## ACKNOWLEDGEMENT

We express our sincere gratitude to Mr. Abhisek Saha, Assistant Professor, Department of Computer Science and Engineering, our project guide and Mr.Avijit Bose, Assistant Professor and Head of Department (CSE) for providing us their guidance and cooperation for the project. We also extend our sincere thanks to all other faculty members of Computer Science & Engineering Department and our friends for their support and encouragement. We will be failing in duty if we do not acknowledge with grateful thanks to the authors of the references and other literatures referred to in this project. Last but never the least we are very much thankful to our parents who guided and supported us in every step which we took.

# CONTENTS

## ABSTRACT

Diabetes is a chronic disease with the potential to cause a worldwide health care crisis. Diabetes mellitus or simply diabetes is a disease caused due to the increase level of blood glucose. Age, obesity, lack of exercise, hereditary diabetes, living style, bad diet, high blood pressure, etc. can cause Diabetes Mellitus (simple diabetes). Diabetes is a major metabolic disorder which can affect entire body system adversely. In addition, undiagnosed diabetes can increase the risk of cardiac stroke, diabetic nephropathy and other disorders. Therefore, early detection of diabetes is very significant so that timely action can be taken and the progression of the disease may be prevented to avoid further complications. Machine Learning algorithm plays a significant role to early prediction of diabetic trends of the patients based on different healthcare parameters. Healthcare industries has large volume databases. Using various Machine Learning algorithms modern researchers study those huge datasets and find hidden information, hidden patterns to discover knowledge from the data and predict outcomes accordingly. In this project, we investigate the relative performance of various machine learning methods such as SVM, Decision Tree, Naïve Bayes, Logistic Regression, and Random Forests for predicting incident diabetes using medical records available in standard medical dataset, which are used to develop trends and detect patterns with risk factors. In addition, we apply different techniques to uncover potential predictors of diabetes.

# 1. INTRODUCTION

## 1.1 ORIGIN OF THE WORK

In today's world, most of the people are unaware of the serious situation that can be caused because of the scattered lifestyle, heavy consumption of junk foods and many more factors. Hats why diseases like diabetes, high blood-pressure, liver/kidney failure, heart-attack rates are increasing frequently. Diabetes is a prolonged disease that happens when the body cannot efficiently use the insulin it generates. As a result, the disease increase the risk of malfunction of different organs, especially the eyes, kidneys, nerves, heart, and blood vessels. According to the report of World Health Organization (WHO) diabetes will be the seventh prominent cause of death by 2030.About 642million adults (1 in 10 adults) are projected to have diabetes in 2040. The deaths of around 1.6 million people were completely affected by diabetes in 2015 and 2.2 million deaths due to high blood glucose in 2012. Diabetes Mellitus do not depend on the age, it can happen with people anytime. There are three types of diabetes: i) Juvenile or childhood diabetes (type 1 diabetes), ii) Type 2 or adult diabetes and iii) Gestational or type 3 diabetes. Gestational diabetes is hyperglycaemia which occurs because of the change in hormones during pregnancy. Generally, type 1 diabetes happens due to the lack of insulin production and it is diagnosed in people of young age. Type 2 is a very familiar form of diabetes, and it contains a huge volume of people from around the world. Type 2 mostly causes surplus body weight and physical disuse. Whatsoever, type 1 and type 2 diabetes cannot be cured properly. But, early diagnosis and simple lifestyle can prevent it. Moreover, there are different new cases of diabetes arises from the developing countries where shocking amounts of diabetes affected people are from Bangladesh which is projected to climb up to more than 16million by 2020.In last few decades, data has been elevated in a vast scale in diverse arenas including medical fields. Machine Learning is a discipline that aims to solve different important biomedical problems .The machine learning based classification techniques are the most operative methods for both real-life and scientific problems. The use of these classification based approaches in the diagnosis and cure of diseases can significantly decrease medical errors and human costs. As described in the study, machine learning based classification techniques have prospective performance in prediction accuracy as compared to other algorithms for data classification. Data classification accuracy may vary conditionally on different machine learning techniques. Many of the researchers have been focused on diabetes from various perspectives of their works where most of the study discussed the classification techniques for diabetes prediction and its accuracy. The motto of this proposed work more efficient results and reduce the cost of diagnosis in the Health Care Services. Therefore, the aim of this study is to evaluate the performance of different machine learning (supervised learning) classification techniques for the classification of diabetic affected or not. We've used six classification techniques Naïve-Bayes (NB), Logistics Regression (LR), Support Vector Machine (SVM), Decision Tree (DT), Random Forest (RF) , k-nearest neighbours algorithm (k-NN). We explore the performance result of different techniques where the performance is evaluated by various standards, such as accuracy, precision, true positive rate (TPR),true negative rate (TNR), F-score. Moreover, the most accurate classification technique is donated for diagnosis of such disease with proposed unified framework.

## 2. LITERATURE SURVEY

This section discusses about the existing techniques and algorithms used for the diagnosis of diabetes mellitus. Each and every algorithm used for the diagnosis of diabetes mellitus has their own limitations and advantages. This section presents an analytical study on the features of the existing techniques.

### A. Naive Bayes Classifier For Diagnosis Diabetes Mellitus

NaiArun et. al. **[1]** in applied an algorithm which classifies the risk of diabetes mellitus using some renowned machine learning classification methods including Logistic Regression and Naive Bayes. The data set was collected from 26 Primary Care Units (PCU) in Sawanpracharak Regional Hospital. In this study, the accuracy of Logistic Regression (LR) and Naïve Bayes (NB) came out as 82.308% & 81.010% respectively**.**

Harry Zhang **[2]** proposed a new explanation on the classification performance of naive Bayes and investigated the optimality of naive Bayes under the Gaussian distribution, and presented the explicit sufficient condition under which naive Bayes is optimal.

Hina S et. al. **[3]** has researched different classifying algorithms such as Naïve Bayes, J.48, Zero R, Random Forest, and Regression to depict the results using the Pima Indians Diabetes

Dataset.The work mainly emphasis on reducing the classification error and for Naïve Bayes the absolute error came out as 0.2841 and Logistic Regression came out as 0.2867.

## **B. Logistic Regression For Diagnosis Diabetes Mellitus**

Pramila M. Chawan et. al. **[4]** developed a system using data mining which has the ability to predict whether the patient has diabetes or not based on three classification methods namely,Support Vector Machine, Logistic regression and Artificial Neural Network algorithms. The accuracy of Logistic Regression in their system is 78%.

Naveen Kishore G et. al. **[5]** used SVM,Decision Tree,KNN,Logistic Regression,Random Forest measuring different parameters within the PIDD dataset where Logistic Regression showed nearly 72.39% accuracy.

## **C. Random Forest For Diagnosis Diabetes Mellitus**

Mani Butwall et. al. in**[6]** used the Pima Indian diabetic database (PIDD) and the tested data mining algorithms to predict their accuracy in diabetic status from the 8 variables given.After analysing the confusion matrix we can say that Random Forest Classifier based approach outperforms better with the accuracy of 99.7%.

Bhatt K. et. al. in**[7]** their work, accuracy of predicting diabetic status on the PIDD was 82.6% on the initial random sample, which exceeds the previously used machine learning algorithms that ranged from 66-81%. Using 576 training instances, the sensitivity and specificity of their algorithm was 76% on the remaining 192 instances.

Huang et. al. in**[8]** their research the data mining approaches in Diabetes diagnosis yields a result of almost 91-92%**.**

Quan Zou et. al. in**[9]**In their study, fivefold cross validation was used to examine the models. They randomly selected 68994 healthy people and diabetic patients' data, respectively as training set. In this study, we used principal component analysis (PCA) and minimum redundancy maximum relevance (mRMR) to reduce the dimensionality. The results showed that prediction with random forest could reach the highest accuracy (ACC = 0.8084) when all the attributes were used.

Anirudh Hebbar P et. al. in**[10]**their work proposes a method called "drap" for detecting diabetes. The proposed algorithm is evaluated on real-life data set and eminal results show the accuracy of 72% for Decision Tree and 76.5% Random Forest respectively.

**D. Decision Tree For Diagnosis Diabetes Mellitus**

Asha Gowda Karegowda et. al. used **[11]** Medical Data mining process of extracting hidden patterns from medical data. The proposed cascaded model with categorical data obtained the classification accuracy of 93.33 % when compared to accuracy of 73.62 % using C4.5 alone for PIMA Indian diabetic dataset.

Hybrid K-means and Decision tree **[12]** achieved the classification accuracy of 92.38% using 10 fold cross validations for continuous data.Later on they have archived classification accuracy of 72.88 % using ANN, 78.21% using DT_ANN where decision tree C4.5 is used to identify relevant features and given as input to ANN using PIDD dataset **[13].**

Anirudh Hebbar P et. al. in **[14]** The proposes a method called “drap” for detecting diabetes. The proposed algorithm is evaluated on real-life data set. Our seminal results show the accuracy of 72% for Decision Tree.

**E. K-nearest Neighbor For Diagnosis Diabetes Mellitus**

IJCST et. al.**[15]** have taken one sample training dataset containing 100 rows and 11 columns of the above mentioned attributes. We have applied K- Nearest neighbor algorithm on the training and test sample data and obtained results for different values of K. Accuracy for K=3,Testdata1=70%,Testdata2=50% and K= 5, Testdata1=69%,Testdata2=63%.

Asha Gowda Karegowda et. al.**[16]**, used cascading K mean ad K nearest neighbor algorithm for categorization of diabetic patients in their proposed work.Accuracy achieved by the proposed system is 82%.

Christobel et. al. (2013) **[17]** have proposed a new class-wise K-Nearest Neighbor (CKNN) classification algorithm for classification of diabetes data. They have used diabetes data set for testing the CKNN algorithm and compared the various cases. The proposed CKNN model gives better classification accuracy of 78.16% compared to simple KNN.

**F. SVM For Diagnosis Diabetes Mellitus**

L.Cheng et. al. **[18]** investigate clinical diagnosis through case study and the Result show the accuracy of 72% for SVM.

S.Ghumbre et. al. **[19]** used pima India dataset [PIDD] and presented an intelligent system based support vector machine along with a radial basis function network for the diagnosis. Results obtained show that SVM performs with accuracy of 97%.

Xiao-Peng et. al. **[20]** designed a cancer predicting medical diagnosis system using SVM model. They used univariate analysis to analyze the relationship between the six image indicators. The accuracy of diabetes predicting status on PIDD was 86%.

# 3. RELATED STUDIES

## 3.1 MACHINE LEARNING

**Machine learning** (**ML**) is the scientific study of algorithms and statistical models that computer systems use to perform a specific task without using explicit instructions. It is seen as a subset of artificial intelligence. Machine learning algorithms build a mathematical model based on sample data, known as “training data", in order to make predictions or decisions without being explicitly programmed to perform the task. Machine learning algorithms are used in a wide variety of applications, such as “email filtering” and “computer vision”, where it is difficult or infeasible to develop a conventional algorithm for effectively performing the task.
Machine learning is closely related to computational statistics, which focuses on making predictions using computers. The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. There are two models of machine learning –

- **Predictive Model**
- **Descriptive Model.**

We’ll be using the **Predictive Modelling** approach.

### 3.1.1 PREDICTIVE MODEL

- A predictive model is used for tasks that involve, as the name implies, the prediction of one value using other values in the dataset.
- The learning algorithm attempts to discover and model the relationship between the target feature (the feature being predicted) and the other features.
- Because predictive models are given clear instruction on what they need to learn and how they are intended to learn it, the process of training a predictive model is known as **Supervised Learning.**

#### 3.1.1.1 SUPERVISED LEARNING

Supervised learning as the name indicates a presence of supervisor as teacher. Basically supervised learning is a learning in which we teach or train the machine using data which is well labeled that means some data is already tagged with correct answer. After that, machine is provided with new set of examples (data) so that supervised learning algorithm analyses the training data (set of training examples) and produces a correct outcome from labeled data. Supervised learning classified into two categories of algorithms: **Classification and Regression**.

In this project, the **Classification Approach** has been taken.

### 3.1.2 NAÏVE-BAYES CLASSIFIER

Naive Bayes classifier is a simple but most operative algorithm for the classification problems. Naive Bayes are statistical classifiers that works by making a hypothesis of conditional independence with the training datasets. Henceforth, Naïve Bayes classifier is the appropriate classification technique that verdicts best solution for a dataset from a pool of different objects.

**• Algorithm:-**

Step 1:Create a frequency table for all the features against the different classes.

Step 2:Draw the likelihood table for the features against the classes.

Step 3:Measure the conditional probabilities for all the classes.

Step 4:Calculate .

### 3.1.3 LOGISTIC REGRESSION

Logistic Regression was mostly used in the biological research and applications in the early 20th century. Logistic Regression (LR) is one of the most used machine learning algorithm that is used where the target variable is categorical. Recently, LR is a popular method for binary classification problems. Moreover, it presents a discrete binary product between 0 and 1. Logistic Regression computes the relationship between the feature variables by assessing probabilities (p) using underlying logistic function.

• **Algorithm:-**

To Implement Logistic Regression, we will use these steps:

Step 1: Data Pre-processing step.

Step 2: Fitting Logistic Regression to the Training set.

Step 3: Predicting the test result.

Step 4: Test accuracy of the result.

Step 5: Visualizing the test set result.

### 3.1.4 RANDOM FOREST

Random forest is a supervised learning algorithm which is used for both classification as well as regression. But however, it is mainly used for classification problems. As we know that a forest is made up of trees and more trees means more robust forest. Similarly, random forest algorithm creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution by means of voting. It is an ensemble method which is better than a single decision tree because it reduces the over-fitting by averaging the result.

• **Algorithm:-**

Step 1:- Select random K data point from the training set.

Step 2:- Build the decision trees associated with the selected data point.(Subset)

Step 3:- Choose the number of N for decision trees that you want to build.

Step 4:- Repeat Step 1 & 2.

Step 5:- For the new data point, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

### 3.1.5 DECISION TREE

Decision tree is the most powerful and popular tool for classification and prediction. A Decision tree is a flowchart like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.

**• Algorithm:-**

Step-1: Begin the tree with the root node, which contains the complete dataset.

Step-2: Find the best attribute in the dataset using Attribute Selection Measure (ASM).

Step-3: Divide the root node into subsets that contains possible values for the best attributes.

Step-4: Generate the decision tree node, which contains the best attribute.

Step-5: Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

### 3.1.6 K-NN

KNN is a method which is used for classifying objects based on closest training examples in the feature space. KNN is the most basic type of instance-based learning or lazy learning. It assumes all instances are points in n-dimensional space. A distance measure is needed to determine the "closeness" of instances. KNN classifies an instance by finding its nearest neighbors and picking the most popular class among the neighbors.

In KNN, the training samples are mainly described by n-dimensional numeric attributes. The training samples are stored in an n dimensional space. When a test sample (unknown class label) is given, k-nearest neighbor classifier starts searching the 'k' training samples which are closest to the unknown sample or test sample. Closeness is mainly defined in terms of Euclidean distance. The Euclidean distance between two points P and Q i.e. P (p1,p2, …. Pn) and Q (q1, q2,..qn) is defined by the following equation:-d(P,Q)= $\sum_{i=1}^{n}(Pi - Qi)^2$

**• Algorithm:-**

Step 1: Take a sample dataset of $n$ columns and $m$ rows named as $R$ .In which $n - 1^{th}$columns are the input vector and $n^{th}$column is the output vector.

Step 2: Take a test dataset of n-1 attributes and y rows named as P.

Step 3: Find the Euclidean distance between every S and T by the help of formula:

➯ Euclidean Distance=$\sqrt{\sum_{i=1}^{y}.\sum_{j=1}^{m}.\sum_{i=1}^{n-1}(R(j,l)-P(i,l))^2}$

Step 4: Then, Decide a random value of K. is the no. of nearest neighbors.

Step 5: Then with the help of these minimum distance and Euclidean distance find out the $n^{th}$ column of each.

Step 6: Find out the same output values.

### 3.1.7 SVM

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

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

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

• **Algorithm:-**

Step1: Data Pre-processing step.

Step2: Fitting the SVM classifier to the training set.

Step3: Predicting the test set result.

Step4: Creating the confusion matrix.

Step5: Visualizing the training set result.

Step6: Visualizing the test set result.

# 4. IMPLEMENTATION OF ALGORITHM

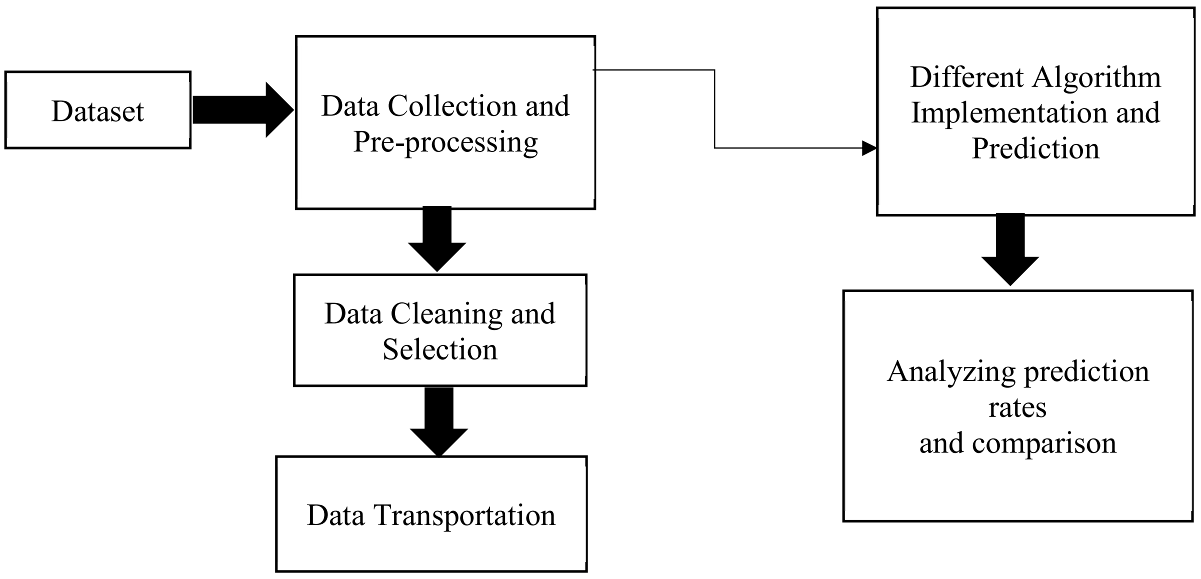
## 4.1 DATASET USED

The PIMA Indian Diabetes Dataset has been used in this study, provided by the UCI Machine Learning Repository. The dataset has been originally collected from the National Institute of Diabetes and Digestive and Kidney Diseases. The dataset consists of some medical distinct variables, such as pregnancy record, BMI, insulin level, age, glucose concentration, diastolic blood pressure, triceps skin fold thickness, diabetes pedigree function [14] etc. This dataset has 768 patient's data where all the patients are female and at least 21 years old. The number of true cases are 268 (34.90%) and the number of false cases are 500 (65.10%), respectively, in the dataset. In the following we chose eight distinct parameters for data prepossessing such as,
This dataset describes the medical records for Pima Indians and whether or not each patient will have an onset of diabetes within the years.
Fields description follow:

- Number of times pregnant
- Plasma glucose concentration a 2 hours in an oral glucose tolerance test
- Diastolic blood pressure (mm Hg)
- Triceps skin fold thickness (mm)
- 2-Hour serum insulin (mu U/ml)
- Body mass index (weight in kg/(height in m)^2)
- Diabetes pedigree function
- Age (years)
- Class variable (1:tested positive for diabetes, 0: tested negative for diabetes)

## 4.2 WORK-FLOW

Our work-flow has been described in the below diagram. Detailed implementation has been shown in the later part.


Dataset
Data Collection and Pre-processing
Data Cleaning and Selection
Data Transportation
Different Algorithm Implementation and Prediction
Analyzing prediction rates and comparison

### 4.2.1 DATA PRE-PROCESSING:-

Data Pre-processing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format(in our case .csv format) which is not good for the analysis.

For achieving optimal and moreover better results from the applied model the format of the data has to be in a proper manner. Some specified Machine Learning model needs information in a specified format as most machine learning algorithms does not support null values, therefore to execute different algorithm null values have to be managed from the original raw data set.

Another aspect is that data set should be formatted in such a way that more than one or more Machine Learning algorithms are executed in one data set, and best out of them is chosen.

## 4.3 TECHNOLOGY USED

This machine learning based project is implemented using **python** because of its simple syntax, modular architecture, rich text processing tools and the ability to work on multiple operating systems. Python language is one of the most flexible languages and can be used for various purposes. The language is great to use when working with machine learning algorithms as it contains special libraries for machine learning.

**Python** is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

### 4.3.1 PYTHON PACKAGES

#### • Numpy

NumPy is the fundamental package for scientific computing with Python. It contains among other things a powerful N-dimensional arrayobject, sophisticated(broadcasting) functions, tools for integrating C/C++ and Fortran code useful linear algebra, Fourier transform, and random number capabilities. Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary datatypes can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

#### • Matplotlib

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

#### • Pandas

*Pandas* is an open source, BSD-licensed library providing high performance, easy- to-use data structures and data analysis tools for the *Python* programming language. Pandas library is well suited for data manipulation and analysis using python. In particular, it offers data structures and operations for manipulating numerical tables and time series.

#### • Scikit-learn

Scikit-learn provides machine learning libraries for python some of the features of Scikit- learn includes:

- Simple and efficient tools for data mining and data analysis.
- Accessible to everybody, and reusable in various contexts.
- Built on NumPy, SciPy, and matplotlib.

- Open source, commercially usable –BSDlicense.

- **Seaborn**

Seaborn is a Python library for making statistical visualizations. It's built to provide eye candy plots and at the same time it makes developers' life easier. We've all been in the situation where we needed to build a simple decent looking histogram and ended up making several function invocations and setting arguments without fully knowing what we're doing. Seaborn tackles this not by reinventing the wheel but by improving it. It is built on top of Matplotlib and provides a high level API that makes "a well-defined set of hard things easy" (as stated in the **docs**), amongst other things by making that its methods work greatly by passing a minimal set of arguments.

## 4.3.2 SOFTWARE AND HARDWARE USED

### 4.3.2.1 HARDWARE REQUIREMENT

- Hard disk – 500 GB
- RAM – 4 GB
- Processor – i3 4thGeneration

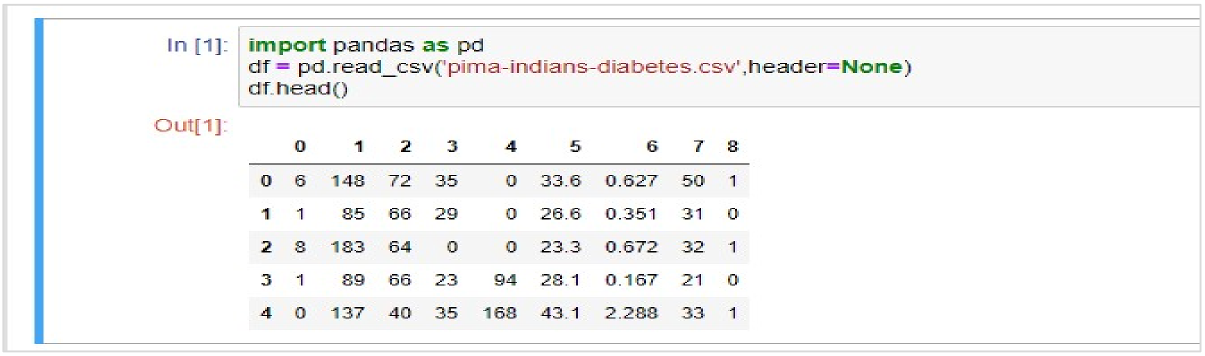
### 4.3.2.2 SOFTWARE SPECIFICATION

- **Anaconda 3**- Anaconda is a free and open distribution of Python programming languages for data science and machine learning related applications (large-scale data processing, predictive analytics, scientific computing), that aims to simplify package management and deployment.

# 5. RESULT & DISCUSSION

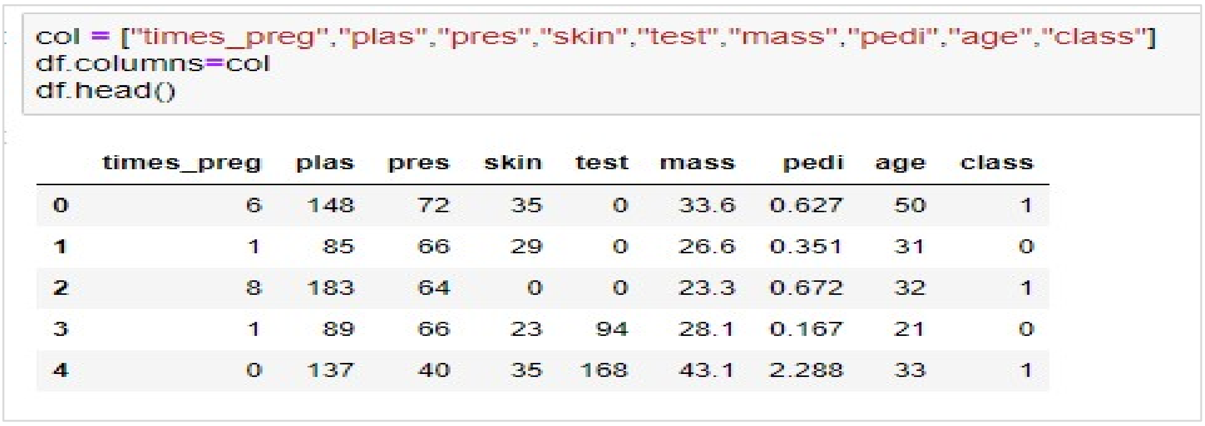
## 5.1 NAÏVE-BAYES AND LOGISTIC REGRESSION

In our project we have processed the raw dataset for Naïve-Bayes and Logistic Regression as given in the snippet below:-

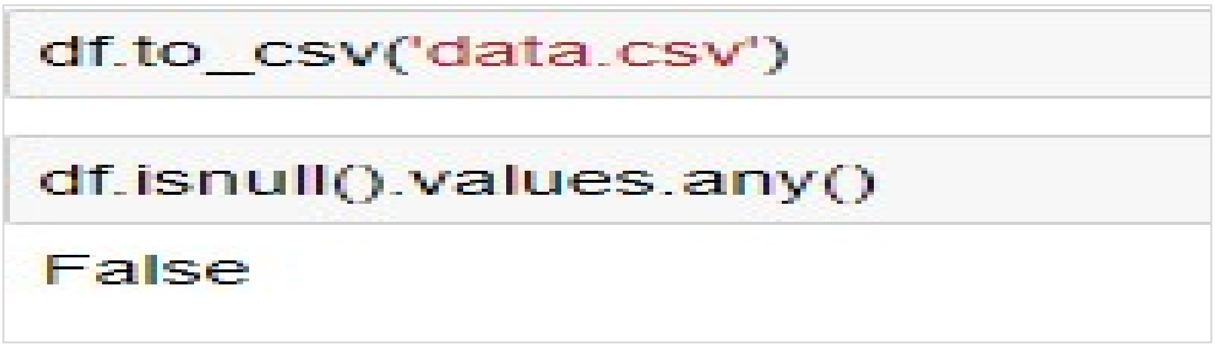


```
In [1]: import pandas as pd
        df = pd.read_csv('pima-indians-diabetes.csv',header=None)
        df.head()

Out[1]:
      0    1   2   3    4     5      6   7  8
   0  6  148  72  35    0  33.6  0.627  50  1
   1  1   85  66  29    0  26.6  0.351  31  0
   2  8  183  64   0    0  23.3  0.672  32  1
   3  1   89  66  23   94  28.1  0.167  21  0
   4  0  137  40  35  168  43.1  2.288  33  1
```

The processed dataset obtained from the above raw dataset given below:-



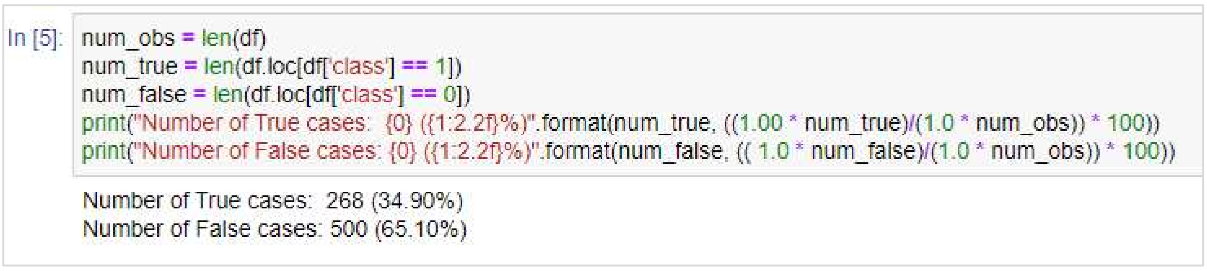
```
col = ["times_preg","plas","pres","skin","test","mass","pedi","age","class"]
df.columns=col
df.head()

   times_preg  plas  pres  skin  test  mass   pedi  age  class
0           6   148    72    35     0  33.6  0.627   50      1
1           1    85    66    29     0  26.6  0.351   31      0
2           8   183    64     0     0  23.3  0.672   32      1
3           1    89    66    23    94  28.1  0.167   21      0
4           0   137    40    35   168  43.1  2.288   33      1
```



```
df.to_csv('data.csv')

df.isnull().values.any()

False
```

Here as we can see we have omitted the numeric numberings and changed non-numeric data entry column into numeric format. Also we've checked is there any null values present in the dataset.

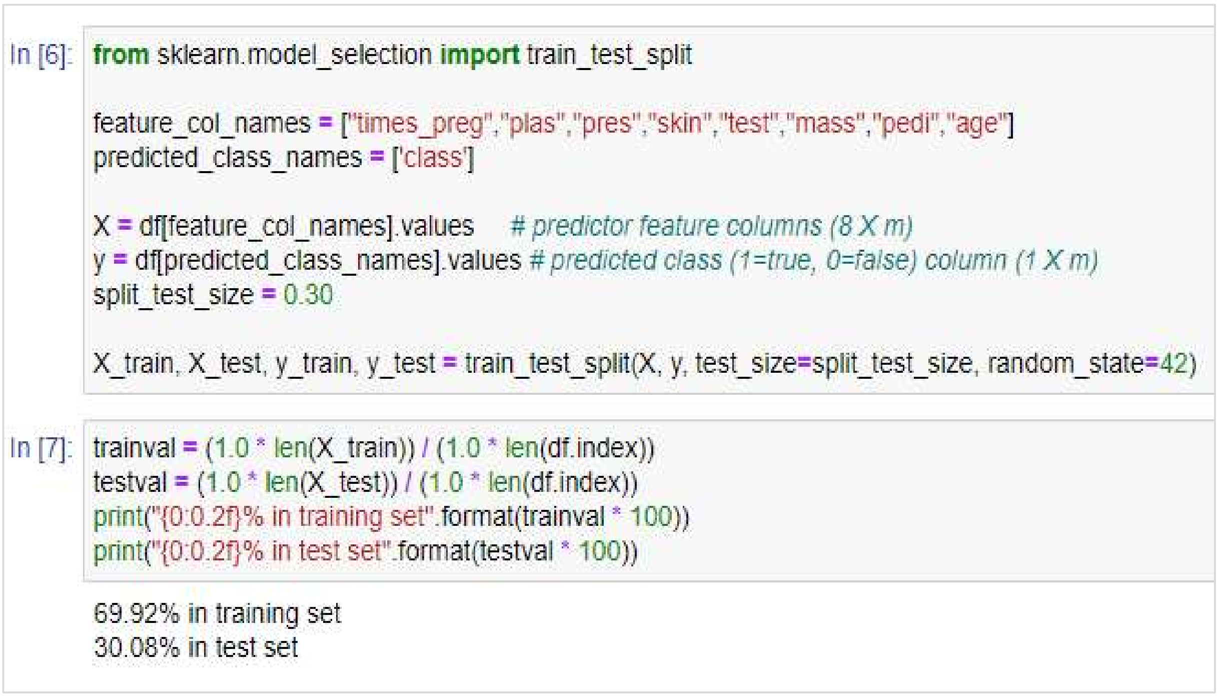
We've counted the number of true and false cases of diabetes prediction from the dataset.



```
In [5]: num_obs = len(df)
        num_true = len(df.loc[df['class'] == 1])
        num_false = len(df.loc[df['class'] == 0])
        print("Number of True cases:  {0} ({1:2.2f}%)".format(num_true, ((1.00 * num_true)/(1.0 * num_obs)) * 100))
        print("Number of False cases: {0} ({1:2.2f}%)".format(num_false, (( 1.0 * num_false)/(1.0 * num_obs)) * 100))

        Number of True cases:  268 (34.90%)
        Number of False cases: 500 (65.10%)
```

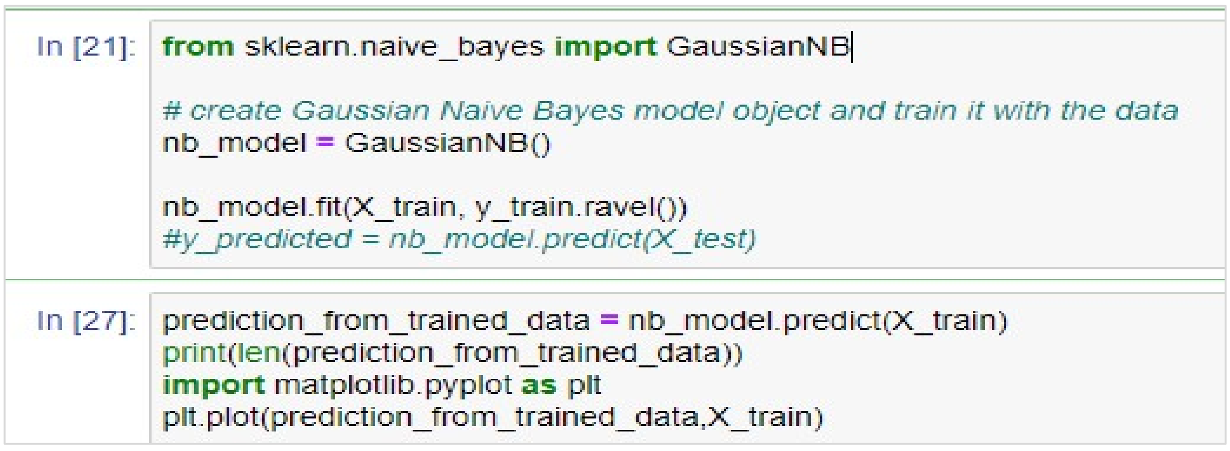
We divided the processed dataset into training and testing part by splitting it. The training part trains the machine with the existing values in the dataset and the testing part is required to check the accuracy after the machine learns from the training part.

Here X_train are the features of the machine and y_train teaches the machine about the output criteria corresponding to the features of X_train.

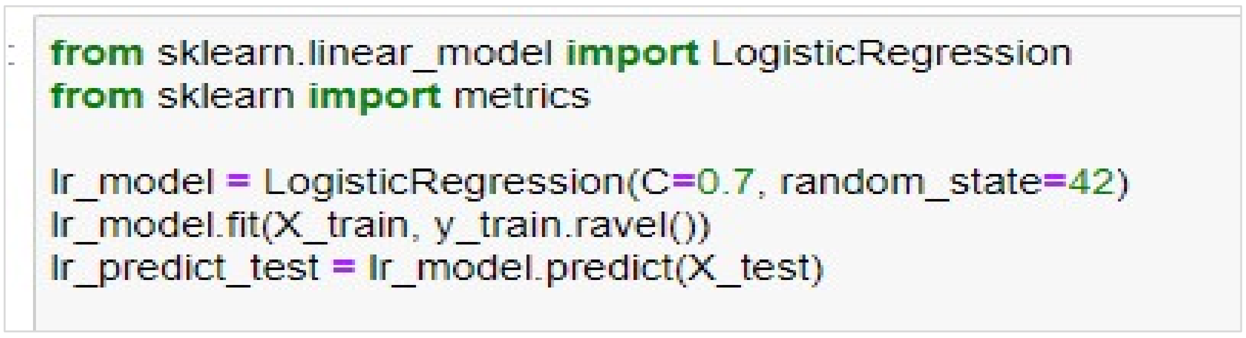


```
In [6]: from sklearn.model_selection import train_test_split

        feature_col_names = ["times_preg","plas","pres","skin","test","mass","pedi","age"]
        predicted_class_names = ['class']

        X = df[feature_col_names].values     # predictor feature columns (8 X m)
        y = df[predicted_class_names].values # predicted class (1=true, 0=false) column (1 X m)
        split_test_size = 0.30

        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=split_test_size, random_state=42)

In [7]: trainval = (1.0 * len(X_train)) / (1.0 * len(df.index))
        testval = (1.0 * len(X_test)) / (1.0 * len(df.index))
        print("{0:0.2f}% in training set".format(trainval * 100))
        print("{0:0.2f}% in test set".format(testval * 100))

        69.92% in training set
        30.08% in test set
```

The below attached code snippet shows how GaussianNB imported from sklearn.naive_bayes and LogisticRegression imported from sklearn.linear_model and also implementation of the same.

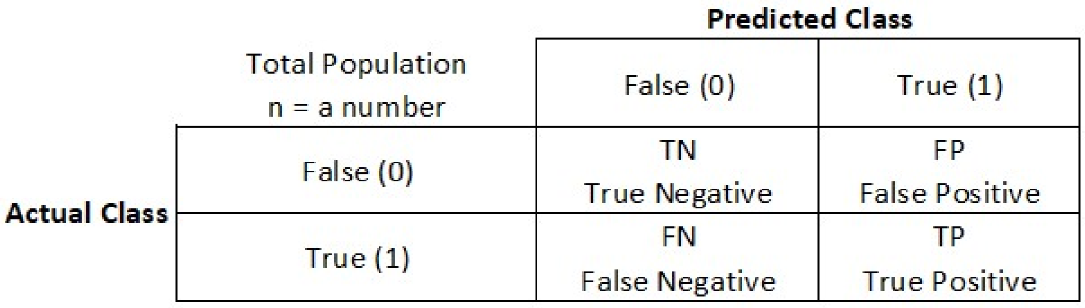


```
In [21]: from sklearn.naive_bayes import GaussianNB

         # create Gaussian Naive Bayes model object and train it with the data
         nb_model = GaussianNB()

         nb_model.fit(X_train, y_train.ravel())
         #y_predicted = nb_model.predict(X_test)

In [27]: prediction_from_trained_data = nb_model.predict(X_train)
         print(len(prediction_from_trained_data))
         import matplotlib.pyplot as plt
         plt.plot(prediction_from_trained_data,X_train)
```



```
: from sklearn.linear_model import LogisticRegression
  from sklearn import metrics

  lr_model = LogisticRegression(C=0.7, random_state=42)
  lr_model.fit(X_train, y_train.ravel())
  lr_predict_test = lr_model.predict(X_test)
```

Performance of all the classification algorithms are assessed by different statistical measurement aspects such as accuracy, recall, precision etc. These classification measurement factors are calculated by the terms: True Positive (TP), False Positive (FP), True Negative (TN) and False Negative (FN).
Let's take a look at the confusion matrix table example and see what the terms mean.



| | | **Predicted Class** | |
|---|---|---|---|
| | Total Population n = a number | False (0) | True (1) |
| **Actual Class** | False (0) | TN True Negative | FP False Positive |
| | True (1) | FN False Negative | TP True Positive |

**True Positive (TP)**: Prediction results are yes and the patient have diabetes.
**True Negative (TN)**: Prediction results are no and the patient do not have diabetes.
**False Positive (FP)**: Prediction results are yes but the patient do not actually have the diabetes (Also known as a "Type 1 error").
**False Negative (FN)**: Prediction results are no but the patient have diabetes (Also known as a "Type2 error").
The computation formula of the measurement factors are as follows,
**Accuracy** in classification problems is the ratio of correct predictions made by the model over all kinds of suitable predictions completed.

**Accuracy** $= \frac{(TP+TN)}{(TP+FP+FN+TN)}$

**True positive rate, sensitivity, or recall** defined here is a measure that tells us what ratio of positive instances that actually have diabetes with the actual positive instances (patient having diabetes are TP and FN).

**TPR= Sensitivity = Recall** $= \frac{(TP)}{(TP+FN)}$

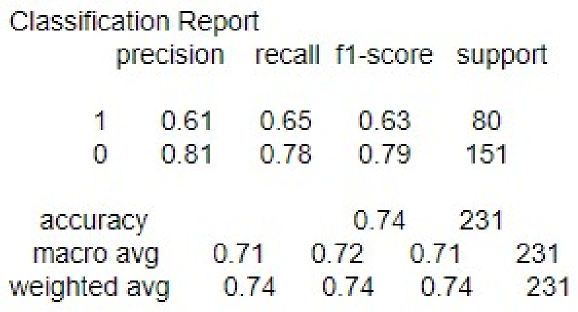
**Positive predictive value or precision** is the number of accurate positive scores divided by the number of positive scores predicted by the classification algorithm.

**Precision** = $\frac{(TP)}{(TP+FP)}$

**F1 measure** is a weighted average of the recall and precision. For the good performance of the classification algorithm, it must be one and for the bad performance, it must be zero.

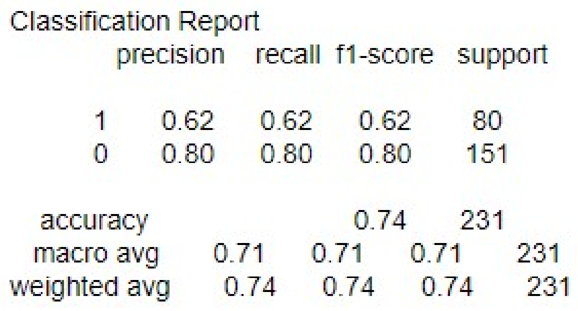
**F1**= $\frac{2*(Recall*Precision)}{Recall+Precision}$

In this experiment, we've planned to conduct different analysis to evaluate the two machine learning classification algorithms for diabetes prediction. From the Pima Indian Data set, **268** true samples and **500** negative samples were taken into analysis. We split the diabetes data set into two parts where the training set contains **69.92%** and the test set contains the remaining **30.08%** of the data, where, **Training True : 188 (35.01%)** ,**Training False: 349 (64.99%), Test True: 80 (34.63%), Test False: 151 (65.37%).** Table 1 & 2 illustrates the **CLASSIFICATION REPORT** of Naïve-Bayes and Logistic Regression mentioning different classification measures such as accuracy, recall, precision, and f1 measure.



```
Classification Report
              precision    recall  f1-score   support

           1       0.61      0.65      0.63        80
           0       0.81      0.78      0.79       151

    accuracy                           0.74       231
   macro avg       0.71      0.72      0.71       231
weighted avg       0.74      0.74      0.74       231
```

Table 1: Classification Report of Naïve-Bayes



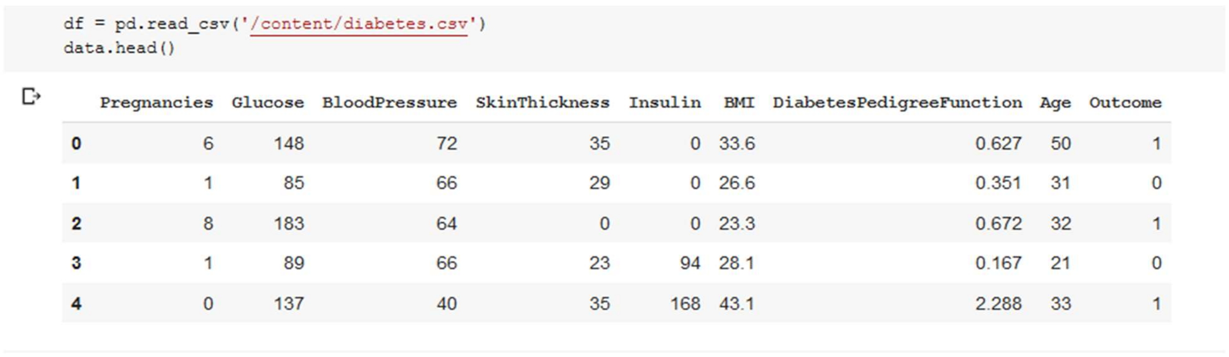
```
Classification Report
              precision    recall  f1-score   support

           1       0.62      0.62      0.62        80
           0       0.80      0.80      0.80       151

    accuracy                           0.74       231
   macro avg       0.71      0.71      0.71       231
weighted avg       0.74      0.74      0.74       231
```

Table 2: Classification Report of Logistic Regression

The Naïve-Bayes and Logistic Regression classifiers shows the accuracy level of nearly **74%**, more precisely **0.7359(Naïve-Bayes)** and **0.7403(Logistic Regression)** which indicates that the performance of these techniques are pretty well. F-1 measure indicates that the classification techniques that we've used almost predict accurate results.

## 5.2 RANDOM FOREST

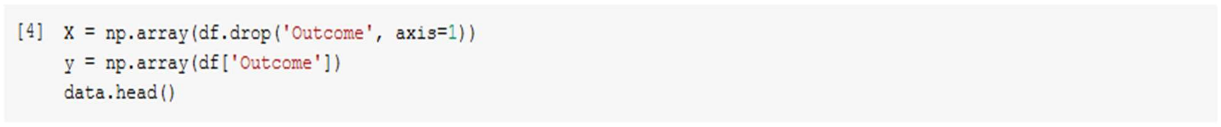
We have processed the raw dataset for Random Forest as given in the snipped below-



```
df = pd.read_csv('/content/diabetes.csv')
data.head()
```

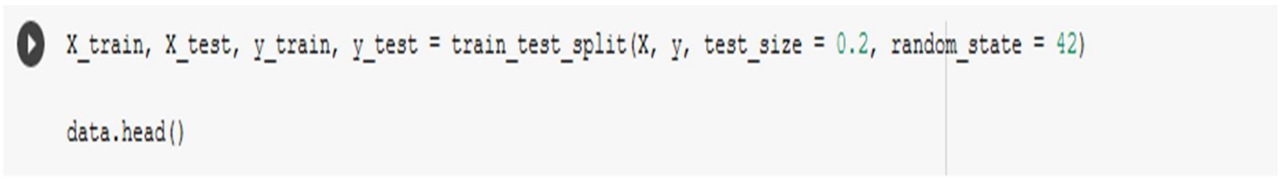
| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 6 | 148 | 72 | 35 | 0 | 33.6 | 0.627 | 50 | 1 |
| 1 | 1 | 85 | 66 | 29 | 0 | 26.6 | 0.351 | 31 | 0 |
| 2 | 8 | 183 | 64 | 0 | 0 | 23.3 | 0.672 | 32 | 1 |
| 3 | 1 | 89 | 66 | 23 | 94 | 28.1 | 0.167 | 21 | 0 |
| 4 | 0 | 137 | 40 | 35 | 168 | 43.1 | 2.288 | 33 | 1 |

Now according to that we have to split the x and y data set.



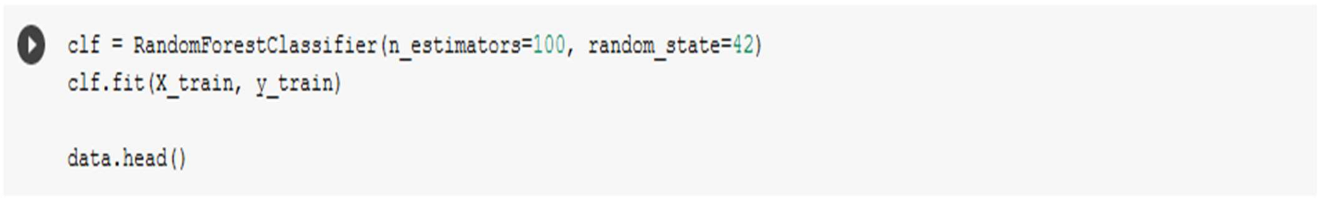
```
[4] X = np.array(df.drop('Outcome', axis=1))
    y = np.array(df['Outcome'])
    data.head()
```

Now we process the data accordingly to the training set & testing set model selection for applying data set in Random Forest.



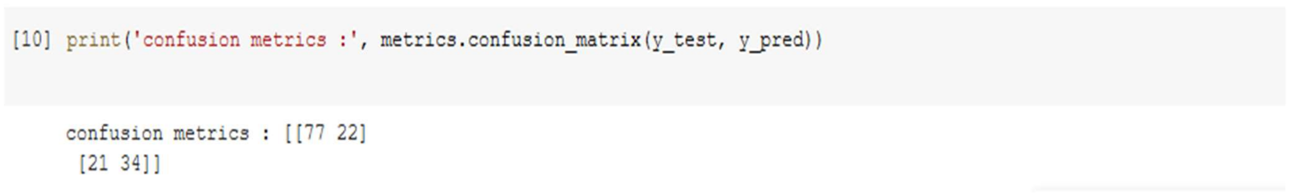
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)

data.head()
```

In this step we feed the data in Random Forest Algorithm.



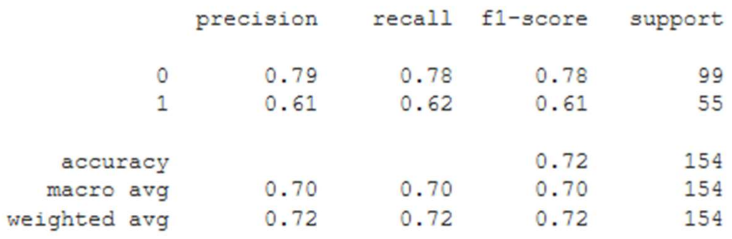
```
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train, y_train)

data.head()
```

After using the technique the resultant of the confusion metrics are shown below.



```
[10] print('confusion metrics :', metrics.confusion_matrix(y_test, y_pred))

     confusion metrics : [[77 22]
      [21 34]]
```

Here is the classification report of Random Forest techniques that measure such as-precision, recall, f1-scope and accuracy.



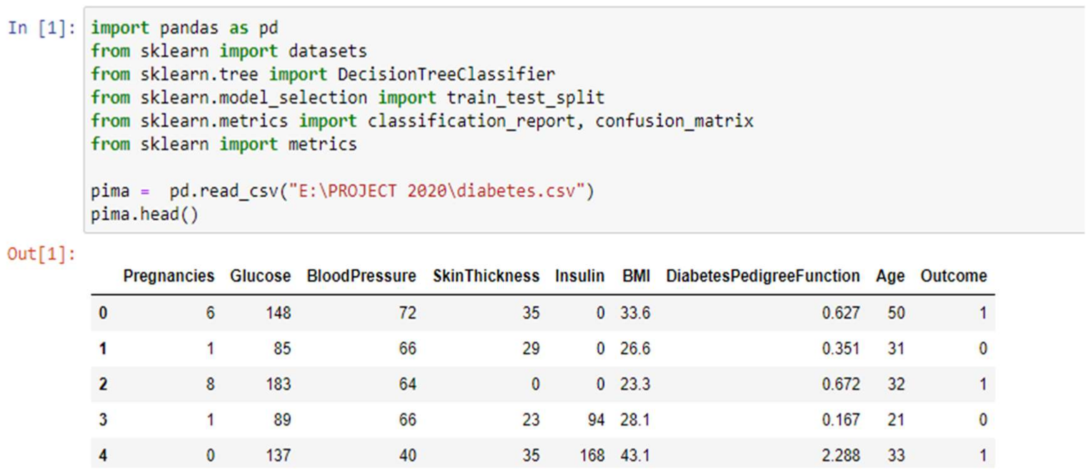
```
              precision    recall  f1-score   support

           0       0.79      0.78      0.78        99
           1       0.61      0.62      0.61        55

    accuracy                           0.72       154
   macro avg       0.70      0.70      0.70       154
weighted avg       0.72      0.72      0.72       154
```

**Classification report**

By using F-1 measure classification technique we have got an accuracy rate of nearly **72%**.

## 5.3 DECISION TREE

In our project we have processed the dataset for Decision tree as given in the snippet below:-

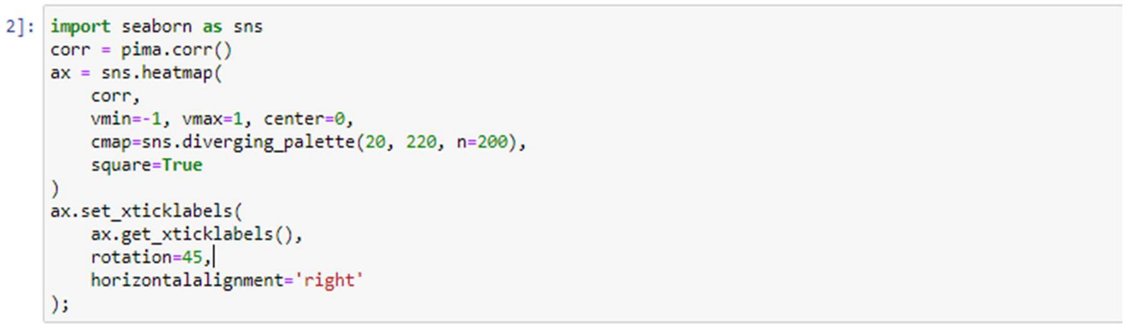


```
In [1]: import pandas as pd
        from sklearn import datasets
        from sklearn.tree import DecisionTreeClassifier
        from sklearn.model_selection import train_test_split
        from sklearn.metrics import classification_report, confusion_matrix
        from sklearn import metrics

        pima =  pd.read_csv("E:\PROJECT 2020\diabetes.csv")
        pima.head()
```

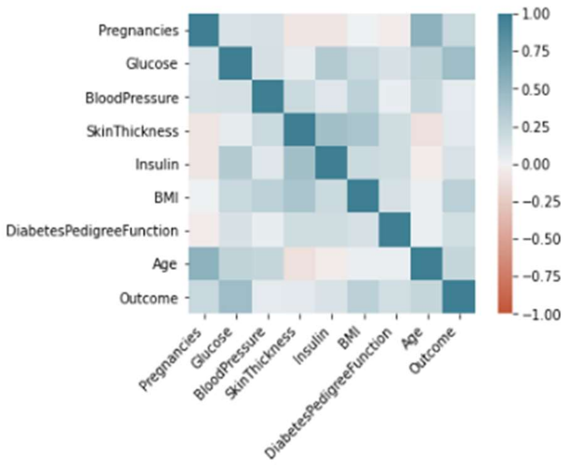
Out[1]:

| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 6 | 148 | 72 | 35 | 0 | 33.6 | 0.627 | 50 | 1 |
| 1 | 1 | 85 | 66 | 29 | 0 | 26.6 | 0.351 | 31 | 0 |
| 2 | 8 | 183 | 64 | 0 | 0 | 23.3 | 0.672 | 32 | 1 |
| 3 | 1 | 89 | 66 | 23 | 94 | 28.1 | 0.167 | 21 | 0 |
| 4 | 0 | 137 | 40 | 35 | 168 | 43.1 | 2.288 | 33 | 1 |

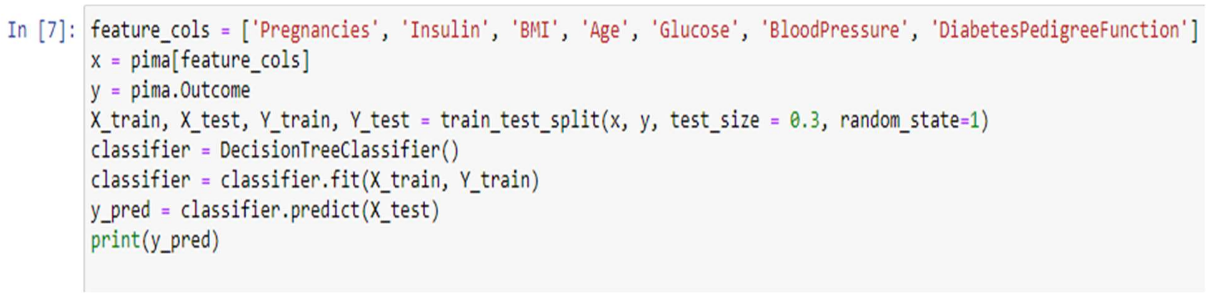
Here we see the heat map of the related dataset for Decision tree as given below:-



```
2]: import seaborn as sns
    corr = pima.corr()
    ax = sns.heatmap(
        corr,
        vmin=-1, vmax=1, center=0,
        cmap=sns.diverging_palette(20, 220, n=200),
        square=True
    )
    ax.set_xticklabels(
        ax.get_xticklabels(),
        rotation=45,
        horizontalalignment='right'
    );
```

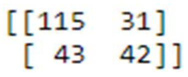


Now we need to divide given columns into two types of variables dependent (or target variable) and independent variable (or feature variables). To understand model performance, dividing the dataset into a training set and a test set is a good strategy. The training part trains the machine with the existing values in the dataset and the testing part is required to check the accuracy after the machine learns from the training part.



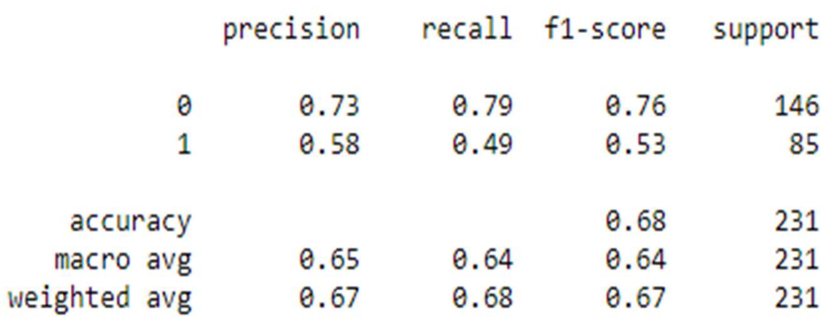
```
In [7]: feature_cols = ['Pregnancies', 'Insulin', 'BMI', 'Age', 'Glucose', 'BloodPressure', 'DiabetesPedigreeFunction']
        x = pima[feature_cols]
        y = pima.Outcome
        X_train, X_test, Y_train, Y_test = train_test_split(x, y, test_size = 0.3, random_state=1)
        classifier = DecisionTreeClassifier()
        classifier = classifier.fit(X_train, Y_train)
        y_pred = classifier.predict(X_test)
        print(y_pred)
```

After using this techniques , the resultent confusion matrix is---



```
[[115  31]
 [ 43  42]]
```

Here, we are showing the classification report of decision tree techniques, that measure such as-precision, recall, f1-measure and accuracy.



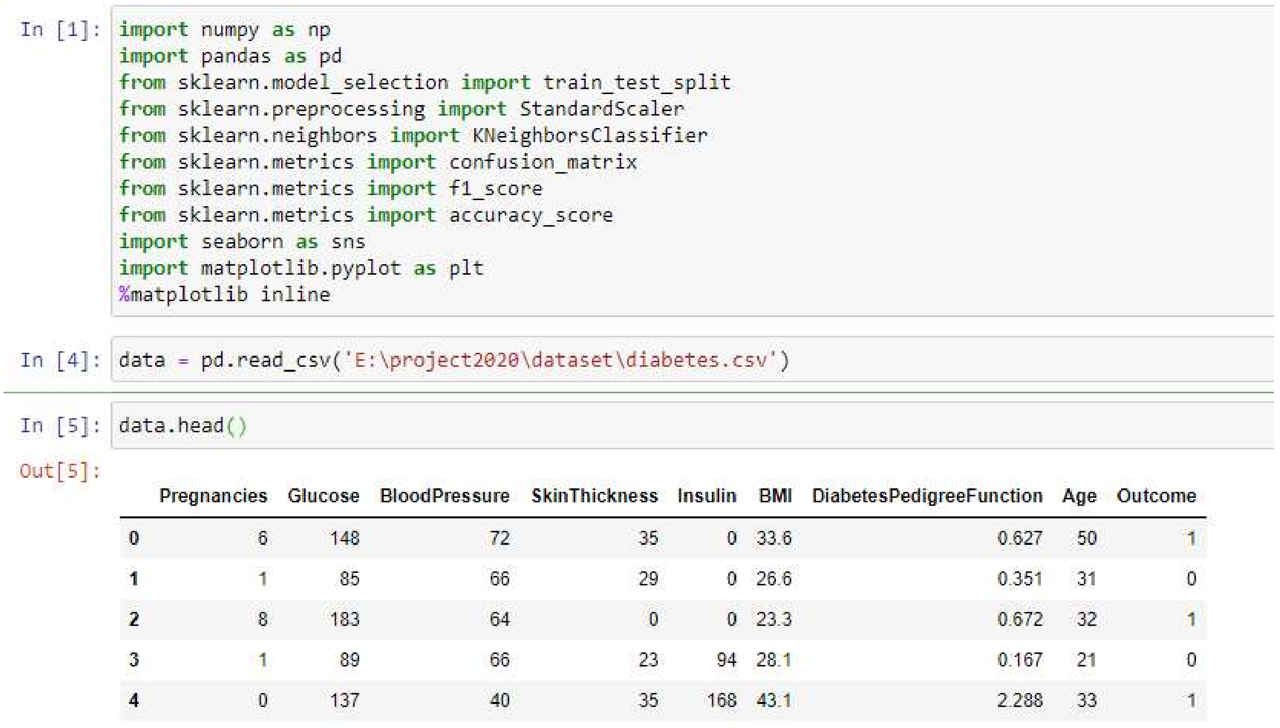
```
              precision    recall  f1-score   support

           0       0.73      0.79      0.76       146
           1       0.58      0.49      0.53        85

    accuracy                           0.68       231
   macro avg       0.65      0.64      0.64       231
weighted avg       0.67      0.68      0.67       231
```

**Classification report**

Accuracy can be computed by comparing actual test set values and predicted values and for this classification technique we've managed to get an accuracy rate of **68 % (0.67965).**

## 5.4 KNN

In our project we have processed the raw dataset for KNN as given in the snippet below:-



```
In [1]: import numpy as np
        import pandas as pd
        from sklearn.model_selection import train_test_split
        from sklearn.preprocessing import StandardScaler
        from sklearn.neighbors import KNeighborsClassifier
        from sklearn.metrics import confusion_matrix
        from sklearn.metrics import f1_score
        from sklearn.metrics import accuracy_score
        import seaborn as sns
        import matplotlib.pyplot as plt
        %matplotlib inline

In [4]: data = pd.read_csv('E:\project2020\dataset\diabetes.csv')

In [5]: data.head()

Out[5]:
```

|   | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 6 | 148 | 72 | 35 | 0 | 33.6 | 0.627 | 50 | 1 |
| **1** | 1 | 85 | 66 | 29 | 0 | 26.6 | 0.351 | 31 | 0 |
| **2** | 8 | 183 | 64 | 0 | 0 | 23.3 | 0.672 | 32 | 1 |
| **3** | 1 | 89 | 66 | 23 | 94 | 28.1 | 0.167 | 21 | 0 |
| **4** | 0 | 137 | 40 | 35 | 168 | 43.1 | 2.288 | 33 | 1 |

The processed dataset obtained from the above raw dataset given below:-

Replace columns like [Glucose, Blood Pressure, Skin Thickness, BMI, Insulin] with Zero as values with mean of respective column.

```
In [7]: zero_not_accepted = ['Glucose','BloodPressure','SkinThickness','BMI','Insulin']
        # for col in zero_not_accepted:
        #     for i in data[col]:
        #         if i==0:
        #             colSum = sum(data[col])
        #             meanCol=colSum/len(data[col])
        #             data[col]=meanCol

        for col in zero_not_accepted:
            data[col]= data[col].replace(0,np.NaN)
            mean = int(data[col].mean(skipna=True))
            data[col] = data[col].replace(np.NaN,mean)
```

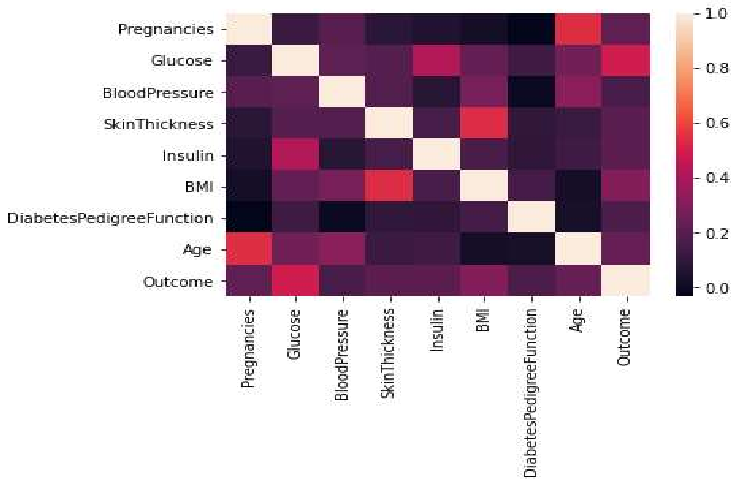
```
In [8]: data.head()
```

Out[8]:

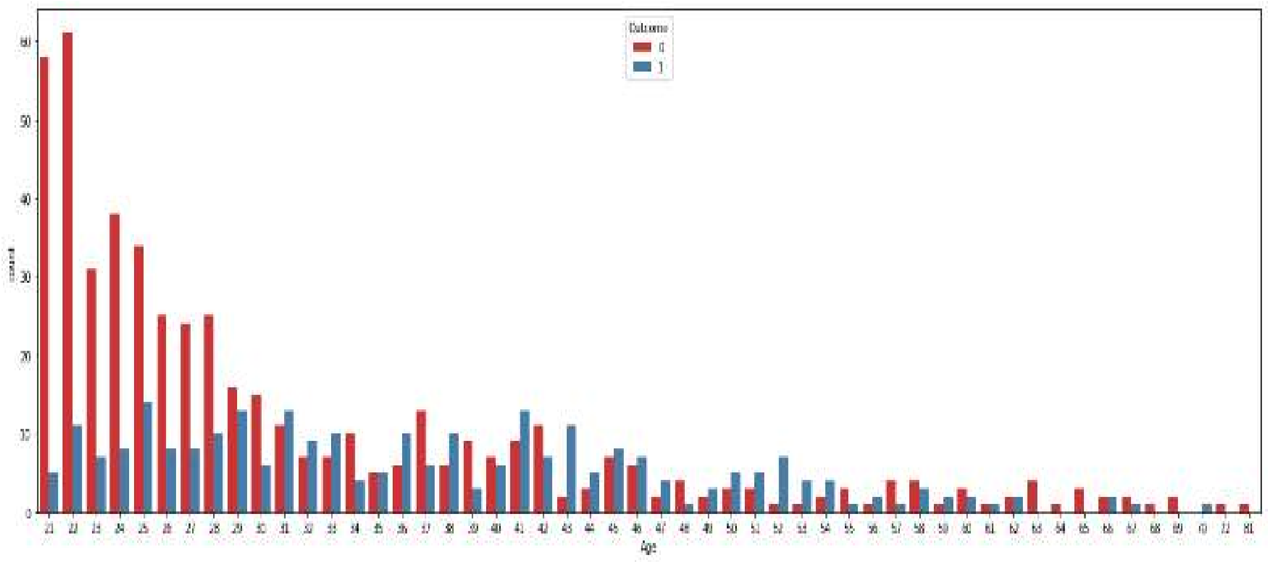
| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 6 | 148.0 | 72.0 | 35.0 | 155.0 | 33.6 | 0.627 | 50 | 1 |
| **1** | 1 | 85.0 | 66.0 | 29.0 | 155.0 | 26.6 | 0.351 | 31 | 0 |
| **2** | 8 | 183.0 | 64.0 | 29.0 | 155.0 | 23.3 | 0.672 | 32 | 1 |
| **3** | 1 | 89.0 | 66.0 | 23.0 | 94.0 | 28.1 | 0.167 | 21 | 0 |
| **4** | 0 | 137.0 | 40.0 | 35.0 | 168.0 | 43.1 | 2.288 | 33 | 1 |

Exploring data to know relation before processing,

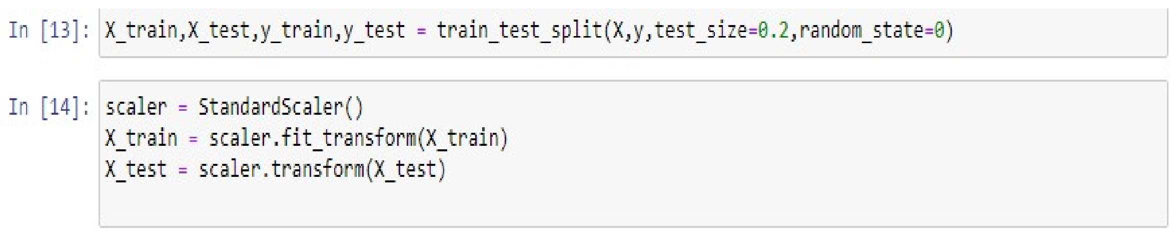
```
In [11]: sns.heatmap(data.corr())
Out[11]: <AxesSubplot:>
```



```
In [12]: plt.figure(figsize=(25,7))
         sns.countplot(x='Age',hue='Outcome',data=data,palette='Set1')
Out[12]: <AxesSubplot:xlabel='Age', ylabel='count'>
```

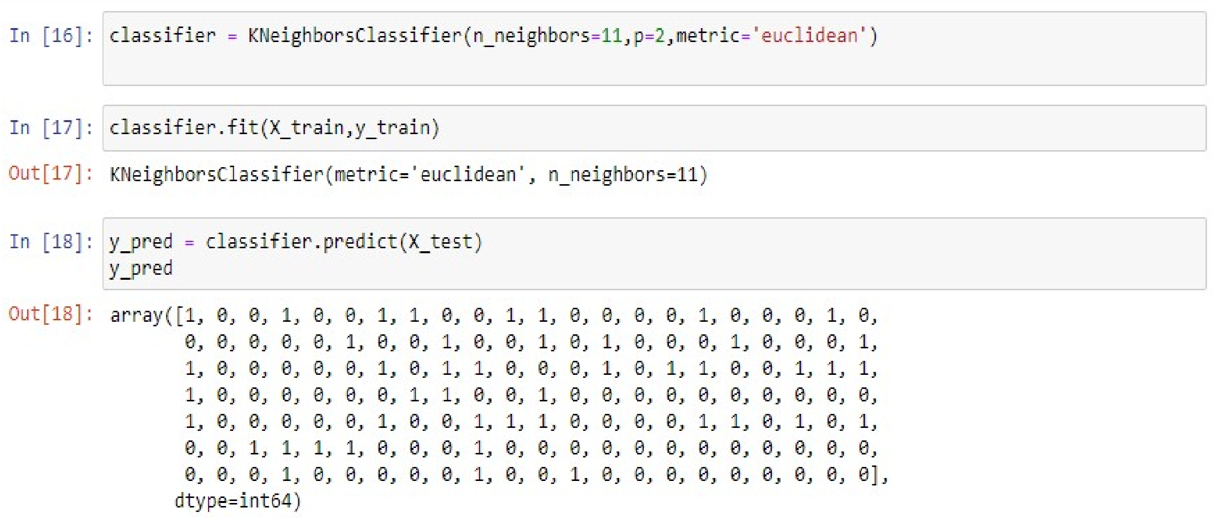


Splitting dataset into training and testing set and feature scaling.

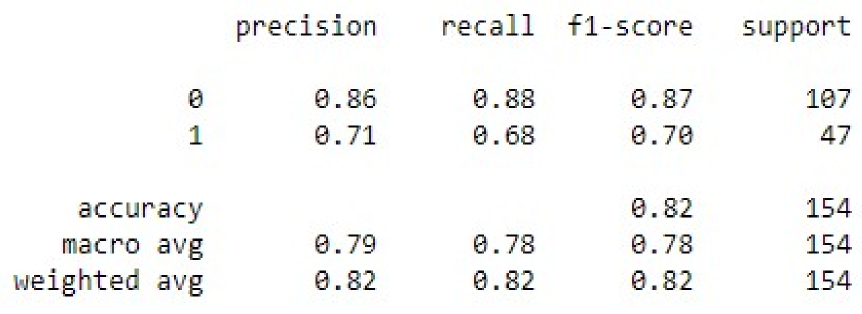


```
In [13]: X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=0)

In [14]: scaler = StandardScaler()
         X_train = scaler.fit_transform(X_train)
         X_test = scaler.transform(X_test)
```

The below attached code snippet and table shows **KNN** loading model and classification report.



```
In [16]: classifier = KNeighborsClassifier(n_neighbors=11,p=2,metric='euclidean')

In [17]: classifier.fit(X_train,y_train)

Out[17]: KNeighborsClassifier(metric='euclidean', n_neighbors=11)

In [18]: y_pred = classifier.predict(X_test)
         y_pred

Out[18]: array([1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0,
                0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1,
                1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1,
                1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1,
                0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
               dtype=int64)
```



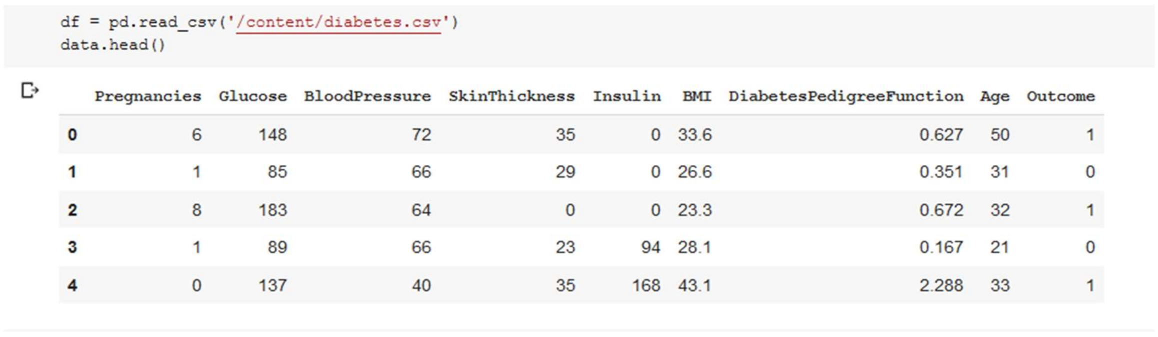
```
              precision    recall  f1-score   support

           0       0.86      0.88      0.87       107
           1       0.71      0.68      0.70        47

    accuracy                           0.82       154
   macro avg       0.79      0.78      0.78       154
weighted avg       0.82      0.82      0.82       154
```

**<u>Classification Report of KNN</u>**

The **k-nearest neighbour (KNN)** classifiers shows the accuracy level of nearly 82%, with **k** value of **11** which indicates that the performance of these techniques are pretty well.

## 5.5 SVM

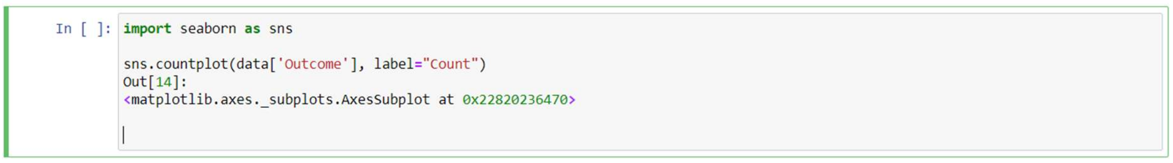
In our project we have processed the raw dataset for SVM as given in the snipped below-



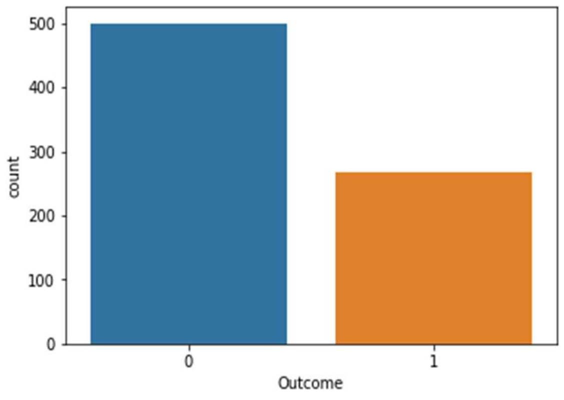
```
df = pd.read_csv('/content/diabetes.csv')
data.head()
```

|   | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 6 | 148 | 72 | 35 | 0 | 33.6 | 0.627 | 50 | 1 |
| **1** | 1 | 85 | 66 | 29 | 0 | 26.6 | 0.351 | 31 | 0 |
| **2** | 8 | 183 | 64 | 0 | 0 | 23.3 | 0.672 | 32 | 1 |
| **3** | 1 | 89 | 66 | 23 | 94 | 28.1 | 0.167 | 21 | 0 |
| **4** | 0 | 137 | 40 | 35 | 168 | 43.1 | 2.288 | 33 | 1 |

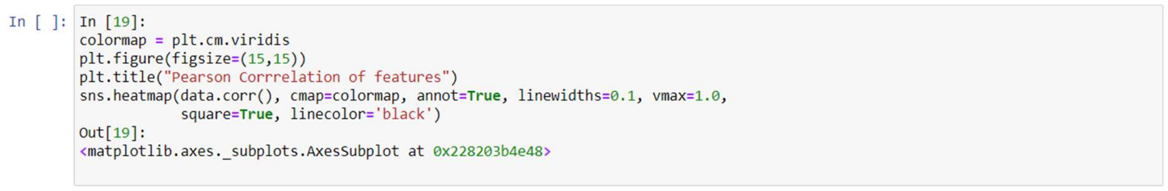
Now we're checking the dimensions of the dataset.



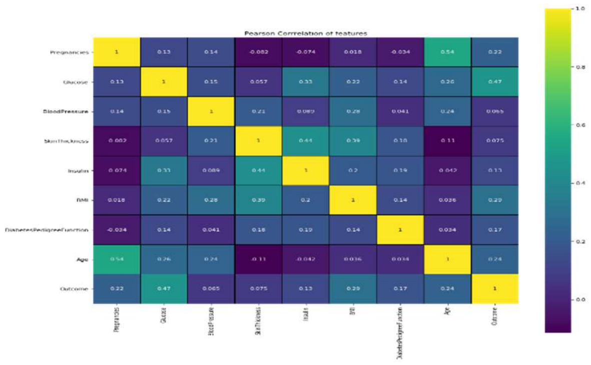
```
In [ ]: import seaborn as sns

        sns.countplot(data['Outcome'], label="Count")
        Out[14]:
        <matplotlib.axes._subplots.AxesSubplot at 0x22820236470>
```



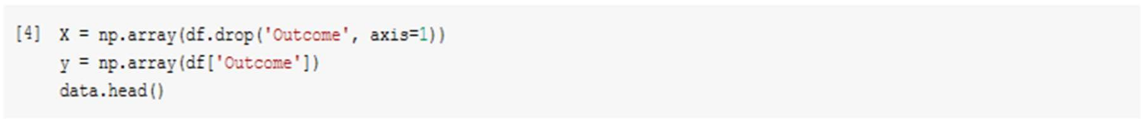
Here we see the heat map of the related dataset for SVM as given below:-



```
In [ ]: In [19]:
        colormap = plt.cm.viridis
        plt.figure(figsize=(15,15))
        plt.title("Pearson Corrrelation of features")
        sns.heatmap(data.corr(), cmap=colormap, annot=True, linewidths=0.1, vmax=1.0,
                    square=True, linecolor='black')
        Out[19]:
        <matplotlib.axes._subplots.AxesSubplot at 0x228203b4e48>
```

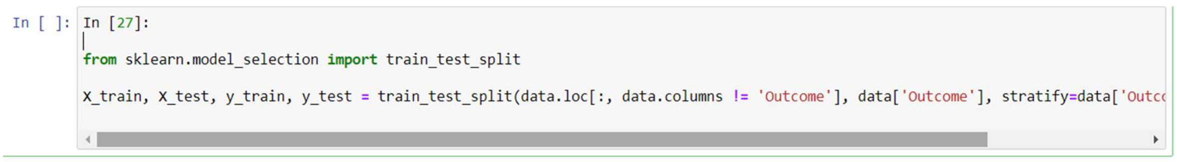


Now according to that we have to split the x and y data set



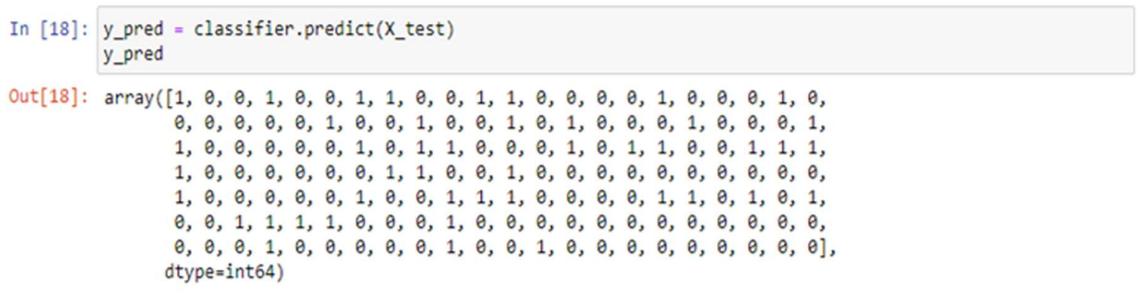
```
[4]  X = np.array(df.drop('Outcome', axis=1))
     y = np.array(df['Outcome'])
     data.head()
```

Now we process the data accordingly to the training set & testing set model selection for applying data set in SVM



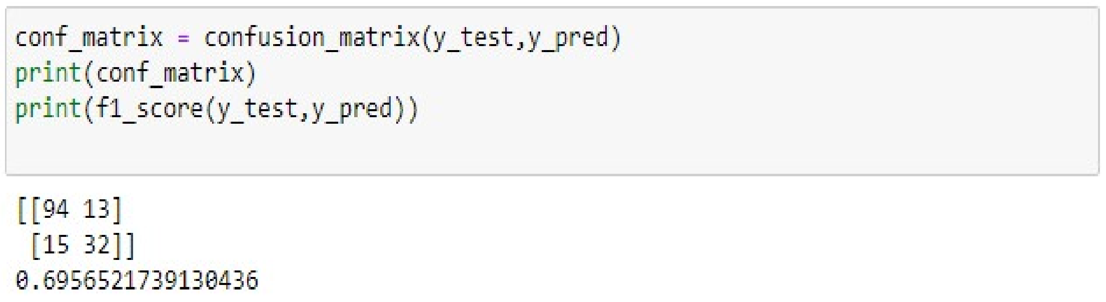
```
In [ ]: In [27]:

        from sklearn.model_selection import train_test_split

        X_train, X_test, y_train, y_test = train_test_split(data.loc[:, data.columns != 'Outcome'], data['Outcome'], stratify=data['Outc
```

Now to understand model performance, dividing the dataset into a training set and a test set is a good strategy. The training part trains the machine with the existing values in the dataset and the testing part is required to check the accuracy after the machine learns from the training part.



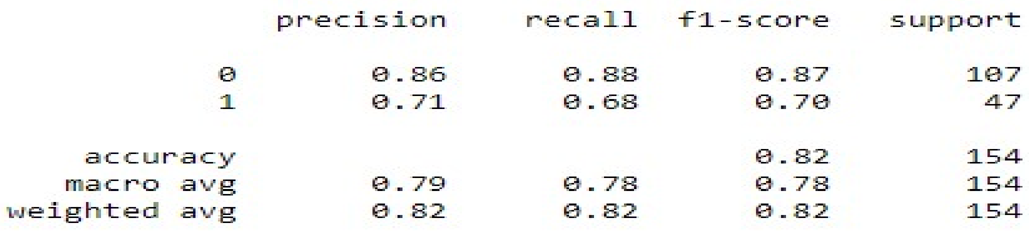
```
In [18]: y_pred = classifier.predict(X_test)
         y_pred

Out[18]: array([1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0,
                0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1,
                1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1,
                1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1,
                0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
               dtype=int64)
```

After using the technique the resultant of the confusion metrics



```
conf_matrix = confusion_matrix(y_test,y_pred)
print(conf_matrix)
print(f1_score(y_test,y_pred))

[[94 13]
 [15 32]]
0.6956521739130436
```

Here is the classification report of SVM techniques with the measures such as-precision, recall, f1-scope and accuracy.



```
              precision    recall  f1-score   support

           0       0.86      0.88      0.87       107
           1       0.71      0.68      0.70        47

    accuracy                           0.82       154
   macro avg       0.79      0.78      0.78       154
weighted avg       0.82      0.82      0.82       154
```

**Classification report**

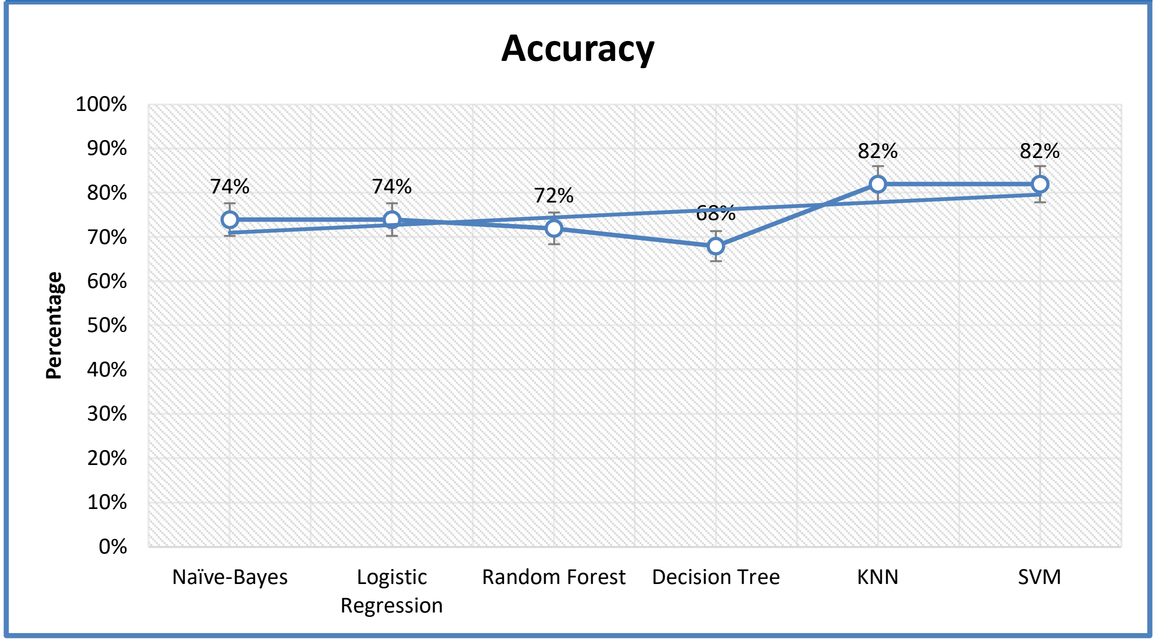
For this SVM classification technique we’re getting an accuracy of **82%**.

# 6. CONCLUSION

The main contribution of this study is to compare the performance of various machine learning classification techniques and evaluated their performance using various technique.

In general, multiple machine learning classifiers should perform better than a single machine learning classifier. We have performed the study on six machine learning classifiers and experimental results show that the highest classification accuracy amongst them is **82%** which we've got from SVM and KNN classification technique.

| **Classification Techniques** | **Accuracy** |
|---|---|
| Naïve-Bayes | 74% |
| Logistic Regression | 74% |
| Random Forest | 72% |
| Decision Tree | 68% |
| KNN | 82% |
| SVM | 82% |



In addition, this application is able to classify the patients based on their diabetes level by collecting real time data from various Health Care Services.

Hence, this study can be a promising tool to aid the stratification of diabetes patients.

# 7. REFERENCES

[1]NaiArun,N.,Moungmai,R.,2015.ComparisonofClassifiersfortheRiskofDiabetesPrediction.procediaComputerScience69,132–142.doi:10.1016/j.procs.2015.10.014.
[2] Harry Zhang, "The Optimality of Naïve Bayes", Faculty of Computer Science at University of New Brunswick
[3] Hina S, Shaikh A, Sattar SA. Analyzing diabetes datasets using data mining. J Basic Appl Sci. 2017;13:466–71.
[4] Prof. Pramila M. Chawan, Tejas N. Joshi. Logistic Regression And Svm Based Diabetes Prediction System. International Journal For Technological Research In Engineering Volume 5, Issue 11, July-2018
[5] Naveen Kishore G, V.Rajesh, A.Vamsi Akki Reddy, K.Sumedh, T.Rajesh Sai Reddy."Prediction Of Diabetes Using Machine Learning Classification Algorithms", International Journal Of Scientific & Technology Research Volume 9, Issue 01, January 2020;ISSN 2277-8616
[6] Mani Butwall, Shraddha Kumar, "A Data Mining Approach for the Diagnosis of Diabetes Mellitus using Random Forest Classifier", International Journal of Computer Applications (0975 – 8887) Volume 120 – No.8, June 2015

[7] Bhatt K., Dalal P., Panwar A., "A Cluster Centres Initialization Method for Clustering Categorical Data Using Genetic Algorithm" International Journal of Digital Application & Contemporary research, 2013, Volume-2 Issue-1

[8] Huang, Zhexue. "A Fast Clustering Algorithm to Cluster Very Large Categorical Data Sets in Data Mining." DMKD. 1997.

[9] Quan Zou, Kaiyang Qu, Yamei Luo, Dehui Yin, Ying Ju, and Hua Tang, "Predicting Diabetes Mellitus With Machine Learning Techniques" Published online 2018 Nov 6. doi: 10.3389/fgene.2018.00515

[10] Anirudh Hebbar P, Manoj Kumar M V, Sanjay H A, "DRAP: Decision Tree and Random Forest Based Classification model to Predict Diabetes" Available at:- https://www.researchgate.net/publication/339175938

[11] Asha Gowda Karegowda, Punya V M.A.Jayaram, A.S .Manjunath,." Rule based Classification for Diabetic Patients using Decision Tree C4.5 ", International Journal of Computer Applications (0975 – 8887) Volume 45– No.12, May 2012

[12] B.M Patil, R.C Joshi, Durga Tosniwal, Hybrid Prediction model for Type-2 Diabetic Patients, Expert System with Applications, 37, 8102-8108 (2010).

[13] Asha Gowda Karegowda, MA.Jayaram, Integrating Decision Tree and ANN for Categorization of Diabetics Data , International Conference on Computer Aided Engineering, December 13-15, IIT Madras, Chennai, India (2007).

[14] Anirudh Hebbar P, Manoj Kumar M V, Sanjay H A,." DRAP: Decision Tree Based Classification model to Predict Diabetes", at:https://www.researchgate.net/publication/339175938

[15] International Journal of Computer Science Trends and Technology (IJCST)

[16] A. G. Karegowda, M. A. Jayaram, and A. S. Manjunath, ―Cascading K-means clustering and K-Nearest Neighbor classifier for categorization of diabetic patients,‖ International Journal of Engineering and Advanced Technology, 1(3):2249– 8958, 2012.

[17] Y. Angeline Christobel, P. Sivaprakasam : A New Classwise k Nearest Neighbor (CKNN) Method for the Classification of Diabetes Dataset, International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, Volume-2 Issue-3, February 2013

[18] Lijun Cheng, Yongsheng Ding , SVM and statistical technique method applying in Primary Open Angle Glaucoma diagnosis, Intelligent Control and Automation (WCICA), 2010 8th World Congress, pp- 2973 – 2978

[19] Shashikant Ghumbre, Chetan Patil, and Ashok Ghatol,(2011) , Heart Disease Diagnosis using Support Vector Machine , International Conference on Computer Science and Information Technology (ICCSIT'2011) Pattaya Dec. 2011 ,pp 84-88.,

[20] Xiao-Peng Zhang*, Zhi-Long Wang, Lei Tang, Ying-Shi Sun, Kun Cao and Yun Gao, (2011) Support vector machine model for diagnosis of lymph node metastasis in gastric cancer with multidetector computed tomography: a preliminary study URL: http://www.biomedcentral.com/1471-2407/11/10