PROFESSIONAL TRAINING REPORT

at

SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
(Deemed to be University)

Submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in Computer Science and Engineering

by

VUSA VAMSI KRISHNA
(40111453)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING SCHOOL OF COMPUTING

SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119, TAMIL NADU

APRIL 2023

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

(Established under Section 3 of UGC Act, 1956)

JEPPIAAR NAGAR, RAJIV GANDHI SALAI

CHENNAI– 600119

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Professional Training Report is the bonafide work of **Mr. VUSA VAMSI KRISHNA (40111453)** who carried out the project entitled "**DIABETES RISK PREDICTION**" under my supervision from January 2023 to April 2023.

Internal Guide

Dr. J. Albert Mayan, M.E., Ph.D.,

Head of the Department

Dr. L. Lakshmanan, M.E., Ph.D.,

Submitted for Viva voce Examination held on ______________

Internal Examiner **External Examiner**

DECLARATION

I, **VUSA VAMSI KRISHNA** hereby declare that the Professional Training Report entitled “**DIABETES RISK PREDICTION”** done by me under the guidance of **Dr. J. Albert Mayan, M.E., Ph.D.,** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in Computer Science and Engineering.

DATE:

PLACE: **SIGNATURE OF THE CANDIDATE**

ACKNOWLEDGEMENT

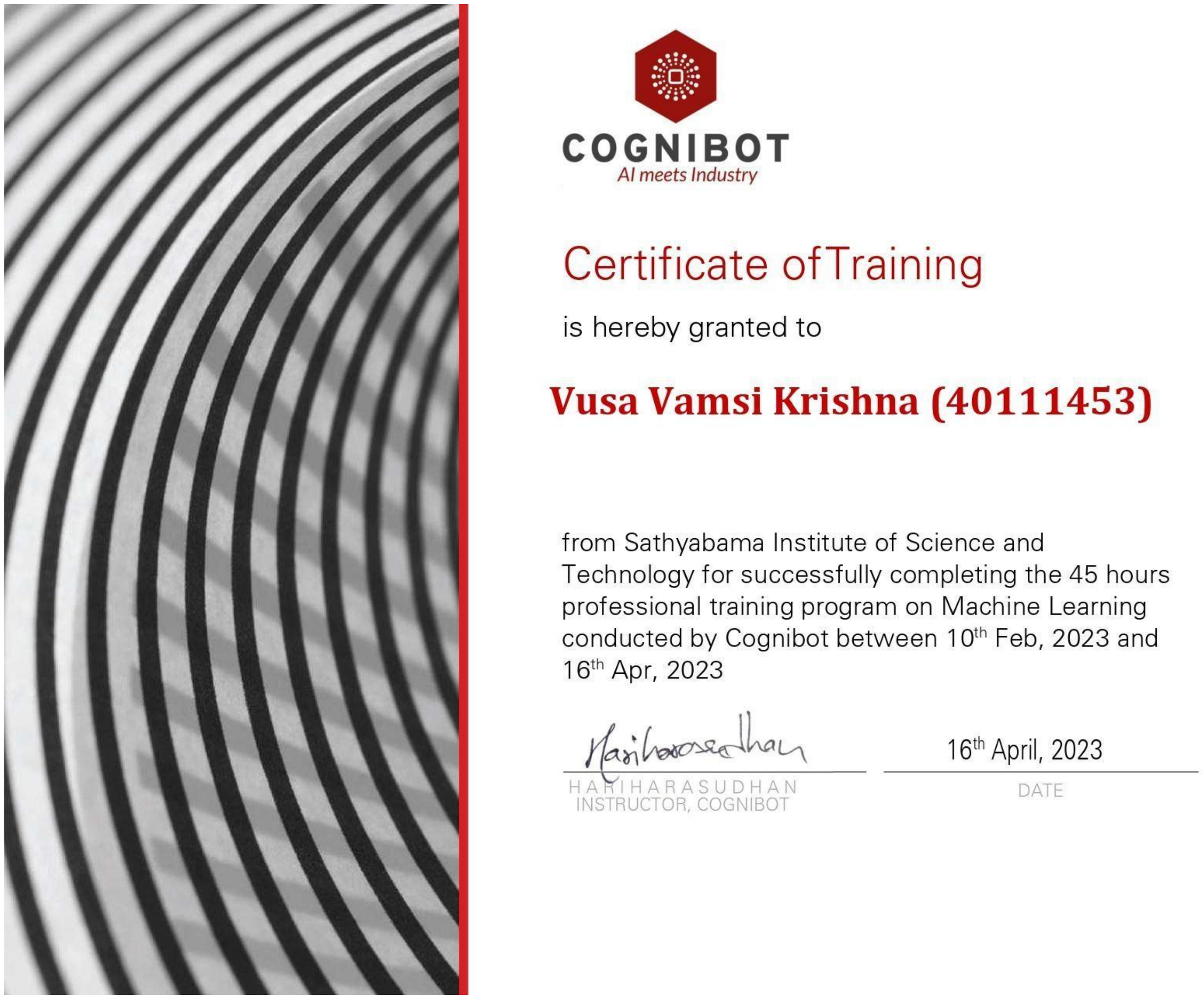
I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

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

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. J. Albert Mayan, M.E., Ph.D.,** for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

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

COURSE CERTIFICATE



COGNIBOT

AI meets Industry

Certificate of Training

is hereby granted to

Vusa Vamsi Krishna (40111453)

from Sathyabama Institute of Science and Technology for successfully completing the 45 hours professional training program on Machine Learning conducted by Cognibot between 10th Feb, 2023 and 16th Apr, 2023

[illegible]

HARIHARASUDHAN
INSTRUCTOR, COGNIBOT

16th April, 2023

DATE

ABSTRACT

Diabetes mellitus (DM) is a metabolic disease characterized by high blood sugar. The main clinical types are type 1 diabetes and type 2 diabetes. Now, the proportion of young people suffering from type 1 diabetes has increased significantly. Type 1 diabetes is chronic when it occurs in childhood and adolescence and has a long incubation period. The early symptoms of the onset are not obvious, which may lead to failure to detect in time and delay treatment. Long-term high blood sugar can cause chronic damage and dysfunction of various tissues, especially eyes, kidneys, heart, blood vessels and nerves. Therefore, the early prediction of diabetes is particularly important. We use supervised machine-learning algorithms like Random- forest classifier to train on the actual data of 520 diabetic patients and potential diabetic patients aged 16 to 90. Through comparative analysis of classification and recognition accuracy, the performance of random forest is the best Jupyter Notebook is utilized as the dominant unit coded in Python language.

Keywords- Machine learning, Jupyter Notebook, Python language.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 ABOUT DIABETES MELLITUS

Diabetes is one of deadliest diseases in the world. It is not only a disease but also a creator of different kind of diseases like heart attack, blindness, kidney diseases, etc. Diabetes Mellitus (DM) is defined as a group of metabolic disorders mainly caused by abnormal insulin secretion and/or action. Insulin deficiency results in elevated blood glucose levels (hyperglycemia) and impaired metabolism of carbohydrates, fat and proteins. DM is one of the most common endocrine disorders, affecting more than 200 million people worldwide. The onset of diabetes is estimated to dramatically in the upcoming years. DM can be divided into several distinct types. However, there are two major clinical types, type 1 diabetes (T1D) and type 2 diabetes (T2D), according to the etiopathology of the disorder. T2D appears to be the most common form of diabetes (90% of all diabetic patients), mainly characterized by insulin resistance. The main causes of T2D include lifestyle, physical activity, dietary habits and heredity, whereas T1D is thought to be due to autoimmune logical destruction of the Langerhans islets hosting pancreatic-β cells. T1D affects almost 10% of all diabetic patients worldwide, with 10% of them ultimately developing idiopathic diabetes. Other forms of DM, classified on the basis of insulin secretion profile and/or onset, include Gestational Diabetes, endocrinopathies, MODY (Maturity Onset Diabetes of the Young), neonatal, mitochondrial, and pregnancy diabetes. The symptoms of DM include polyuria, polydipsia, and significant weight loss among others. Diagnosis depends on blood glucose levels (fasting plasma glucose = 7.0 mmol/L).

1.2 MACHINE LEARNING

Machine learning is the scientific field dealing with the ways in which machines learn from experience. For many scientists, the term “machine learning” is identical to the “artificial intelligence” given that the possibility of learning is the main characteristic of an entity called intelligent in the broadest sense of the word. The purpose of machine learning is the construction of computer systems that can adapt and learn from their experience. A more detailed and formal definition of machine learning is given by Mitchel: A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E. With the rise of Machine Learning approaches, we have the ability to find a solution to this issue, we have developed a system using data mining which has the ability to predict whether the patient has diabetes or not. Furthermore, predicting the disease early leads to treating the patients before it becomes critical. Data mining has the ability to extract hidden knowledge from a huge amount of diabetes-related data.

Because of that, it has a significant role in diabetes research, now more than ever. The aim of this research is to develop a system which can predict the diabetic risk level of a patient with a higher accuracy. This research has focused on developing a system based on Random Forest Classifier Machine learning techniques.

1.2.1 SUPERVISED LEARNING

In supervised learning, the system must which is an expression of a model describing the data. The objective function is used to predict the value of a variable, called dependent variable or output variable, from a set of variables, called independent variables or input variables or characteristics or features. The set of possible input values of the function, i.e. its domain, are called instances. Each case is described by a set of characteristics (attributes or features). A subset of all cases, for which the output variable value is known, is called training data or examples. In order to infer the best target function, the learning system, given a training set, takes into consideration alternative functions, called hypothesis and denoted by h. In supervised learning, there are two kinds of learning tasks: classification and regression. Classification models try to predict distinct classes, such as e.g. blood groups, while regression models predict numerical values. Some of the most common techniques are Decision Trees (DT), Rule Learning, and Instance Based Learning (IBL), such as k-Nearest Neighbours (k-NN), Genetic Algorithms (GA), Artificial Neural Networks (ANN), and Support Vector Machines (SVM).

1.2.2 UNSUPERVISED LEARNING

In unsupervised learning, the system tries to discover the hidden structure of data or associations between variables. In that case, training data consists of instances without any corresponding labels. Association Rule Mining appeared much later than machine learning and is subject to greater influence from the research area of databases. Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense or another) to each other than to those in other groups (clusters). It is a main task of exploratory data mining, and a common technique for statistical data analysis, used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, bioinformatics, data compression, and computer graphics.

1.2.3 REINFORCEMENT LEARNING

The term Reinforcement Learning is a general term given to a family of techniques, in which the system attempts to learn through direct interaction with the environment so as to maximize some notion of cumulative reward. It is important to mention that the system has no prior knowledge about the behavior of the environment and the only way to find out is through trial and failure (trial and error). Reinforcement learning is mainly applied to autonomous systems, due to its independence in relation to its environment

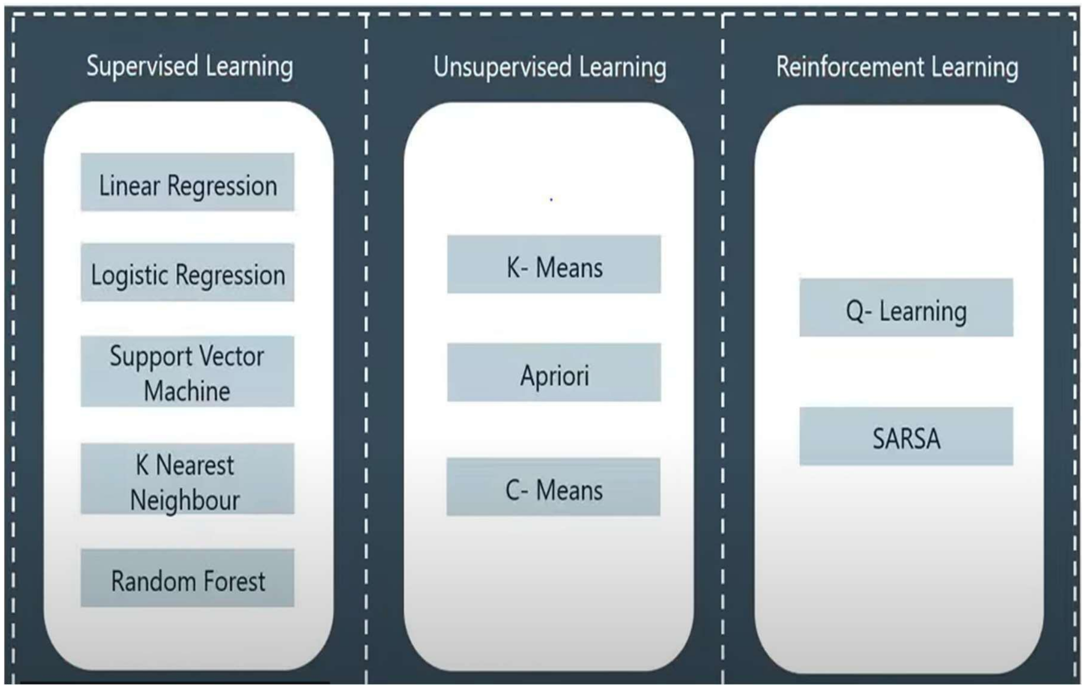


Fig.1.2. Types of Machine Learning Algorithms

The above figure describes the detail information about the types of machine learning Algorithms and their classification into supervised, unsupervised and reinforcement learning.

CHAPTER 2
LITERATURE SURVEY

a. REQUIREMENTS

SOFTWARE COMPONENTS

JUPYTER NOTEBOOK:

The JUPYTER Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative texts. uses-includes: data cleaning and transformation, numerical simulations, statistical modeling, data visualizations, machine learning and much more.

PYTHON:

Python code is understandable by humans, which makes it easier to build models for machine learning. Since Python is a general-purpose language, it can do a set of complex machine learning tasks and enable you to build prototypes quickly that allow you to test your product for machine learning purposes. And detailed history and features are explained below.

HARDWARE COMPONENTS

- **OPERATING SYSTEM**
- **GPU**

b. PYTHON FEATURES

For machine learning we use python because of its various features which makes building models easily below mentioned some of the features.

Python's features include:

0. Easy-to-learn: Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
1. Easy-to-read: Python code is more clearly defined and visible to the eyes.
2. A broad standard library: Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, Macintosh.
3. Interactive Mode: Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
4. Portable: Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
5. Extendable: You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
6. Databases: Python provides interfaces to all major commercial databases.
7. GUI Programming: Python supports GUI applications that can be created and ported to many system calls, libraries, and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
8. Scalable: Python provides a better structure and support for large programs than shell scripting.
9. Databases: Python provides interfaces to all major

commercial databases.

10. GUI Programming: Python supports GUI applications that can be created and ported to many system calls, libraries, and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
11. Scalable: Python provides a better structure and support for large programs than shell scripting.
12. Large standard libraries to solve common tasks: Python has a number of standard libraries which makes the life of a programmer.
13. A high-level, interpreted language: Unlike C/C++, you don’t have to worry about daunting tasks like memory management, garbage collection and so on. Likewise, when you run Python code, it automatically converts your code to the language your computer understands. You don't need to worry about any lower-level operations

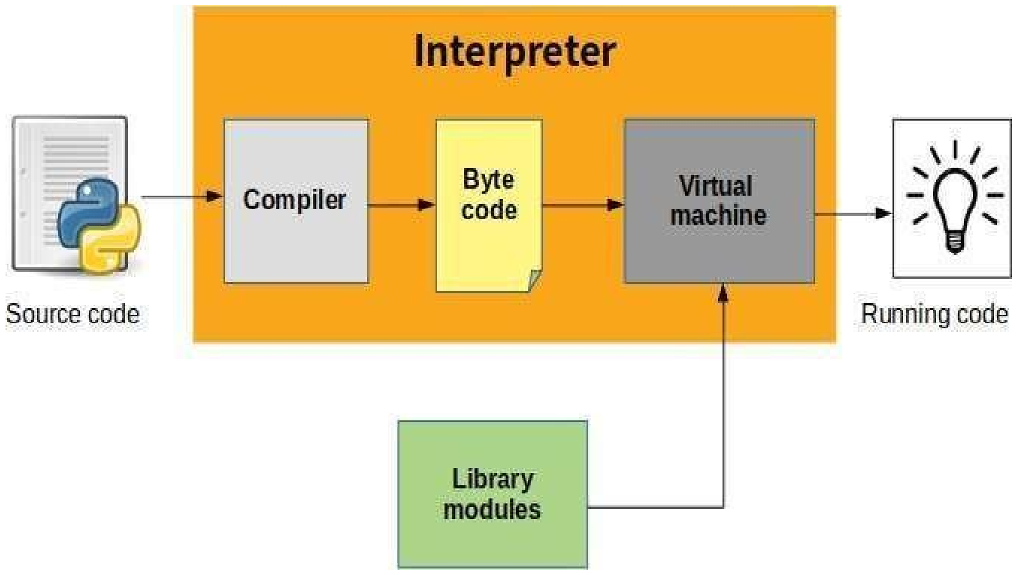


FIg.2.2. Execution of Source Code

c. METHODOLOGIES

For this project implementation the first and foremost thing is to choose the best classification model (or) algorithm. The algorithm we are going to apply should be a classifier algorithm because in our dataset the prediction attribute consists of discrete class labels. We have so many classifier algorithms like RandomForestClassifier SVM,Decision Tree Classifier,Knn etc---so the choosing of algorithms becomes important. Here we are using RandomForestClassifier and Decision Tree But when both are compared Random Forest Classifier has high accuracy in our project because randomforestclassifier is an ensemble learning model that operates by constructing a multitude of decision trees at training time. Further details will be explained below.

CHAPTER 3
AIM AND SCOPE

AIM AND OBJECTIVE

The Main aim of this project is to predict diabetes using machine learning and to showcase the different dimensions in it and with machine learning what we can achieve and it helps us to solve real world problems.

For this project we used machine learning using python because ML provides better results for prediction by constructing different models from the datasets collected from various patients**.**

The Objective of this project is to develop a system which can perform early prediction of diabetes for a patient with a higher accuracy by machine learning techniques. To achieve this goal, we gathered a diagnostic dataset from UCI repository. It contains 520 instances and 17 attributes with a few missing values which have been pre-processed by ignoring the tuples with incomplete values. The dataset is summarized in Table 1.

Fig.3.1. ATTRIBUTE DESCRIPTION

Attributes	**Description**
Age	20 years - 65 years
Sex	1.Male, 2.Female
Polyuria	1.Male, 2.Female
Polydipsia	1.Yes, 2.No.
Sudden Weight loss	1.Yes, 2.No.
Weakness	1.Yes, 2.No.
Polyphagia	1.Yes, 2.No.
Genital thrush	1.Yes, 2.No.
Visual blurring	1.Yes, 2.No.
Itching	1.Yes, 2.No.

Irritability	1.Yes, 2.No.
Delayed Healing	1.Yes, 2.No.
Partial Paresis	1.Yes, 2.No.
Muscle Stiffness	1.Yes, 2.No.
Alopecia	1.Yes, 2.No.
Obesity	1.Yes, 2.No.
Class	1.Positive, 2.Negative

MODULES

In our project we will use so many modules(libraries) which will be explained below.

NUMPY- NUMPY which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed. In machine learning we use this a lot in conversion of 1D TO 2D arrays.

PANDAS- Pandas is an open-source Python package that is most widely used for data science/data analysis and machine learning tasks. It is built on top of another package named Numpy, which provides support for multi- dimensional arrays. The two primary data structures of pandas are **Series** (1- dimensional) and **DataFrame** (2-dimensional).we mainly use dataframe in our machine learning task.

SKLEARN - Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering, and dimensionality reduction via a consistent interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib

MATPLOTLIB - Matplotlib is one of the most popular Python packages used for data visualization. It is a cross-platform library for making 2D plots from data in arrays. It provides an object-oriented API that helps in embedding plots in applications using Python GUI toolkits such as PyQt, WxPythonotTkinter. It can be used in Python and IPython shells, Jupyter notebook and web application servers also.

SEABORN - Seaborn is a data visualization library built on top of matplotlib and closely integrated with pandas data structures in Python. Visualization is the central part of Seaborn which helps in exploration and understanding of data.

INTRODUCTION TO RANDOM FOREST

Random Forest is supervised learning, used for both classification and Regression.It is an ensemble learning model that operates by constructing a multitude of decision trees at training time. The logic behind the random forest is a bagging technique to create random sample features. The difference between the decision tree and the random forest is the process of finding the root node and splitting the feature node will run randomly. The Steps are given below

1. Load the data where it consists of a number of features representing behavior of the dataset.
2. The training algorithm of random forest is called bootstrap algorithm or bagging technique to select n feature randomly from m features, i.e. to create random samples, this model trains the new sample to out of bag sample (1/3rd of the data) used to determine the unbiased OOB error.

3. Calculate the total number of votes of each tree for the predicting target. The highest voted class is the final prediction of the random forest.

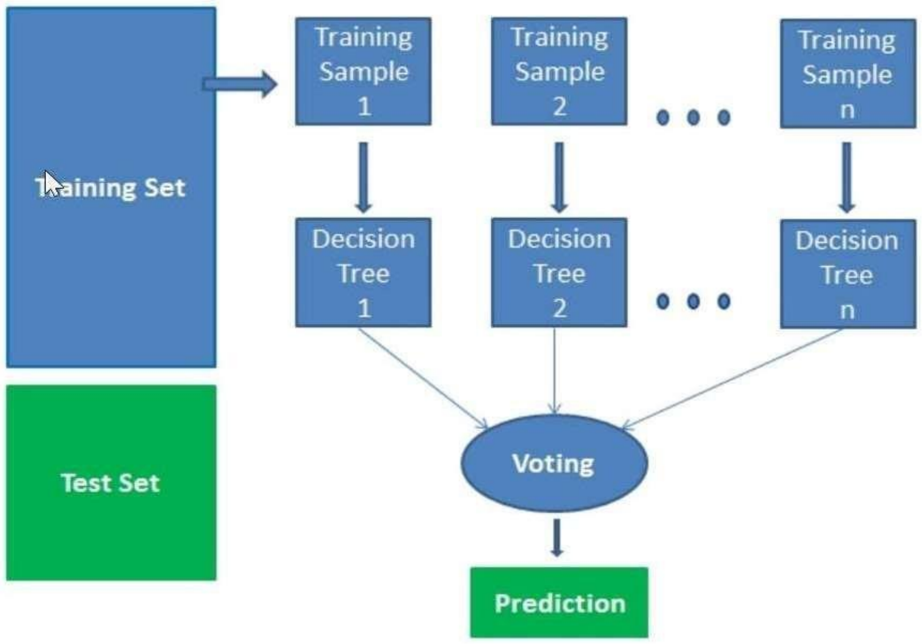


Fig.3.3. Working model of Random Forest algorithm

3.3.1 Understanding the model algorithm

Before building the model, we need to understand the model 's nature. Random forest classifier ML algorithm has parameters called hyperparameters which can be tuned to achieve maximum performance. The hyperparameters used in model model are

1. **n_estimators**: The number of decision trees in the forest

   (int, default=100).

2. **bootstrap**: Whether bootstrap samples are used when building

   trees. If false, the whole dataset is used to build each tree (bool, default=True).

3. **max_depth**: The maximum depth of the tree.If None,then nodes

   are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples. (int, default=None).

4. **random_state**: Controls both the randomness of the bootstrapping

   of the samples used when building trees and the sampling of the it features to consider when looking for the best split at each node (int, default=None).

Here the n_estimators ,bootstrap and random state are constant n_estimator represents no. of decision trees, random state is given to test values for the same shuffle data .

SYSTEM ARCHITECTURE

The work flow of this project will be mentioned in the below

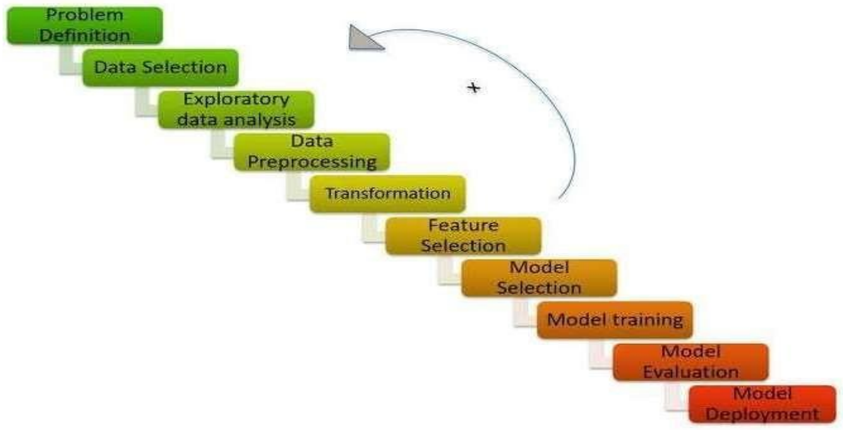


Fig. 3.4. System Architecture.

The detailed explanation of the flow diagram will be explained below.

3.5 DECISION TREE

A decision tree is a non-parametric supervised learning algorithm, which is utilized for both classification and regression tasks. It has a hierarchical tree structure,it is which consists of a root node, branches, internal nodes and leaf nodes**.**

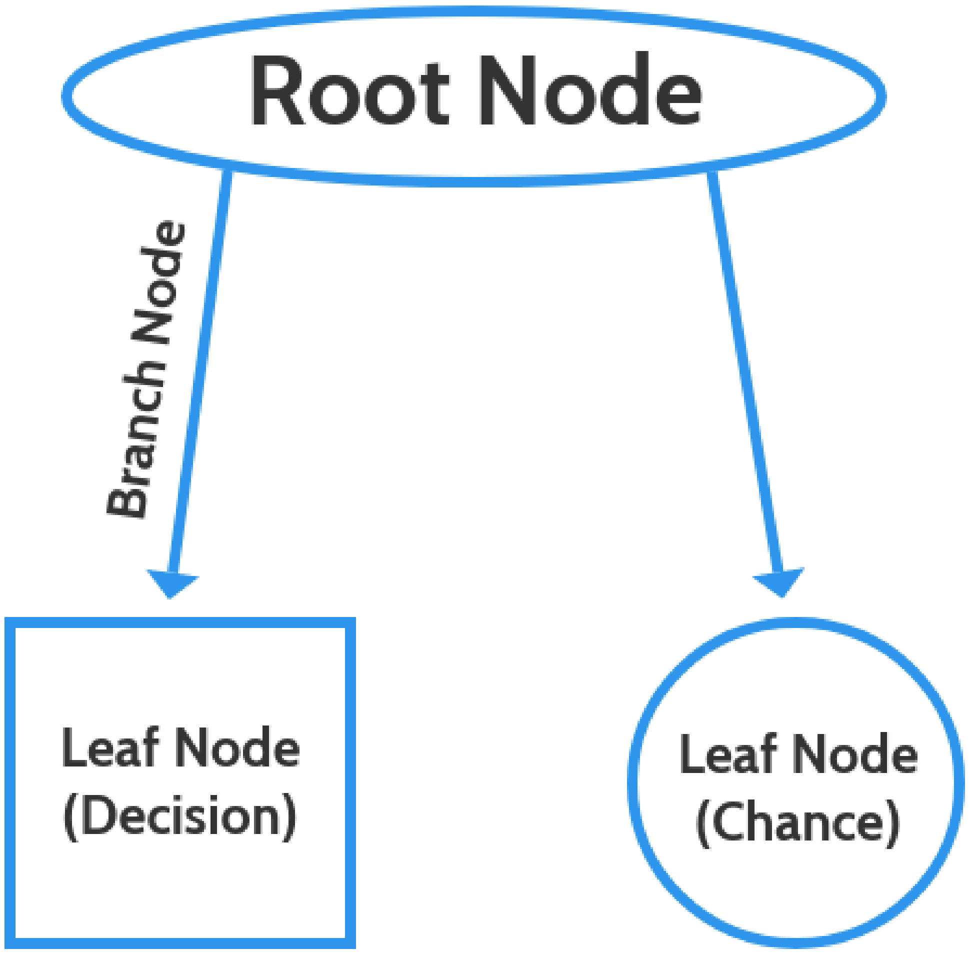
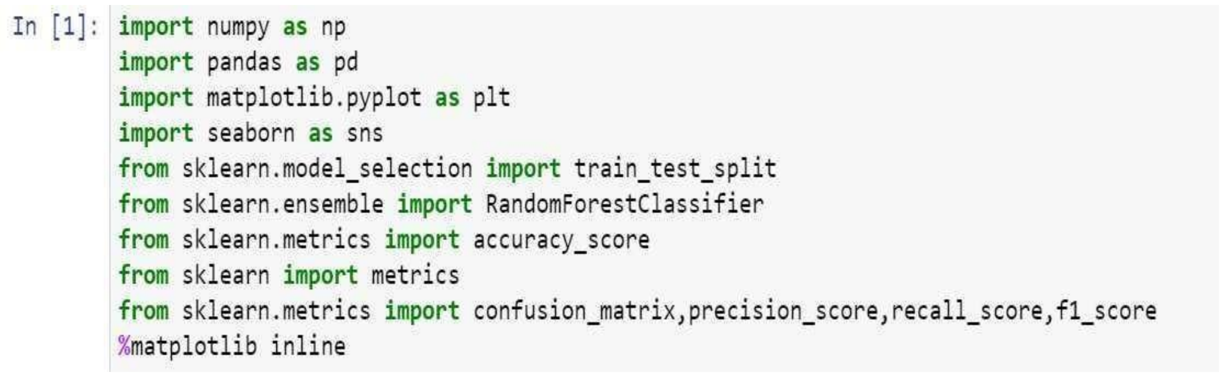


Fig. 3.5 Structure Of Decision Tree

CHAPTER 4

PROPOSED METHODOLOGY

4.1 IMPORT THE REQUIRED LIBRARIES – The first and fore most step in executing any machine learning project is to import all the required libraries which we will use through out the project. The Libraries we will use in these project are displayed below.

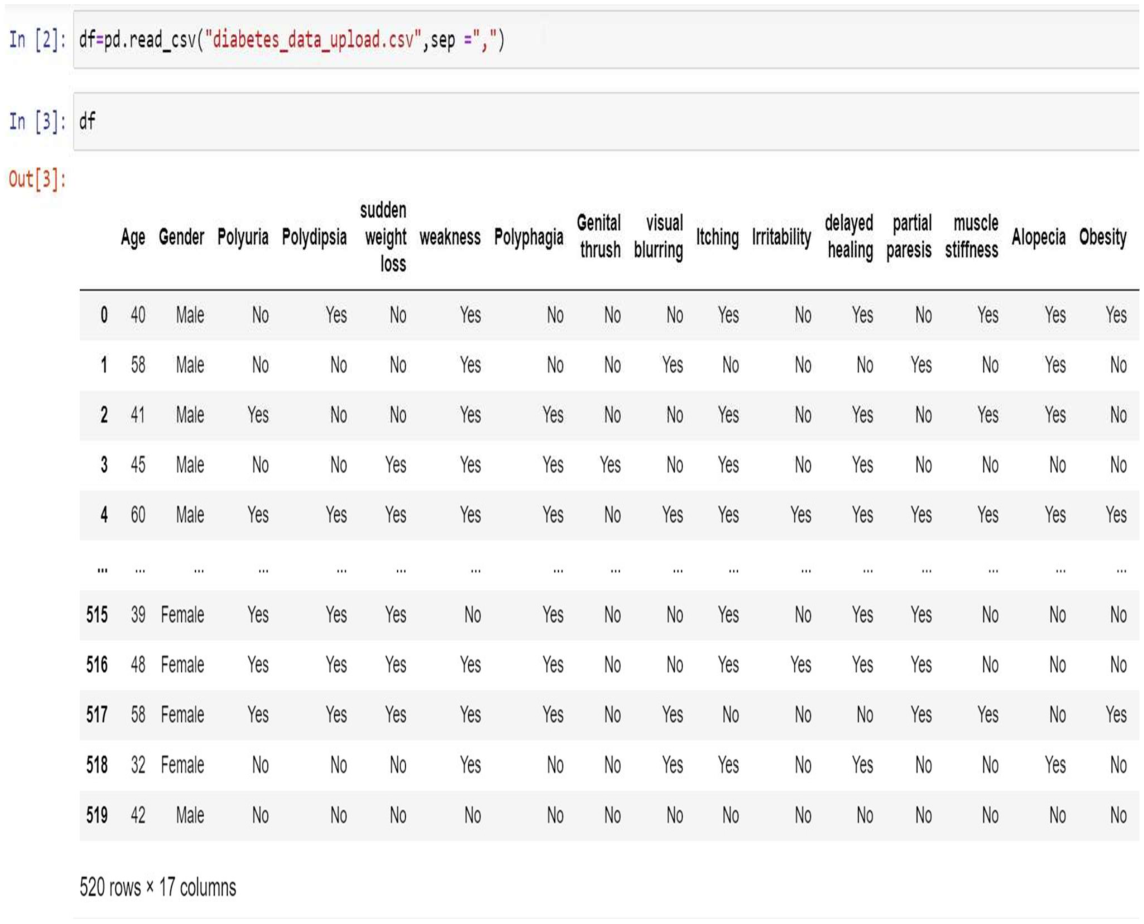


```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
        from sklearn.model_selection import train_test_split
        from sklearn.ensemble import RandomForestClassifier
        from sklearn.metrics import accuracy_score
        from sklearn import metrics
        from sklearn.metrics import confusion_matrix,precision_score,recall_score,f1_score
        %matplotlib inline
```

Fig. 4.1. Import libraries

4.2 EXPLORATORY DATA ANALYSIS

4.2.1 LOAD THE DATASET- Loading the data into the pandas data frame is certainly one of the most important steps in EDA and also in our project, as we can see that the value from the data set is comma-separated. So all we have to do is to just read the CSV into a data frame and pandas data frame does the job for us.



```
In [2]: df=pd.read_csv("diabetes_data_upload.csv",sep =",")
```

```
In [3]: df
```

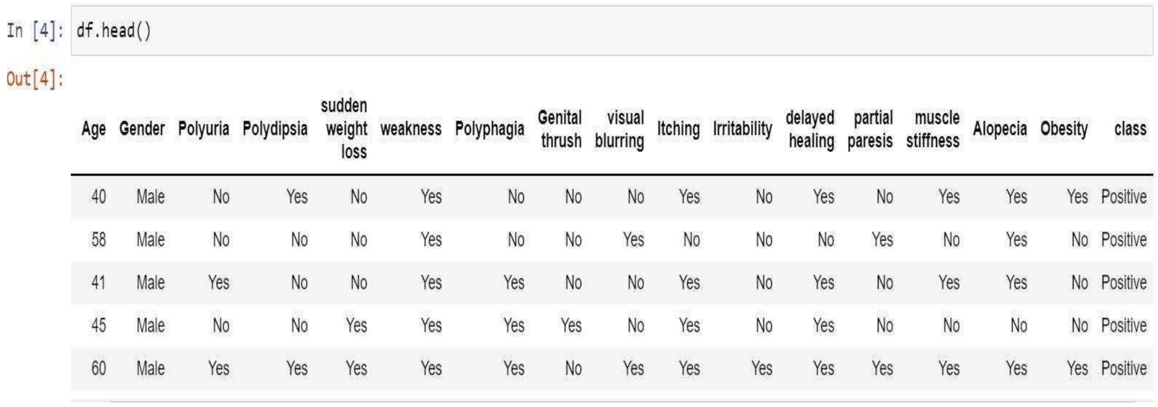
Out[3]:

	Age	Gender	Polyuria	Polydipsia	sudden weight loss	weakness	Polyphagia	Genital thrush	visual blurring	Itching	Irritability	delayed healing	partial paresis	muscle stiffness	Alopecia	Obesity
0	40	Male	No	Yes	No	Yes	No	No	No	Yes	No	Yes	No	Yes	Yes	Yes
1	58	Male	No	No	No	Yes	No	No	Yes	No	No	No	Yes	No	Yes	No
2	41	Male	Yes	No	No	Yes	Yes	No	No	Yes	No	Yes	No	Yes	Yes	No
3	45	Male	No	No	Yes	Yes	Yes	Yes	No	Yes	No	Yes	No	No	No	No
4	60	Male	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
515	39	Female	Yes	Yes	Yes	No	Yes	No	No	Yes	No	Yes	Yes	No	No	No
516	48	Female	Yes	Yes	Yes	Yes	Yes	No	No	Yes	Yes	Yes	Yes	No	No	No
517	58	Female	Yes	Yes	Yes	Yes	Yes	No	Yes	No	No	No	Yes	Yes	No	Yes
518	32	Female	No	No	No	Yes	No	No	Yes	Yes	No	Yes	No	No	Yes	No
519	42	Male	No	No	No	No	No	No	No	No	No	No	No	No	No	No

520 rows × 17 columns

Fig. 4.2.1. Loading dataset

df.head()-It displays the first five rows of the dataset

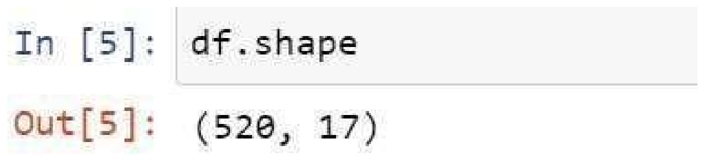


```
In [4]: df.head()
```

Out[4]:

Age	Gender	Polyuria	Polydipsia	sudden weight loss	weakness	Polyphagia	Genital thrush	visual blurring	Itching	Irritability	delayed healing	partial paresis	muscle stiffness	Alopecia	Obesity	class
40	Male	No	Yes	No	Yes	No	No	No	Yes	No	Yes	No	Yes	Yes	Yes	Positive
58	Male	No	No	No	Yes	No	No	Yes	No	No	No	Yes	No	Yes	No	Positive
41	Male	Yes	No	No	Yes	Yes	No	No	Yes	No	Yes	No	Yes	Yes	No	Positive
45	Male	No	No	Yes	Yes	Yes	Yes	No	Yes	No	Yes	No	No	No	No	Positive
60	Male	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Positive

df.shape()-It tells about how many rows and columns that our dataset contains.

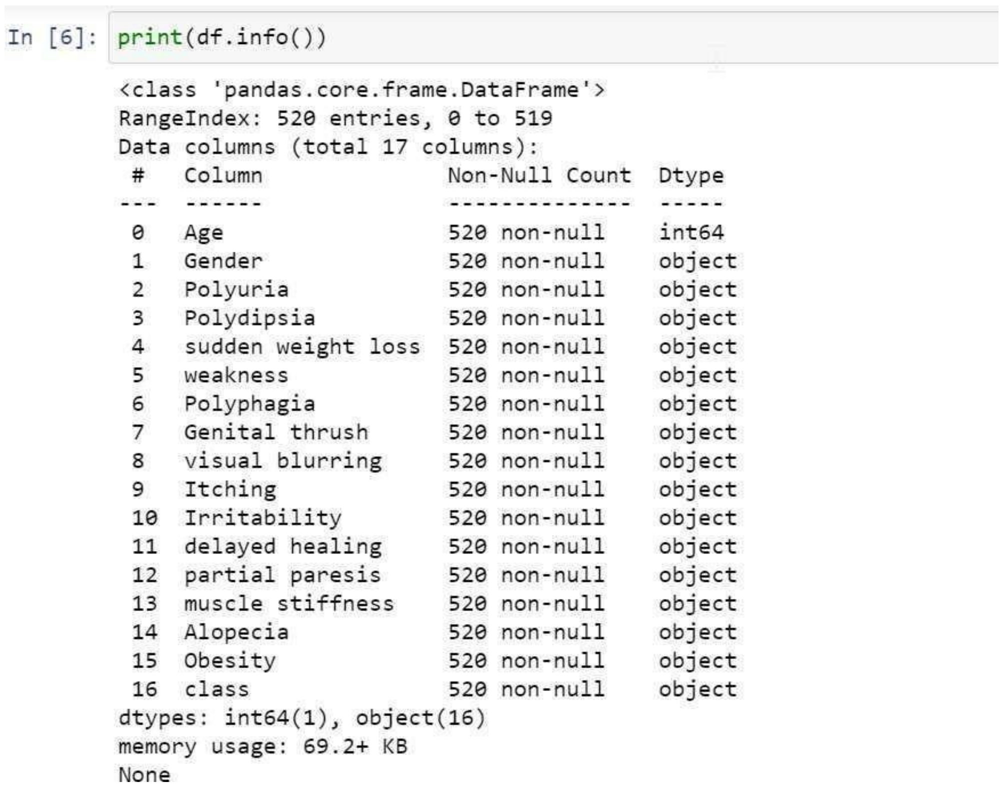


```
In [5]: df.shape

Out[5]: (520, 17)
```

Our dataset contains 520 rows and 17 columns.

4.2.1 Checking the types of data

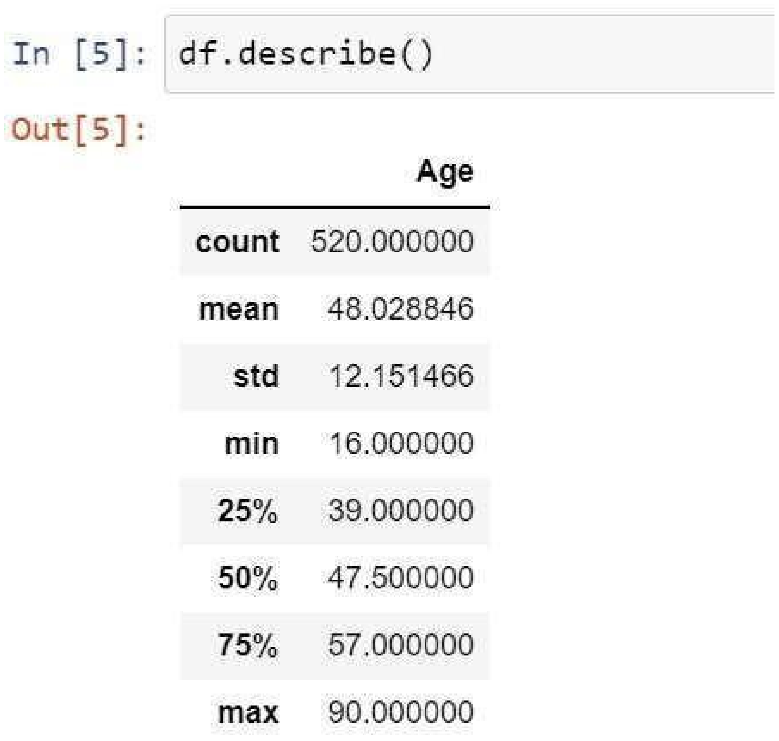
df.info ()-Here we can check for the datatype of the attributes in our dataset**.** This helps a lot from this we can convert if an attributes has a wrong datatype to correct one and it also tells about how many non-null values that each attribute contains in our dataset. Here, in this case, the data is already in integer format so nothing to worry .



```
In [6]: print(df.info())

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 520 entries, 0 to 519
Data columns (total 17 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Age                 520 non-null    int64
 1   Gender              520 non-null    object
 2   Polyuria            520 non-null    object
 3   Polydipsia          520 non-null    object
 4   sudden weight loss  520 non-null    object
 5   weakness            520 non-null    object
 6   Polyphagia          520 non-null    object
 7   Genital thrush      520 non-null    object
 8   visual blurring     520 non-null    object
 9   Itching             520 non-null    object
 10  Irritability        520 non-null    object
 11  delayed healing     520 non-null    object
 12  partial paresis     520 non-null    object
 13  muscle stiffness    520 non-null    object
 14  Alopecia            520 non-null    object
 15  Obesity             520 non-null    object
 16  class               520 non-null    object
dtypes: int64(1), object(16)
memory usage: 69.2+ KB
None
```

Fig. 4.2.2 Displaying datatypes.

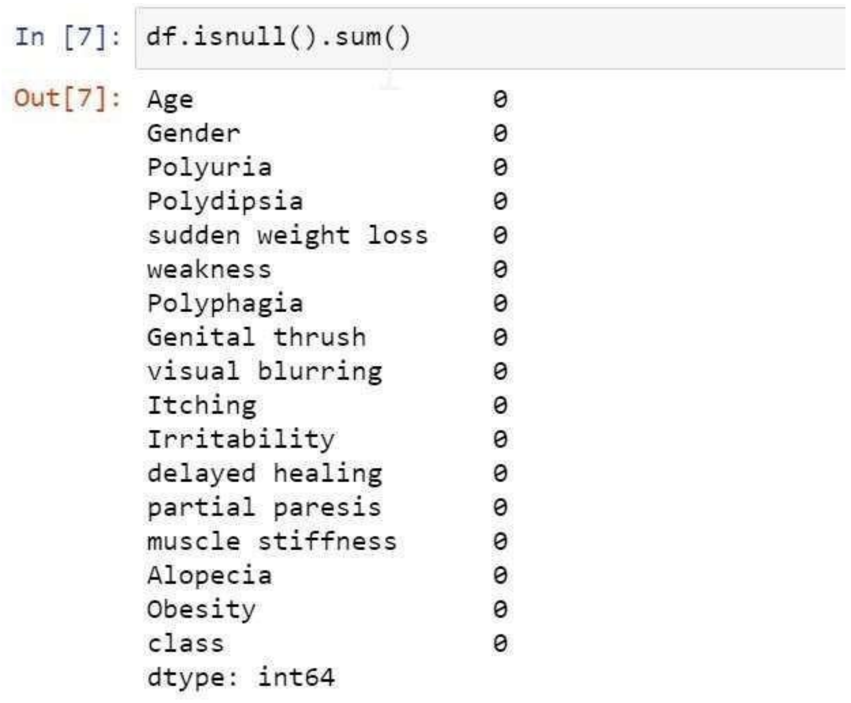
df.describe()-The describe() method is used for calculating some statistical data like **percentile, mean** and **std** of the numerical values of the Series or DataFrame. It analyzes both numeric and object series and also the DataFrame column sets of mixed data types.



```
In [5]: df.describe()
Out[5]:
```

	Age
count	520.000000
mean	48.028846
std	12.151466
min	16.000000
25%	39.000000
50%	47.500000
75%	57.000000
max	90.000000

df.isnull().sum() - It tells about how many null values that each attribute contains.Here,in this case there are no null values. If there are any null values then we have replace those values with the mean values of each attribute.



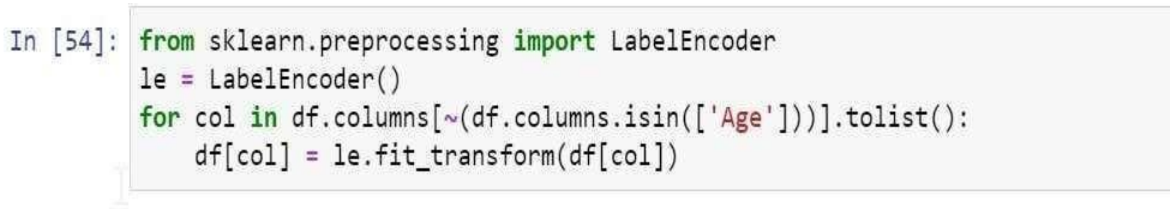
```
In [7]: df.isnull().sum()

Out[7]: Age                   0
        Gender                0
        Polyuria              0
        Polydipsia            0
        sudden weight loss    0
        weakness              0
        Polyphagia            0
        Genital thrush        0
        visual blurring       0
        Itching               0
        Irritability          0
        delayed healing       0
        partial paresis       0
        muscle stiffness      0
        Alopecia              0
        Obesity               0
        class                 0
        dtype: int64
```

4.3 DATA PREPROCESSING

Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model. When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So for this, we use data preprocessing task. Therefore the dataset we are using is a clean dataset because there are no null values, missing values (or) any other duplicate values.

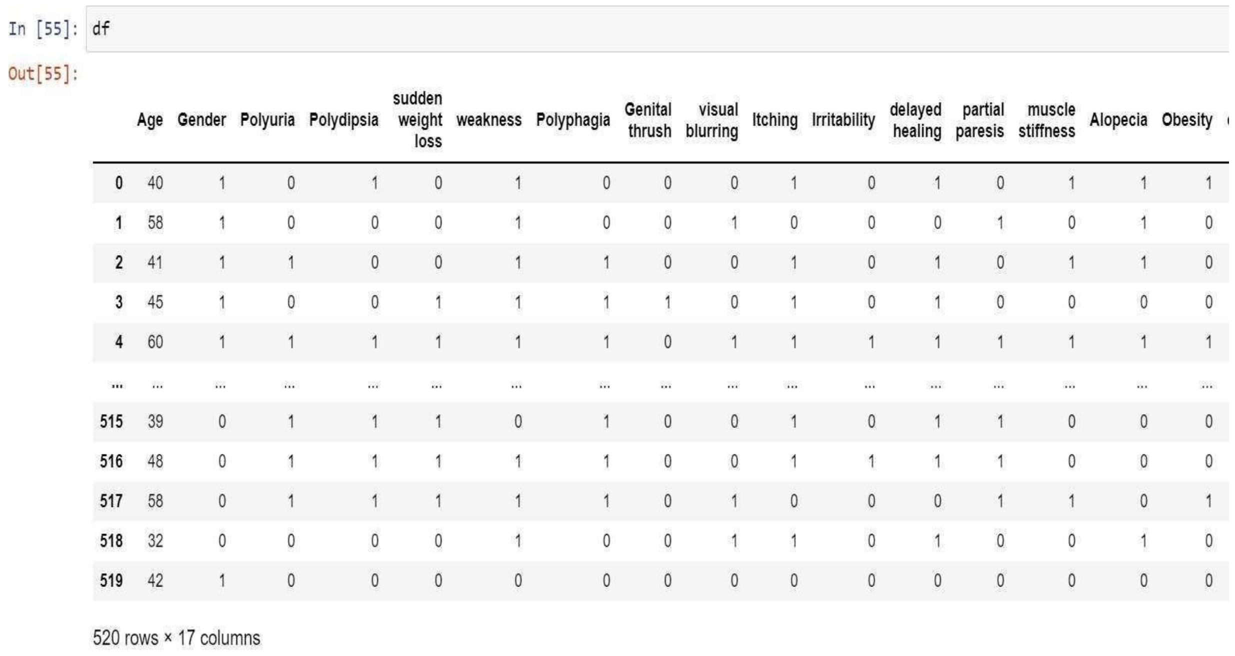
In machine learning, we usually deal with datasets that contain multiple labels in one or more than one column. These labels can be in the form of words or numbers. To make the data understandable or in human-readable form, the training data is often labeled in words. For this there are two methods to do. They are Labelencoder and One Hot Encoder. We are using Label Encoder in this project.

- **LABEL ENCODER**- Label Encoding refers to converting the labels into a numeric form so as to convert them into the machine-readable form. Machine learning algorithms can then decide in a better way how those labels must be operated. It is an important pre-processing step for the structured dataset in supervised learning. After applying this the attributes has changed to either 0 or 1.The code for the label encoder is



```
In [54]: from sklearn.preprocessing import LabelEncoder
         le = LabelEncoder()
         for col in df.columns[~(df.columns.isin(['Age']))].tolist():
             df[col] = le.fit_transform(df[col])
```

We will import label Encoder from sklearn library .After applying this code the dataset becomes as mentioned below.

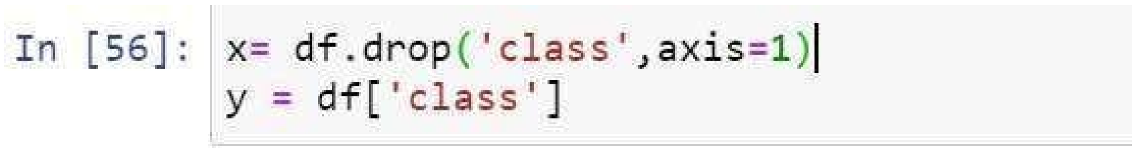


```
In [55]: df
Out[55]:
```

	Age	Gender	Polyuria	Polydipsia	sudden weight loss	weakness	Polyphagia	Genital thrush	visual blurring	Itching	Irritability	delayed healing	partial paresis	muscle stiffness	Alopecia	Obesity
0	40	1	0	1	0	1	0	0	0	1	0	1	0	1	1	1
1	58	1	0	0	0	1	0	0	1	0	0	0	1	0	1	0
2	41	1	1	0	0	1	1	0	0	1	0	1	0	1	1	0
3	45	1	0	0	1	1	1	1	0	1	0	1	0	0	0	0
4	60	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
515	39	0	1	1	1	0	1	0	0	1	0	1	1	0	0	0
516	48	0	1	1	1	1	1	0	0	1	1	1	1	0	0	0
517	58	0	1	1	1	1	1	0	1	0	0	0	1	1	0	1
518	32	0	0	0	0	1	0	0	1	1	0	1	0	0	1	0
519	42	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

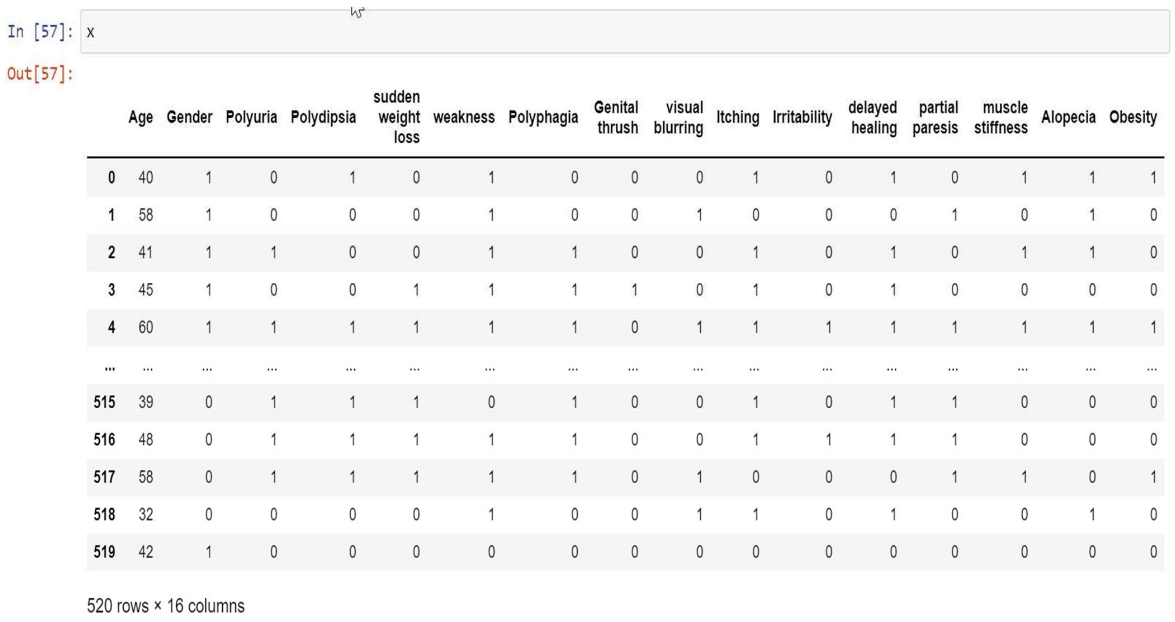
520 rows × 17 columns

For training the model first we have to separate our dataset into features and labels(prediction attribute).To do this we have to drop the label attribute from the dataset and store it in a variable and remaining all attributes in another attribute. Below mentioned code describes about it.



```
In [56]: x= df.drop('class',axis=1)
         y = df['class']
```

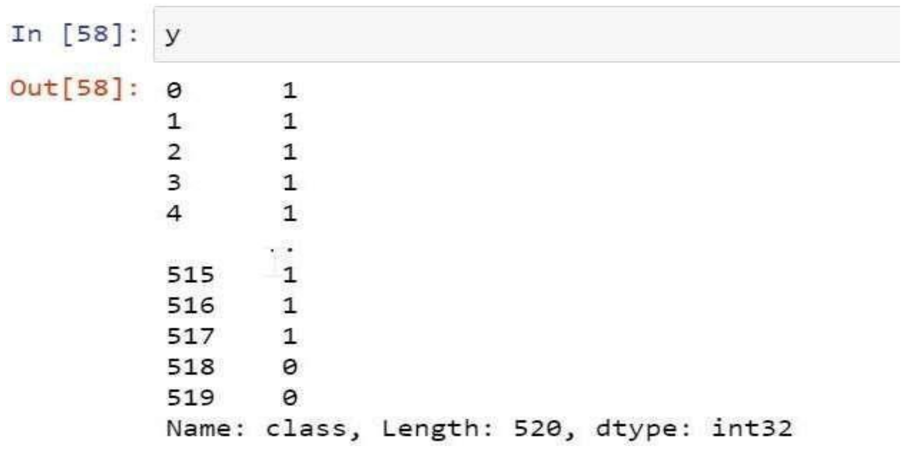
X variables looks like



```
In [57]: x
Out[57]:
```

	Age	Gender	Polyuria	Polydipsia	sudden weight loss	weakness	Polyphagia	Genital thrush	visual blurring	Itching	Irritability	delayed healing	partial paresis	muscle stiffness	Alopecia	Obesity
0	40	1	0	1	0	1	0	0	0	1	0	1	0	1	1	1
1	58	1	0	0	0	1	0	0	1	0	0	0	1	0	1	0
2	41	1	1	0	0	1	1	0	0	1	0	1	0	1	1	0
3	45	1	0	0	1	1	1	1	0	1	0	1	0	0	0	0
4	60	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
515	39	0	1	1	1	0	1	0	0	1	0	1	1	0	0	0
516	48	0	1	1	1	1	1	0	0	1	1	1	1	0	0	0
517	58	0	1	1	1	1	1	0	1	0	0	0	1	1	0	1
518	32	0	0	0	0	1	0	0	1	1	0	1	0	0	1	0
519	42	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

520 rows × 16 columns



```
In [58]: y

Out[58]: 0      1
         1      1
         2      1
         3      1
         4      1
               ..
         515    1
         516    1
         517    1
         518    0
         519    0
         Name: class, Length: 520, dtype: int32
```

Fig .4.3. Images of x& y attributes.

4.4 DATA VISUALIZATION

Data visualization is an important skill in applied statistics and machine learning. Statistics does indeed focus on quantitative descriptions and estimations of data. Data visualization provides an important suite of tools for gaining a qualitative understanding. This can be helpful when exploring and getting to know a dataset and can help with identifying patterns, corrupt data, outliers, and much more With the help of data visualization, we can see how the data looks like and what kind of correlation is held by the attributes of data. It is the fastest way to see if the features correspond to the output

There are five key plots that you need to know well for basic data visualization. They are:

- Bar Chart
- Count plot
- Histogram Plot
- Box and Whisker Plot
- Scatter Plot

Here we have shown two visualizations about the dataset. first one is count plot for using count plot we should import it from seaborn. Seaborn is an amazing visualization library for statistical graphics plotting in Python. It provides beautiful default styles and color palettes to make statistical plots more attractive. It is built on the top of matplotlib library and also closely integrated to the data structures from pandas.
Seaborn.countplot() method is used to Show the counts of observations in each categorical bin using bars. Here we have done count plot for all the attributes of the dataset.

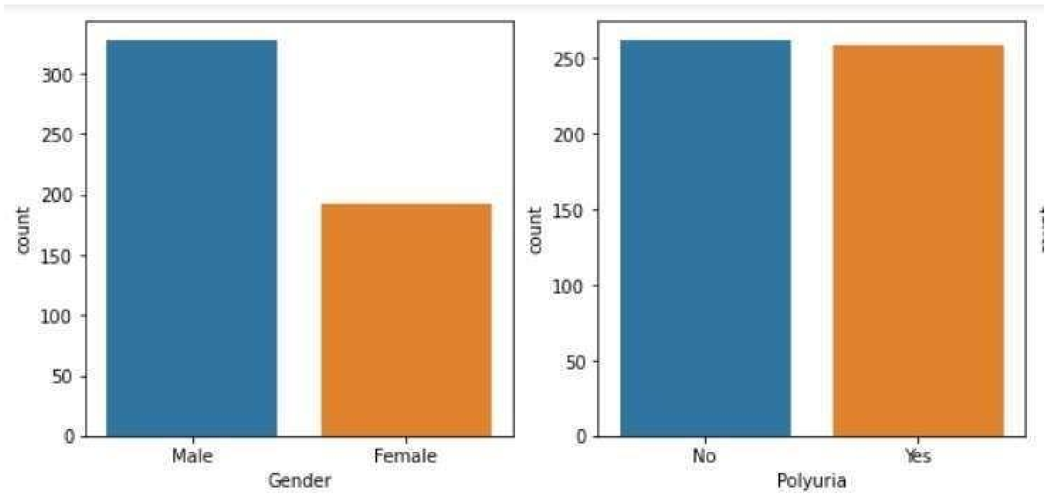


Fig. 4.4.1.a. Count Plot of All Attributes

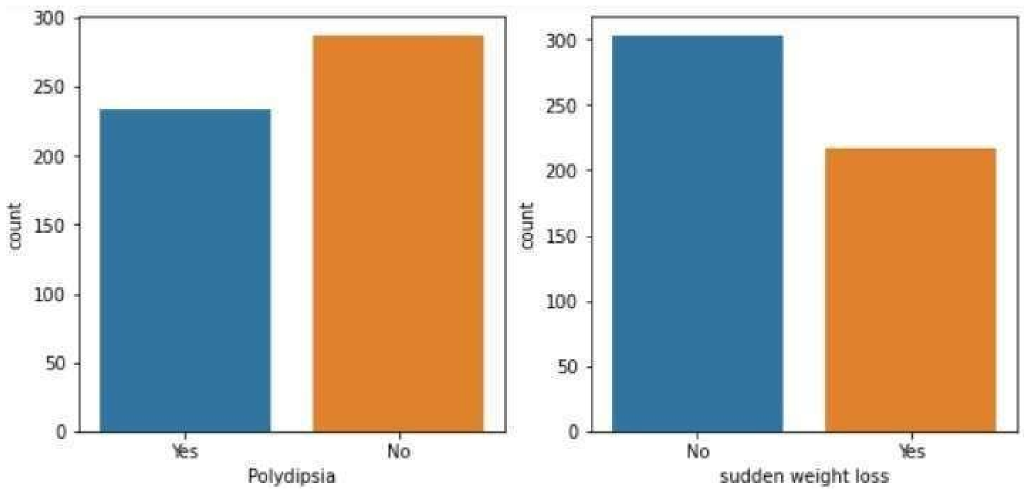


Fig. 4.4.1.b. Count Plot of All Attributes

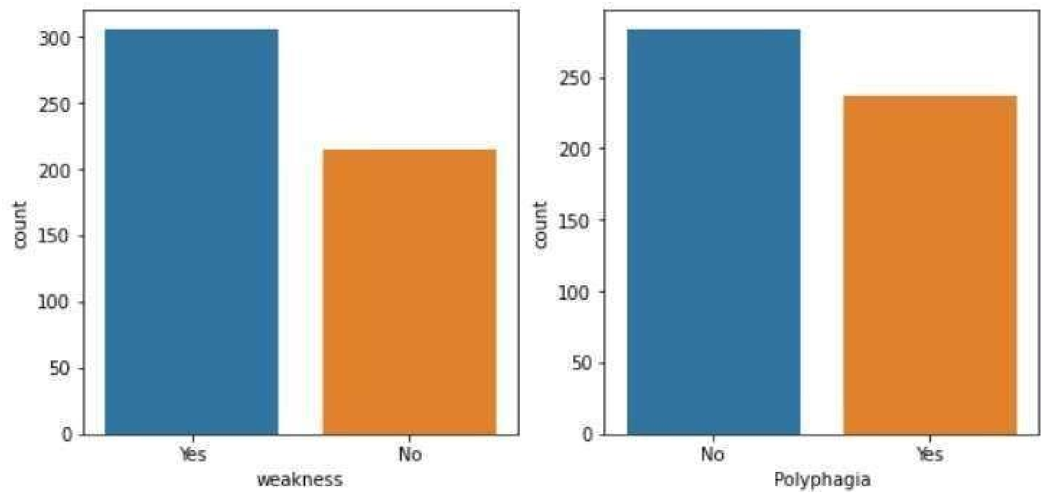


Fig. 4.4.1.c. Count Plot of All Attributes

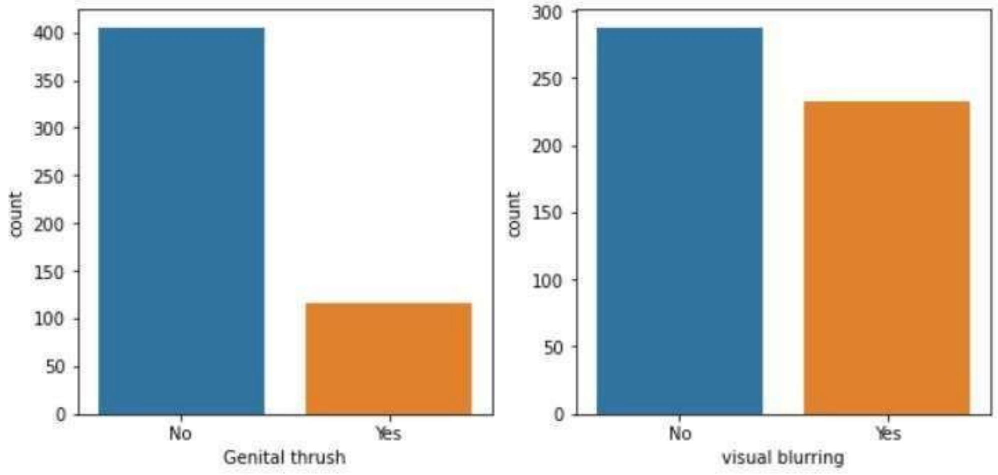


Fig .4.4.1.d. Count Plot of All Attributes

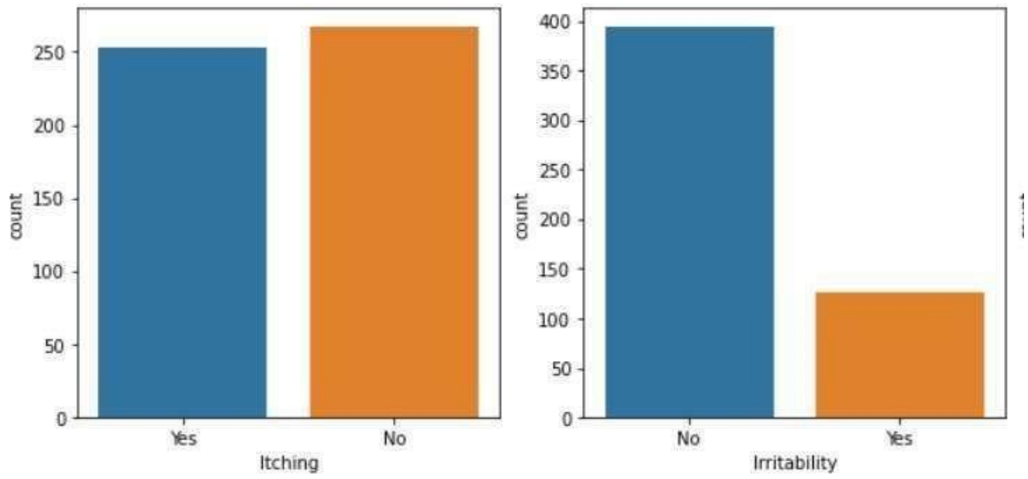


Fig. 4.4.1.e.Count Plot of All Attributes

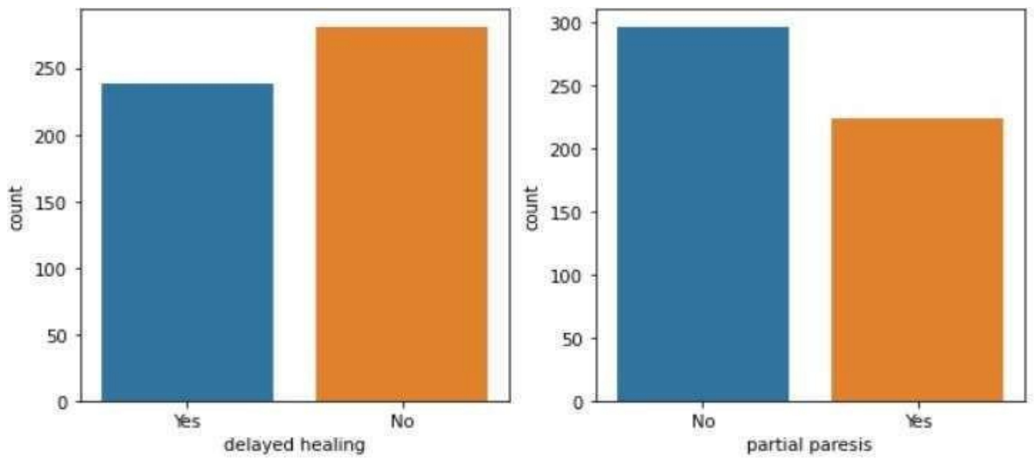


Fig. 4.4.1.f. Count Plot of All Attributes

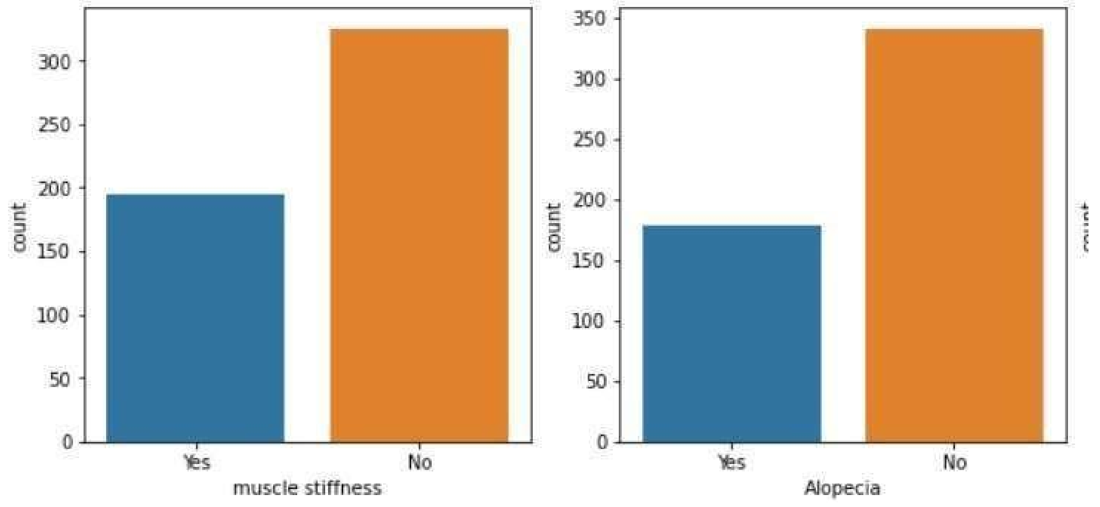


Fig. 4.4.1.g. Count Plot of All Attributes

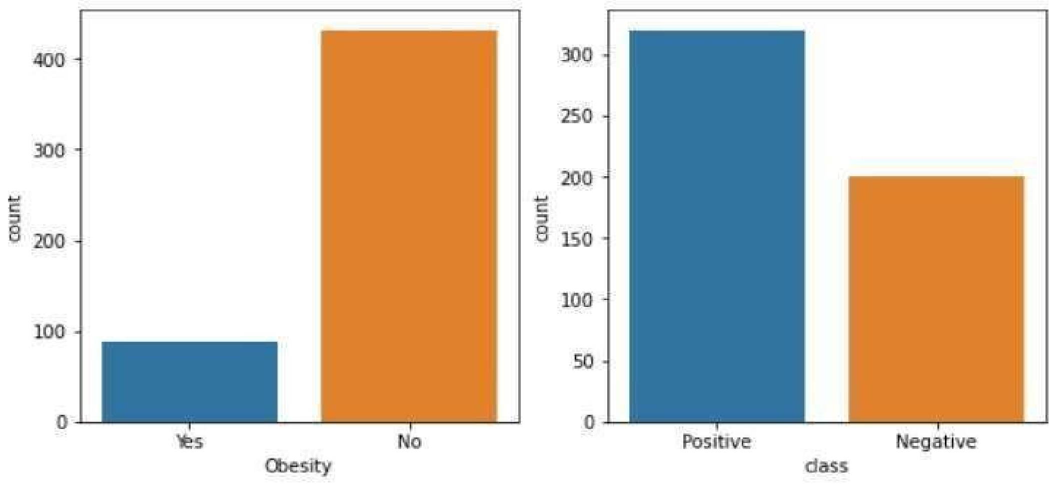


Fig. 4.4.1.h. Count Plot of All Attributes

4.5 MODEL

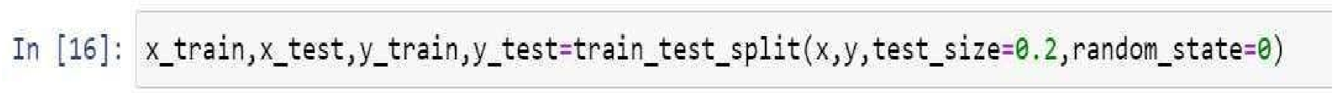
4.5.1 TRAIN TEST SPLIT – The train-test split is a technique for evaluating the performance of a machine learning algorithm. It can be used for classification or regression problems and can be used for any supervised learning algorithm. The procedure involves taking a dataset and dividing it into two subsets. The first subset is used to fit the model and is referred to as the training dataset. The second subset is not used to train the model; instead, the input element of the dataset is provided to the model, then predictions are made and compared to the expected values. This second dataset is referred to as the test dataset.

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

The objective is to estimate the performance of the machine learning model on new data (data not used to train the model). This is how we expect to use the model in practice. Namely, to fit it on available data with known inputs and outputs, then make predictions on new examples in the future where we do not have the expected output or target values. Common split-up percentage includes :

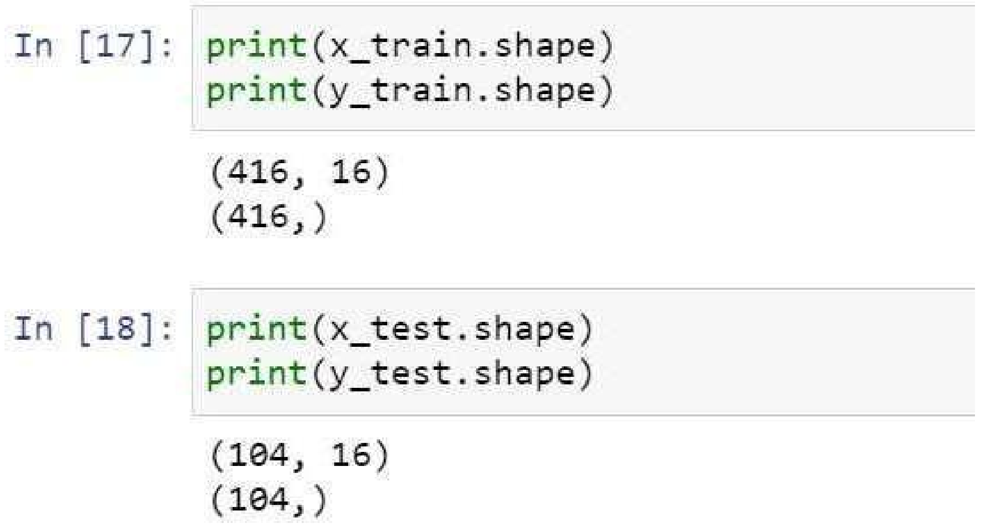
- Train: 70% and Test: 30%
- Train: 80% and Test: 20%
- Train: 90% and Test: 10%

Here we used Train: 80% and Test: 20% split-up.



```
In [16]: x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

The below mentioned figure tells us about the shape of the train and test model.
Shape indicates both instances and attributes.

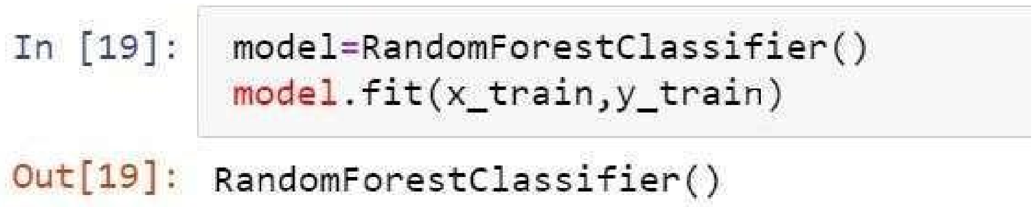


```
In [17]: print(x_train.shape)
         print(y_train.shape)

(416, 16)
(416,)
```

```
In [18]: print(x_test.shape)
         print(y_test.shape)

(104, 16)
(104,)
```

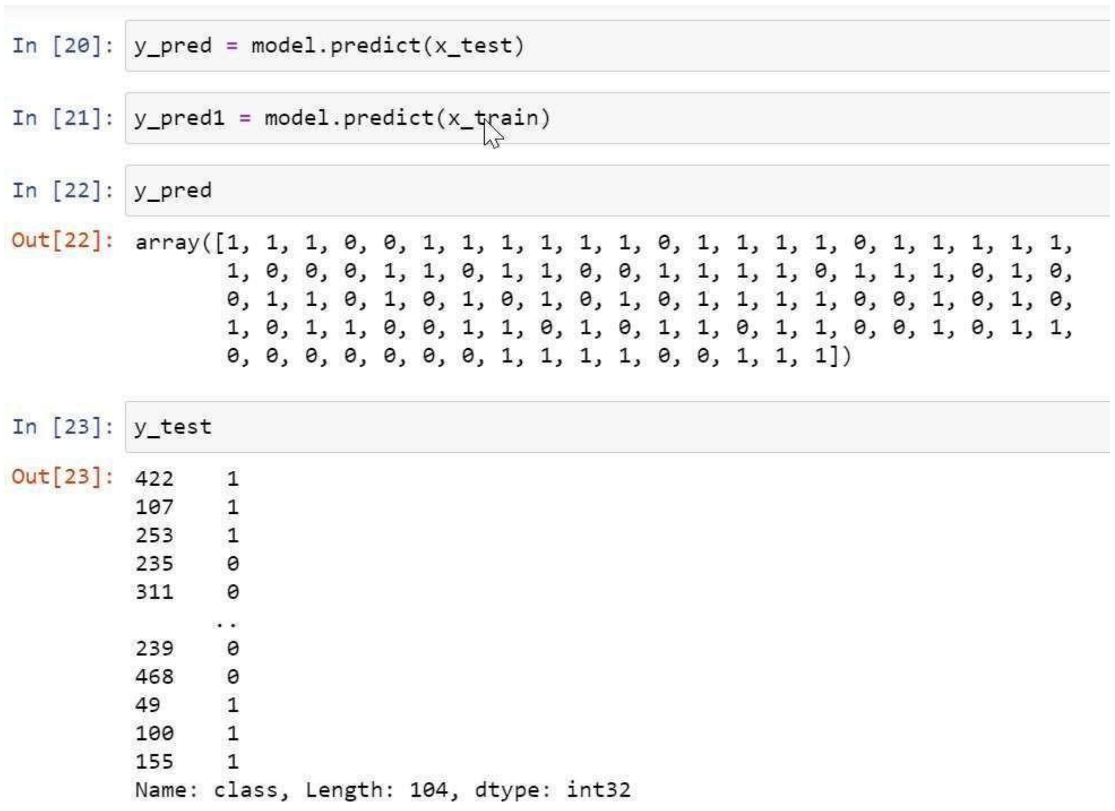
4.5.2 MODEL SELECTION – Here we are using RandomForestClassifer model because our dataset comes under classical problem because our predicting attribute has a categorical value. From Sklearn.ensemble we will import RandomForestClassifier. A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting.

Model.fit(x_train,y_train)-It is an inbuilt method which helps for training the model, already we have divided the dataset into train (80%) and test(20%).This command takes 80% values and train the model while training it analyses various patterns that the dataset contains after that we will predict from the remaining 20% values.

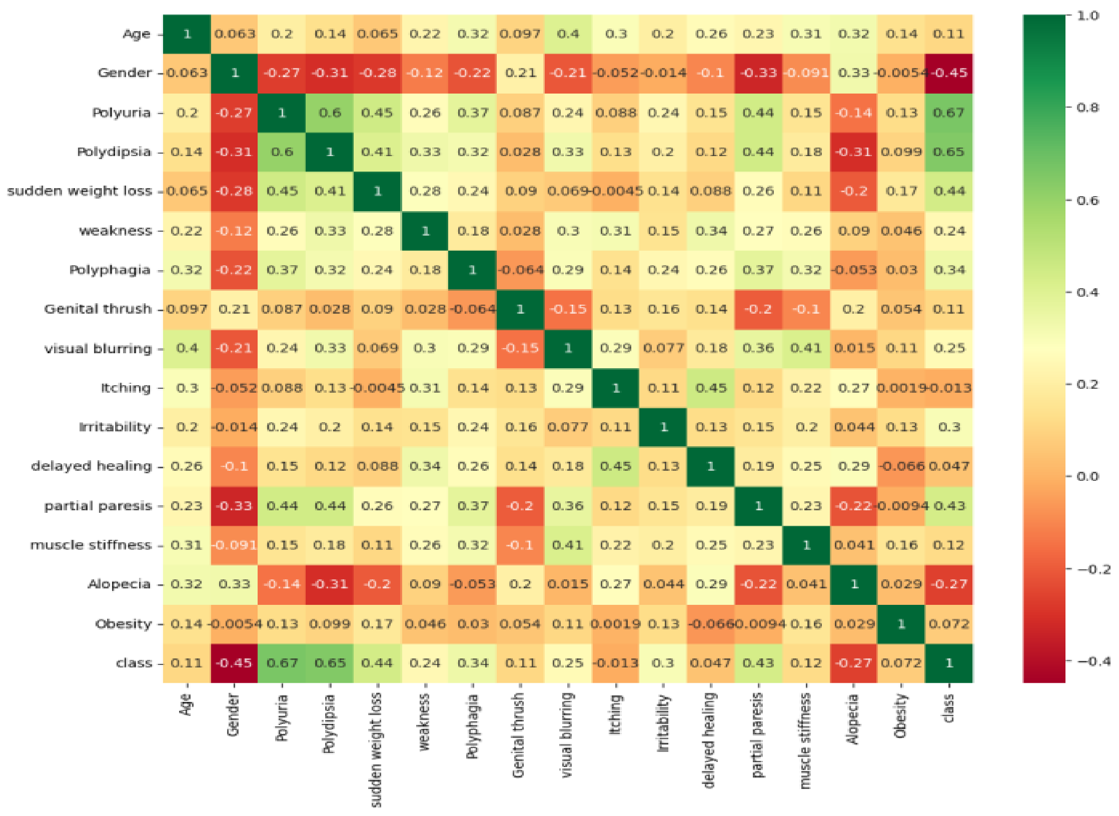


```
In [19]: model=RandomForestClassifier()
         model.fit(x_train,y_train)

Out[19]: RandomForestClassifier()
```

The next figures describes about how I will make predictions from the test model and how the values look like .



```
In [20]: y_pred = model.predict(x_test)

In [21]: y_pred1 = model.predict(x_train)

In [22]: y_pred

Out[22]: array([1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
                1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0,
                0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0,
                1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1,
                0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1])

In [23]: y_test

Out[23]: 422    1
         107    1
         253    1
         235    0
         311    0
               ..
         239    0
         468    0
         49     1
         100    1
         155    1
         Name: class, Length: 104, dtype: int32
```



	Age	Gender	Polyuria	Polydipsia	sudden weight loss	weakness	Polyphagia	Genital thrush	visual blurring	Itching	Irritability	delayed healing	partial paresis	muscle stiffness	Alopecia	Obesity	class
Age	1	0.063	0.2	0.14	0.065	0.22	0.32	0.097	0.4	0.3	0.2	0.26	0.23	0.31	0.32	0.14	0.11
Gender	0.063	1	-0.27	-0.31	-0.28	-0.12	-0.22	0.21	-0.21	-0.052	-0.014	-0.1	-0.33	-0.091	0.33	-0.0054	-0.45
Polyuria	0.2	-0.27	1	0.6	0.45	0.26	0.37	0.087	0.24	0.088	0.24	0.15	0.44	0.15	-0.14	0.13	0.67
Polydipsia	0.14	-0.31	0.6	1	0.41	0.33	0.32	0.028	0.33	0.13	0.2	0.12	0.44	0.18	-0.31	0.099	0.65
sudden weight loss	0.065	-0.28	0.45	0.41	1	0.28	0.24	0.09	0.069	-0.0045	0.14	0.088	0.26	0.11	-0.2	0.17	0.44
weakness	0.22	-0.12	0.26	0.33	0.28	1	0.18	0.028	0.3	0.31	0.15	0.34	0.27	0.26	0.09	0.046	0.24
Polyphagia	0.32	-0.22	0.37	0.32	0.24	0.18	1	-0.064	0.29	0.14	0.24	0.26	0.37	0.32	-0.053	0.03	0.34
Genital thrush	0.097	0.21	0.087	0.028	0.09	0.028	-0.064	1	-0.15	0.13	0.16	0.14	-0.2	-0.1	0.2	0.054	0.11
visual blurring	0.4	-0.21	0.24	0.33	0.069	0.3	0.29	-0.15	1	0.29	0.077	0.18	0.36	0.41	0.015	0.11	0.25
Itching	0.3	-0.052	0.088	0.13	-0.0045	0.31	0.14	0.13	0.29	1	0.11	0.45	0.12	0.22	0.27	0.0019	-0.013
Irritability	0.2	-0.014	0.24	0.2	0.14	0.15	0.24	0.16	0.077	0.11	1	0.13	0.15	0.2	0.044	0.13	0.3
delayed healing	0.26	-0.1	0.15	0.12	0.088	0.34	0.26	0.14	0.18	0.45	0.13	1	0.19	0.25	0.29	-0.066	0.047
partial paresis	0.23	-0.33	0.44	0.44	0.26	0.27	0.37	-0.2	0.36	0.12	0.15	0.19	1	0.23	-0.22	-0.0094	0.43
muscle stiffness	0.31	-0.091	0.15	0.18	0.11	0.26	0.32	-0.1	0.41	0.22	0.2	0.25	0.23	1	0.041	0.16	0.12
Alopecia	0.32	0.33	-0.14	-0.31	-0.2	0.09	-0.053	0.2	0.015	0.27	0.044	0.29	-0.22	0.041	1	0.029	-0.27
Obesity	0.14	-0.0054	0.13	0.099	0.17	0.046	0.03	0.054	0.11	0.0019	0.13	-0.066	-0.0094	0.16	0.029	1	0.072
class	0.11	-0.45	0.67	0.65	0.44	0.24	0.34	0.11	0.25	-0.013	0.3	0.047	0.43	0.12	-0.27	0.072	1

CHAPTER 5
RESULTS AND DISCUSSIONS

5.1 CLASSIFICATION METRICS

The model Evaluation is done based on the scores we receive from the various metrices , after validating them we may come to a conclusion that our model is good or bad and our model is underfitting or overfitting. Here we performed Random Forest Classifier algorithm the performance of this algorithm can be validate after seeing the scores of some of the classification metrices.
Some of the classification metrices are

ACCURACY- Accuracy is one of the metric for evaluating classification models.

Informally, **accuracy** is the fraction of predictions our model got right. Formally, accuracy has the following definition:

Accuracy=Number of correct predictions/Total number of predictions

Our model got an accuracy score equal to 99% from this we can say that our model is pretty much good.

CONFUSION MATRIX - A Confusion matrix is an N x N matrix used for

evaluating the performance of a classification model, where N is the number of target classes. The matrix compares the actual target values with those predicted by the machine learning model. This gives us a holistic view of how well our classification model is performing and what kinds of errors it is making. For a binary classification problem, we would have a 2 x 2 matrix as shown below in

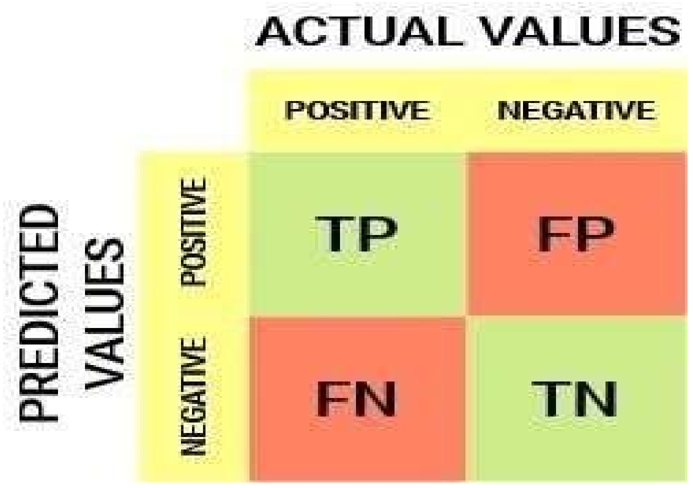


Fig .5.1. Confusion matrix Image

Let's decipher the matrix:

The target variable has two values: **Positive** or **Negative**

The **columns** represent the **actual values** of the target variable

The **rows** represent the **predicted values** of the target variable

PRECISION - Precision tells us how many of the correctly predicted cases actually turned out to be positive.

Here's how to calculate Precision:

Precision = TP/TP+FP .

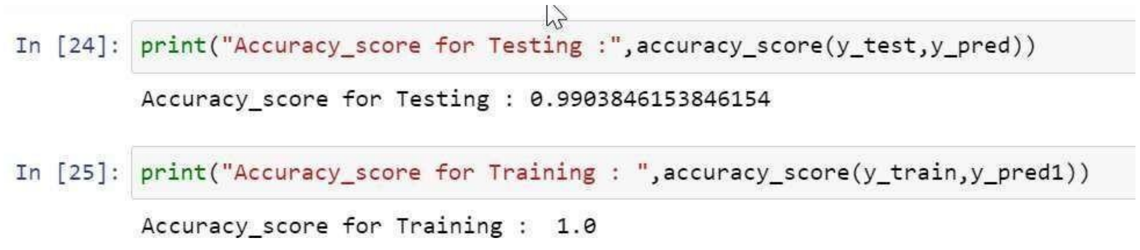
RECALL- Recall tells us how many of the actual positive cases we were able to predict correctly with our model.

And here's how we can calculate Recall:

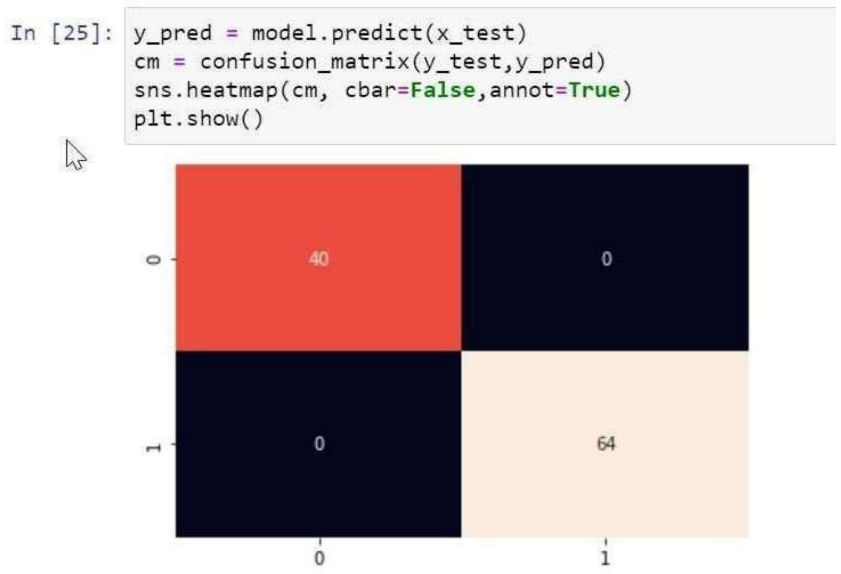
Recall = TP/TP +FN .

F1 SCORE - The F1 Score is the 2*((precision*recall)/(precision+recall)). It is also called the F Score or the F Measure. Put another way, the F1 score conveys the balance between the precision and the recall.

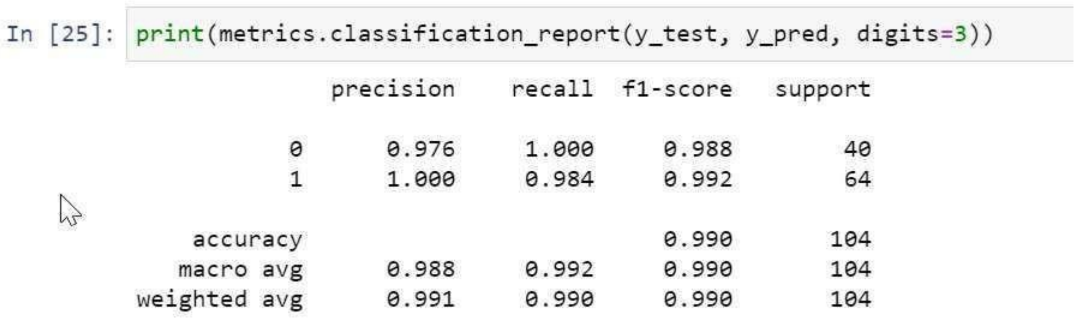
Therefore, we can say that our model is a good model because we got good scores in all the metrices.



```
In [24]: print("Accuracy_score for Testing :",accuracy_score(y_test,y_pred))

         Accuracy_score for Testing : 0.9903846153846154

In [25]: print("Accuracy_score for Training : ",accuracy_score(y_train,y_pred1))

         Accuracy_score for Training :  1.0
```



```
In [25]: y_pred = model.predict(x_test)
         cm = confusion_matrix(y_test,y_pred)
         sns.heatmap(cm, cbar=False,annot=True)
         plt.show()
```



```
In [25]: print(metrics.classification_report(y_test, y_pred, digits=3))

                       precision    recall  f1-score   support

                  0        0.976     1.000     0.988        40
                  1        1.000     0.984     0.992        64

           accuracy                            0.990       104
          macro avg        0.988     0.992     0.990       104
       weighted avg        0.991     0.990     0.990       104
```

Fig .5.2 Result images.

```
[ ]: #Building the model using DecisionTree

     from sklearn.tree import DecisionTreeClassifier

     dtree = DecisionTreeClassifier()
     dtree.fit(x_train, y_train)
```

```
[ ]: DecisionTreeClassifier()
```

```
[ ]: predictions = dtree.predict(x_test)
```

```
[ ]: #Getting the accuracy score for Decision Tree

     from sklearn import metrics

     print("Accuracy Score =", format(metrics.accuracy_score(y_test,predictions)))
```

```
Accuracy Score = 0.9711538461538461
```

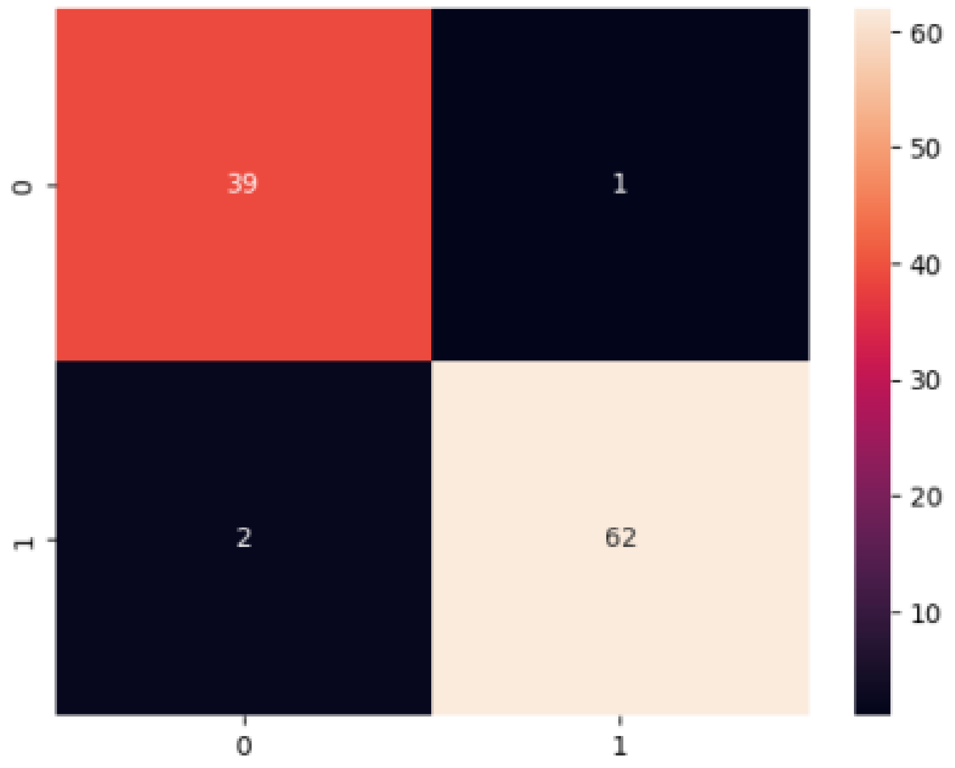
```
[ ]: from sklearn.metrics import classification_report, confusion_matrix

     print(confusion_matrix(y_test, predictions))
```

```
print(classification_report(y_test,predictions))
```

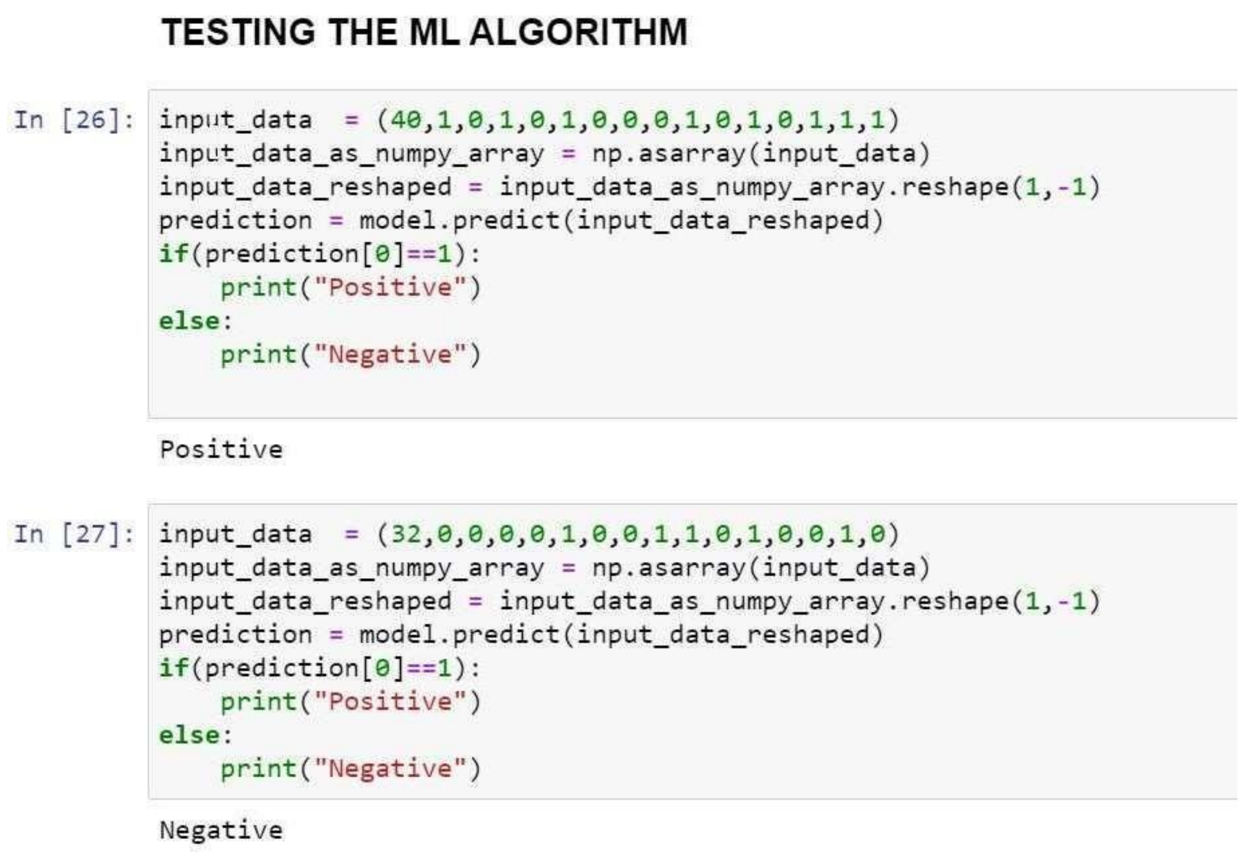
```
[[39  1]
 [ 2 62]]
              precision    recall  f1-score   support

           0       0.95      0.97      0.96        40
           1       0.98      0.97      0.98        64

    accuracy                           0.97       104
   macro avg       0.97      0.97      0.97       104
weighted avg       0.97      0.97      0.97       104
```

```
[ ]: # Heatmap of Confusion matrix
     cm=confusion_matrix(y_test, predictions)
     sns.heatmap(pd.DataFrame(cm), annot=True)
     plt.show()
```

```
[ ]: # Heatmap of Confusion matrix
     cm=confusion_matrix(y_test, predictions)
     sns.heatmap(pd.DataFrame(cm), annot=True)
     plt.show()
```



```
[ ]: #Plotting feature importances
     (pd.Series(rfc.feature_importances_, index=x.columns)
        .plot(kind='barh'))
```



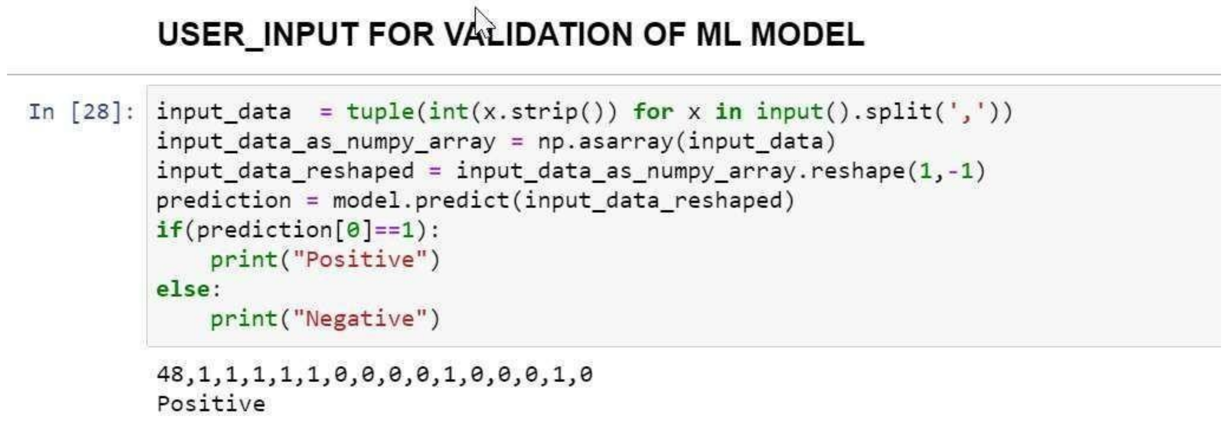
TESTING THE ML ALGORITHM

```
In [26]: input_data  = (40,1,0,1,0,1,0,0,0,1,0,1,0,1,1,1)
         input_data_as_numpy_array = np.asarray(input_data)
         input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)
         prediction = model.predict(input_data_reshaped)
         if(prediction[0]==1):
             print("Positive")
         else:
             print("Negative")
```

```
Positive
```

```
In [27]: input_data  = (32,0,0,0,0,1,0,0,1,1,0,1,0,0,1,0)
         input_data_as_numpy_array = np.asarray(input_data)
         input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)
         prediction = model.predict(input_data_reshaped)
         if(prediction[0]==1):
             print("Positive")
         else:
             print("Negative")
```

```
Negative
```



USER_INPUT FOR VALIDATION OF ML MODEL

```
In [28]: input_data  = tuple(int(x.strip()) for x in input().split(','))
         input_data_as_numpy_array = np.asarray(input_data)
         input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)
         prediction = model.predict(input_data_reshaped)
         if(prediction[0]==1):
             print("Positive")
         else:
             print("Negative")
```

```
48,1,1,1,1,1,0,0,0,0,1,0,0,0,1,0
Positive
```

CHAPTER 6
CONCLUSION AND FUTURE WORK

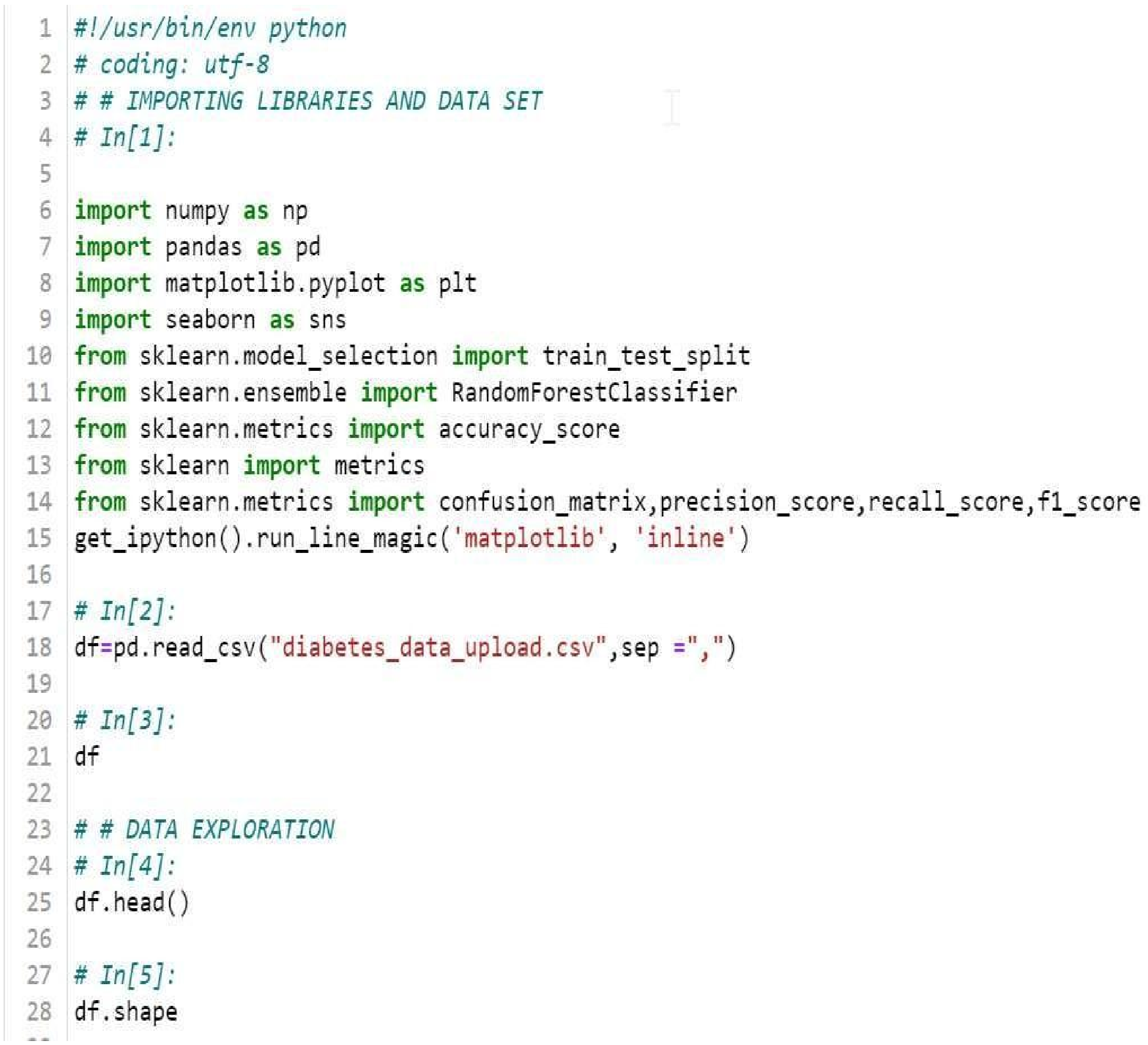
Machine learning approaches work well for diagnosis of diabetes. It performs very well with medical datasets. Prediction of diabetes at early stage helps the patient in order to provide appropriate treatment. I had taken a wide range of research about different ML Algorithms and finally choose random forest classifier approach to solve this problem. In the process of research, I came to know about the various ML techniques and its different algorithms which helps to solve real-world problems. In this project I had implement random forest classifier approach and I observed that it given a good accuracy and performed well. From this project I understood what we can achieve by using machine learning Concepts and algorithms. I gained a lot of knowledge about all the concepts and how to solve and which algorithm to apply when a problem is given. Further in future we can extend this work by adding few more attributes and instances with the development of technology with the new inventions of algorithm we can explore them and get comparatively good scores in all the metrices and it also strengthen our model therefore predictions become quite easy which helps to solve our problems.

REFERENCES

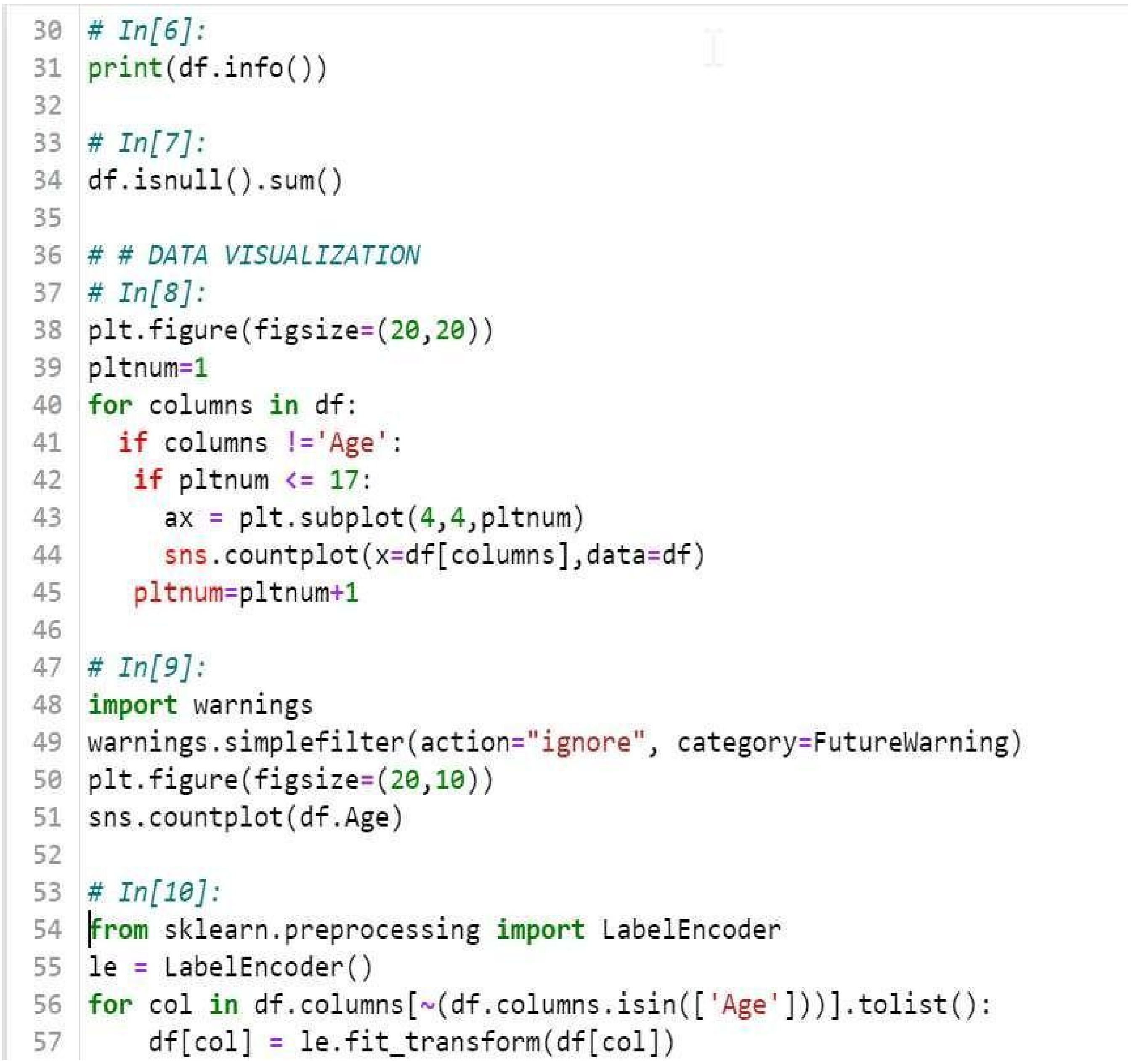
1. https://archive.ics.uci.edu/ml/datasets/Early+stage+diabetes+risk+prediction+dat aset.
2. https://www.healthline.com/health/diabetes.
3. https://www.python.org/doc/
4. https://scikit-learn.org/stable/.
5. https://towardsdatascience.com/understanding-random-forest-58381e0602d2.
6. https://yourdataguy.org/machine-learning-for-diabetes-prediction-in-python/.
7. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6232260/
8. https://cppsecrets.com/users/25231009710711510497105115991111111086410 3109971051084699111109/Diabetes-Prediction-Using-Machine-Learning.php
9. Scikit-learn: Machine Learning in Python,Pedregosa *et al.*, JMLR 12, pp. 2825-2830,2011
10. https://scikitlearn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifi er.html.
11. Matplotlib.pyplot. matplotlib.pyplot - Matplotlib 3.4.3 documentation. (n.d.).Retrived November 2,2021, from https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.html

APPENDIX

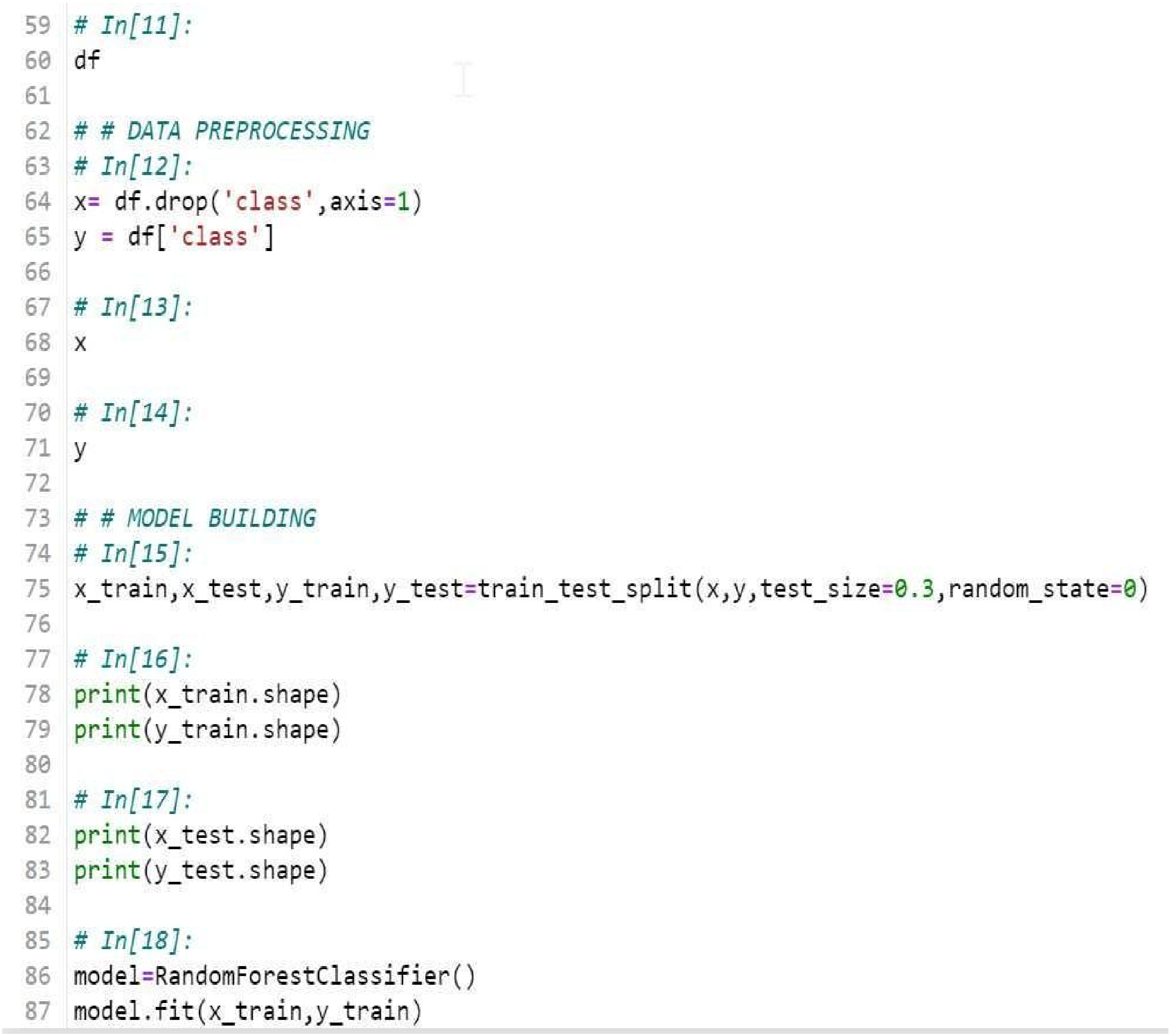
SOURCE CODE :



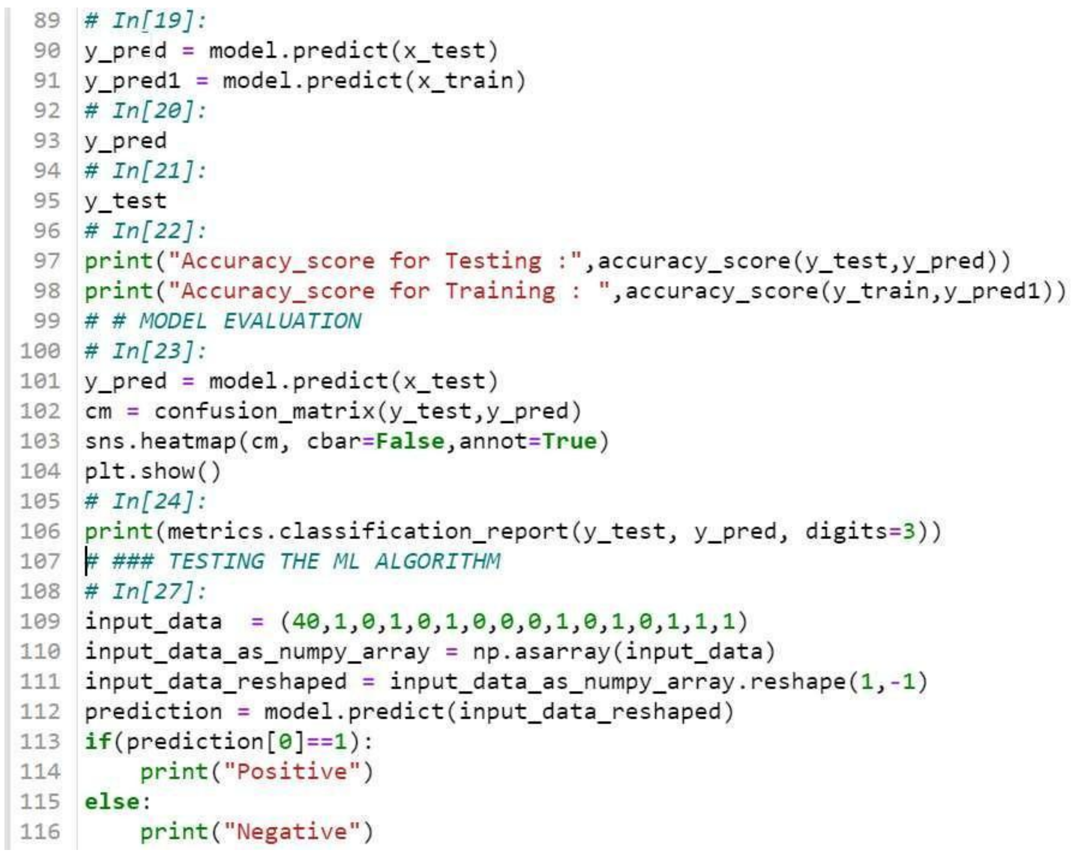
```
#!/usr/bin/env python
# coding: utf-8
# # IMPORTING LIBRARIES AND DATA SET
# In[1]:

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn import metrics
from sklearn.metrics import confusion_matrix,precision_score,recall_score,f1_score
get_ipython().run_line_magic('matplotlib', 'inline')

# In[2]:
df=pd.read_csv("diabetes_data_upload.csv",sep =",")

# In[3]:
df

# # DATA EXPLORATION
# In[4]:
df.head()

# In[5]:
df.shape
```



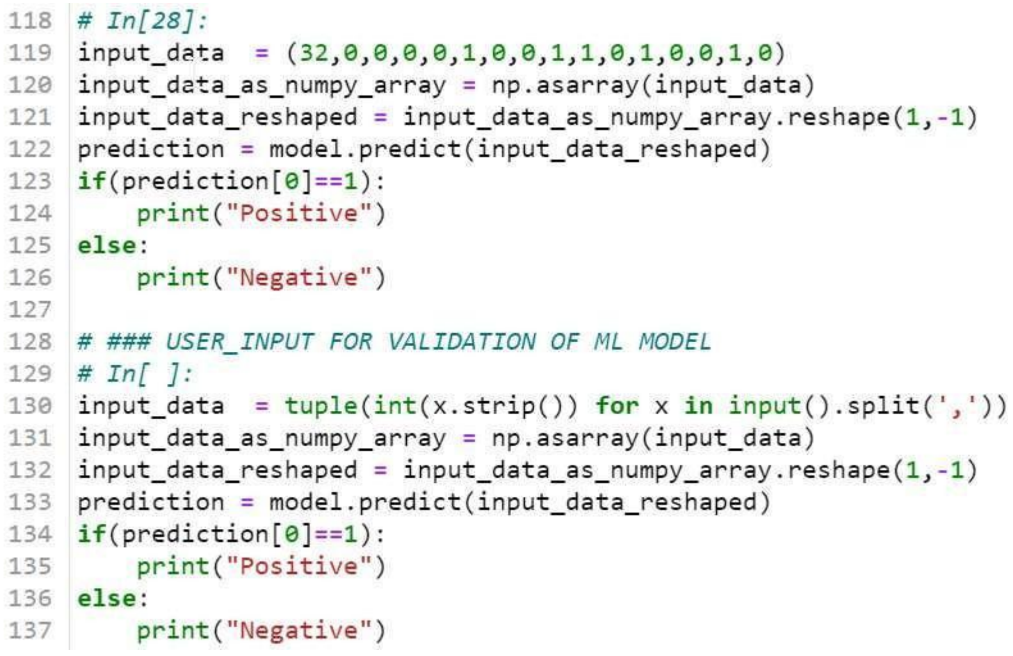
```
# In[6]:
print(df.info())

# In[7]:
df.isnull().sum()

# # DATA VISUALIZATION
# In[8]:
plt.figure(figsize=(20,20))
pltnum=1
for columns in df:
  if columns !='Age':
   if pltnum <= 17:
     ax = plt.subplot(4,4,pltnum)
     sns.countplot(x=df[columns],data=df)
   pltnum=pltnum+1

# In[9]:
import warnings
warnings.simplefilter(action="ignore", category=FutureWarning)
plt.figure(figsize=(20,10))
sns.countplot(df.Age)

# In[10]:
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
for col in df.columns[~(df.columns.isin(['Age']))].tolist():
    df[col] = le.fit_transform(df[col])
```



```python
# In[11]:
df

# # DATA PREPROCESSING
# In[12]:
x= df.drop('class',axis=1)
y = df['class']

# In[13]:
x

# In[14]:
y

# # MODEL BUILDING
# In[15]:
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=0)

# In[16]:
print(x_train.shape)
print(y_train.shape)

# In[17]:
print(x_test.shape)
print(y_test.shape)

# In[18]:
model=RandomForestClassifier()
model.fit(x_train,y_train)
```



```python
# In[19]:
y_pred = model.predict(x_test)
y_pred1 = model.predict(x_train)
# In[20]:
y_pred
# In[21]:
y_test
# In[22]:
print("Accuracy_score for Testing :",accuracy_score(y_test,y_pred))
print("Accuracy_score for Training : ",accuracy_score(y_train,y_pred1))
# # MODEL EVALUATION
# In[23]:
y_pred = model.predict(x_test)
cm = confusion_matrix(y_test,y_pred)
sns.heatmap(cm, cbar=False,annot=True)
plt.show()
# In[24]:
print(metrics.classification_report(y_test, y_pred, digits=3))
# ### TESTING THE ML ALGORITHM
# In[27]:
input_data  = (40,1,0,1,0,1,0,0,0,1,0,1,0,1,1,1)
input_data_as_numpy_array = np.asarray(input_data)
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)
prediction = model.predict(input_data_reshaped)
if(prediction[0]==1):
    print("Positive")
else:
    print("Negative")
```



```python
# In[28]:
input_data  = (32,0,0,0,0,1,0,0,1,1,0,1,0,0,1,0)
input_data_as_numpy_array = np.asarray(input_data)
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)
prediction = model.predict(input_data_reshaped)
if(prediction[0]==1):
    print("Positive")
else:
    print("Negative")

# ### USER_INPUT FOR VALIDATION OF ML MODEL
# In[ ]:
input_data  = tuple(int(x.strip()) for x in input().split(','))
input_data_as_numpy_array = np.asarray(input_data)
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)
prediction = model.predict(input_data_reshaped)
if(prediction[0]==1):
    print("Positive")
else:
    print("Negative")
```