random Forest

Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction

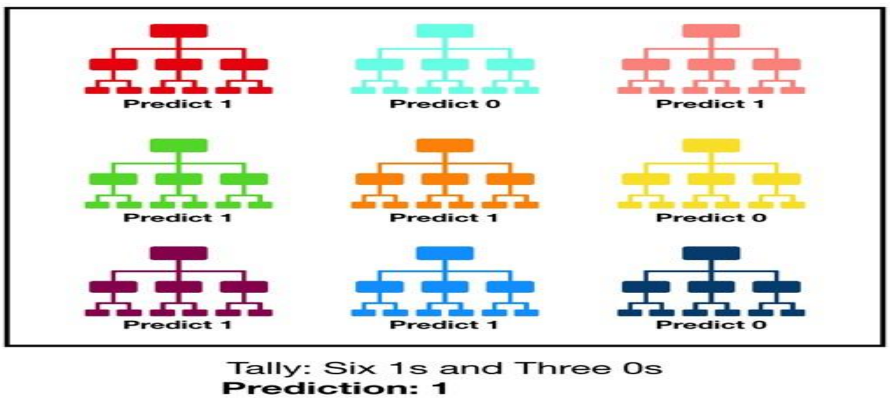


Fig 5.3: Visualization of a Random Forest Model Making a Prediction

The fundamental concept behind random forest is a simple but powerful one. The wisdom of crowds. In data science speak, the reason that the random forest model works so well is a large number of relatively uncorrelated models (trees) operating as committee will outperform any of the individual constituent models. The low correlation between models is the key. Just like how investments with low correlations (like stocks and bonds) come together to form a portfolio that is greater than the sum of its parts, uncorrelated models can produce ensemble predictions that are more accurate than any of the individual predictions. The reason for this wonderful effect is that the trees protect each other from their individual errors (as long as they don't constantly all err in the same direction). While some trees may be wrong, many other trees will be right, so as a group the trees are able to move in the correct direction. So the prerequisites for random forest to perform well are:

1. There needs to be some actual signal in our features so that models built using those features do better than random guessing.
2. The predictions (and therefore the errors) made by the individual trees need to have low correlations with each other.

5.3 Libraries Used

Pandas

Pandas is a fast, powerful, flexible, and easy to use open-source data analysis and manipulation tool, built on top of the Python programming language. The pandas is used in the entire project and the command to install the package is pip install pandas.

import pandas as pd

Matplotlib

Matplotlib is a collection of command style functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc. This is used in the project and the command to install the package is pip install matplotlib.

import matplotlib.pyplot as plt

NumPy

NumPy is a general-purpose array-processing package. It provides a high performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. This is used in the project and the command to install the package is pip install NumPy.

import numpy as np

SK Learn

The train-test split procedure is used to estimate the performance of machine learning algorithms when they are used to make predictions on data not used to train the model. It is a fast and easy procedure to perform, the results of which allow you to compare the performance of machine learning algorithms for your predictive modeling problem. Although simple to use and interpret, there are times when the procedure should not be used, such as when you have a small dataset and situations where additional configuration is required, such as when it is used for classification and the dataset is not balanced.

● Train Dataset: Used to fit the machine learning model.

● Test Dataset: Used to evaluate the fit machine learning model.

from sklearn.model_selection import train_test_split

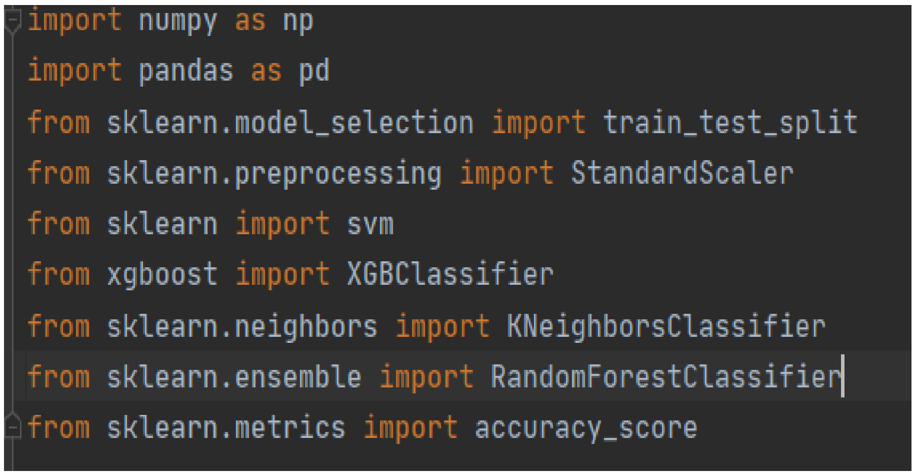
5.4 Code Implementation

Below are the some steps required to practice Python Machine Learning Project

Step 1:

Importing the necessary Librarires and modules:

All the required libraries like Sklearn and Modules like Numpy , Pandas are imported into the pycharm. The implementation of these will be as follows:

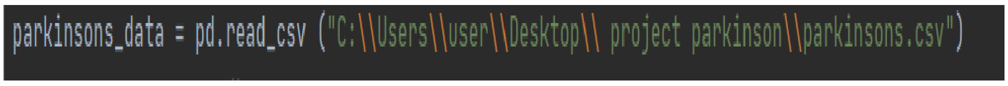


```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn import svm
from xgboost import XGBClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
```

Fig 5.4 : Importing the libraries

Step 2:

Loading the data from CSV

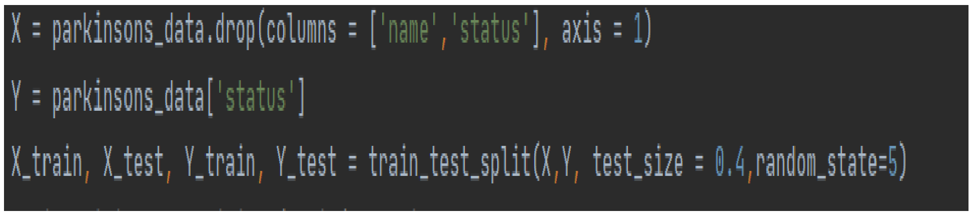


```
parkinsons_data = pd.read_csv ("C:\\Users\\user\\Desktop\\ project parkinson\\parkinsons.csv")
```

Fig 5.5 : Loading the data

Step 3:

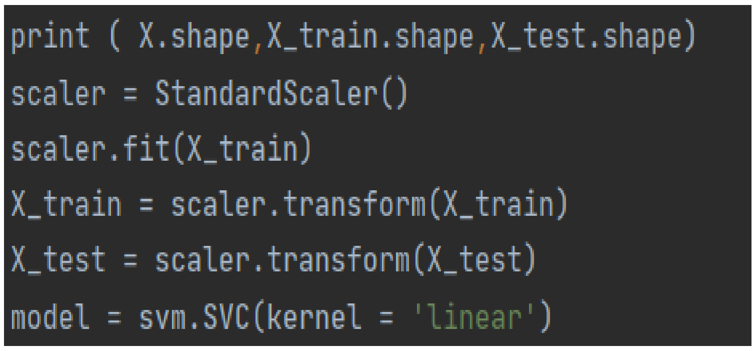
It is important to know about the information of each and every attribute,such information can be easily predicted and is analysed as follows: Pandas head() method is used to return top n (5 by default) rows



```
X = parkinsons_data.drop(columns = ['name','status'], axis = 1)
Y = parkinsons_data['status']
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size = 0.4,random_state=5)
```

Step 4:

Visualization



```
print ( X.shape,X_train.shape,X_test.shape)
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
model = svm.SVC(kernel = 'linear')
```

CHAPTER - 6

TESTING

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not. Testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements. It involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration.

It involves identifying bug/error/defect in a software without correcting it. Normally professionals with a quality assurance background are involved in bugs entification. Testing is performed in the testing phase. We have different types of testing like unit testing, integration testing, validation testing, system testing. manual testing.

6.1 Unit Testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration

6.2 Integration Testing

Integration tests are designed to test integrated software components to determine if they actually run as one program Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

As a rule, system testing takes, as its input, all the "integrated" software components that have successfully passed integration testing and also the software system itself integrated with any applicable hardware system. System testing is a more limited type of testing; it seeks to detect defects both within the "inter assemblages" and also within the system as a whole. System testing is performed on the entire system in the context of a Functional Requirement Specification (FRS) or System Requirement Specification (SRS).

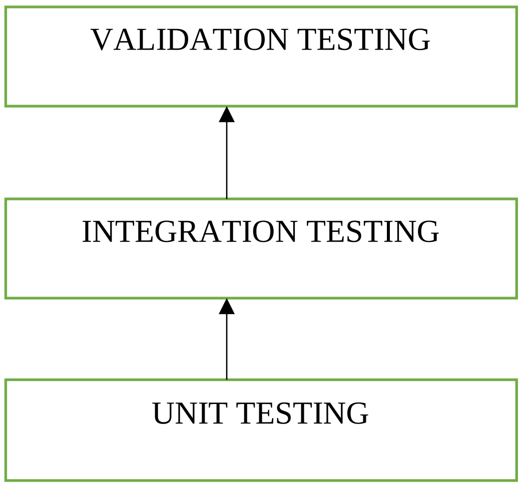
6.3 Validation Testing

An engineering validation test (EVT) is performed on first engineering prototypes, to ensure that the basic unit performs to design goals and specifications.

It is important in identifying design problems and solving them as early in the design cycle as possible, is the key to keeping projects on time and within budget. Too often, product design and performance problems are not detected until late in the product development cycle-when the product is ready to be shipped. The old adage holds true: It costs a penny to make a change in engineering, a dime in production and a dollar after a product is in the field.

Verification is a Quality control process that is used to evaluate whether or not a product, service, or system complies with regulations, specifications, or conditions imposed at the start of a development phase. Verification can be in development, scale-up, or production. This is often an internal process. Validation is a Quality assurance process of establishing evidence that provides a high degree of assurance that a product, service, or system accomplishes its intended requirements, This often involves acceptance of fitness for purpose with end users and other product stakeholders.

The testing process overview is as follows:



6.4 System Testing

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic.

As a rule, system testing takes, as its input, all the "integrated" software components that have successfully passed integration testing and also the software system itself integrated with any applicable hardware system. System testing is a more limited type of testing; it seeks to detect defects both within the "inter assemblages" and also within the system as a whole. System testing is performed on the entire system in the context of a Functional Requirement Specification (FRS) or System Requirement Specification (SRS).

6.5 Manual Testing

Manual testing is the process of manually testing software for defects. It requires a tester to play the role of an end user whereby they use most of the application's features to ensure correct behavior. To guarantee completeness of testing, the tester often follows a written test plan that leads them through a set of important test cases.

Advantages:

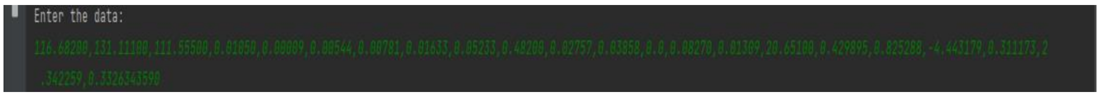
- Low-cost operation as no software tools are used.

- Most of the bugs are caught by manual testing.
- Humans observe and judge better than the automated tools.

RESULT

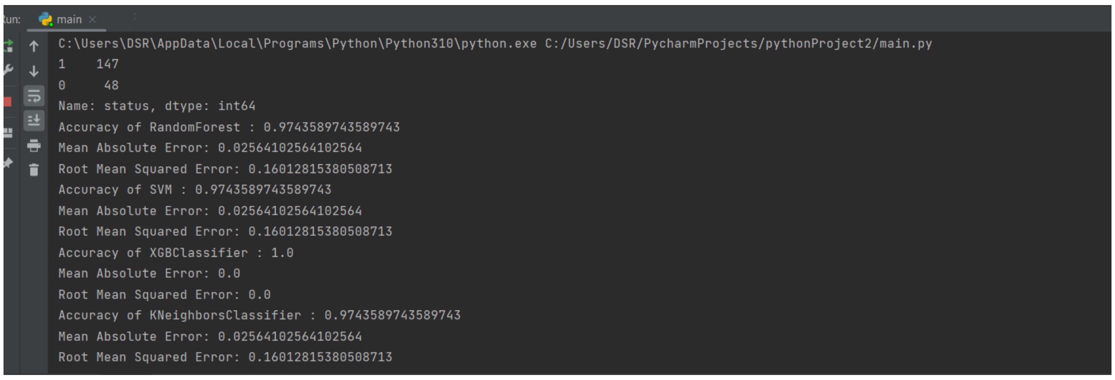
After writing the code related to the four Algorithms, we will get the accuracy of Parkinson disease and it will display whether a person is affected by the Parkinson disease or not.

Input:

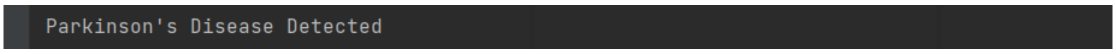


```
Enter the data:
116.68200,131.11100,111.55500,0.01050,0.00009,0.00544,0.00781,0.01633,0.05233,0.48200,0.02757,0.03858,0.0,0.08270,0.01309,20.65100,0.429895,0.825288,-4.443179,0.311173,2
.342259,0.3326343598
```

Accuracy of Four Algorithms:

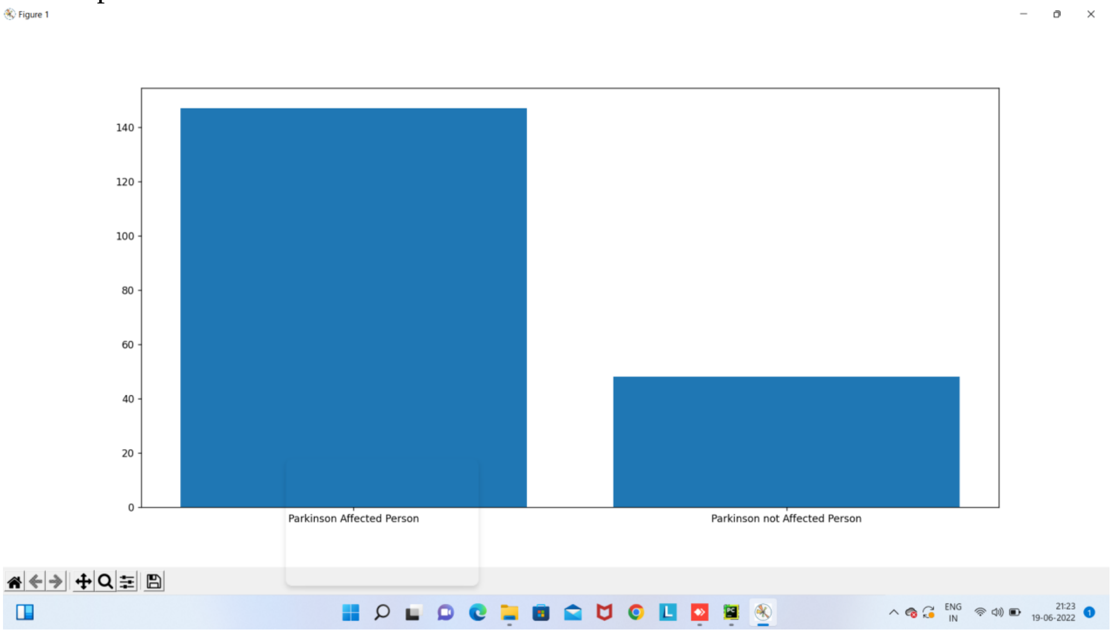


```
C:\Users\DSR\AppData\Local\Programs\Python\Python310\python.exe C:/Users/DSR/PycharmProjects/pythonProject2/main.py
1    147
0     48
Name: status, dtype: int64
Accuracy of RandomForest : 0.9743589743589743
Mean Absolute Error: 0.02564102564102564
Root Mean Squared Error: 0.16012815380508713
Accuracy of SVM : 0.9743589743589743
Mean Absolute Error: 0.02564102564102564
Root Mean Squared Error: 0.16012815380508713
Accuracy of XGBClassifier : 1.0
Mean Absolute Error: 0.0
Root Mean Squared Error: 0.0
Accuracy of KNeighborsClassifier : 0.9743589743589743
Mean Absolute Error: 0.02564102564102564
Root Mean Squared Error: 0.16012815380508713
```



```
Parkinson's Disease Detected
```

BarGraph:



This is the result of our project Whether the person is having Parkinson Disease or not. It is showing YES in this case.Here it says that the person is not affected by the Parkinson Disease.

CONCLUSION

So, Finally We conclude by saying that this project Parkinson Disease Detection using machine learning is very much useful in everyone's day to day life and it is mainly more important for the healthcare sector, because they are the one that daily uses these systems to detect the Parkinson disease of the patients based on their general. information and their symptoms that they are been through. Now a day's health industry plays major role in curing the diseases of the patients so this is also some kind of help for the health industry to tell the user and also it is useful for the user in case he/she doesn't want to go to the hospital or any other clinics, so just by entering the symptoms or information the user can get to know whether he/she having Parkinson disease or not, the health industry can also get benefit from this system by just asking the symptoms from the user and entering in the system and in just few seconds they can tell the exact and up to some extent the accurate values.

If health industry adopts this project, then the work of the doctors can be reduced. The Parkinson Disease is to provide Detect for the disease that when unchecked and sometimes ignored can turns into fatal disease and cause lot of problem to the patient and as well as their family members.

REFERENCES

[1] D. Heisters, “Parkinson’s: symptoms, treatments and research,” British Journal of Nursing *where the paper published*, vol. 20, no. 9, pp. 548–554, 2011.

[2] “Parkinson Disease Detection” , Kaggle.com
Available in : https://www.kaggle.com/nidaguler/park...

[3] L., Naranjo, C.J., Pérez, J., Martín, Y., Campos-Roca, "A two-stage variable selection and classification approach for Parkinson’s disease detection by using voice recording replications", COMPUT METH PROG BIO, vol. 142, pp.147-156, 2017.

[4] H., Lee, L.,Guan, I., Lee,"Video analysis of human gait and posture to determine neurological disorders", EURASIP J. Image Video PROC., vol. 1, pp. 380867, 2008.