Internship Report

(Project Work)

On

DIABETES PREDICTION USING MACHINE LARNING

Submitted to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY ANANTAPUR, ANANTHAPURAMU

In Partial Fulfillment of the Requirements for the Award of the Degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE & TECHNOLOGY

Submitted By

THONDAMALLA SNEHA - (21695A2804)

Under the Guidance of

Dr. K. Sree Divya., Ph. D

Assistant Professor

Department of Computer Science & Technology

MADANAPALLE INSTITUTE OF TECHNOLGY & SCIENCE

(UGC – AUTONOMOUS)

(Affiliated to JNTUA, Ananthapuramu)

Accredited by NBA, Approved by AICTE, New Delhi)

AN ISO 9001:2008 Certified Institution

P. B. No: 14, Angallu, Madanapalle, Annamayya – 517325

MADANAPALLEINSTITUTEOF TECHNOLOGY &SCIENCE
(UGC-AUTONOMOUS INSTITUTION)
Affiliated to JNTUA, Ananthapuramu & Approved by AICTE,NewDelhi
NAAC Accredited with A+ Grade, NIRF India Rankings 2021-Band: 201-250 (Engg.)
NBA Accredited -B.Tech. (CIVIL, CSE, ECE, EEE, MECH), MBA & MCA

DEPARTMENT OF COMPUTER SCIENCE & TECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that the **SUMMER INTERNSHIP-1 (20CST701)** entitled **"DIABETES PREDICTION"** is a bonafide work carried out by

THONDAMALLA SNEHA - (21695A2804)

Submitted in partial fulfillment of the requirements for the award of degree **Bachelor of Technology** in the stream of **Computer Science & Technology** in **Madanapalle Institute of Technology & Science, Madanapalle,** affiliated to **Jawaharlal Nehru Technological University Anantapur, Ananthapuramu** during the academic year 2022-2023

Guide
Dr. K. Sree Divya., Ph.D
Assistant Professor,
Department of CST

Head of the Department
Dr.M. Sreedevi ., Ph.D
Professor and Head
Department of CST

INTERNSHIP CERTIFICATE

CERTIFICATE NO: PS-APSSDC-MLNTERN-1044

CERTIFICATE

OF INTERNSHIP

THIS IS TO CERTIFY THAT

Skill AP APSSDC

Pantech e Learning
DIGITAL LEARNING SIMPLIFIED

PANTECH SOLUTIONS
Technology Beyond The Dreams

NAME Ms.THONDAMALLA SNEHA

ORGANISATION MADANAPALLE INSTITUTE OF TECHNOLOGY AND SCIENCE

has successfully completed

30 DAYS INTERNSHIP ON MACHINE LEARNING

at Pantech E Learning Pvt Ltd, Chennai

FROM 30.05.2022 TO 28.06.2022

M.Malaiyappan
Director
Pantech e Learning

Dr.Ravi Gujjala
Chief General Manager (Technical)
APSSDC

DECLARATION

We hereby declare that the results embodied in this **SUMMER INTERNSHIP-1 (20CST701) "Diabetes Prediction Using Machine Learning"** by me under the guidance of **Dr. K. Sree Divya, Ph.D. AssistantProfessor, Dept. of CST** in partial fulfillment of the award of **Bachelor of Technology** in **Computer Science & Technology** from **Jawaharlal Nehru Technological University Anantapur, Ananthapuramu** and I have not submitted the same to any other University/institute for award of any other degree.

Date : 09-JAN-2023

Place : MADANAPALLE

THONDAMALLA SNEHA
(21695A2804)

I certify that above statement made by the students is correct to the best of my knowledge.

Date : 09-01-2023 **Guide** Dr. K. Sree Divya., Ph.D.

TABLE OF CONTENT

ABSTRACT

Diabetes is an illness caused because of high glucose level in a human body. Diabetes should not be ignored if it is untreated then Diabetes may cause some major issues in a person like: heart related problems, kidney problem, blood pressure, eye damage and it can also affect other organs of human body. Diabetes can be controlled if it is predicted earlier. To achieve this goal this project work we will do early prediction of Diabetes in a human body or a patient for a higher accuracy through applying, Various Machine Learning Techniques. Machine learning techniques Provide better result for prediction by constructing models from datasets collected from patients. The current work uses the Machine Learning techniques, like Classification and ensemble methods on the dataset, gathered from Minister of Health department, to predict diabetic patients. Classification techniques are Logistic Regression (LR), Decision Tree (DT), Support Vector Machine (SVM). Ensembling techniques are Gradient Boosting (GB) and Random Forest (RF). The accuracy is different for every model when compared to other models. The result shows that Random Forest algorithm achieves high accuracy for the selected data set when compared with other machine learning techniques.

List of Figures

LIST OF ABBREVIATIONS

SVM	Support Vector Machines
RF	Random Forest
CSV	Comma Separated Value

CHAPTER 1

INTRODUCTION

1.1 About Industry or Organization Details:

Pantech Solutions Private Limited is an unlisted private company incorporated on 04 August, 2005. It is classified as a private limited company and is located in Chennai 600017, Tamil Nadu. It's authorized share capital is INR 5.00 lac and the total paid-up capital is INR 1.00 lac.The current status of Pantech Solutions Private Limited is- Active.

The last reported AGM (Annual General Meeting) of Pantech Solutions Private Limited, per our records, was held on 29 September, 2017. Also, as per our records, its last balance sheet was prepared for the period ending on 31 March, 2017.

Pantech Solutions Private Limited has two directors - Malaiyappan Maruthamuthu and Srinivasan Natrajan.

The Corporate Identification Number (CIN) of Pantech Solutions Private Limited is U72200TN2005PTC057109. The registered office of Pantech Solutions Private Limited is at 3/2, RAMACHANDRAN STREETNORTH USMAN ROAD, T NAGAR, CHENNAI 600017, Tamil Nadu.

1.2 My Personal Benefits:

This internship provides you with firsthand experience, professional opportunities and personal growth.

It will also make you more competitive when applying for jobs. As an intern, you'll gain relevant skills to showcase on your resume.

These are the My personal Benefits:

1. Gain valuable work experience.
2. Explore a career path.
3. Network with professionals in the field.
4. Gain confidence.
5. The organization certified me with an internship completed certificate.

1.3 Objective of the Project:

The goal of this project is to monitor the personal health of a person in the aspect of diabetes

The project is the process of predicting the probability of a person to get the diabetes

It warns the person to be away and take necessary practices to be healthy

Identification of the dominant characteristics that may lead to diabetic complications using feature selection methods.

Implementation and evaluation of traditional and ensemble machine learning models to predict eight complications in diabetic patients by utilizing a comprehensive UAE based dataset

1.4 Limitations of Project:

1. As we know that the system will work based on the training data set, so that we have to give large size of data to get the 100% of accuracy.
2. Cost of the sensors are going to be used is also very high.
3. Proper maintenance is needed for every limited period of time.
4. The sensors are not available at all places so it can’t Reach all the areas

CHAPTER 2
SYSTEM ANALYSIS

2.1 Introduction:

1.we use the Pima Indian Diabetes Dataset; we apply various Machine Learning classification and ensemble Techniques to predict diabetes.

2. Machine Learning Is a method that is used to train computers or machines explicitly.

3. we apply popular classification and ensemble methods on dataset for prediction.

4. Implementation and evaluation of traditional and ensemble machine learning models to predict eight complications in diabetic patients by utilizing a comprehensive UAE based dataset.

2.2 Existing System:

1. The existing systems going to use Manual blood samples from the patients every time for checkup

2. It need large amount of time to get more and more accuracy

3. They are following more time to give the results

4. It uses complex procedures

2.3 Disadvantages of Existing System:

1. As we discussed above the cost of the algorithms are high.
2. Maximum datasets are needed.
3. Maintenance is Needed.
4. Consumes more compiler time
5. The late by the people affects the health of patients who are waiting for the results

2.4 Proposed Systems

1. The proposed System is based on “SUPPORT VECTOR SYSTEM”

2. The proposed System Don’t need any manual blood Samples from the patients

3. It works based on the Previous health progress and health analytics of particular patients

4. Svm creates a hyperplane that separate two classes. It can create a hyperplane or set of hyperplanes in high dimensional space.

5. This hyper plane can be used for classification or regression also.

6. Svm differentiates instances in specific classes and can also classify the entities which are not sup- ported by data

7. Separation is done by through hyperplane performs the separation to the closest training point of any class.

2.5 Advantages over Existing System:

1. It is based on the previous health reports of the patient
2.It can’t of depends on the blood samples of patient every time
3.It takes less computational time
4.It Don’t follow more complex procedures
5.It provide better accuracy than previously existing systems with less effort

CHAPTER 3

SYSTEM SPECIFICATION

3.1 Hardware Requirement Specification:

1.Operating systems: Windows * 7 or later and ubuntu

2.processor: Intel core i3 or later

3.Disk space: 4 to 8 GB

3.2 Software Requirement Specification:

1. Programing Language: Python

2. Database: Microsoft Access

3. Operating System: Windows

About Machine Learning

What is Machine Learning?

The area of Machine Learning deals with the design of programs that can learn rules from data, adapt to changes, and improve performance with experience. In addition to being one of the initial dreams of Computer Science, Machine Learning has become crucial as computers are expected to solve increasingly complex problems and become more integrated into our daily lives. Writing a computer program is a bit like writing down instructions for an extremely literal child who just happens to be millions of times faster than you. Yet many of the problems we now want computers to solve are no longer tasks we know how to explicitly tell a computer how to do. These include identifying faces in images, autonomous driving in the desert, finding relevant documents in a database (or throwing out irrelevant ones, such as spam email), finding patterns in large volumes of scientific data, and adjusting internal parameters of systems to optimize performance. That is, we may ourselves be good at identifying people in photographs, but we do not know how to directly tell a computer how to do it. Instead, methods that take labeled training data (images labeled by who is in them, or email messages labeled by whether or not they are spam) and then learn appropriate rules from the data, seem to be the best approaches to solving these problems. Furthermore, we need systems that can adapt to changing conditions, that can be user-friendly by adapting to needs of their individual users, and that can improve performance over time. I would like to thank Dick Karp, Neha Dave, Michael Kearns, and Tom Mitchell for

comments and suggestions on earlier drafts of this essay.

I)Supervised Learning

II) Unsupervised Learning

SUPERVISED LEARNING:

The supervised learning can be achieved by a model if the data provided is labelled.Classification and regression problems can be solved by using supervised learning. • Classification: In this, we need to categorize the given set of data into classes. Examples: Logistic Regression, Classification trees, Support Vector Machines, Random Forests, Artificial Neural Networks etc. • Regression: In this, we analyse the effect of the independent variable on dependent variables. Examples: Linear Regression, Decision Trees, Bayesian Networks, Fuzzy Classification etc. own confidence or that can optimize multiple objectives. One would like models that capture

UNSUPERVISED LEARNING

The unsupervised learning can be achieved by a model if the data provided is unlabelled.Clustering and Dimension reduction problems can be solved by using unsupervised learning. • Clustering: In this, we take the unlabelled data and understand it's features and group them accordingly.Examples: K-means Clustering, Hierarchal Clustering, Gaussian Mixture models,Genetic Algorithms etc. • Dimension Reduction: This is the process of reducing the number of random variables under consideration to make the classification simple. Examples: Principal Component Analysis, Tensor Decomposition, Multi dimension Statistics, Random Projections etc.

What is Classification:

In machine learning, Classification, as the name suggests, classifies data into different parts/classes/groups. It is used to predict from which dataset the input data belongs to.For example, if we are taking a dataset of scores of a cricketer in the past few matches, along with average, strike rate, not outs etc, we can classify him as "in form" or "out of form". Classification is the process of assigning new input variables (X) to the class they most likely belong to, based

on a classification model, as constructed from previously labeled training data. Data with labels is used to train a classifier such that it can perform well on data without labels (not yet labeled). This process of continuous classification, of previously known classes, trains a machine. If the classes are discrete, it can be difficult to perform classification tasks

Types of Classification

There are two types of classifications;

• Binary classification

• Multi-class classification

Binary classification

It is a process or task of classification, in which a given data is being classified into two classes. It's basically a kind of prediction about which of two groups the thing belongs to. Let us suppose, two emails are sent to you, one is sent by an insurance company that keeps sending their ads, and the other is from your bank regarding your credit card bill. The email service provider will classify the two emails, the first one will be sent to the spam folder and the second one will be kept in the primary one. This process is known as binary classification, as there are two discrete classes, one is spam and the other is primary. So, this is a problem of binary classification.Binary classification uses some algorithms to do the task, some of the most common algorithms used by binary classification are

• Logistic Regression

• k-Nearest Neighbors

• Decision Trees

• Support Vector Machine

• Naive Bayes

Multiclass Classification:

Multi-class classification is the task of classifying elements into different classes. Unlike binary, it doesn't restrict itself to any number of classes.

Examples of multi-class classification are

• classification of news in different categories,

• classifying books according to the subject,

• classifying students according to their streams etc.

In these, there are different classes for the response variable to be classified in and thus according to the name, it is a Multi-class classification.

Can a classification possess both binary or multi-class?

Let us suppose we have to do sentiment analysis of a person, if the classes are just "positive" and "negative", then it will be a problem of binary class. But if the classes are "sadness", happiness", "disgusting", "depressed", then it will be called a problem of Multi-class classification.

Logistic Regression:

o Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables. o Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it

gives the probabilistic values which lie between 0 and 1. o Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas Logistic regression is used for solving the classification problems. o In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1). o The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc. o Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets

Type of Logistic Regression:

Assumptions for Logistic Regression:

The dependent variable must be categorical in nature. o The independent variable should not have multi-collinearity. On the basis of the categories, Logistic Regression can be classified into three types: o Binomial: In binomial Logistic regression, there can be only two possible types of the dependent variables, such as 0 or 1, Pass or Fail, etc. o Multinomial: In multinomial Logistic regression, there can be 3 or more possible unordered types of the dependent variable, such as "cat", "dogs", or "sheep" o Ordinal: In ordinal Logistic regression, there can be 3 or more possible ordered types of dependent variables, such as "low", "Medium", or "High"

K-Nearest Neighbor(KNN) Algorithm for Machine Learning:

o K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique. o K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories. o K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm. o K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.

o K-NN is a non-parametric algorithm, which means it does not make any assumption on

underlying data. o It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset. o KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data

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

Decision Tree Classification Algorithm

o Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

o In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

o The decisions or the test are performed on the basis of features of the given dataset.

o It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.

o It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure. o In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm. o A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.

PYTHON MODULES:

A module allows you to logically organize your Python code. Grouping related code into a module makes the code easier to understand and use. A module is a Python object with arbitrarily named attributes that you can bind and reference.

Simply, a module is a file consisting of Python code. A module can define functions, classes and variables. A module can also include runnable code.

There are actually three different ways to define a **module** in Python:

A module can be written in Python itself.

A module can be written in C and loaded dynamically at run-time, like the **re** (regular expression) module.

A **built-in** module is intrinsically contained in the interpreter, like the itertools module.

The *import* statement:

We can use any Python source file as a module by executing an import statement in some other Python source file. When interpreter encounters an import statement, it imports the module if the module is present in the search path. A search path is a list of directories that the interpreter searches for importing a module.

Libraries Required:

1.numpy

2. pandas

3. NumPy

4.seaborn

NUMPY:

NumPy is a general-purpose array-processing package. It provides a high performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python.

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely. NumPy stands for Numerical Python.

It contains various features including these important ones:

A powerful N-dimensional array object.

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 multidimensional container of generic data. Arbitrary data-types can be defined using NumPy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

Installation of NumPy:

pip install numpy

PANDAS:

Pandas is an open source library that allows to you perform data manipulation in Python. Pandas library is built on top of Numpy, meaning Pandas needs Numpy to operate. Pandas provide an easy way to create, manipulate and wrangle the data. Pandas is also an elegant solution for time series data.

Pandas is a Python library used for working with data sets. It has functions for analyzing, cleaning,

exploring, and manipulating data. The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008

The two primary data structures of pandas, Series (1-dimensional) and Data Frame (2dimensional), handle the vast majority of typical use cases in finance, statistics, social science, and many areas of engineering.

Advantages of Pandas:

Easily handles missing data

It uses Series for one-dimensional data structure and Data Frame for multi- dimensional data structure

It provides an efficient way to slice the data

It provides a flexible way to merge, concatenate or reshape the data

In a nutshell, Pandas is a useful library in data analysis. It can be used to perform data manipulation and analysis. Pandas provide powerful and easy-to-use data structures, as well as the means to quickly perform operations on these structures.

Installation of Pandas:

anaconda install pandas (or)
pip install pandas

MATPLOTLIB:

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002.

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible. Create publication quality plots. Make interactive figures that can zoom, pan, update

One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.

Installation of Matplotlib:

python -mpip install -U matplotlib

SEABORN:

Seaborn is an amazing visualization library for statistical graphics plotting in Python. It provides beautiful default styles and colour 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 aims to make visualization the central part of exploring and understanding data. It provides dataset-oriented APIs, so that we can switch between different visual representations for same variables for better understanding of dataset.

Plots are basically used for visualizing the relationship between variables. Those variables can be either be completely numerical or a category like a group, class or division. Seaborn divides plot into the below categories –

Relational plots: This plot is used to understand the relation between two variables.

Categorical plots: This plot deals with categorical variables and how they can be visualized.

Distribution plots: This plot is used for examining univariate and bivariate distributions

Regression plots: The regression plots in seaborn are primarily intended to add a visual guide that helps to emphasize patterns in a dataset during exploratory data analyses.

Matrix plots: A matrix plot is an array of scatterplots.

Multi-plot grids: It is used draw multiple instances of the same plot on

CHAPTER 4

SYSTEM DESIGN

4.1 System Architecture:

Steps for training a classifier for Diabetes prediction:

Firstly, data have to be prepared in order to obtain a data set namely, the training set, by means of pre-processing and feature selection methods.

Then, such a data set is involved in the learning step, which uses ML algorithms and yields a trained classifier. Finally, the classifier has to be tested on a different data set namely, the test set.

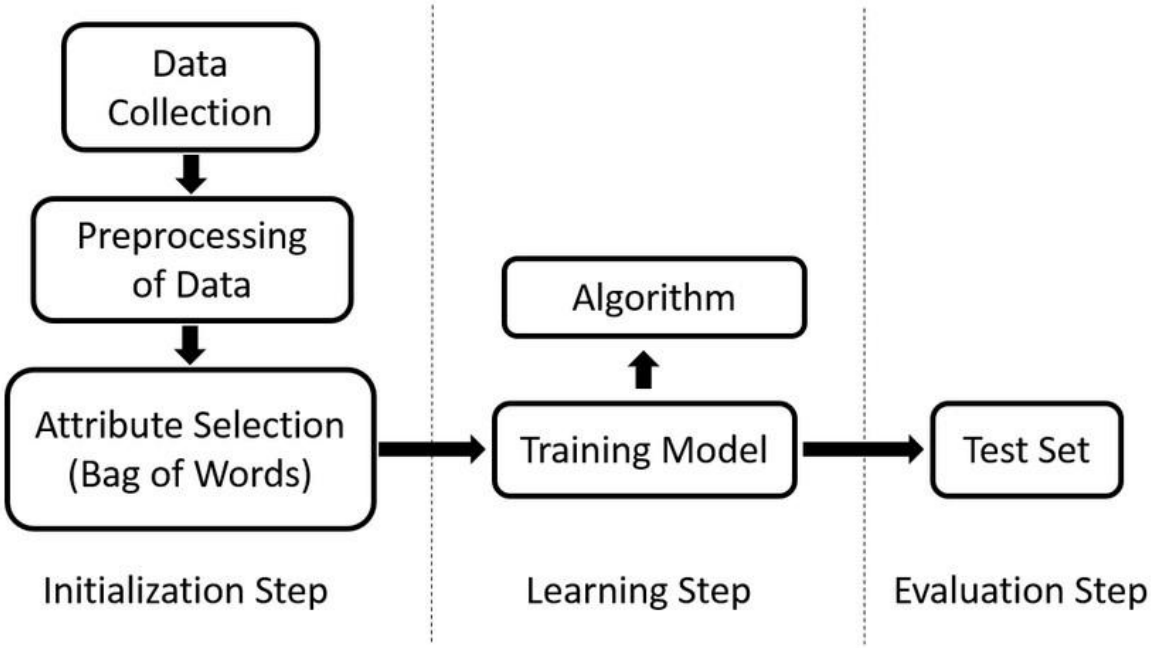
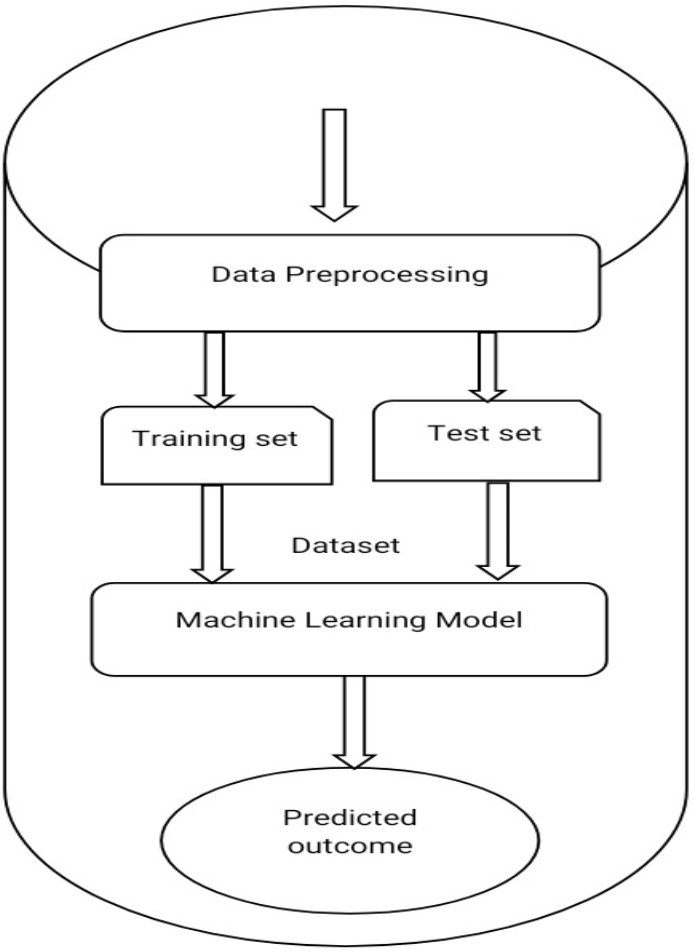


Fig.no: 1 System Architecture

4.1 Block diagram:



4.1 System Flow Diagram:

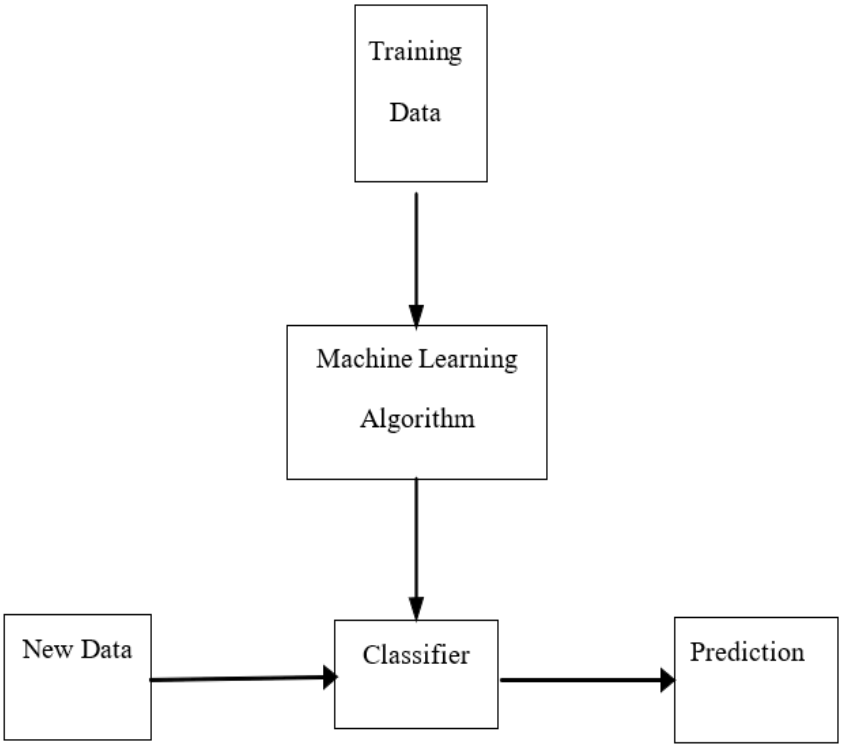


Figure 5. System Flow diagram

CHAPTER 5

IMPLEMENTATION AND RESULTS

INTRODUCTION:

We experimented with different classification and ensemble algorithms to predict diabetes.

In the following, we briefly discuss the phases

5.1. Dataset Description:

1. The data is gathered from UCI repository which is named as Pima Indian Diabetes Dataset. The dataset has many attributes of 768 patients.

2. The 9th attribute is class variable of each data points. This class variable shows the outcome 0 and 1 for diabetics which indicates positive or negative for diabetics

3. Distribution of Diabetic patient

We made a model to predict diabetes however the dataset was slightly imbalanced having around 500 classes labeled as 0 means negative means no diabetes and 268 labeled as 1 means positive means diabetic

```
diabetes_df.info()
```

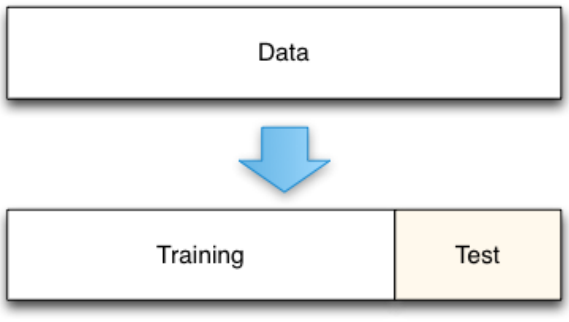
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

5.2 Data Preprocessing

Data preprocessing is most important process. Mostly healthcare related data contains missing vale and other impurities that can cause effective- ness of data. To improve quality and effectiveness obtained after mining process, Data preprocessing is done. To use Machine Learning Techniques on the dataset effectively this process is essential for accurate result and successful prediction. For Pima Indian diabetes dataset, we need to perform preprocessing in two steps

1. Missing Values removal- Remove all the instances that have zero (0) as worth. Having zero as worth is not possible. Therefore, this instance is eliminated. Through eliminating irrelevant features/instances we make feature subset and this process is called features subset selection, which reduces dimensionality of data and help to work

2. Splitting of data- After cleaning the data, data is normalized in training and testing the model. When data is spitted then we train algorithm on the training data set and keep test data set aside. This training process will produce the training model based on logic and algorithms and values of the feature in training data. Basically, aim of normalization is to bring all the attributes under same scale.



5.3 SPLITTING OF DATA:

The Processed data is splitted into the training set and testing set based on the size of the dataset

5.4. Apply Machine Learning:

When data has been ready, we apply Machine Learning Technique. We use different classification and ensemble techniques, to predict diabetes. The methods applied on Pima Indians diabetes dataset.

Main objective to apply Machine Learning Techniques to analyze the performance of these methods and find accuracy of them, and also been able to figure out the responsible/important feature which play a major role in prediction. The Techniques are follows

5.4.1. Support Vector Machine

Support Vector Machine also known as svm is a supervised machine learning algorithm.

Svm is most popular classification technique. Svm creates a hyperplane that separate two classes. It can create a hyperplane or set of hyperplane in high dimensional space.

This hyper plane can be used for classification or regression also.

Svm differentiates instances in specific classes and can also classify the entities which are not supported by data. Separation is done by through hyperplane performs the separation to the closest training point of any class.

Algorithm:

- Select the hyper plane which divides the class better.
- To find the better hyper plane you have to calculate the distance between the planes and the data which is called Margin
- If the distance between the classes is low then the chance of miss conception is high and vice versa.
- Select the class which has the high margin.
- Margin = distance to positive point + Distance to negative point

5.4.2. Decision Tree:

Decision tree is a basic classification method. It is supervised learning method. Decision tree used when response variable is categorical. Decision tree has tree like structure-based model which describes classification process based on input feature. Input variables are any types like graph, text, discrete, continuous etc.

Steps for Decision Tree Algorithm:

Construct tree with nodes as input feature.

Select feature to predict the output from input feature whose information gain is highest.

The highest information gain is calculated for each attribute in each node of tree.

Repeat step 2 to form a subtree using the feature which is not used in above node.

5.4.3. Random Forest-

It is type of ensemble learning method and also used for classification and regression tasks. The accuracy it gives is grater then compared to other models. This method can easily handle large datasets. Random Forest is developed by Leo Bremen. It is popular ensemble Learning Method. Random Forest Improve Performance of Decision Tree by reducing variance. It operates by constructing a multitude of decision trees at training time and outputs the class that is the mode of the classes or classification or mean prediction (regression) of the individual trees.

Algorithm:

- The first step is to select the R features from the total features m where R<<M.
- Among the R features, the node using the best split point
- Split the node into sub nodes using the best split.
- Repeat a to c steps until l number of nodes has been reached
- Built forest by repeating steps a to do for a number of times to create n number of trees.
- The random forest finds the best split using the Gin-Index Cost Function which is given by the following
- The first step is to need the take a glance at choices and use the foundations of each indiscriminately created decision tree to predict the result and stores the anticipated outcome at intervals the target place. Secondly, calculate the votes for each predicted target and ultimately, admit the high voted predicted target as a result of the ultimate prediction from the random forest formula. Some of the options of Random Forest does correct predictions result for a spread of applications are offered

Implementation of code:

```
import NumPy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()

from pandas.plotting import scatter_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.metrics import classification_report
import warnings

warnings.filterwarnings('ignore')
%matplotlib inline
diabetes_df = pd.read_csv("Documents\\diabetes.csv")
diabetes_df.head()
diabetes_df.columns
diabetes_df.info()
diabetes_df.describe()
diabetes_df.describe().T
diabetes_df.isnull().head(10)
diabetes_df.isnull().sum()
diabetes_df_copy = diabetes_df.copy(deep = True)
diabetes_df_copy[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']] =
diabetes_df_copy[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']].repl
ace(0,np.NaN)


# Showing the Count of NANs
print(diabetes_df_copy.isnull().sum())
diabetes_df_copy['Glucose'].fillna(diabetes_df_copy['Glucose'].mean(), inplace =
True)
diabetes_df_copy['BloodPressure'].fillna(diabetes_df_copy['BloodPressure'].mea
n(), inplace = True)
```

```
diabetes_df_copy['SkinThickness'].fillna(diabetes_df_copy['SkinThickness'].med
ian(), inplace = True)
diabetes_df_copy['Insulin'].fillna(diabetes_df_copy['Insulin'].median(), inplace =
True)
diabetes_df_copy['BMI'].fillna(diabetes_df_copy['BMI'].median(), inplace =
True)
color_wheel = {1: "#0392cf", 2: "#7bc043"}
colors = diabetes_df["Outcome"].map(lambda x: color_wheel.get(x + 1))
print(diabetes_df.Outcome.value_counts())
p=diabetes_df.Outcome.value_counts().plot(kind="bar")
plt.subplot(121), sns.distplot(diabetes_df['Insulin'])
plt.subplot(122), diabetes_df['Insulin'].plot.box(figsize=(16,5))
plt.show()
plt.figure(figsize=(12,10))


# seaborn has an easy method to showcase heatmap
p = sns.heatmap(diabetes_df.corr(), annot=True,cmap ='RdYlGn')
sc_X = StandardScaler()
X =  pd.DataFrame(sc_X.fit_transform(diabetes_df_copy.drop(["Outcome"],axis
= 1),), columns=['Pregnancies',
'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI',
'DiabetesPedigreeFunction', 'Age'])
X.head()
y = diabetes_df_copy.Outcome
y
X = diabetes_df.drop('Outcome', axis=1)
y = diabetes_df['Outcome']
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.33,
                                    random_state=7)
from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier(n_estimators=200)
rfc.fit(X_train, y_train)
rfc_train = rfc.predict(X_train)
from sklearn import metrics

print("Accuracy_Score =", format(metrics.accuracy_score(y_train, rfc_train)))
from sklearn import metrics
```

```
predictions = rfc.predict(X_test)
print("Accuracy_Score =", format(metrics.accuracy_score(y_test, predictions)))
from sklearn.metrics import classification_report, confusion_matrix

print(confusion_matrix(y_test, predictions))
print(classification_report(y_test,predictions))
#decision tree
from sklearn.tree import DecisionTreeClassifier

dtree = DecisionTreeClassifier()
dtree.fit(X_train, y_train)
from sklearn import metrics

predictions = dtree.predict(X_test)
print("Accuracy Score =", format(metrics.accuracy_score(y_test,predictions)))
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, predictions))
print(classification_report(y_test,predictions))

#support vector machine
from sklearn.svm import SVC
svc_model = SVC()
svc_model.fit(X_train, y_train)
svc_pred = svc_model.predict(X_test)
from sklearn import metrics


print("Accuracy Score =", format(metrics.accuracy_score(y_test, svc_pred)))
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, svc_pred))
print(classification_report(y_test,svc_pred))
rfc.feature_importances_
(pd.Series(rfc.feature_importances_, index=X.columns).plot(kind='barh'))
import pickle

# Firstly we will be using the dump() function to save the model using pickle
saved_model = pickle.dumps(rfc)

# Then we will be loading that saved model
```

```
rfc_from_pickle = pickle.loads(saved_model)

# lastly, after loading that model we will use this to make predictions
rfc_from_pickle.predict(X_test)
rfc.predict([[0,137,40,35,168,43.1,2.228,33]]) #4th patient
```

Execution:

```
[27] import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns
     sns.set()
```

```
[28] from pandas.plotting import scatter_matrix
     from sklearn.preprocessing import StandardScaler
     from sklearn.model_selection import train_test_split
     from sklearn.neighbors import KNeighborsClassifier
     from sklearn.metrics import confusion_matrix
     from sklearn import metrics
     from sklearn.metrics import classification_report
     import warnings
     warnings.filterwarnings('ignore')
     %matplotlib inline
```

```
diabetes_df = pd.read_csv("diabetes.csv")
diabetes_df.head()
diabetes_df.columns
diabetes_df.info()
diabetes_df.describe()
diabetes_df.describe().T
diabetes_df.isnull().head(10)
diabetes_df.isnull().sum()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
```

check 0s completed

File Edit View Insert Runtime Tools Help Last saved at 2:12 PM

+ Code + Text Connect Editing

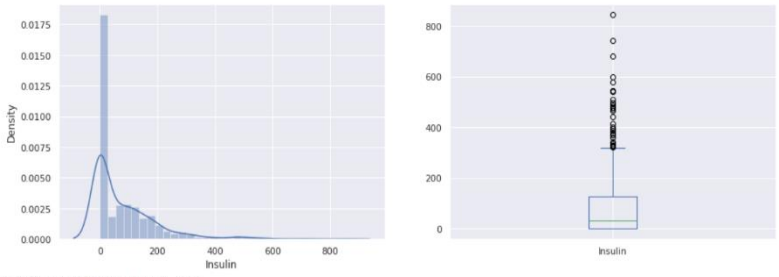
```
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

```
[ ] diabetes_df_copy = diabetes_df.copy(deep = True)
    diabetes_df_copy[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']] = diabetes_df_copy[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']].replace(0,np.NaN)
```

```
File  Edit  View  Insert  Runtime  Tools  Help  Last saved at 2:12 PM
+ Code  + Text                                                    Connect  Editing

Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64

[ ] diabetes_df_copy = diabetes_df.copy(deep = True)
    diabetes_df_copy[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']] = diabetes_df_copy[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']].replace(0,np.NaN)
```

```
[ ] print(diabetes_df_copy.isnull().sum())

Pregnancies                 0
Glucose                     5
BloodPressure              35
SkinThickness             227
Insulin                   374
BMI                        11
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64

[11] diabetes_df_copy['Glucose'].fillna(diabetes_df_copy['Glucose'].mean(), inplace = True)
     diabetes_df_copy['BloodPressure'].fillna(diabetes_df_copy['BloodPressure'].mean(), inplace = True)
     diabetes_df_copy['SkinThickness'].fillna(diabetes_df_copy['SkinThickness'].median(), inplace = True)
     diabetes_df_copy['Insulin'].fillna(diabetes_df_copy['Insulin'].median(), inplace = True)
     diabetes_df_copy['BMI'].fillna(diabetes_df_copy['BMI'].median(), inplace = True)
     color_wheel = {1: "#0392cf", 2: "#7bc043"}
     colors = diabetes_df["Outcome"].map(lambda x: color_wheel.get(x + 1))

     print(diabetes_df.Outcome.value_counts())

0    500
1    268
Name: Outcome, dtype: int64
```

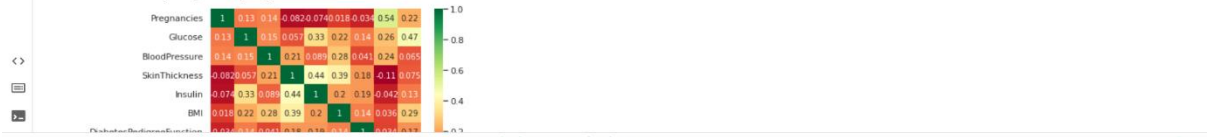
```
p=diabetes_df.Outcome.value_counts().plot(kind="bar")
plt.subplot(121), sns.distplot(diabetes_df['Insulin'])
plt.subplot(122), diabetes_df['Insulin'].plot.box(figsize=(16,5))
plt.show()
plt.figure(figsize=(12,10))
```



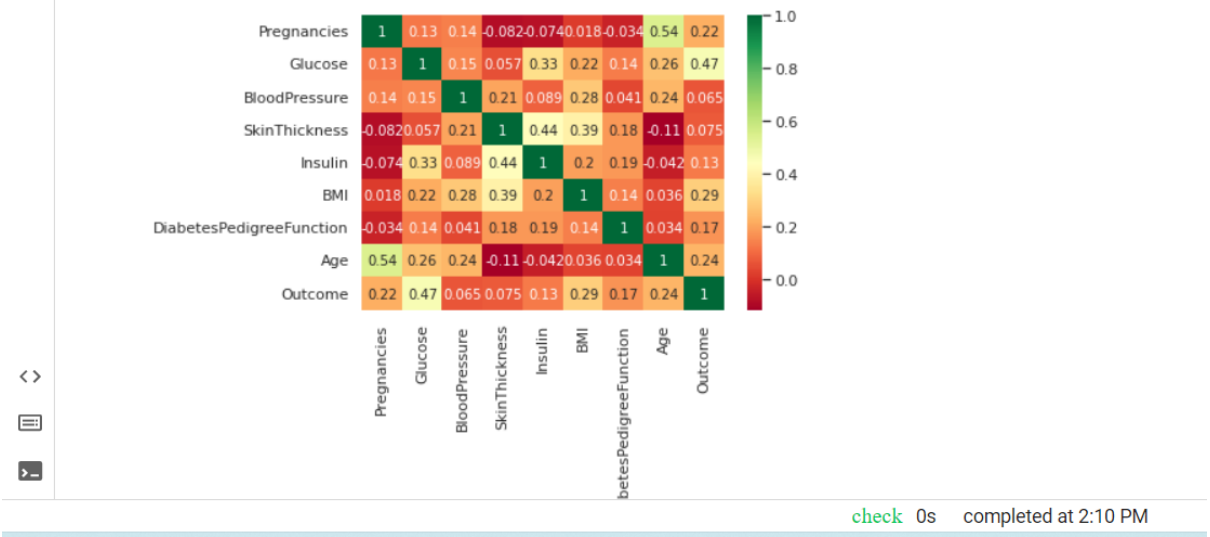

```
<Figure size 864x720 with 0 Axes>
<Figure size 864x720 with 0 Axes>

p = sns.heatmap(diabetes_df.corr(), annot=True,cmap ='RdYlGn')
sc_X = StandardScaler()
X =  pd.DataFrame(sc_X.fit_transform(diabetes_df_copy.drop(["Outcome"],axis = 1),), columns=['Pregnancies',

check  0s  completed at 2:10 PM
```

```
p = sns.heatmap(diabetes_df.corr(), annot=True,cmap ='RdYlGn')
sc_X = StandardScaler()
X =  pd.DataFrame(sc_X.fit_transform(diabetes_df_copy.drop(["Outcome"],axis = 1),), columns=['Pregnancies',
'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age'])
X.head()
y = diabetes_df_copy.Outcome
y
```

```
0      1
1      0
2      1
3      0
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 768, dtype: int64
```



check 0s completed at 2:10 PM



check 0s completed at 2:10 PM

```
[20] X = diabetes_df.drop('Outcome', axis=1)
     y = diabetes_df['Outcome']
     from sklearn.model_selection import train_test_split

     X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.33,
                                                         random_state=7)
     from sklearn.ensemble import RandomForestClassifier
     rfc = RandomForestClassifier(n_estimators=200)
     rfc.fit(X_train, y_train)
     rfc_train = rfc.predict(X_train)
     from sklearn import metrics
     print("Accuracy_Score =", format(metrics.accuracy_score(y_train, rfc_train)))
     from sklearn import metrics
```

```
Accuracy_Score = 1.0
```

```
predictions = rfc.predict(X_test)
print("Accuracy_Score =", format(metrics.accuracy_score(y_test, predictions)))
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, predictions))
print(classification_report(y_test,predictions))
```

```
Accuracy_Score = 0.7637795275590551
[[135  27]
 [ 33  59]]
              precision    recall  f1-score   support

           0       0.80      0.83      0.82       162
```

check 0s completed at 2:10 PM

```
predictions = rfc.predict(X_test)
print("Accuracy_Score =", format(metrics.accuracy_score(y_test, predictions)))
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, predictions))
print(classification_report(y_test,predictions))
```

```
Accuracy_Score = 0.7637795275590551
[[135  27]
 [ 33  59]]
              precision    recall  f1-score   support

           0       0.80      0.83      0.82       162
           1       0.69      0.64      0.66        92

    accuracy                           0.76       254
   macro avg       0.74      0.74      0.74       254
weighted avg       0.76      0.76      0.76       254
```

```
[22] from sklearn.tree import DecisionTreeClassifier
dtree = DecisionTreeClassifier()
dtree.fit(X_train, y_train)
from sklearn import metrics
predictions = dtree.predict(X_test)
print("Accuracy Score =", format(metrics.accuracy_score(y_test,predictions)))
```

check 0s completed at 2:10 PM

25°C Sunny | Search | ENG IN | 14:14 09-01-2023

```
print("Accuracy Score =", format(metrics.accuracy_score(y_test,predictions)))
from sklearn.metrics import classification_report, confusion_matrix
```

```
Accuracy Score = 0.7007874015748031
```

```
print(confusion_matrix(y_test, predictions))
print(classification_report(y_test,predictions))
#support vector machine
from sklearn.svm import SVC
```

```
[[127  35]
 [ 41  51]]
              precision    recall  f1-score   support

           0       0.76      0.78      0.77       162
           1       0.59      0.55      0.57        92

    accuracy                           0.70       254
   macro avg       0.67      0.67      0.67       254
weighted avg       0.70      0.70      0.70       254
```

+ Code + Text | RAM Disk | Editing

```
[23]
    accuracy                           0.70       254
   macro avg       0.67      0.67      0.67       254
weighted avg       0.70      0.70      0.70       254
```

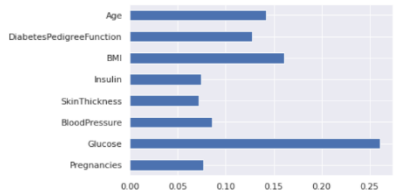
```
[24] svc_model = SVC()
svc_model.fit(X_train, y_train)
svc_pred = svc_model.predict(X_test)
from sklearn import metrics
print("Accuracy Score =", format(metrics.accuracy_score(y_test, svc_pred)))
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, svc_pred))
print(classification_report(y_test,svc_pred))
```

```
Accuracy Score = 0.7480314960629921
[[145  17]
 [ 47  45]]
              precision    recall  f1-score   support

           0       0.76      0.90      0.82       162
           1       0.73      0.49      0.58        92

    accuracy                           0.75       254
   macro avg       0.74      0.69      0.70       254
weighted avg       0.74      0.75      0.73       254
```

```
rfc.feature_importances_
(pd.Series(rfc.feature_importances_, index=X.columns).plot(kind='barh'))
import pickle
```

check 0s completed at 2:10 PM

```
[25] rfc.feature_importances_
     (pd.Series(rfc.feature_importances_, index=X.columns).plot(kind='barh'))
     import pickle
     saved_model = pickle.dumps(rfc)
     rfc_from_pickle = pickle.loads(saved_model)
```




```
rfc_from_pickle.predict(X_test)
rfc.predict([[0,137,40,35,168,43.1,2.228,33]])
```

```
array([1])
```

```
[ ]
```

CHAPTER 6
TESTING AND RESULTS

ACCURACY FOR RANDOM FOREST MODEL:

```
from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier(n_estimators=200)
rfc.fit(X_train, y_train)
```

```
RandomForestClassifier(n_estimators=200)
```

```
rfc_train = rfc.predict(X_train)
from sklearn import metrics

print("Accuracy_Score =", format(metrics.accuracy_score(y_train, rfc_train)))
```

```
Accuracy_Score = 1.0
```

```
from sklearn import metrics

predictions = rfc.predict(X_test)
print("Accuracy_Score =", format(metrics.accuracy_score(y_test, predictions)))
```

```
Accuracy_Score = 0.7598425196850394
```

ACCURACY FOR DECISION TREE MODEL

```
#decision tree
from sklearn.tree import DecisionTreeClassifier

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

```
DecisionTreeClassifier()
```

```
from sklearn import metrics

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

```
Accuracy Score = 0.7047244094488189
```

ACCURACY FOR SUPPORT VECTOR MACHINE MODEL

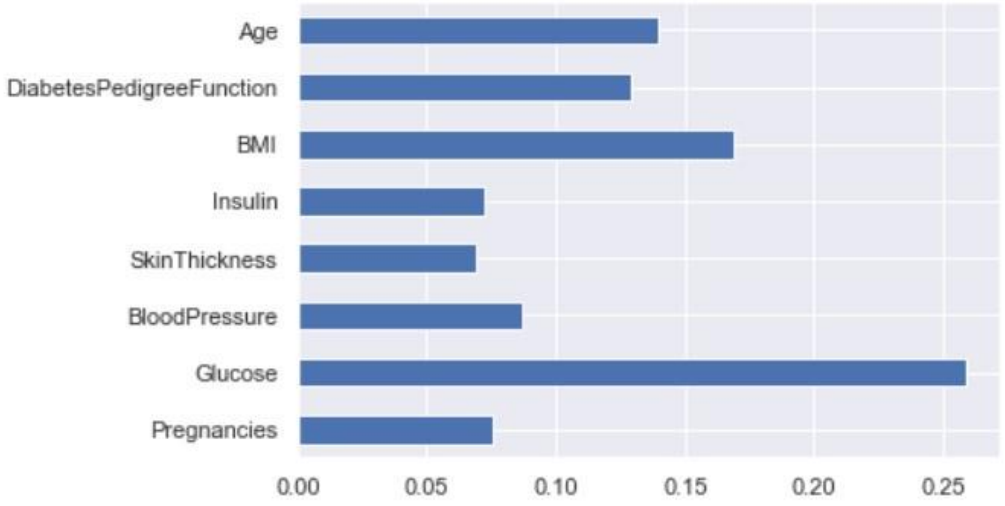
```
#support vector machine
from sklearn.svm import SVC

svc_model = SVC()
svc_model.fit(X_train, y_train)
```

```
SVC()
```

```
svc_pred = svc_model.predict(X_test)
```

```
from sklearn import metrics

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

```
Accuracy Score = 0.7480314960629921
```



4.By comparing all models on each other, RANDOM FOREST MODEL has more accuracy in predicting diabetes.

Classifier	Accuracy
Random forest	0.75984
Decision Tree	0.70472
Support Vector Machine	0.7480

Table 1: Evaluation of Models

TESTED OUTCOMES

```
rfc.predict([[0,137,40,35,168,43.1,2.228,33]]) #4th patient
```

```
array([1], dtype=int64)
```

```
rfc.predict([[10,101,76,48,180,32.9,0.171,63]])  # 763 th patient
```

```
array([0], dtype=int64)
```

CHAPTER 7

CONCLUSION

CONCLUSION AND FUTURE SCOPE

The main aim of this project was to design and implement Diabetes Prediction Using Machine Learning Methods and Performance Analysis of that methods and it has been achieved successfully. The proposed approach uses various classification and ensemble learning method in which SVM, Knn, Random Forest, Decision Tree, Logistic Regression and Gradient Boosting classifiers are used. And 77% classification accuracy has been achieved. The Experimental results can be asst health care to take early prediction and make early decision to cure diabetes and save humans life. In future, the designed system with the used machine learning classification algorithms can be used to predict or diagnose other diseases. The work can be extended and improved for the automation of diabetes analysis including some other machine learning algorithms

REFERENCES

[1] Debadri Dutta, Debpriyo Paul, Parthajeet Ghosh,
"Analyzing Feature Importances for Diabetes Prediction using Machine Learning". IEEE, pp 942-928, 2018.

[2] K.VijiyaKumar, B.Lavanya, I.Nirmala, S.Sofia Caroline, "Random Forest Algorithm for the Prediction of Diabetes ".Proceeding of International Conference on Systems Computation Automation and Networking, 2019.

[3] Md. Faisal Faruque, Asaduzzaman, Iqbal H. Sarker, "Perfor- mance Analysis of Machine Learning Techniques to Predict Diabetes Mellitus". International Conference on Electrical, Computer and Communication Engineering (ECCE), 7-9 February, 2019.

[4] Tejas N. Joshi, Prof. Pramila M. Chawan, "Diabetes Prediction Using Machine Learning Techniques".Int. Journal of Engineering Research and Application, Vol. 8, Issue 1, (Part -II) Janu- ary 2018, pp.-09-13

[5] Nonso Nnamoko, Abir Hussain, David England, "Predicting Diabetes Onset: an Ensemble Supervised Learning Approach ". IEEE Congress on Evolutionary Computation (CEC), 2018.

[6] Deeraj Shetty, Kishor Rit, Sohail Shaikh, Nikita Patil, "Diabe- tes Disease Prediction Using Data Mining ".International Con- ference on Innovations in Information, Embedded and Com- munication Systems (ICIIECS), 2017.

[7] Nahla B., Andrew et al,"Intelligible support vector machines for diagnosis of diabetes mellitus. Information Technology in Biomedicine", IEEE Transactions. 14, (July. 2010), 1114-20.

[8] A.K., Dewangan, and P., Agrawal, Classification of Diabetes Mellitus Using Machine Learning Techniques, International Journal of Engineering and Applied Sciences, vol. 2, 2015.

Thank you