

A
PROJECT REPORT
ON
“HEART DISEASE PREDICTION”
Under
The Guidance
OF
“EICT Academy ,IIT Roorkee”



Submitted By,

Nikhil Laxman Palve
nikpalve@gmail.com
9922995092

Submitted To,

“EICT Academy ,IIT Roorkee”

ABSTRACT

Disease Prediction using Machine Learning is a system which predicts the disease based on the information or the symptoms he/she enter into the system and provides the accurate results based on that information. If the patient is not much serious and the user just wants to know the type of disease, he/she has been through. It is a system which provides the user the tips and tricks to maintain the health system of the user and it provides a way to find out the disease using this prediction.

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 and all other useful information the user can get to know the disease he/she is suffering from and 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 diseases.

This Disease Prediction Using Machine Learning is completely done with the help of Machine Learning and Python Programming language with Tkinter Interface for it and also using the dataset that is available previously by the hospitals using that we will predict the disease.

BRIEF ON THE PROJECT:-

Cardiovascular diseases (CVDs) are the number 1 cause of death globally, taking an estimated 17.9 million lives each year, which accounts for 31% of all deaths worldwide. Heart failure is a common event caused by CVDs and this dataset contains 12 features that can be used to predict mortality by heart failure. Most cardiovascular diseases can be prevented by addressing behavioural risk factors such as tobacco use, unhealthy diet and obesity, physical inactivity and harmful use of alcohol using population-wide strategies.

People with cardiovascular disease or who are at high cardiovascular risk (due to the presence of one or more risk factors such as hypertension, diabetes, hyperlipidaemia or already established disease) need early detection and management wherein a machine learning model can be of great help.

DELIVERABLES OF THE PROJECT:-

The Heart failure.csv dataset can get from a URL source or directly from a kaggle website. After getting a dataset first all process like data cleaning, Data Reduction, Data Analysis, Feature Engineering etc should be done. By observing we can assume that a classification modeling can be done on this dataset. And after successful modeling we can get accuracy score of the trained and test model. From this modeling score we can easily assume how a heart can get fail. For example: A man of age 75 not having anemia, 582 creatinine, with 20 ejection fraction, having high blood pressure, platelets of 265000, serum creatinine of 1.9 , serum sodium of 130, and and no smoking this patient have a chance of heart failure and can die. If a man of age 49, having anemia, having a 80 creatinine, no diebities, having 30 ejection fraction, having high blood pressure with 427000 platelets, having serus cretinine 1, serum sodium 138, and also she don't smokes and she have no chance of death from heart failure. This type of data when trained we can get a model from which we can further assume that a man or women have a tendency of heart failure or not and he or she have risk of death or not.

The more accuracy we get the better the model we have created. This model can easily be used for example a Insurance Company, where they can find if there is any chance of pre mature death from heart failure, before giving a policy to someone. This can be done by asking a question to someone like:

“ Do you have anemia? What is your Blood Platelets? Do you have Diabetes? What is your Cretinine level? What is your Ejection Fraction? Do you have blood pressure or not? What is your Serum Sodium and Cretinine? What is your age and do you smoke or not?”

This type of Question can be asked to a person and the answer get from them can help him predicting the chance of heart failure in future. For this puropose the Algorithm we made can be very useful.

Dataset Source:- <https://www.kaggle.com/andrewmvd/heart-failure-clinical-data>

CONTENTS

ABSTRACT	II
LIST OF FIGURES	V
LIST OF TABLES	VII
1. INTRODUCTION	
1.1. DISEASE PREDICTION	1
1.2. PROBLEM DEFINITION	2
1.3. PROJECT PURPOSE	3
1.4. PROJECT FEATURES	3
2. LITERATURE SURVEY	
2.1. MACHINE LEARNING	5
2.1.1. FEATURES OF MACHINE LEARNING	6
2.2. EXISTING SYSTEM	8
2.3. PROPOSED SYSTEM	9
2.4. SOFTWARE DESCRIPTION	10
2.4.1. PYTHON	10
2.4.2. BENEFITS OF PYTHON	10
2.4.3. TKINTER INTERFACE	11
3. REQUIREMENT ANALYSIS	
3.1. FUNCTIONAL REQUIREMENTS	12
3.2. NON-FUNCTIONAL REQUIREMENTS	13
3.3. HARDWARE REQUIREMENTS	15
3.4. SOFTWARE REQUIREMENTS	15
4. DESIGN	
4.1. DESIGN GOALS	16
4.2. SYSTEM ARCHITECTURE	17
4.3. DATA FLOW DIAGRAM	18

4.4. CLASS DIAGRAM	19
4.5. SEQUENCE DIAGRAM	20
4.6. USE CASE DIAGRAMS	21
4.7. ACTIVITY DIAGRAM	22
4.8. COMPONENT DIAGRAM	23
4.9. STATE CHART DIAGRAM	24
4.10. COLLABORATION DIAGRAM	25
4.11. DEPLOYMENT DIAGRAM	26
4.12. INTERFACE AND FRAMEWORK DIAGRAM	27
5. IMPLEMENTATION	
5.1 OVERVIEW	29
5.2 LOGISTIC REGRESSION ALGORITHM	31
5.3 KNN ALGORITHM	44
6. TESTING	
6.1 UNIT TESTING	56
6.2 INTEGRATION TESTING	56
6.3 VALIDATION TESTING	56
6.4 SYSTEM TESTING	57
6.5 TESTING OF INITIALIZATION AND UI COMPONENTS	58
7. CONCLUSION AND FUTURE ENHANCEMENT	
7.1 CONCLUSION	61
7.2 FUTURE ENHANCEMENT	61
REFERENCES	62

LIST OF FIGURES

Figure No	Figure Name	Page No
2.1.1	Traditional Programming vs Machine Learning	7
2.1.2	Machine Learning Model	7
4.2	System Architecture	17
4.3	Data Flow Diagram	18
4.4	Class Diagram	19
4.5	Sequence Diagram	20
4.6	Use Case Diagram	21
4.7	Activity Diagram	22
4.8	Component Diagram	23
4.9	State Chart Diagram	24
4.10	Collaboration Diagram	25
4.11	Deployment Diagram	26
4.12	Interface and Framework Diagram	27
5.2.1	Logistic Regression	31
5.3.1	KNN Classifier	44
6.1	The Testing Process	57

LIST OF TABLES

Table No	Table Name	Page No
6.5.1	Test Case for User Registration	58
6.5.2	Test Case for User Login	59
6.5.3	Test Case for Prediction Result	60

CHAPTER 1

INTRODUCTION

1.1 DISEASE PREDICTION

Disease Prediction using Machine Learning is a system which predicts the disease based on the information provided by the user. It also predicts the disease of the patient or the user based on the information or the symptoms he/she enter into the system and provides the accurate results based on that information. If the patient is not much serious and the user just wants to know the type of disease, he/she has been through. It is a system which provides the user the tips and tricks to maintain the health system of the user and it provides a way to find out the disease using this prediction. 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 and all other useful information the user can get to know the disease he/she is suffering from and 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 diseases. This DPUML is previously done by many other organizations but our intention is to make it different and beneficial for the users who are using this system. This Disease Prediction Using Machine Learning is completely done with the help of Machine Learning and Python Programming language with Tkinter Interface for it and also using the dataset that is available previously by the hospitals using that we will predict the disease. Now a day's doctors are adopting many scientific technologies and methodology for both identification and diagnosing not only common disease, but also many fatal diseases. The successful treatment is always attributed by right and accurate diagnosis. Doctors may sometimes fail to take accurate decisions while diagnosing the disease of a patient, therefore disease prediction systems which use machine learning algorithms assist in such cases to get accurate results. The project disease prediction using machine learning is developed to overcome general disease 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 who ignores about general disease which leads to harmful disease later. The main reason of ignorance is laziness to consult a doctor and time concern the peoples have involved themselves so much that they have no time to take an appointment and consult the doctor which later results into fatal disease. According to research there are 70% peoples in India suffers from general disease and 25% of peoples face death due to early ignorance the main motive to develop this project is that a user can sit at their convenient place and have a check-up of their health the UI is designed in such a simple way that everyone can easily operate on it and can have a check-up.

1.2 PROBLEM DEFINITION

Now a day's 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 also 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 there 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 many 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 disease he/she is encountered with so this will lead to various diseases in future. So, to avoid that and get to know the disease in early stages of the symptoms this disease prediction will help a lot to the various people's ranging from children to teenagers to adults and also the senior citizens.

1.3 PROJECT OBJECTIVE

To Create Classification filter using (Logistic Regression & KNN Classification Algorithm) to predict Heart Failure .Compare Performance of the filter.

The purpose of making this project called “Disease Prediction Using Machine Learning” is to predict the accurate disease of the patient using all their general information’s and also the symptoms. Using this information, there we will compare with our previous datasets of the patients and predicts the disease of the patient he/she is been through. If this Prediction is done at the early stages of the disease with the help of this project and all other necessary measure the disease can be cured and in general this prediction system can also be very useful in health industry. If health industry adopts this project then the work of the doctors can be reduced and they can easily predict the disease of the patient. The general purpose of this Disease prediction is to provide prediction for the various and generally occurring diseases that when unchecked and sometimes ignored can turn into fatal disease and cause a lot of problem to the patient and as well as their family members. This system will predict the most possible disease based on the symptoms. The health industry in information yet and knowledge poor and this industry is very vast industry which has a 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 people who are in the need.

1.4 PROJECT FEATURES

The features of Disease Prediction Using Machine Learning are as follows.

- This Project will predict the diseases of the patients based on the symptoms and other general information using the datasets.
- This is done based on the previous datasets of the hospitals so after comparing it can provide up to 80% of accurate results, and the project is still developing further to get the 100% accurate results.
- With the help of Disease prediction, it can predict the disease of the patient and can solve various problems and prevent from various aspects.
- It provides security for the system so that no one can break into that and no one can make any changes in the system.

- The disease is predicted using the algorithms and the user has to enter the symptoms from the given drop-down menu, in order to get correct accuracy, the user has to enter all the symptoms.
- Here we can easily prepare the data and transform that data into algorithm, which will reduce the overall work of the project.
- To make user more application friendly rather than discussing with others for their disease.
- It provides the necessary options to choose from the types and attributes.
- Here the user has to register first, in order to use the prediction and then login to the system using the credentials such as username and password.
- Once user open the system to login user needs to register by clicking on register/signup button.
- After which user needs to provide some basic details of signup and then the details of user are saved in system

CHAPTER 2

LITERATURE SURVEY

2.1 MACHINE LEARNING

Tom Mitchell states machine learning as “A computer program is said to learn from experience and from some tasks and some performance on, as measured by, improves with experience”. Machine Learning is combination of correlations and relationships, most machine learning algorithms in existence are concerned with finding and/or exploiting relationship between datasets. Once Machine Learning Algorithms can pinpoint on certain correlations, the model can either use these relationships to predict future observations or generalize the data to reveal interesting patterns. In Machine Learning there are various types of algorithms such as Regression, Linear Regression, Logistic Regression, Naive Bayes Classifier, Bayes theorem, KNN (K-Nearest Neighbor Classifier), Decision Tress, Entropy, ID3, SVM (Support Vector Machines), K-means Algorithm, Random Forest and etc.,

The name machine learning was coined in 1959 by Arthur Samuel. Machine learning explores the study and construction of algorithms that can learn from and make predictions on data Machine learning is closely related to (and often overlaps with) computational statistics, which also focuses on prediction-making through the use of computers. It has strong ties to mathematical optimization, which delivers methods, theory and application domains to the field. Machine learning is sometimes conflated with data mining, where the latter subfield focuses more on exploratory data analysis and is known as unsupervised learning.

Within the field of data analytics, machine learning is a method used to devise complex models and algorithms that lend themselves to prediction; in commercial use, this is known as predictive analytics. These analytical models allow researchers, data scientists, engineers, and analysts to "produce reliable, repeatable decisions and results" and uncover "hidden insights" through learning from historical relationships and trends in the data.

Machine learning tasks Machine learning tasks are typically classified into several broad categories:

Supervised learning: The computer is presented with example inputs and their desired outputs, given by a "teacher", and the goal is to learn a general rule that maps inputs to outputs. As special cases, the input signal can be only partially available, or restricted to special feedback.

Semi-supervised learning: The computer is given only an incomplete training signal: a training set with some (often many) of the target outputs missing.

Active learning: The computer can only obtain training labels for a limited set of instances (based on a budget), and also has to optimize its choice of objects to acquire labels for. When used interactively, these can be presented to the user for labelling.

Unsupervised learning: No labels are given to the learning algorithm, leaving it on its own to find structure in its input. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end (feature learning).

Reinforcement learning: Data (in form of rewards and punishments) are given only as feedback to the program's actions in a dynamic environment, such as driving a vehicle or playing a game against an opponent.

2.1.1 FEATURES OF MACHINE LEARNING

- It is nothing but automating the Automation.
- Getting computers to program themselves.
- Writing Software is bottleneck.
- Machine leaning models involves machines learning from data without the help of humans or any kind of human intervention.
- Machine Learning is the science of making of making the computers learn and act like humans by feeding data and information without being explicitly programmed.

- Machine Learning is totally different from traditionally programming, here data and output is given to the computer and in return it gives us the program which provides solution to the various problems. Below is the figure.

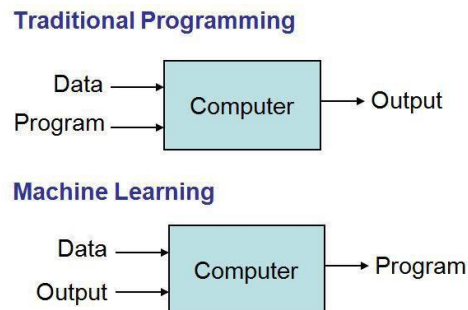
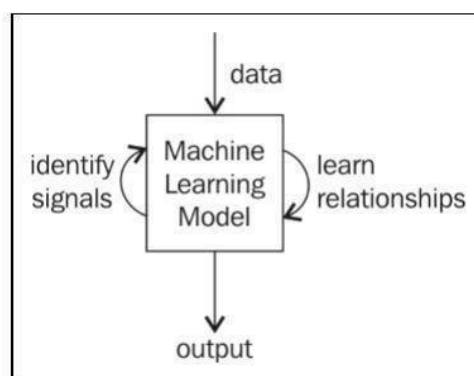


Fig 2.1.1 Traditional Programming vs Machine Learning

- Machine Learning is a combination of Algorithms, Datasets, and Programs.
- There are Many Algorithms in Machine Learning through which we will provide us the exact solution in predicting the disease of the patients.
- How Does Machine Learning Works?
- Solution to the above question is Machine learning works by taking in data, finding relationships within that data and then giving the output.



An overview of machine learning models

Fig 2.1.2 Machine Learning Model

- There are various applications in which machine learning is implemented such as Web search, computing biology, finance, e-commerce, space exploration, robotics, social networks, debugging and much more.
- There are 3 types of machine learning supervised, unsupervised, and reinforcement.

2.2 EXISTING SYSTEM

Prediction using traditional methods and models involves various risk factors and it consists of various measures of algorithms such as datasets, programs and much more to add on. High-risk and Low-risk patient classification is done on the basis of the tests that are done in group. But these models are only valuable in clinical situations and not in big industry sector. So, to include the disease predictions in various health related industries, we have used the concepts of machine learning and supervised learning methods to build the predictions system.

After doing the research and comparison of all the algorithms and theorems of machine learning we have come to conclusion that all those algorithms such as Decision Tree, KNN, Naïve Bayes, Regression and Random Forest Algorithm all are important in building a disease prediction system which predicts the disease of the patients from which he/she is suffering from and to do this we have used some performance measures like ROC, KAPPA Statistics, RMSE, MEA and various other tools. After using various techniques such as neural networks to make predictions of the diseases and after doing that we come to conclusion that it can predicts up to 90% accuracy rate after doing the experimentation and verifying the results. The information of patient statistics, results, disease history in recorded in EHR, which enables to identify the potential data centric solution, which reduces the cost of medical case studies. Existing system can predict the disease but not the sub type of the disease and it fails to predict the condition of the people, the predictions of disease have been indefinite and non-specific.

2.3 PROPOSED SYSTEM

The proposed system of disease prediction using machine learning is that we have used many techniques and algorithms and all other various tools to build a system which predicts the 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 disease we will predict the accurate percentage 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 feature selection is done by the user where he 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, Naïve Bayes, Random Forest and etc. Then the data goes in the recommendation model, there it shows the risk analysis that is involved in the system and it also provides the probability estimation of the system such that it shows the various probability like how the system behaves when there are n number of predictions are done and it also does the recommendations for the patients from their final result and also from their symptoms like it can show what to use and what not to use from the given datasets and the final results. Here we have combined the overall structure and unstructured form of data for the overall risk analysis that is required for doing the prediction of the disease. Using the structured analysis, we can identify the chronic types of disease in a particular region and particular community. In unstructured analysis we select the features automatically with the help of algorithms and techniques. This system takes symptoms from the user and predicts the disease accordingly based on the symptoms that it takes and also from the previous datasets, it also helps in continuous evaluation of viral diseases, heart rate, blood pressure, sugar level and much more which is in the system and along with other external symptoms it predicts the appropriate and accurate disease.

2.4 SOFTWARE DESCRIPTION

2.4.1 PYTHON

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by metaprogramming and metaobjects. Many other paradigms are supported via extensions, including design by contract and logic programming. Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

Python's developers strive to avoid premature optimization, and reject patches to non-critical parts of CPython that would offer marginal increases in speed at the cost of clarity. When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C, or use PyPy, a just-in-time compiler. Cython is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter. An important goal of Python's developers is keeping it fun to use. Python's design offers some support for functional programming in the Lisp tradition. It has filter, map, and reduce functions, list comprehensions, dictionaries, sets, and generator expressions. The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML.

2.4.2 BENEFITS OF PYTHON

- Presence of Third-Party Modules
- Extensive Support Libraries
- Open Source and Community Development
- Learning Ease and Support Available
- User-friendly Data Structures
- Productivity and Speed
- Highly Extensible and Easily Readable Language.

2.4.3 TKINTER INTERFACE

Tkinter is a Python binding to the Tk GUI toolkit. It is the standard Python interface to the Tk GUI toolkit and is Python's de facto standard GUI. Tkinter is included with standard Linux, Microsoft Windows and Mac OS X installs of Python. The name Tkinter comes from Tk interface. Tkinter was written by Fredrik Lundh. Tkinter is free software released under a Python license.

As with most other modern Tk bindings, Tkinter is implemented as a Python wrapper around a complete Tool Command Language (TCL) interpreter embedded in the Python interpreter. Tkinter calls are translated into Tcl commands which are fed to this embedded interpreter, thus making it possible to mix Python and TCL in a single application. In Tkinter, the Frame widget is the basic unit of organization for complex layouts. A frame is a rectangular area that can contain other widgets. When any widget is created, a parent-child relationship is created. For example, if you place a text label inside a frame, the frame is the parent of the label.

Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter outputs the fastest and easiest way to create the GUI applications.

To create a tkinter:

Importing the module – tkinter

Create the main window (container)

Add any number of widgets to the main window

Apply the event Trigger on the widgets.

Importing tkinter is same as importing any other module in the python code. Note that the name of the module in Python 2.x is 'Tkinter' and in Python 3.x is 'tkinter'.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FUNCTIONAL REQUIREMENTS

A Functional requirement defines a function of a system or its component. A function is described as a set of inputs, the behaviour, and outputs. Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish. Behavioural requirements describing all cases where the system uses the functional requirements are captured in use cases. Functional requirements are supported by non-functional requirements (also known as quality requirements), which impose constraints on the design or implementation (such as performance requirements, security, or reliability).

As defined in requirements engineering, functional requirements specify particular results of a system. This should be contrasted with non-functional requirements which specify overall characteristics such as cost and reliability. Functional requirements drive the application architecture of a system, while non-functional requirements drive the technical architecture of a system.

- Functional Requirements concerns with the specific functions delivered by the system. So, Functional requirements are statements of the services that the system must provide.
- The functional requirements of the system should be both complete and consistent
- Completeness means that all the services required by the user should be defined.
- Consistency means that requirements should not have any contradictory definitions.
- The requirements are usually described in a fairly abstract way. However, functional system requirements describe the system function in details, its inputs and outputs, exceptions and soon.
- Take user id and password match it with corresponding file entries. If a match is found then continue else raise an error message.

3.2 NON-FUNCTIONAL REQUIREMENTS

- Non-functional Requirements refer to the constraints or restrictions on the system. They may relate to emergent system properties such as reliability, response time and store occupancy or the selection of language, platform, implementation techniques and tools.

- The non-functional requirements can be built on the basis of needs of the user, budget constraints, organization policies and etc.

1. **Performance requirement:** All data entered shall be up to mark and no flaws shall be there for the performance to be 100%.
2. **Platform constraints:** The main target is to generate an intelligent system to predict the adult height.
3. **Accuracy and Precision:** Requirements are accuracy and precision of the data
4. **Modifiability:** Requirements about the effort required to make changes in the software. Often, the measurement is personnel effort (person- months).
5. **Portability:** Since mobile phone is handy so it is portable and can be carried and used whenever required.
6. **Reliability:** Requirements about how often the software fails. The definition of a failure must be clear. Also, don't confuse reliability with availability which is quite a different kind of requirement. Be sure to specify the consequences of software failure, how to protect from failure, a strategy for error Prediction, and a strategy for correction.
7. **Security:** One or more requirements about protection of your system and its data.
8. **Usability:** Requirements about how difficult it will be to learn and operate the system. The requirements are often expressed in learning time or similar metrics.

ACCESSIBILITY:

Accessibility is a general term used to describe the degree to which a product, device, service, or environment is accessible by as many people as possible. In our project people who have registered with the cloud can access the cloud to store and retrieve their data with the help of a secret key sent to their email ids. User interface is simple and efficient and easy to use.

MAINTAINABILITY:

In software engineering, maintainability is the ease with which a software product can be modified in order to include new functionalities can be added in the project based on the user requirements just by adding the appropriate files to existing project using .net and programming languages. Since the programming is very simple, it is easier to find and correct the defects and to make the changes in the project.

SCALABILITY:

System is capable of handling increase total throughput under an increased load when resources (typically hardware) are added. System can work normally under situations such as low bandwidth and large number of users.

PORTABILITY:

Portability is one of the key concepts of high-level programming. Portability is the software code base feature to be able to reuse the existing code instead of creating new code when moving software from an environment to another. Project can be executed under different operation conditions provided it meet its minimum configurations. Only system files and dependant assemblies would have to be configured in such case.

VALIDATION:

It is the process of checking that a software system meets specifications and that it fulfils its intended purpose. It may also be referred to as software quality control. It is normally the responsibility of software testers as part of the software development lifecycle. Software validation checks that the software product satisfies or fits the intended use (high-level checking), i.e., the software meets the user requirements, not as specification artefacts or as needs of those who will operate the software only; but, as the needs of all the stakeholders.

3.3 HARDWARE REQUIREMENTS

- ❖ System : Pentium 4, Intel Core i3, i5, i7 or Higher versions and 2GHz Minimum
- ❖ RAM : 512Mb or above
- ❖ Hard Disk : 10 GB or above
- ❖ Input Device : Keyboard and Mouse
- ❖ Output Device : Monitor or PC

3.4 SOFTWARE REQUIREMENTS

- ❖ Operating System : Windows 7, 10 or Higher Versions
- ❖ Platform : Jupyter Notebook
- ❖ Front End : Python Tkinter
- ❖ Back End : Python and Files
- ❖ Programming Lang : Python

CHAPTER 4

DESIGN

4.1 DESIGN GOALS

The Design goals consist of various design which we have implemented in our system disease 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.

We have designed our system in such a way that whenever user log in into the system, the user has to register to the system, and new user cannot use the system without registering in the system. After that for registration the user requires basic credentials such as username, age, email, phone, password. Then the user has to login to the system using the same username and password. Here are the things that this system can perform.

- a. Entering Symptoms
- b. Disease Prediction

Entering Symptoms: Once user successfully logged in to the system then he/she has to select the symptoms from the given drop-down menu.

Disease prediction: The predictive model predicts the disease of a person he might have, based on the user entered symptoms.

.

4.2 SYSTEM ARCHITECTURE

Disease prediction using machine learning predicts the presence of the disease for the user based on various symptoms and the information the user gives such as sugar level, haemoglobin level and many more such general information through the symptoms. The architecture of the system disease prediction using machine learning consist of various datasets 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 disease prediction model using all the inputs from the user that is mentioned above. Then after user entering the above information and overall processed data combines and compares in the prediction model of the system and finally predicts the disease. 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.

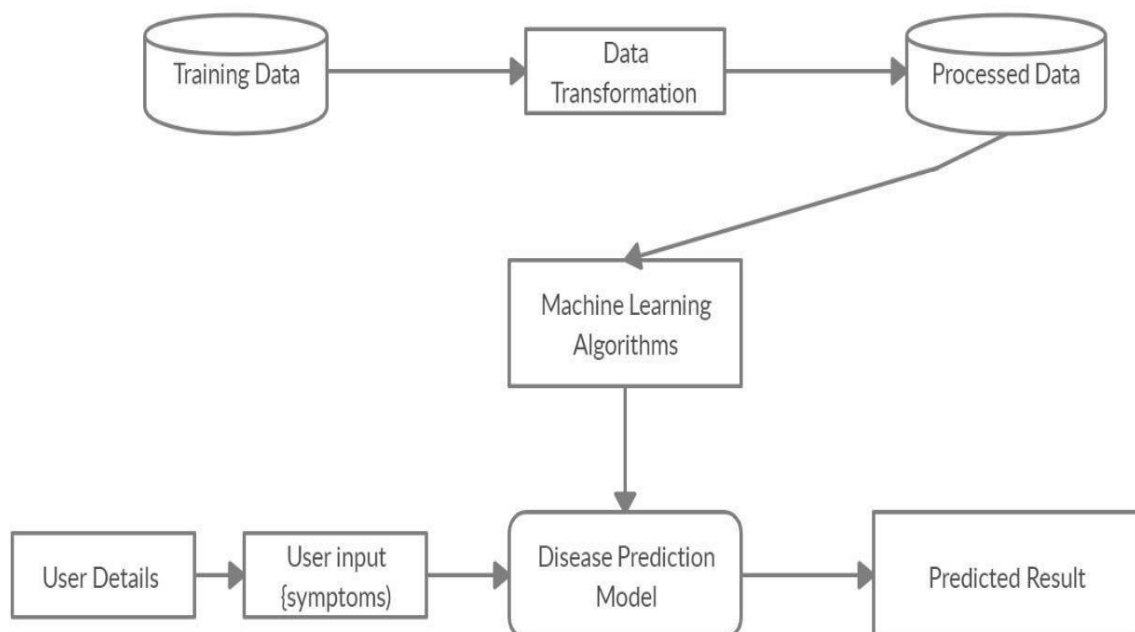


Fig 4.2 System Architecture

4.3 DATA FLOW DIAGRAM

The dataflow diagram of the project disease prediction using machine learning consist of all the various aspects a normal flow diagram requires. This dataflow diagram shows how from starting the model flows from one step to another, like he enter into the system then enters all the information's and all other general information along with the symptoms that goes into the system, compares with the prediction model and if true is predicts the appropriate results otherwise it shows the details where the user if gone wrong while entering the information.

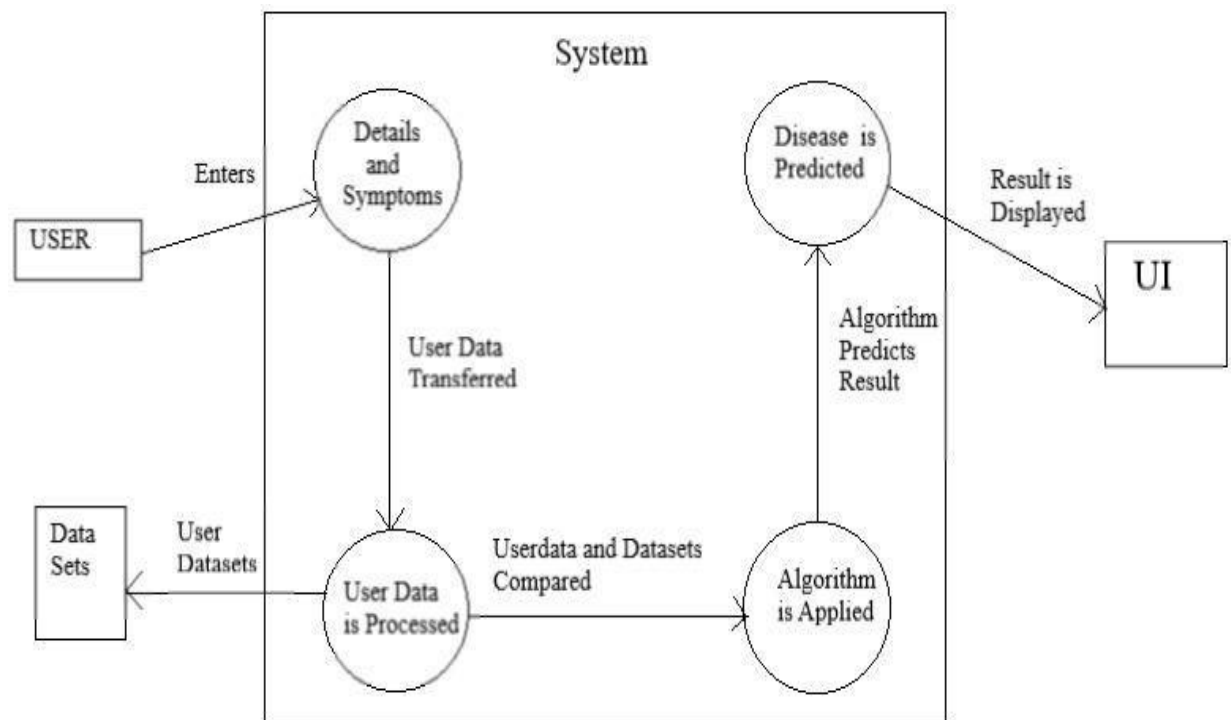


Fig 4.3 Data Flow Diagram

4.4 CLASS DIAGRAM

Disease prediction using machine learning consist of class diagram that all the other application that consists the basic class diagram, here the class diagram is the basic entity that is required in order to carry on with the project. Class diagram consist information about all the classes that is used and all the related datasets, and all the other necessary attributes and their relationships with other entities, all these information is necessary in order to use the concept of the prediction, where the user will enter all necessary information such as user name, email, phone number, and many more attributes that is required in order to login into the system and using the files concept we will store the information of the users who are registering into the system and retrieves those information later while logging into the system.

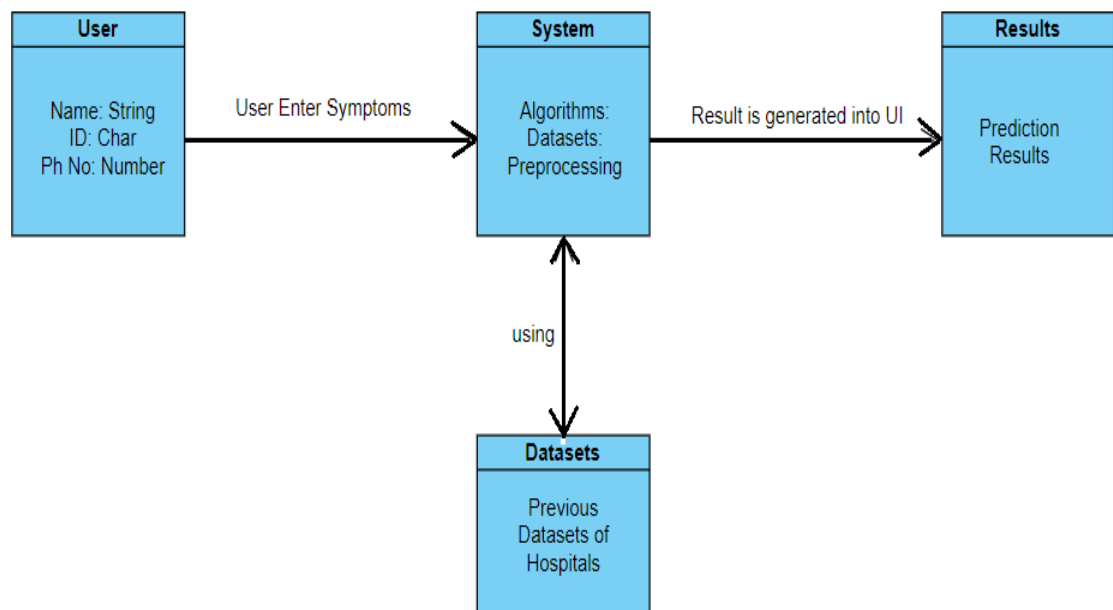


Fig 4.4 Class Diagram

4.5 SEQUENCE DIAGRAM

The Sequence diagram of the project disease 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 enter into the system then enters all the information's and all other general information along with the symptoms that goes into the system, compares with the prediction model and if true is predicts the appropriate results otherwise it shows the details where the user if gone wrong while entering the information's and it also shows the appropriate precautionary measure for the user to follow. Here the sequence of all the entities are linked to each other where the user gets started with the system.

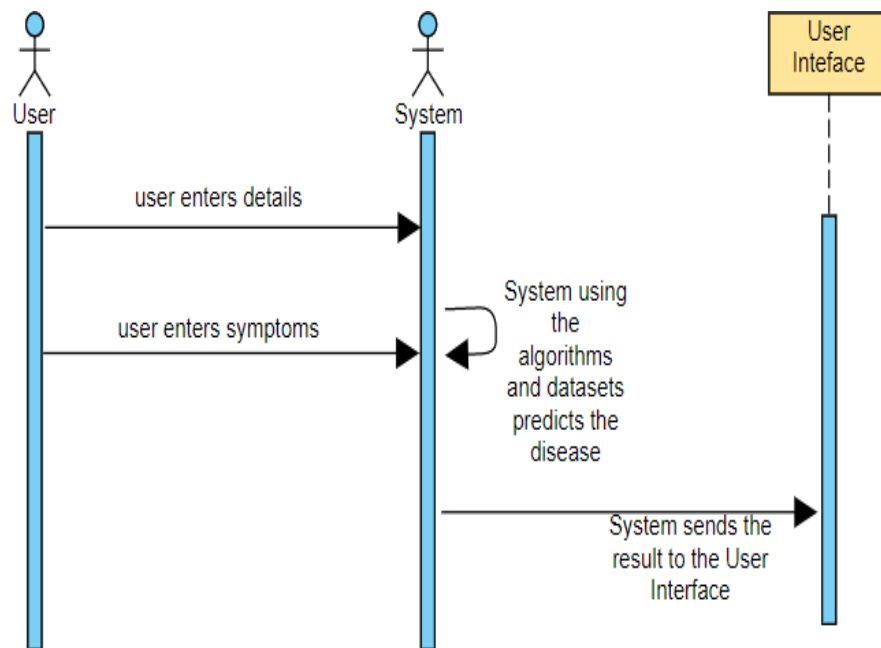


Fig 4.5 Sequence Diagram

4.6 USE CASE DIAGRAM

The Use Case diagram of the project disease prediction using machine learning consist of all the various aspects a normal use case diagram requires. This use case diagram shows how from starting the model flows from one step to another, like he enter into the system then enters all the information's and all other general information along with the symptoms that goes into the system, compares with the prediction model and if true is predicts the appropriate results otherwise it shows the details where the user if gone wrong while entering the information's and it also shows the appropriate precautionary measure for the user to follow. Here the use case diagram of all the entities are linked to each other where the user gets started with the system.

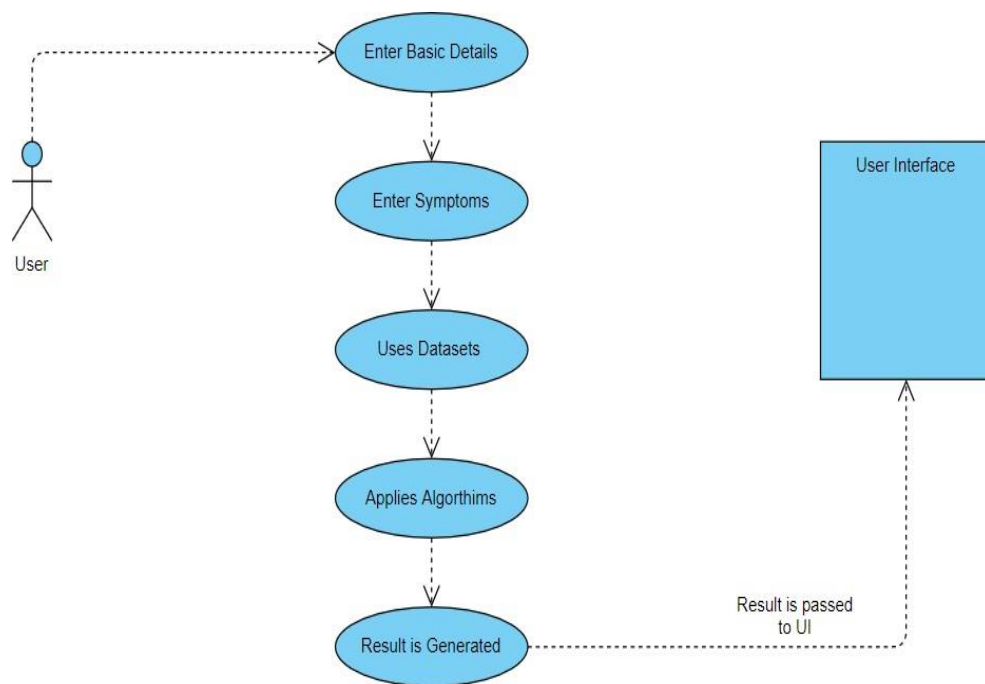


Fig4.6 Use Case Diagram

4.7 ACTIVITY DIAGRAM

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. Here in this diagram the activity starts from user where the user registers into the system then login using the credentials and then the credentials are matched in the system and if its true, then the user proceeds to the prediction phase where the prediction happens. Then finally after processing the data from datasets the analysis will happen then the correct result will be displayed that is nothing but the Output.

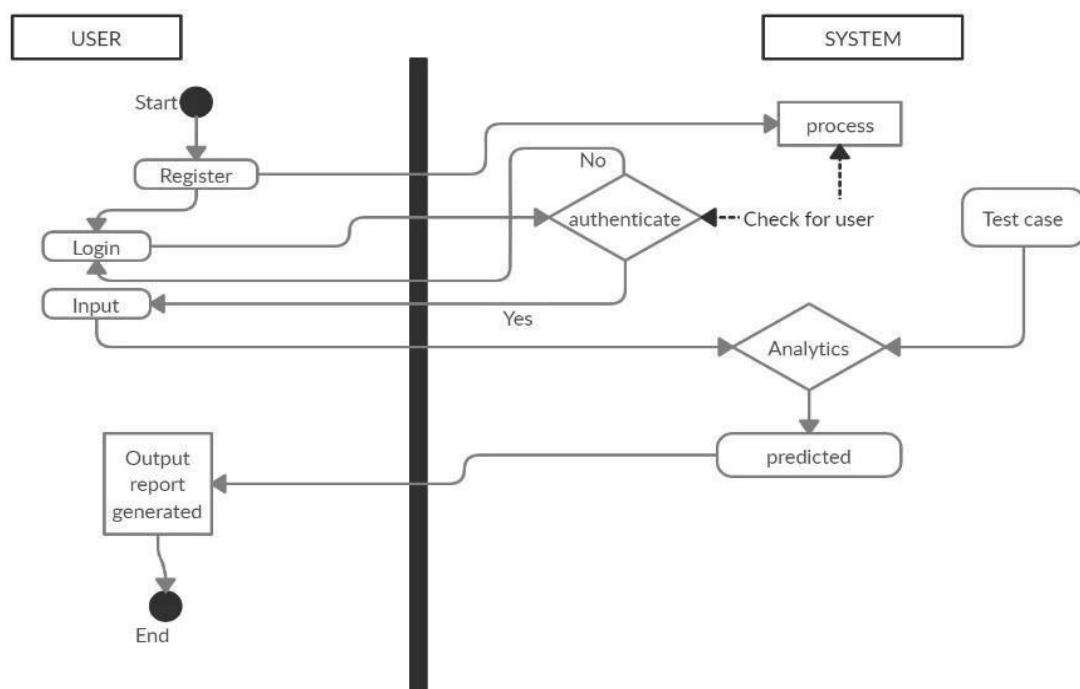


Fig 4.7 Activity Diagram

4.8 COMPONENT DIAGRAM

A component diagram, also known as a UML component diagram, describes the organization and wiring of the physical components in a system. Component diagrams are often drawn to help model implementation details and double-check that every aspect of the system's required function is covered by planned development. Here component diagram consists of all major components that is used to built a system. So, Design, Algorithm, File System and Datasets all are linked to one another. Datasets are used to compare the results and algorithm is used to process those results and give a correct accuracy and design UI is used to show the result in an appropriate way in the system and file system is used to store the user data. So, like this all components are interlinked to each other.

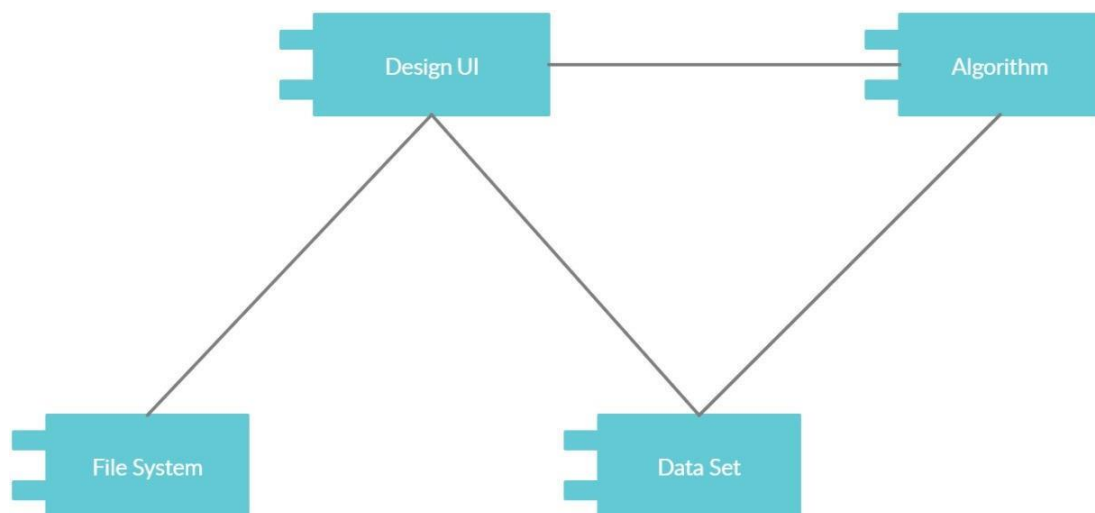


Fig 4.8 Component Diagram

4.9 STATE CHART DIAGRAM

A State chart diagram describes the behaviour of a single object in response to a series of events in a system. Sometimes it's also known as a Harel state chart or a state machine diagram. This UML diagram models the dynamic flow of control from state to state of a particular object within a system. It is similar to activity diagram but here there are only few rules like how it starts and how it end all are denoted with the help of the symbol given below, the system starts with the registration and then login comes, if the login is successful then it will go to the next step and if login is incorrect then comes back to same page stating incorrect details. After the successful login the user needs to enter the symptoms and then press the prediction button, at the same time the backend process will do their work and the correct result is predicted.

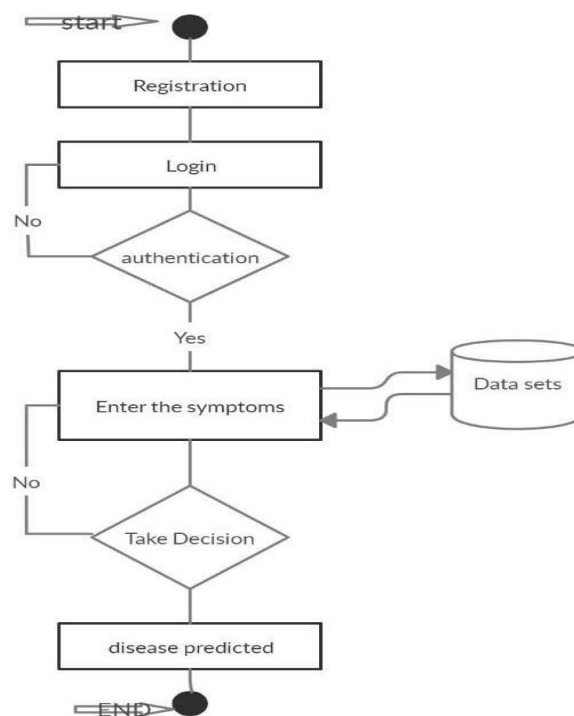


Fig 4.9 State Chart Diagram

4.10 COLLABORATION DIAGRAM

A collaboration diagram, also known as a communication diagram, is an illustration of the relationships and interactions among software objects in the Unified Modelling Language (UML). These diagrams can be used to portray the dynamic behaviour of a particular use case and define the role of each object. Here this diagram shows how all the models are connected to show the correct result starting from user, where he opens the system then using the system he does registration and that registration data is saved into file system and the using those data user logs in to the system and then he provides all the necessary information in order to get the accurate result, then system evaluates the user entered information and finally gives the correct result.

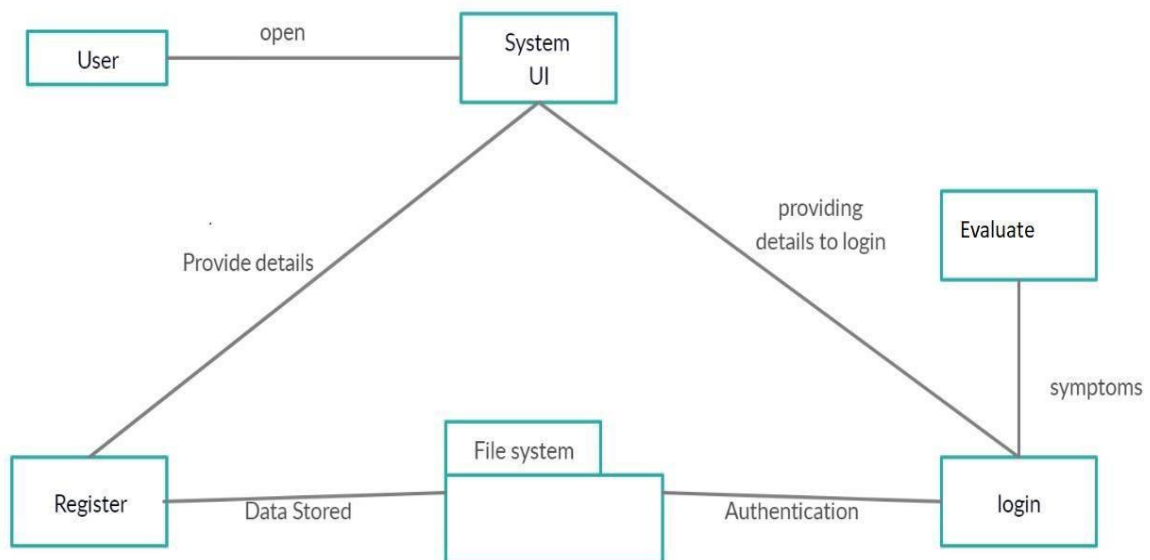


Fig 4.10 Collaboration Diagram

4.11 DEPLOYMENT DIAGRAM

A deployment diagram shows the configuration of run time processing nodes and the components that live on them. Deployment diagrams is a kind of structure diagram used in modelling the physical aspects of an object-oriented system. Here the deployment diagram show the final stage of the project and it also shows how the model looks like after doing all the processes and deploying in the machine. Starting from the system how it processes the user entered information and then comparing that information with the help of datasets, then training and testing those data using the algorithms such as decision tree, naïve Bayes, random forest. Then finally processing all those data and information the system gives the desired result in the interface.

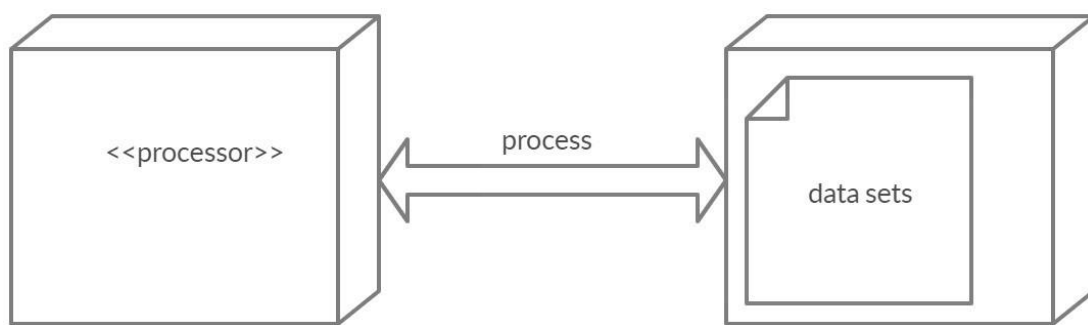


Fig 4.11 Deployment Diagram

4.12 INTERFACE AND FRAMEWORK DIAGRAM

- Below is the structure which we will use in our project Disease Prediction using Machine learning.
- The User Interface of this system consists of Python’s library interface called tkinter.
- Then it goes into the framework model where all the actions and services are combined and then the result is processed.
- It also consists of file system where all the user related information is stored such as username, password, age, phone, email.
- Below is the structure of the User Interface along with necessary implementations.

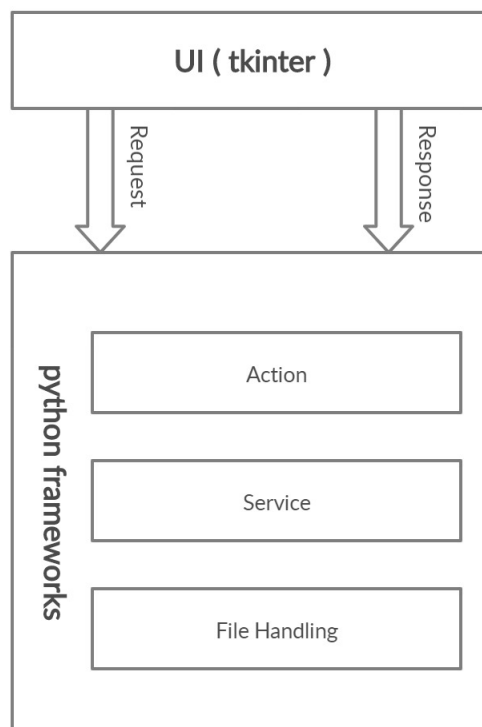


Fig 4.12 Interface and Framework Diagram

- After the User Interface it consist of the framework in which the system works accordingly using all the technologies, algorithms and various tools in which the project works accordingly.

- The framework consists of all the modules starting from the data preparation, data building and assessment stage.
- All these three factors are then going into the data collection phase, where the data is classified accordingly using the appropriate models and algorithms such as decision tree, naïve bayes, random forest.
- Then all those algorithms use the datasets and it forms the sets where all the previous data is stored, then using that data it compares with the new data and result is generated.
- Then pre-processing work will happen to reduce and analyze the data that is present in the system.
- Then with the help of UI the data is transferred into the main screen.
- Then later all those data are analyzed and validated then the final result is generated.
- Finally after user enters the symptoms, all backend mechanisms works and the predicted result is displayed in the User Interface.

CHAPTER 5

IMPLEMENTATION

.1 OVERVIEW

The project Disease Prediction using Machine Learning is developed to overcome general disease 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 who ignores about general disease which leads to harmful disease later. The Project “Disease Prediction using Machine Learning” is implemented using python completely. Even the interface of this project is done using python’s library interface called Tkinter. Here first the user needs to register into the system in order to use the prediction, user needs to register with username, email-id, phone, age and password. All these values are stored into the file system respectively, then user has option to move forward or leave, then user needs to login to the system using the username and password which he/she provided during the time of registration. If he/she enter incorrect username and correct password then the error message will prompt stating incorrect username and if he/she enters incorrect password and correct username then the error message will prompt stating incorrect password, so both username and password is necessary in order to login to the system. After logging in the user needs to the name and needs to select the symptoms from given drop-down menu, 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 2 algorithms of machine learning such as Logistic Regression & KNN Classifier .

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

- Once user open the system to login user needs to register by clicking on register/signup button
- After which user needs to provide some basic details of signup and then the details of user are saved in system
- Then user needs to login to have a checkup of his/her health
- When user tries to login if he provides wrong user name the system will provide a prompt message stating that the user is not found
- And if user tries to enter the wrong password the system will prompt stating that password is incorrect hence the user needs to enter the correct user id and password to get in to the system
- After user enters the 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.
- Developing a probabilistic model and deep learning approach (RNN) for Disease Prediction in this step probabilistic model and deep learning approach based on RNN is to be developed it will run effectively on extensive databases of healthcare. And generate decision tree also it can deal with a huge number of information variables without variable deletion.
- Training and experimentation on datasets The Disease Prediction model will be trained on the dataset of diseases to do the prediction accurately and produce Confusion matrix. In this project different algorithms were used –
 - Logistics Regression
 - KNN Classification

5.2 LOGISTIC REGRESSION :-

- Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.
- Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.
- Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas Logistic regression is used for solving the classification problems.
- In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).
- The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc.
- Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets.
- Logistic Regression can be used to classify the observations using different types of data and can easily determine the most effective variables used for the classification. The below image is showing the logistic function:

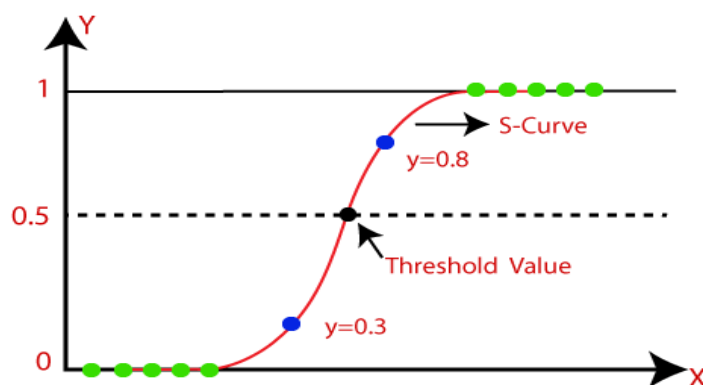


Fig 5.2.1 Logistic Regression

Logistic Function (Sigmoid Function) :-

- The sigmoid function is a mathematical function used to map the predicted values to probabilities.
- It maps any real value into another value within a range of 0 and 1.
- The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the "S" form. The S-form curve is called the Sigmoid function or the logistic function.
- In logistic regression, we use the concept of the threshold value, which defines the probability of either 0 or 1. Such as values above the threshold value tends to 1, and a value below the threshold values tends to 0.

Assumptions for Logistic Regression:-

- The dependent variable must be categorical in nature.
- The independent variable should not have multi-collinearity.

Logistic Regression Equation:-

The Logistic regression equation can be obtained from the Linear Regression equation.

The mathematical steps to get Logistic Regression equations are given below:

- We know the equation of the straight line can be written as:

$$y = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \dots + b_nx_n$$

- In Logistic Regression y can be between 0 and 1 only, so for this let's divide the above equation by (1-y):

$$\frac{y}{1-y} ; 0 \text{ for } y=0, \text{ and infinity for } y=1$$

- But we need range between -[infinity] to +[infinity], then take logarithm of the equation it will become:

$$\log \left[\frac{y}{1-y} \right] = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \dots + b_nx_n$$

Type of Logistic Regression:-

On the basis of the categories, Logistic Regression can be classified into three types:

- Binomial: In binomial Logistic regression, there can be only two possible types of the dependent variables, such as 0 or 1, Pass or Fail, etc.
- Multinomial: In multinomial Logistic regression, there can be 3 or more possible unordered types of the dependent variable, such as "cat", "dogs", or "sheep"
- Ordinal: In ordinal Logistic regression, there can be 3 or more possible ordered types of dependent variables, such as "low", "Medium", or "High".

Python Implementation of Logistic Regression (Binomial):-

To understand the implementation of Logistic Regression in Python, we will use the below example:

Example: There is a dataset given which contains the information of various users obtained from the social networking sites. There is a car making company that has recently launched a new SUV car. So the company wanted to check how many users from the dataset, wants to purchase the car.

For this problem, we will build a Machine Learning model using the Logistic regression algorithm. The dataset is shown in the below image. In this problem, we will predict the purchased variable (Dependent Variable) by using age and salary (Independent variables).

User ID	Gender	Age	EstimatedSalary	Purchased
15624510	Male	19	19000	0
15810944	Male	35	20000	0
15668575	Female	26	43000	0
15603246	Female	27	57000	0
15804002	Male	19	76000	0
15728773	Male	27	58000	0
15598044	Female	27	84000	0
15694829	Female	32	150000	1
15600575	Male	25	33000	0
15727311	Female	35	65000	0
15570769	Female	26	80000	0
15606274	Female	26	52000	0
15746139	Male	20	86000	0
15704987	Male	32	18000	0
15628972	Male	18	82000	0
15697686	Male	29	80000	0
15733883	Male	47	25000	1
15617482	Male	45	26000	1
15704583	Male	46	28000	1
15621083	Female	48	29000	1
15649487	Male	45	22000	1
15736760	Female	47	49000	1

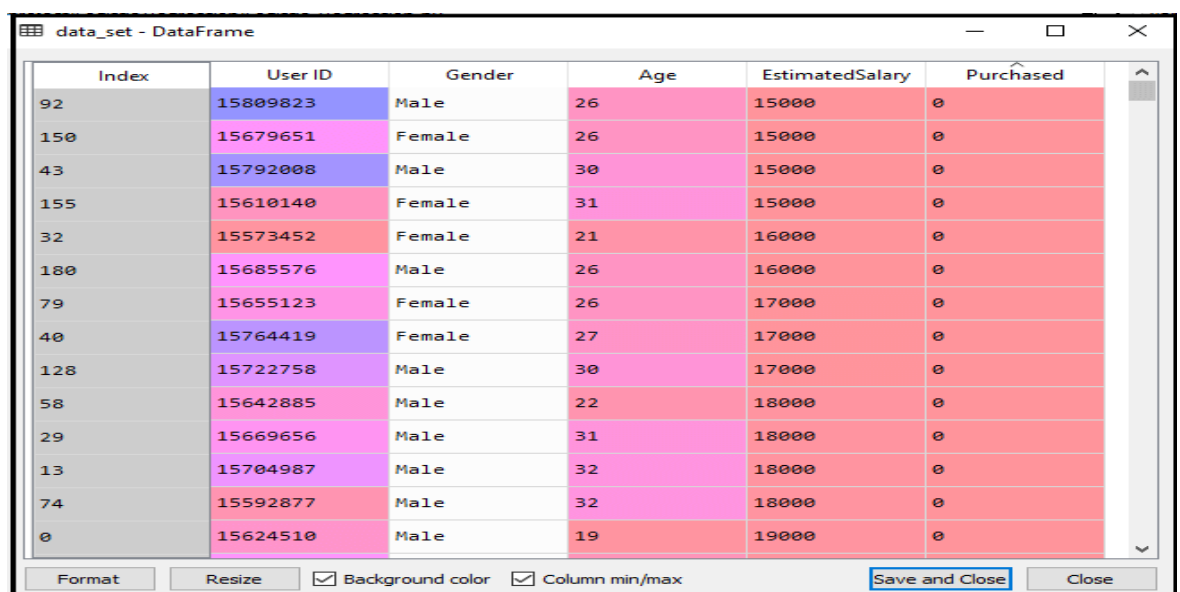
Steps in Logistic Regression:

To implement the Logistic Regression using Python, we will use the same steps as we have done in previous topics of Regression. Below are the steps:

- Data Pre-processing step
- Fitting Logistic Regression to the Training set
- Predicting the test result
- Test accuracy of the result(Creation of Confusion matrix)
- Visualizing the test set result.

Data Pre-processing step: In this step, we will pre-process/prepare the data so that we can use it in our code efficiently. It will be the same as we have done in Data pre-processing topic. The code for this is given below:

- Data Pre-processing Step
- importing libraries
- `import numpy as nm`
- `import matplotlib.pyplot as mtp`
- `import pandas as pd`
- importing datasets
- `data_set= pd.read_csv ('user_data.csv')`
- By executing the above lines of code, we will get the dataset as the output. Consider the given image:



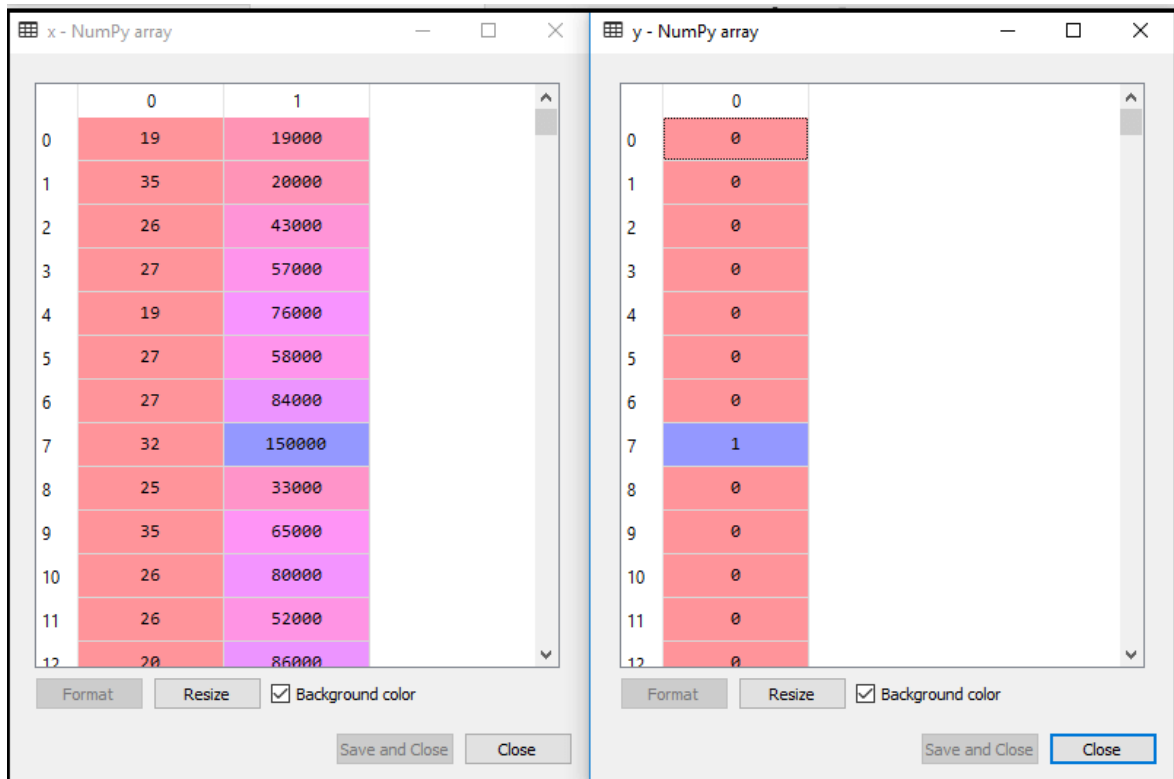
The screenshot shows a Jupyter Notebook interface with a DataFrame named 'data_set'. The DataFrame contains 15 rows of user data. The columns are Index, User ID, Gender, Age, EstimatedSalary, and Purchased. The data is displayed in a table with alternating row colors (pink and light blue). The 'Purchased' column shows values 0 or 1.

Index	User ID	Gender	Age	EstimatedSalary	Purchased
92	15809823	Male	26	15000	0
150	15679651	Female	26	15000	0
43	15792008	Male	30	15000	0
155	15610140	Female	31	15000	0
32	15573452	Female	21	16000	0
180	15685576	Male	26	16000	0
79	15655123	Female	26	17000	0
40	15764419	Female	27	17000	0
128	15722758	Male	30	17000	0
58	15642885	Male	22	18000	0
29	15669656	Male	31	18000	0
13	15704987	Male	32	18000	0
74	15592877	Male	32	18000	0
0	15624510	Male	19	19000	0

Now, we will extract the dependent and independent variables from the given dataset. Below is the code for it:

```
#Extracting Independent and dependent Variable  
x= data_set.iloc[:, [2,3]].values  
y= data_set.iloc[:, 4].values
```

In the above code, we have taken [2, 3] for x because our independent variables are age and salary, which are at index 2, 3. And we have taken 4 for y variable because our dependent variable is at index 4. The output will be:



	0	1
0	19	19000
1	35	20000
2	26	43000
3	27	57000
4	19	76000
5	27	58000
6	27	84000
7	32	150000
8	25	33000
9	35	65000
10	26	80000
11	26	52000
12	20	86000

	0
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	1
8	0
9	0
10	0
11	0
12	0

Now we will split the dataset into a training set and test set. Below is the code for it:
Splitting the dataset into training and test set.

```
from sklearn.model_selection import train_test_split  
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.25, random_state=0)
```

The output for this is given below:

	0	1
0	-0.804802	0.504964
1	-0.0125441	-0.567782
2	-0.309641	0.157046
3	-0.804802	0.273019
4	-0.309641	-0.567782
5	-1.1019	-1.43758
6	-0.70577	-1.58254
7	-0.210609	2.15757
8	-1.99319	-0.0459058
9	0.878746	-0.770734
10	-0.804802	-0.596776
11	-1.00287	-0.422817
12	-0.111576	-0.422817

	0
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	1
8	0
9	0
10	0
11	0
12	0

For training set:

	0	1
0	-0.804802	0.504964
1	-0.0125441	-0.567782
2	-0.309641	0.157046
3	-0.804802	0.273019
4	-0.309641	-0.567782
5	-1.1019	-1.43758
6	-0.70577	-1.58254
7	-0.210609	2.15757
8	-1.99319	-0.0459058
9	0.878746	-0.770734
10	-0.804802	-0.596776
11	-1.00287	-0.422817
12	-0.111576	-0.422817

	0
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	1
8	0
9	0
10	0
11	0
12	0

in logistic regression, we will do feature scaling because we want accurate result of predictions. Here we will only scale the independent variable because dependent variable have only 0 and 1 values. Below is the code for it:

```
feature Scaling
from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
x_test= st_x.transform(x_test)
```

The scaled output is given below:

	0	1
0	-0.804802	0.504964
1	-0.0125441	-0.567782
2	-0.309641	0.157046
3	-0.804802	0.273019
4	-0.309641	-0.567782
5	-1.1019	-1.43758
6	-0.70577	-1.58254
7	-0.210609	2.15757
8	-1.99319	-0.0459058
9	0.878746	-0.770734
10	-0.804802	-0.596776
11	-1.00287	-0.422817
12	-0.111576	-0.422817

	0	1
0	0.581649	-0.886707
1	-0.606738	1.46174
2	-0.0125441	-0.567782
3	-0.606738	1.89663
4	1.37391	-1.40858
5	1.47294	0.997847
6	0.0864882	-0.799728
7	-0.0125441	-0.248858
8	-0.210609	-0.567782
9	-0.210609	-0.190872
10	-0.309641	-1.29261
11	-0.309641	-0.567782
12	0.383585	0.0990599

2. Fitting Logistic Regression to the Training set:

We have well prepared our dataset, and now we will train the dataset using the training set. For providing training or fitting the model to the training set, we will import the LogisticRegression class of the **sklearn** library. After importing the class, we will create a classifier object and use it to fit the model to the logistic regression. Below is the code for it:

```
#Fitting Logistic Regression to the training set
from sklearn.linear_model import LogisticRegression
classifier= LogisticRegression(random_state=0)
classifier.fit(x_train, y_train)
```

Output: By executing the above code, we will get the below output:

Out[5]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                    intercept_scaling=1, l1_ratio=None, max_iter=100,
                    multi_class='warn', n_jobs=None, penalty='l2',
                    random_state=0, solver='warn', tol=0.0001, verbose=0,
                    warm_start=False)
```

Hence our model is well fitted to the training set.

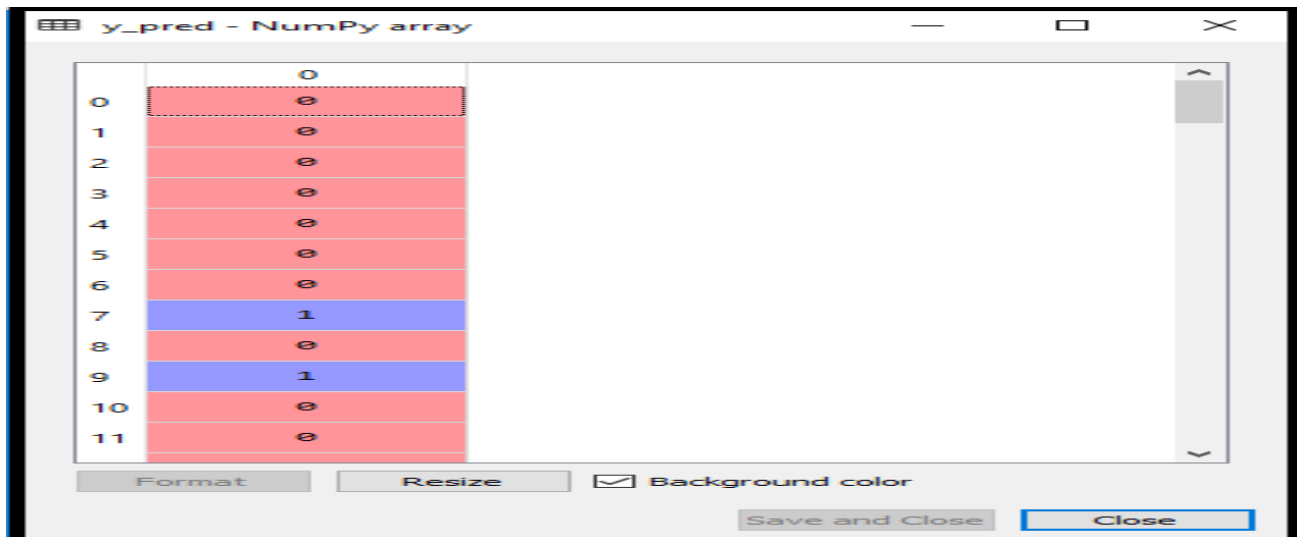
3. Predicting the Test Result

Our model is well trained on the training set, so we will now predict the result by using test set data. Below is the code for it:

```
#Predicting the test set result
y_pred= classifier.predict(x_test)
```

In the above code, we have created a y_pred vector to predict the test set result.

Output: By executing the above code, a new vector (y_pred) will be created under the variable explorer option. It can be seen as:



The above output image shows the corresponding predicted users who want to purchase or not purchase the car.

4. Test Accuracy of the result

Now we will create the confusion matrix here to check the accuracy of the classification. To create it, we need to import the **confusion_matrix** function of the sklearn library. After importing the function, we will call it using a new variable **cm**. The function takes two parameters, mainly **y_true** (the actual values) and **y_pred** (the targeted value return by the classifier). Below is the code for it:

#Creating the Confusion matrix

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

```
cm= confusion_matrix()
```

Output:

By executing the above code, a new confusion matrix will be created. Consider the below image:



We can find the accuracy of the predicted result by interpreting the confusion matrix. By above output, we can interpret that $65+24= 89$ (Correct Output) and $8+3= 11$ (Incorrect Output).

5. Visualizing the training set result

Finally, we will visualize the training set result. To visualize the result, we will use ListedColormap class of matplotlib library. Below is the code for it:

```
#Visualizing the training set result
from matplotlib.colors import ListedColormap
x_set, y_set = x_train, y_train
x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() -
1, stop = x_set[:, 0].max() + 1, step =0.01),
nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shape),
alpha = 0.75, cmap = ListedColormap(('purple','green' )))
mtp.xlim(x1.min(), x1.max())
mtp.ylim(x2.min(), x2.max())
for i, j in enumerate(nm.unique(y_set)):
mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
c = ListedColormap(('purple', 'green'))(i), label = j)
mtp.title('Logistic Regression (Training set)')
mtp.xlabel('Age')
mtp.ylabel('Estimated Salary')
mtp.legend()
```

In the above code, we have imported the ListedColormap class of Matplotlib library to create the colormap for visualizing the result. We have created two new variables x_set and y_set to replace x_train and y_train. After that, we have used the command to create a rectangular grid, which has a range of -1(minimum) to 1 (maximum). The pixel points we have taken are of 0.01 resolution.

To create a filled contour, we have used mtp.contourf command, it will create regions of provided colors (purple and green). In this function, we have passed the classifier.predict to show the predicted data points predicted by the classifier.

Output: By executing the above code, we will get the below output:



The graph can be explained in the below points:-

- In the above graph, we can see that there are some **Green points** within the green region and **Purple points** within the purple region.
- All these data points are the observation points from the training set, which shows the result for purchased variables.
- This graph is made by using two independent variables i.e., **Age on the x-axis** and **Estimated salary on the y-axis**.
- The **purple point observations** are for which purchased (dependent variable) is probably 0, i.e., users who did not purchase the SUV car.
- The **green point observations** are for which purchased (dependent variable) is probably 1 means user who purchased the SUV car.
- We can also estimate from the graph that the users who are younger with low salary, did not purchase the car, whereas older users with high estimated salary purchased the car.
- But there are some purple points in the green region (Buying the car) and some green points in the purple region (Not buying the car). So we can say that younger users with a high estimated salary purchased the car, whereas an older user with a low estimated salary did not purchase the car.

The goal of the classifier:

We have successfully visualized the training set result for the logistic regression, and our goal for this classification is to divide the users who purchased the SUV car and who did not purchase the car. So from the output graph, we can clearly see the two regions (Purple and Green) with the observation points. The Purple region is for those users who didn't buy the car, and Green Region is for those users who purchased the car.

Linear Classifier:

As we can see from the graph, the classifier is a Straight line or linear in nature as we have used the Linear model for Logistic Regression. In further topics, we will learn for non-linear Classifiers.

Visualizing the test set result:

Our model is well trained using the training dataset. Now, we will visualize the result for new observations (Test set). The code for the test set will remain same as above except that here we will use **x_test** and **y_test** instead of **x_train** and **y_train**. Below is the code for it:

```
Visulaizing the test set result
from matplotlib.colors import ListedColormap
x_set, y_set = x_test, y_test
x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1, step = 0.01),
nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shape),
alpha = 0.75, cmap = ListedColormap(('purple', 'green' )))
mtp.xlim(x1.min(), x1.max())
mtp.ylim(x2.min(), x2.max())
for i, j in enumerate(nm.unique(y_set)):
mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
c = ListedColormap(('purple', 'green'))(i), label = j)
mtp.title('Logistic Regression (Test set)')
mtp.xlabel('Age')
mtp.ylabel('Estimated Salary')
mtp.legend()
mtp.show()
```

Output:



The above graph shows the test set result. As we can see, the graph is divided into two regions (Purple and Green). And Green observations are in the green region, and Purple observations are in the purple region. So we can say it is a good prediction and model. Some of the green and purple data points are in different regions, which can be ignored as we have already calculated this error using the confusion matrix (11 Incorrect output). Hence our model is pretty good and ready to make new predictions for this classification problem.

Steps in Logistic Regression: To implement the Logistic Regression using Python, we will use the same steps as we have done in previous topics of Regression. Below are the steps:

- Data Pre-processing step
- Fitting Logistic Regression to the Training set
- Predicting the test result
- Test accuracy of the result (Creation of Confusion matrix)
- Visualizing the test set result.

The goal of the classifier: We have successfully visualized the training set result for the logistic regression, and our goal for this classification is to divide the users who purchased the SUV car and who did not purchase the car. So from the output graph, we can clearly see the two regions (Purple and Green) with the observation points. The Purple region is for those users who didn't buy the car, and Green Region is for those users who purchased the car.

5.3 KNN CLASSIFICATION :-

- 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.

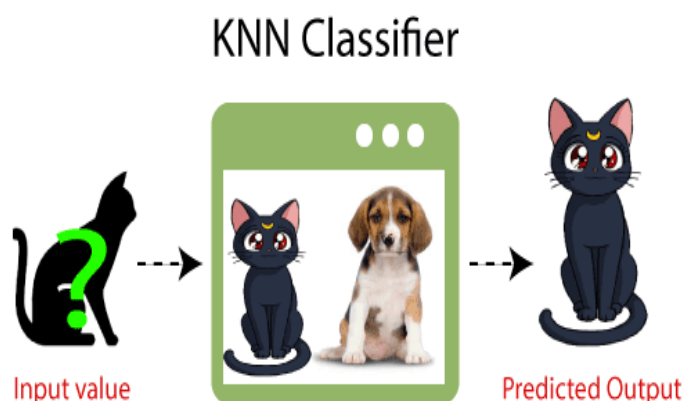
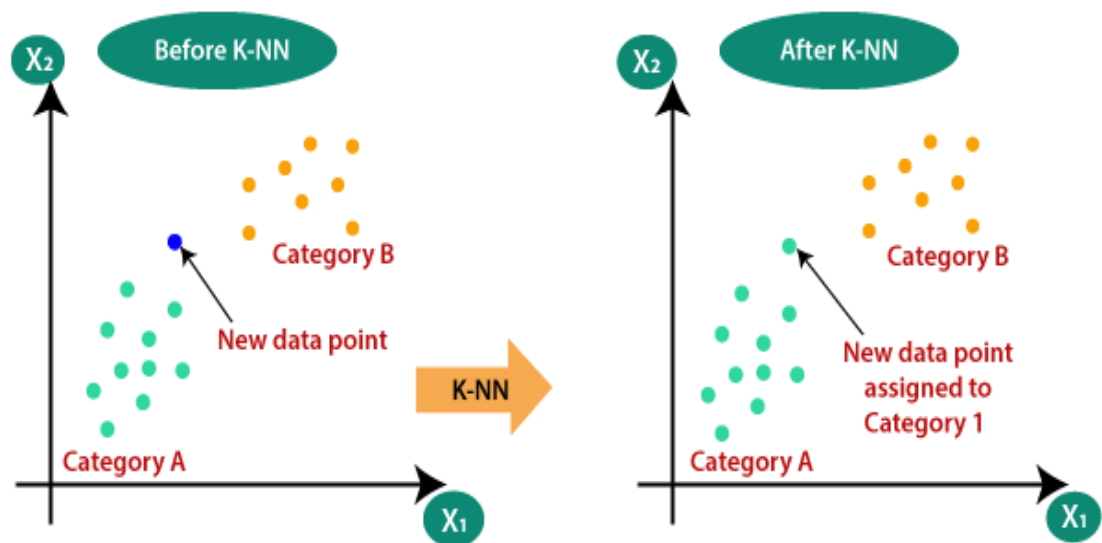


Fig 5.3.1 KNN Classifier

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 x_1 , 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:

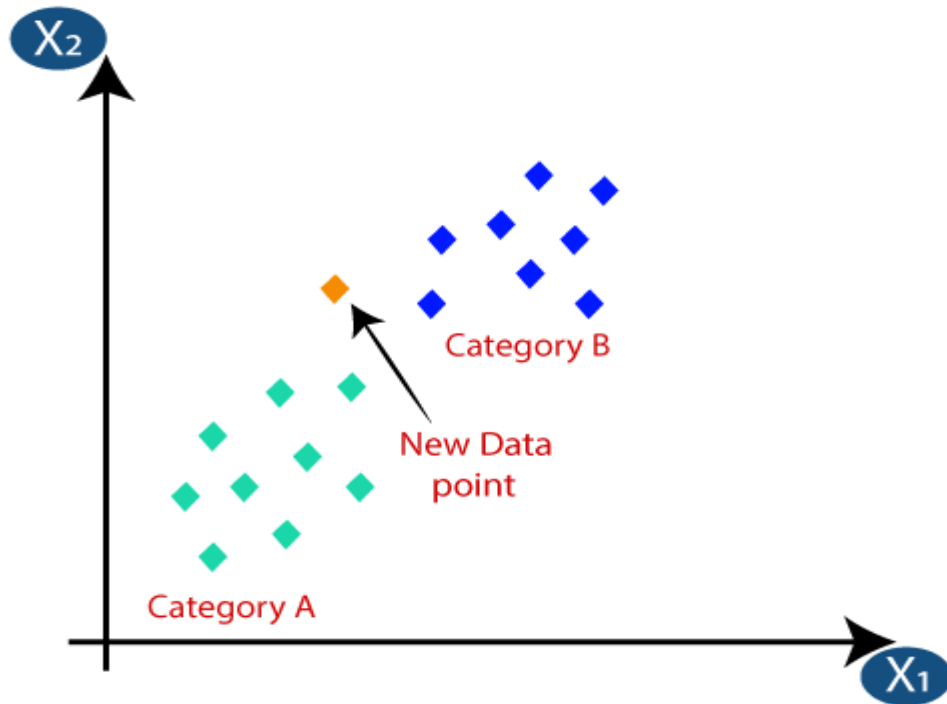


How does K-NN work?

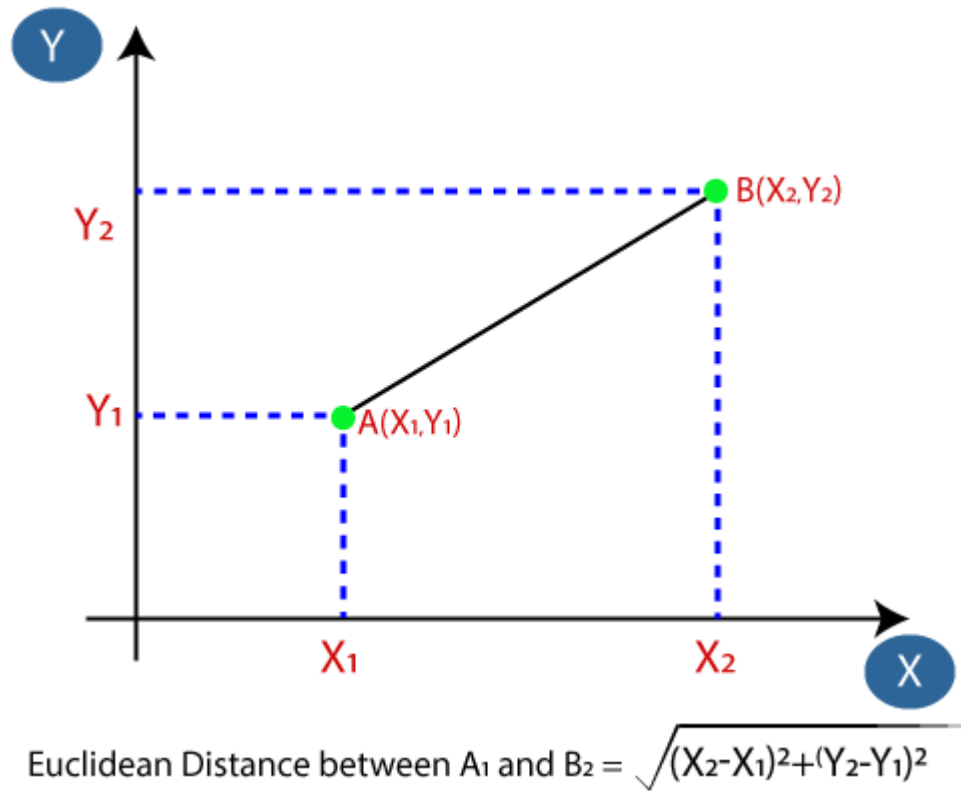
The K-NN working can be explained on the basis of the below algorithm:

- **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:



- 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:



- As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.

How to select the value of K in the K-NN Algorithm?

Below are some points to remember while selecting the value of K in the K-NN algorithm:

- There is no particular way to determine the best value for "K", so we need to try some values to find the best out of them. The most preferred value for K is 5.
- A very low value for K such as $K=1$ or $K=2$, can be noisy and lead to the effects of outliers in the model.
- Large values for K are good, but it may find some difficulties.

Advantages of KNN Algorithm:

- It is simple to implement.
- It is robust to the noisy training data
- It can be more effective if the training data is large.

Disadvantages of KNN Algorithm:

- Always needs to determine the value of K which may be complex some time.
- The computation cost is high because of calculating the distance between the data points for all the training samples.

Python implementation of the KNN algorithm

To do the Python implementation of the K-NN algorithm, we will use the same problem and dataset which we have used in Logistic Regression. But here we will improve the performance of the model. Below is the problem description:

Problem for K-NN Algorithm: There is a Car manufacturer company that has manufactured a new SUV car. The company wants to give the ads to the users who are interested in buying that SUV. So for this problem, we have a dataset that contains multiple user's information through the social network. The dataset contains lots of information but the **Estimated Salary** and **Age** we will consider for the independent variable and the **Purchased variable** is for the dependent variable. Below is the dataset:

User ID	Gender	Age	EstimatedSalary	Purchased
15624510	Male	19	19000	0
15810944	Male	35	20000	0
15668575	Female	26	43000	0
15603246	Female	27	57000	0
15804002	Male	19	76000	0
15728773	Male	27	58000	0
15598044	Female	27	84000	0
15694829	Female	32	150000	1
15600575	Male	25	33000	0
15727311	Female	35	65000	0
15570769	Female	26	80000	0
15606274	Female	26	52000	0
15746139	Male	20	86000	0
15704987	Male	32	18000	0
15628972	Male	18	82000	0
15697686	Male	29	80000	0
15733883	Male	47	25000	1
15617482	Male	45	26000	1
15704583	Male	46	28000	1
15621083	Female	48	29000	1
15649487	Male	45	22000	1
15736760	Female	47	49000	1

Steps to implement the K-NN algorithm:

- Data Pre-processing step
- Fitting the K-NN algorithm to the Training set
- Predicting the test result
- Test accuracy of the result(Creation of Confusion matrix)
- Visualizing the test set result.

Data Pre-Processing Step:

The Data Pre-processing step will remain exactly the same as Logistic Regression. Below is the code for it:

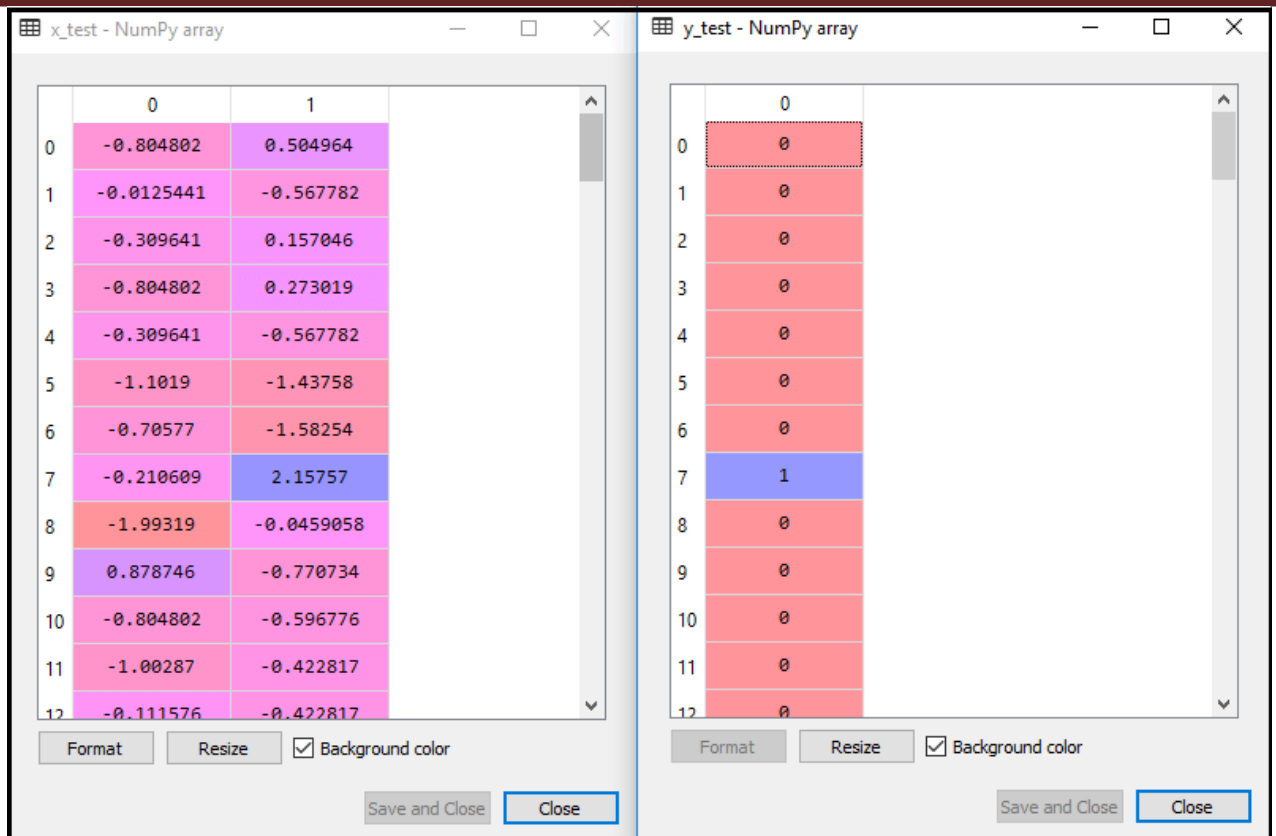
```
# importing libraries
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd

#importing datasets
data_set= pd.read_csv('user_data.csv')

#Extracting Independent and dependent Variable
x= data_set.iloc[:, [2,3]].values
y= data_set.iloc[:, 4].values
# Splitting the dataset into training and test set.
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.25, random_state=0)

#feature Scaling
from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
x_test= st_x.transform(x_test)
```

By executing the above code, our dataset is imported to our program and well pre-processed. After feature scaling our test dataset will look like:



From the above output image, we can see that our data is successfully scaled.

Fitting KNN Classifier to the Training data:-

Now we will fit the K-NN classifier to the training data. To do this we will import the KNeighbors Classifier class of Sklearn Neighbors library. After importing the class, we will create the Classifier object of the class. The Parameter of this class will be

- `n_neighbors`: To define the required neighbors of the algorithm. Usually, it takes 5.
- `metric='minkowski'`: This is the default parameter and it decides the distance between the points.
- `p=2`: It is equivalent to the standard Euclidean metric.

And then we will fit the classifier to the training data. Below is the code for it:

```
Fitting K-NN classifier to the training set
from sklearn.neighbors import KNeighborsClassifier
classifier= KNeighborsClassifier(n_neighbors=5, metric='minkowski', p=2 )
classifier.fit(x_train, y_train)
```

Output: By executing the above code, we will get the output as:

Out[10]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',  
                    metric_params=None, n_jobs=None, n_neighbors=5, p=2,  
                    weights='uniform')
```

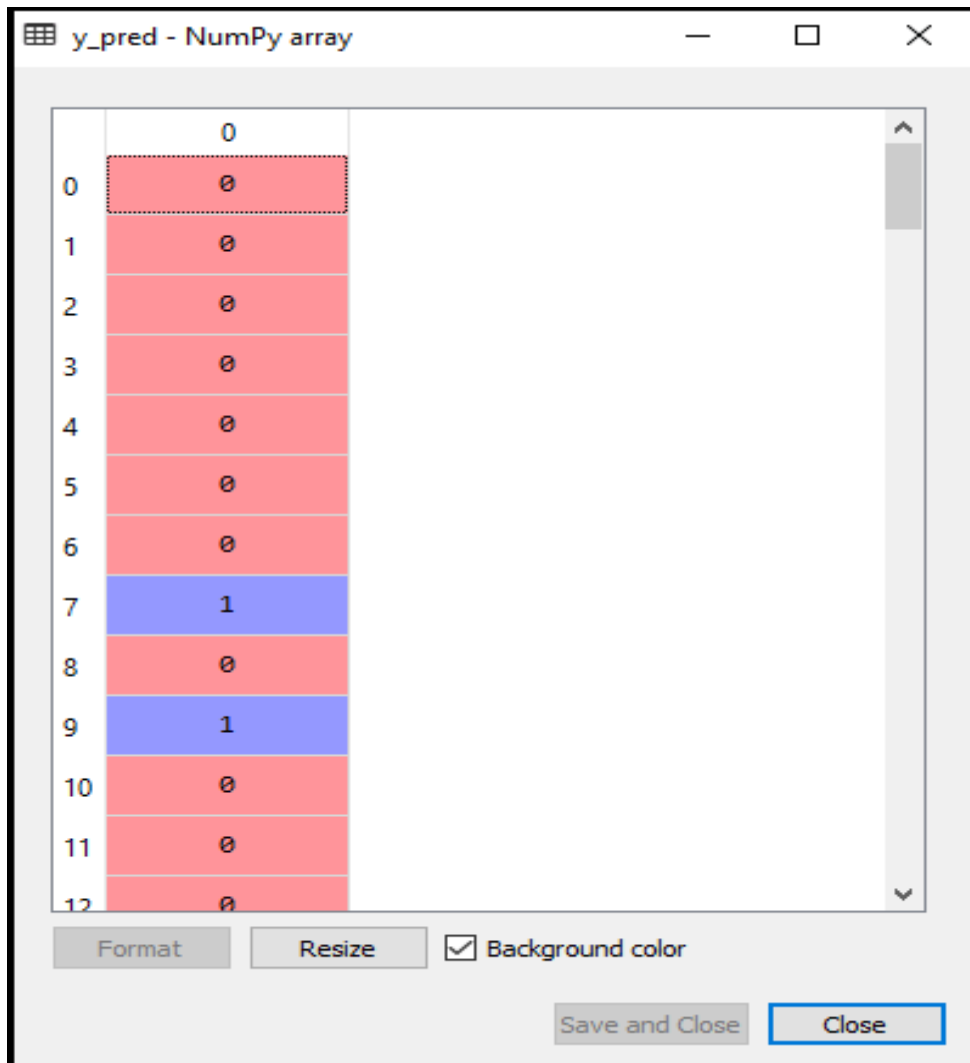
- **Predicting the Test Result:** To predict the test set result, we will create a **y_pred** vector as we did in Logistic Regression. Below is the code for it:

Predicting the test set result

```
y_pred= classifier.predict(x_test)
```

Output:

The output for the above code will be:



○

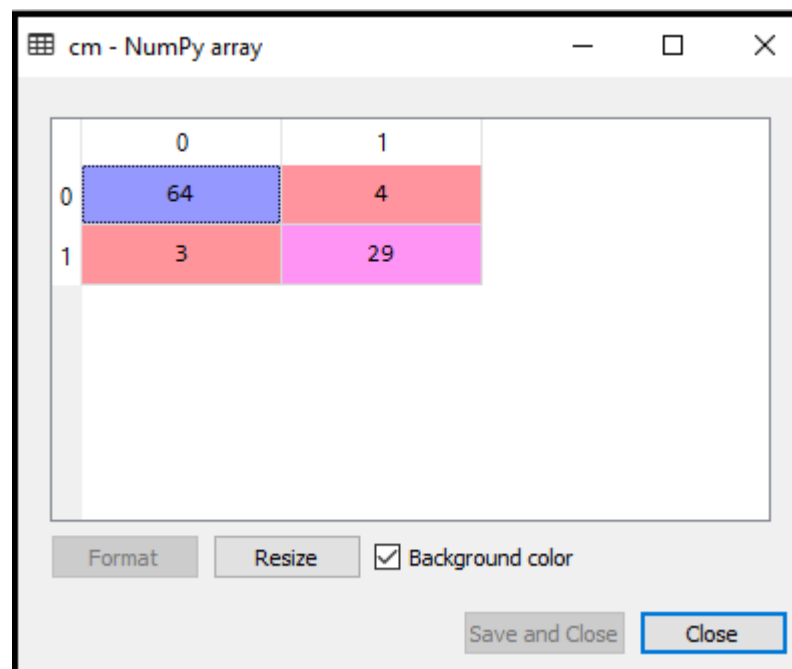
- **Creating a confusion matrix:-**

Now we will create the Confusion Matrix for our K-NN model to see the accuracy of the classifier. Below is the code for it:

```
#Creating the Confusion matrix  
from sklearn.metrics import confusion_matrix  
cm= confusion_matrix(y_test, y_pred)
```

In above code, we have imported the confusion_matrix function and called it using the variable cm.

Output: By executing the above code, we will get the matrix as below:



In the above image, we can see there are $64+29= 93$ correct predictions and $3+4= 7$ incorrect predictions, whereas, in Logistic Regression, there were 11 incorrect predictions. So we can say that the performance of the model is improved by using the K-NN algorithm.

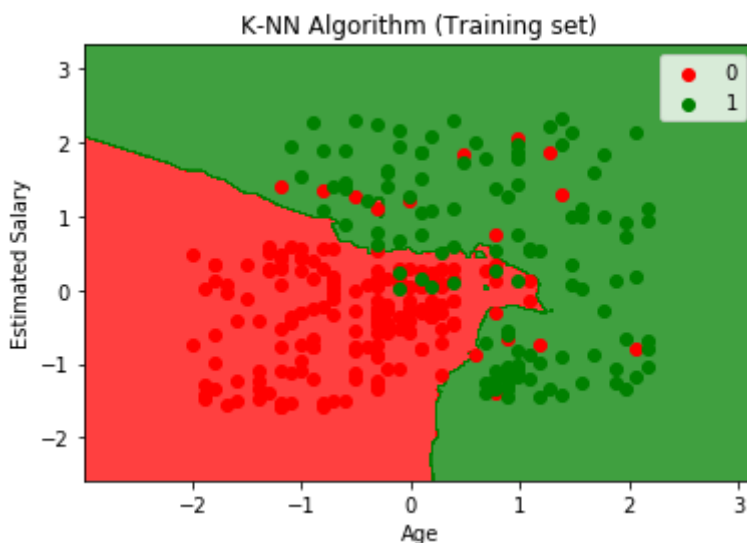
Visualizing The Training set result:-

Now, we will visualize the training set result for K-NN model. The code will remain same as we did in Logistic Regression, except the name of the graph. Below is the code for it:

```
#Visualizing the training set result
from matplotlib.colors import ListedColormap
x_set, y_set = x_train, y_train
x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() -
1, stop = x_set[:, 0].max() + 1, step = 0.01),
nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1.s
hape),
alpha = 0.75, cmap = ListedColormap(('red','green' )))
mtp.xlim(x1.min(), x1.max())
mtp.ylim(x2.min(), x2.max())
for i, j in enumerate(nm.unique(y_set)):
mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
c = ListedColormap(('red', 'green'))(i), label = j)
mtp.title('K-NN Algorithm (Training set)')
mtp.xlabel('Age')
mtp.ylabel('Estimated Salary')
mtp.legend()
mtp.show()
```

Output:

By executing the above code, we will get the below graph:



The output graph is different from the graph which we have occurred in Logistic Regression. It can be understood in the below points:

- As we can see the graph is showing the red point and green points. The green points are for Purchased(1) and Red Points for not Purchased(0) variable.

- The graph is showing an irregular boundary instead of showing any straight line or any curve because it is a K-NN algorithm, i.e., finding the nearest neighbor.
- The graph has classified users in the correct categories as most of the users who didn't buy the SUV are in the red region and users who bought the SUV are in the green region.
- The graph is showing good result but still, there are some green points in the red region and red points in the green region. But this is no big issue as by doing this model is prevented from overfitting issues.
- Hence our model is well trained.

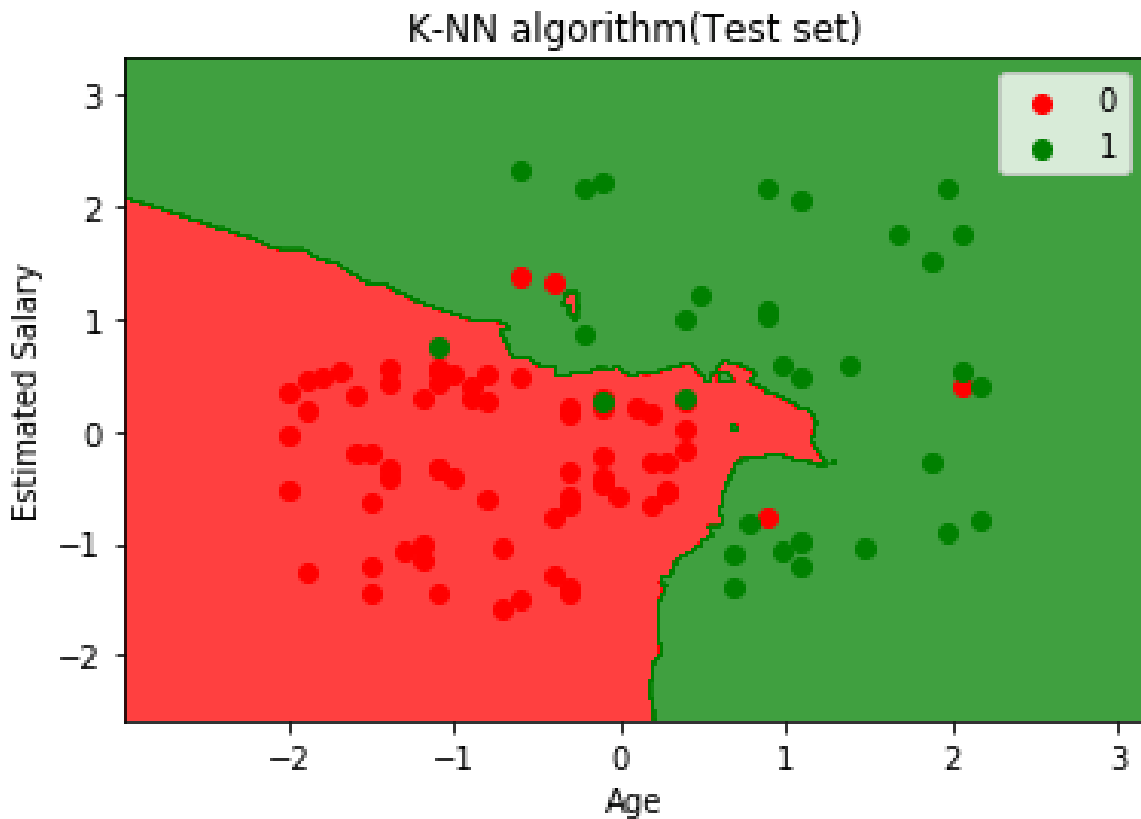
Visualizing the Test set result:

After the training of the model, we will now test the result by putting a new dataset, i.e., Test dataset. Code remains the same except some minor changes: such as **x_train** and **y_train** will be replaced by **x_test** and **y_test**.

Below is the code for it:

```
#Visualizing the test set result
from matplotlib.colors import ListedColormap
x_set, y_set = x_test, y_test
x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1, step = 0.01),
nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shape),
alpha = 0.75, cmap = ListedColormap(('red', 'green' )))
mtp.xlim(x1.min(), x1.max())
mtp.ylim(x2.min(), x2.max())
for i, j in enumerate(nm.unique(y_set)):
mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
c = ListedColormap(('red', 'green'))(i), label = j)
mtp.title('K-NN algorithm(Test set)')
mtp.xlabel('Age')
mtp.ylabel('Estimated Salary')
mtp.legend()
mtp.show()
```

Output:



The above graph is showing the output for the test data set. As we can see in the graph, the predicted output is well good as most of the red points are in the red region and most of the green points are in the green region.

However, there are few green points in the red region and a few red points in the green region. So these are the incorrect observations that we have observed in the confusion matrix(7 Incorrect output).

CHAPTER 6

TESTING

TYPES OF TESTS

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.

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:

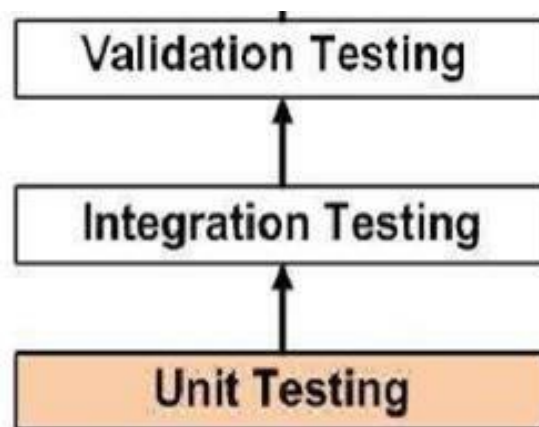


Fig 6.1 The Testing Process

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 of 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 TESTING OF INITIALIZATION AND UICOMPONENTS

Serial Number of Test Case	TC 01
Module Under Test	User Registration
Description	A user enters their details for registering themselves to the System
Input	Details of Users such as username, email, phone, age, password.
Output	If the user's details are correct, user is registered. If the user's details are incorrect, Displays error message. If the user is already registered, Displays error message.
Remarks	Test Successful.

Table 6.5.1 Test Case for User Registration

Serial Number of Test Case	TC 02
Module Under Test	User Login
Description	When the user tries to log in, details of user are verified in the system
Input	Username and Password.
Output	If the login details are correct, the user is logged in and user page is displayed. If the login details are incorrect, Displays error message.
Remarks	Test Successful.

Table 6.5.2 Test Case for User Login

Serial Number of Test Case	TC 03
Module Under Test	Prediction Result
Description	User needs to enter the name and symptoms to get the prediction result.
Input	Name and Symptoms
Output	If user enters all 5 correct symptoms then the accuracy will be high. If user enters only few symptoms then accuracy will be low.
Remarks	Test Successful.

Table 6.5.3 Test Case for Prediction Result

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

7.1 CONCLUSION

So, Finally I conclude by saying that, this project Disease prediction 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 predict the diseases of the patients based on their general information and there 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 and all other useful information the user can get to know the disease he/she is suffering from and 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 diseases. If health industry adopts this project then the work of the doctors can be reduced and they can easily predict the disease of the patient. The Disease prediction is to provide prediction for the various and generally occurring diseases 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.

7.2 FUTURE ENHANCEMENT

- 7.2.1 Facility for modifying user detail.
- 7.2.2 More interactive user interface.
- 7.2.3 Facilities for Backup creation.
- 7.2.4 Can be done as Web page.
- 7.2.5 Can be done as Mobile Application.
- 7.2.6 More Details and Latest Diseases.

REFERENCE

- Machine learning can predict survival of patients with heart failure from serum creatinine and ejection fraction alone. (03 February 2020)
<https://doi.org/10.1186/s12911-020-1023-5>
- Disease Prediction and Doctor Recommendation System by www.irjet.net
- GDPS - General Disease Prediction System by www.irjet.net
- Disease Prediction Using Machine Learning by International Research Journal of Engineering and Technology (IRJET).
- Kaveeshwar, S.A., and Cornwall, J., 2014, “The current state of disease mellitus in India”. AMJ, 7(1), pp. 45-48.
- Dean, L., McEntyre, J., 2004, “The Genetic Landscape of Disease [Internet]. Bethesda (MD): National Center for Biotechnology Information (US); Chapter 1, Introduction to Disease. 2004 Jul 7.
- Machine Learning Methods Used in Disease by www.wikipedia.com
- https://www.researchgate.net/publication/325116774_disease_prediction_using_machine_learning_techniques
- https://ieeexplore.ieee.org/document/8819782/disease_prediction
- Algorithms Details from www.dataspirant.com
- https://www.youtube.com/disease_prediction
- https://www.slideshare.com/disease_prediction

- [https:// en.wikipedia.org/machine learning algorithms](https://en.wikipedia.org/machine_learning_algorithms)
- [https://en.wikipedia.org/wiki/Python \(programming language\)](https://en.wikipedia.org/wiki/Python_(programming_language))
- <https://wiki.python.org/TkInter>

