

PREDICTION OF DIABETES USING DATA SCIENCE TECHNIQUE

Submitted in partial fulfilment of the requirements for the
award of
Bachelor of Engineering degree in Computer Science and Engineering

By

PERCY PAULIN. J (38110403)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade "A" by NAAC

**JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI - 600 119**

MAY - 2022



SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **PERCY PAULIN. J (38110403)** who carried out the project entitled "**PREDICTION OF DIABETES USING DATA SCIENCE TECHNIQUE**" under my supervision from September 2019 to April 2020.

Internal Guide

Dr. S. L. JANY SHABU M.Phil., Ph.D.,

Head of the Department

Dr. S. VIGNESWARI M.E., Ph.D.,

Submitted for Viva voce Examination held on_____

Internal Examiner

External Examiner

DECLARATION

I **PERCY PAULIN J (38110403)** hereby declare that the Project Report entitled **“PREDICTION OF DIABETES USING DATA SCIENCE TECHNIQUE”** done by me under the guidance of **Dr. S. L. JANY SHABU M.Phil., Ph.D.**, and Sathyabama Institute of Science and Technology is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

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

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

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. S. L. JANY SHABU M.Phil., Ph.D.**, for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

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

ABSTRACT

Diabetes is a chronic (long-lasting) health condition that affects how your body turns food into energy. Most of the food you eat is broken down into sugar (also called glucose) and released into your bloodstream. When your blood sugar goes up, it signals your pancreas to release insulin. Nowadays machine learning is applied to healthcare system where there is a chance of predicting the disease early. The main necessity of Artificial intelligence is data. The past dataset is collected and that dataset is used to build a machine learning model. The necessary pre-processing techniques are applied like univariate analysis and bivariate analysis are implemented. The data is visualised for better understanding of the features and based on that a classification model is built by using machine learning algorithm and comparison of algorithms are done based on their performance metrics like accuracy, F1 score recall etc.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	V
	LIST OF FIGURES	VIII
	LIST OF ABBREVIATION	IX
1	INTRODUCTION	1
1	1.1 Data Science	
2	1.1.1 Data Scientist	
3	1.2 Artificial Intelligence	3
3	1.3 Machine learning	
4	1.4 Overview	
5	2 LITERATURE SURVEY	
8	3 AIM AND SCOPE OF THE PROJECT	
8	3.1 Aim	
8	3.2 Scope	
8	3.3 Objective	

	3.4 Proposed system	
8		
	3.4.1 Advantage	
9		
	4	ALGORITHM AND METHODS
10		
	4.1 Model description	
10		
	4.1.1 Data Pre-processing	
10		
	4.1.2 Data Visualization	
11		
	4.1.3 Comparing Algorithm	
12		
	4.1.4 Deployment using Flask	
33		
	vi	
	4.2 Requirement and technology used	34
	4.2.1 Functional requirement	
34		
	4.2.2 Non Functional requirement	
34		
	4.2.3 Environmental requirement	
35		
5	CONCLUSION AND FUTURE WORK	36
	5.1 Conclusion	
36		
	5.2 Application	
36		

37	5.2 Future work
38	REFERENCE
39	APPENDIX
39	A. Sample Code
56	B. Screen short

LIST OF FIGURES		
FIGURE NO	TITLE	PAGE NO

	3.1	System Architecture	9
10	4.1	After inputting all the library and data	
	4.2	Bar graph of Age wise positive cases	11
12	4.3	Positive and Negative cases in heat map	
	4.4	Positive and Negative predictive value of Logistic Regression.	16
	4.5	Accuracy result of Logistic Regression	17
19	4.6	Positive and Negative predictive value of Decision Tree.	
21	4.7	Accuracy result of Decision Tree.	
	4.8	Positive and Negative predictive value of Random Forest.	23
	4.9	Accuracy result of Random Forest.	25
	4.10	Positive and Negative predictive value of Support Vector Machine.	29
	4.11	Accuracy result of Support vector machine	30
	5.1	Pie chart of positive and negative cases	36

LIST OF ABBREVIATION

- | | |
|----------|-----------------------------------|
| 1 | AI - Artificial intelligence |
| 2 | ML - Machine Learning |
| 3 | SML – Supervised Machine Learning |

CHAPTER 1

INTRODUCTION

1.1 DATA SCIENCE

Data science is an interdisciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from structured and unstructured data, and apply knowledge and actionable insights from data across a broad range of application domains. The term "data science" has been traced back to 1974, when Peter Naur proposed it as an alternative name for computer science. In 1996, the International Federation of Classification Societies became the first conference to specifically feature data science as a topic. However, the definition was still in flux.

The term "data science" was first coined in 2008 by D.J. Patil, and Jeff Hammerbacher, the pioneer leads of data and analytics efforts at LinkedIn and Facebook. In less than a decade, it has become one of the hottest and most trending professions in the market. Data science is the field of study that combines domain expertise, programming skills, and knowledge of mathematics and statistics to extract meaningful insights from data. Data science can be defined as a blend of mathematics, business acumen, tools, algorithms and machine learning techniques, all of which help us in finding out the hidden insights or patterns from raw data which can be of major use in the formation of big business decisions.

1.1.1 Data Scientist:

Data scientists examine which questions need answering and where to find the related data. They have business acumen and analytical skills as well as the ability to mine, clean, and present data. Businesses use data scientists to source, manage, and analyse large amounts of unstructured data.

1.2 ARTIFICIAL INTELLIGENCE

Artificial intelligence (AI) refers to the simulation of human intelligence in machines that are programmed to think like humans and mimic their actions. The term may also be applied to any machine that exhibits traits associated with a human mind such as learning and problem-solving. Artificial intelligence (AI) is intelligence demonstrated by machines, as opposed to the natural intelligence displayed by humans or animals. Leading AI textbooks define the field as the study of "intelligent agents" any system that perceives its environment and takes actions that maximize its chance of achieving its goals.

Learning processes. This aspect of AI programming focuses on acquiring data and creating rules for how to turn the data into actionable information. The rules, which are called algorithms, provide computing devices with step-by-step instructions for how to complete a specific task.

Reasoning processes. This aspect of AI programming focuses on choosing the right algorithm to reach a desired outcome.

Self-correction processes. This aspect of AI programming is designed to continually fine-tune algorithms and ensure they provide the most accurate results possible.

1.3 MACHINE LEARNING

Machine learning is to predict the future from past data. Machine learning (ML) is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of Computer Programs that can change when exposed to new data and the basics of Machine Learning, implementation of a

simple machine learning algorithm using python. Process of training and prediction involves use of specialized algorithms. It feed the training data to an algorithm, and the algorithm uses this training data to give predictions on a new test data. Machine learning can be roughly separated in to three categories. There are supervised learning, unsupervised learning and reinforcement learning. Supervised learning program is both given the input data and the corresponding labeling to learn data has to be labeled by a human being beforehand. Unsupervised learning is no labels. It provided to the learning algorithm.

This algorithm has to figure out the clustering of the input data. Classification is the process of predicting the class of given data points. Classes are sometimes called as targets/ labels or categories. Classification predictive modeling is the task of approximating a mapping function from input variables(X) to discrete output variables(y). In machine learning and statistics, classification is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observation. This data set may simply be bi-class (like identifying whether the person is male or female or that the mail is spam or non-spam) or it may be multi-class too. Some examples of classification problems are: speech recognition, handwriting recognition, bio metric identification, document classification etc.

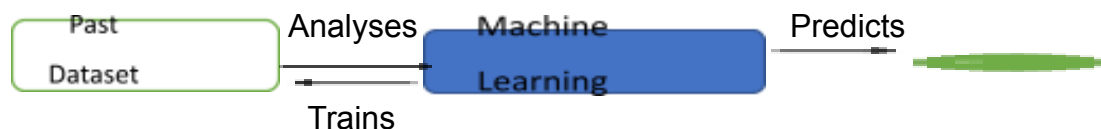


Fig: Process of Machine learning

Supervised Machine Learning is the majority of practical machine learning uses supervised learning. Supervised learning is where have input variables (X) and an output variable (y) and use an algorithm to learn the mapping function from the input to the output is $y = f(X)$. The goal is to approximate the mapping function so well

that when you have new input data (X) that you can predict the output variables (y) for that data. Techniques of Supervised Machine Learning algorithms include logistic regression, multi-class classification, Decision Trees and support vector machines etc. Supervised learning requires that the data used to train the algorithm is already labeled with correct answers. Supervised learning problems can be further grouped into Classification problems. This problem has as goal the construction of a succinct model that can predict the value of the dependent attribute from the attribute variables. The difference between the two tasks is the fact that the dependent attribute is numerical for regression and categorical for classification. A classification model attempts to draw some conclusion from observed values. Given one or more inputs a classification model will try to predict the value of one or more outcomes. A classification problem is when the output variable is a category, such as “red” or “blue”.

1.4 OVERVIEW

To understand why insulin is important in diabetes, it helps to know more about how the body uses food for energy. Your body is made up of millions of cells. To make energy, these cells need food in a very simple form. When you eat or drink, much of your food is broken down into a simple sugar called "glucose." Then, glucose is transported through the bloodstream to the cells of your body where it can be used to provide some of the energy your body needs for daily activities.

- Define a problem
- Preparing data
- Evaluating algorithms
- Improving results
- Predicting results

CHAPTER 2

LITERATURE SURVEY

2.1 SURVEY 1

Title : A Predictive model for diabetes using Machine Learning techniques.

Author: A E. Ewiekpaefe, Nafisat Abdulkadir.

Year : 2021

Diabetes Mellitus (DM) which refers to a metabolic disorder that occurs when the level of blood sugar in the body is considered high, which could be a resulting effect of inadequate availability of insulin in the body. It is a chronic disease which may lead to myriads of complications in the body system. Statistics by the World Health Organization (WHO) in 2013, indicated that DM was the cause of death of over 1.5 million people around the world and in 2016, 8.5% of adults within age seventeen (17) and above were reported to be diabetic and diabetic patients have continued to increase in recent years.

2.2 SURVEY 2

Title A Survey: Detection and Prediction of Diabetes Using Machine Learning Techniques

Author : Priyanka Indoria , Yogesh Kumar Rathore,.

Year : March-2018

Diabetes is a one of the leading cause of blindness, kidney failure, amputations, heart failure and stroke. When we eat, our body turns food into

sugars, or glucose. At that point, our pancreas is supposed to release insulin. Insulin serves as a “key” to open our cells, to allow the glucose to enter -- and allow us to use the glucose for energy. But with diabetes, this system does not work. Several major things can go wrong – causing the onset of diabetes. Type 1 and type 2 diabetes are the most common forms of the disease, but there are also other kinds, such as gestational diabetes, which occurs during pregnancy, as well as other forms. This paper focuses on recent developments in machine learning which have made significant impacts in the detection and diagnosis of diabetes.

2.3 SURVEY 3

Title Prediction of Diabetics using Machine Learning

Author: G. Geetha, K.Mohana Prasad

Year : May 2011

Around 50.9 Million People in India suffer from diabetics and Tamil Nadu stands second in the list of Indian states. The main objective of this paper is to develop prediction modeling of the given medical data of patients with and without diabetics. Through this paper, we aim to create hybrid models that can be easily used by doctors to treat patients with diabetics. Naïve Bayes and Random forest algorithms are used to predict whether a person having diabetics or not, by keeping his health conditions in mind. The Random Forest algorithm is used here in order to perform feature selection. It takes n inputs from the dataset and builds numerous uncorrelated decision trees during the time of training. It then displays the class that is the mode of all of the class outputs by individual trees.

2.4 SURVEY 4

Title : Effective Prediction of Diabetes Mellitus using Nine different Machine Learning Techniques and their Performances

Author : Sarvesh Vishwakarma, Anupam Agrawal

Year : May 2011

Diabetes is a disease where the predominant finding is high blood sugar. The high blood sugar may either be because of deficient insulin production (Type 1) or insulin resistance in peripheral tissue cells (Type 2). Many problems occur if diabetes remains untreated and unidentified. Positive likelihood ratio, Negative likelihood ratio, Positive predictive value, Negative predictive value, Disease prevalence, Specificity, Precision, Recall, F1-Score, True positive rate, False positive rate of the applied algorithms is discussed and compared. Diabetes is growing at an increasing in the world and it requires continuous monitoring. To check this we use Logical regression, Random forest, Logical regression CV, Support Vector Machine, Artificial Neural Network (ANN), Decision Tree, k-nearest neighbors (KNN), XGB classifier.

2.5 SURVEY 5

Title : Literature Survey On Different Techniques Used For Predicting Diabetes Mellitus

Author : Shashank Joshi, Vijayendra Gaikwad, Sairam Rathod,

Year : , June 2020

Diabetes is a disease where the predominant finding is high blood sugar. The high blood sugar may either be because of deficient insulin production (Type 1) or insulin resistance in peripheral tissue cells (Type 2). Many problems occur if diabetes remains untreated and unidentified. Nine different machine learning techniques are used in this research work for prediction of diabetes. A dataset of diabetic patient's is taken and nine different machine learning techniques are applied on the dataset. Positive

likelihood ratio, Negative likelihood ratio, Positive predictive value, Negative predictive value, Disease prevalence, Specificity, Precision, Recall, F1-Score ,True positive rate, False positive rate of the applied algorithms is discussed and compared. Diabetes is growing at an increasing in the world and it requires.

CHAPTER 3

AIM AND SCOPE OF THE PROJECT

3.1 Aim:

The main Aim of the project is to Predict the peoples who affected by diabetes, comparing the algorithm and taking the best accuracy result.

3.2 Scope:

The scope of this project is to investigate a diabetes dataset to find the diabetes using machine learning technique. To find the diabetes is based on the people's health actual conditions it's not easy one to predict, but using the AI machine learning technique compare with different algorithms we can build a predictive models. Using metrics (accuracy, precision, recall, etc..) we can validate the model accuracy.

3.3 Objective:

The goal is to develop a machine learning model for Predicting diabetes using to potentially replace the updatable supervised machine learning classification models by predicting results in the form of best accuracy by comparing supervised algorithm.

3.4 Proposed system:

The proposed method is to build a machine learning model for predicting Diabetes. The process carries from data collection where the past data related to Diabetes are collected. Data mining is a commonly used technique for processing enormous data in the healthcare domain. The Diabetes if found before proper treatment can save lives. Machine learning is now applied and mostly used in health care where it reduces the manual effort and better model makes error less which leads in saving the life. The data analysis is done on the dataset proper variable identification done that is both the dependent variables and independent variables are found. Then proper machine learning algorithm are applied on the dataset where the pattern of data is learnt. After applying different algorithms a better algorithm is used for the prediction of outcome.

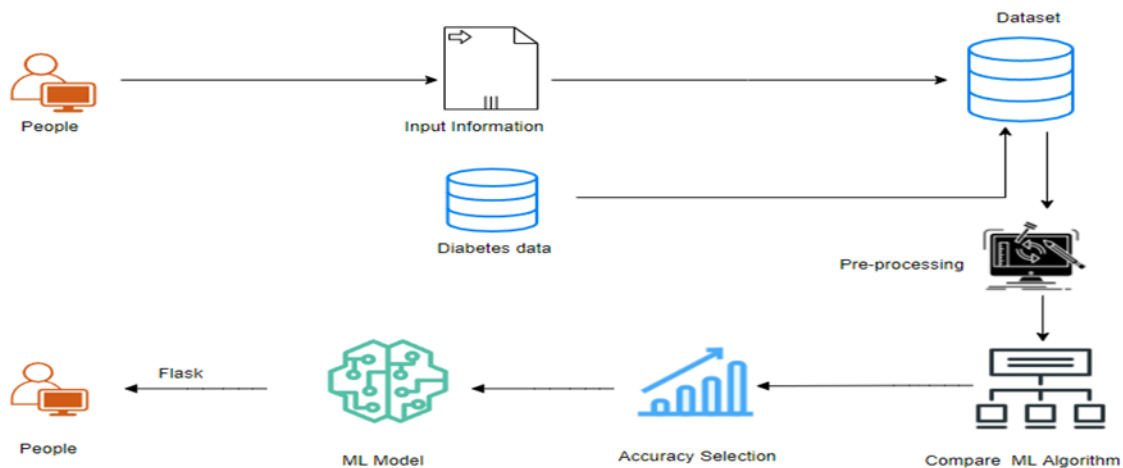


Fig 3.1 Architecture Diagram

3.4.1 Advantages:

These reports are to the investigation of applicability of machine learning techniques for Diabetes prediction in operational conditions.

Finally, it highlights some observations on future research issues, challenges, and needs.

CHAPTER 4

ALGORITHM AND METHODS

4.1 Model description:

4.1.1 Data Pre-processing

Validation techniques in machine learning are used to get the error rate of the Machine Learning (ML) model, which can be considered as close to the true error rate of the dataset. If the data volume is large enough to be representative of the population, you may not need the validation techniques. However, in real-world scenarios, to work with samples of data that may not be a true representative of the population of given dataset. To finding the missing value, duplicate value and description of data type whether it is float variable or integer. The sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyper parameters.

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

Fig 4.1 Dataset of Diabetes

The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration. The validation set is used to evaluate a given model, but this is for frequent evaluation. It as machine learning engineers use this data to fine-tune the model hyper parameters. Data collection, data analysis, and the process of addressing data content, quality, and structure can add up to a time-consuming to-do list. During the process of data identification, it helps to understand your data and its properties; this knowledge will help you choose which algorithm to use to build your model.

4.1.2 Data visualization

Data visualization is an important skill in applied statistics and machine learning. Statistics does indeed focus on quantitative descriptions and estimations of data. Data visualization provides an important suite of tools for gaining a qualitative understanding. This can be helpful when exploring and getting to know a dataset and can help with identifying patterns, corrupt data, outliers, and much more. With a little domain knowledge, data visualizations can be used to express and demonstrate key relationships in plots and charts that are more visceral and stakeholders than measures of association or significance. Data visualization and exploratory data analysis are whole fields themselves and it will recommend a deeper dive into some the books

mentioned.

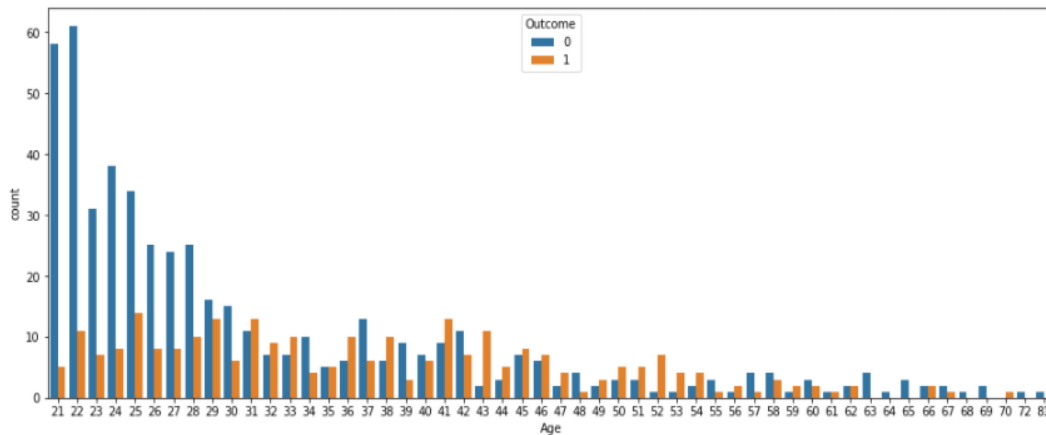


Fig 4.2 Bar Graph of Age wise Positive cases

False Positives (FP): A person who will pay predicted as defaulter. When actual class is no and predicted class is yes. E.g. if actual class says this passenger did not survive but predicted class tells you that this passenger will survive.

False Negatives (FN): A person who default predicted as payer. When actual class is yes but predicted class in no. E.g. if actual class value indicates that this passenger survived and predicted class tells you that passenger will die.

True Positives (TP): A person who will not pay predicted as defaulter. These are the correctly predicted positive values which means that the value of actual class is yes and the value of predicted class is also yes. E.g. if actual class value indicates that this passenger survived and predicted class tells you the same thing.

True Negatives (TN): A person who default predicted as payer. These are the correctly predicted negative values which means that the value of actual class is no and value of predicted class is also no. E.g. if actual class says this passenger did not survive and predicted class tells you the same thing.

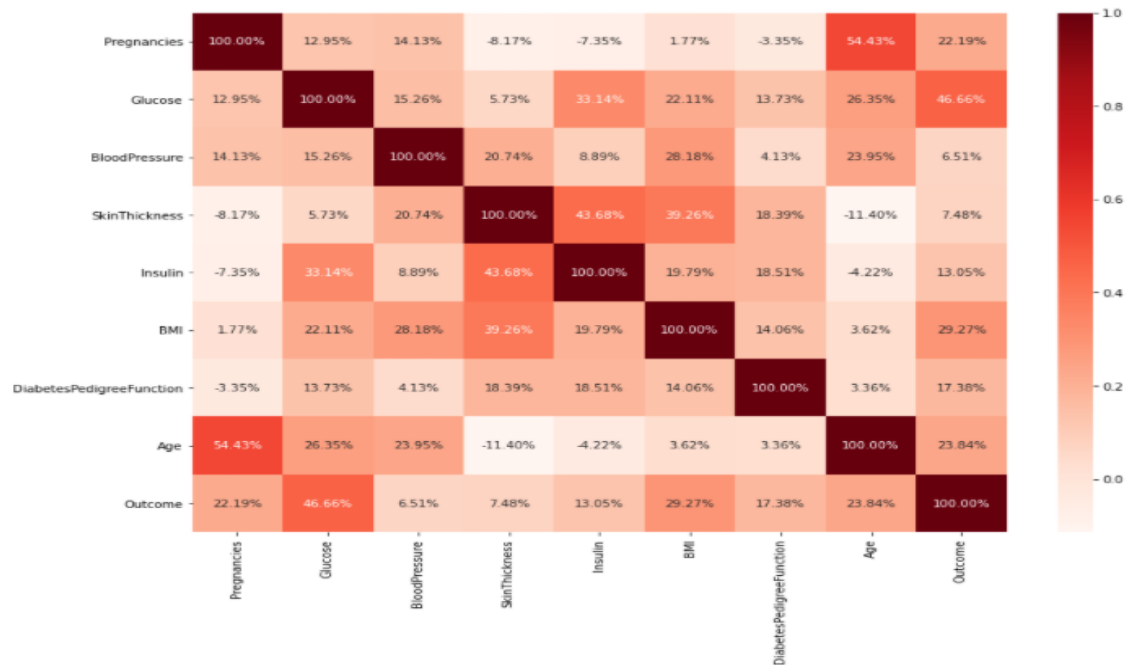


Fig 4.3 Positive and negative case in heat map

4.1.3 Comparing Algorithm with prediction in the form of best accuracy result

It is important to compare the performance of multiple different machine learning algorithms consistently and it will discover to create a test harness to compare multiple different machine learning algorithms in Python with scikit-learn. It can use this test harness as a template on your own machine learning problems and add more and different algorithms to compare. Each model will have different performance characteristics. When have a new dataset, it is a good idea to visualize the data using different techniques in order to look at the data from different perspectives. The same idea applies to model selection. You should use a number of different ways of looking at the estimated accuracy of your machine learning algorithms in order to choose the one or two to finalize. A way to do this is to use different visualization methods to show the average accuracy, variance and other properties of the distribution of model accuracies.

In the example below 4 different algorithms are compared:

Logistic Regression
Decision Tree
Random Forest
Support Vector Machine

The K-fold cross validation procedure is used to evaluate each algorithm, importantly configured with the same random seed to ensure that the same splits to the training data are performed and that each algorithm is evaluated in precisely the same way. Before that comparing algorithm, Building a Machine Learning Model using install Scikit-Learn libraries. In this library package have to done preprocessing, linear model with logistic regression method, cross validating by KFold method, ensemble with random forest method and tree with decision tree classifier. Additionally, splitting the train set and test set. To predicting the result by comparing accuracy.

Prediction result by accuracy:

Logistic regression algorithm also uses a linear equation with independent predictors to predict a value. The predicted value can be anywhere between negative infinity to positive infinity. It need the output of the algorithm to be classified variable data. Higher accuracy predicting result is logistic regression model by comparing the best accuracy.

True Positive Rate(TPR) = $TP / (TP + FN)$

False Positive rate(FPR) = $FP / (FP + TN)$

Accuracy: The Proportion of the total number of predictions that is correct otherwise overall how often the model predicts correctly defaulters and non-defaulters.

Accuracy calculation:

Accuracy = $(TP + TN) / (TP + TN + FP + FN)$

Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. One may

think that, if we have high accuracy then our model is best. Yes, accuracy is a great measure but only when you have symmetric datasets where values of false positive and false negatives are almost same.

Precision: The proportion of positive predictions that are actually correct. (When the model predicts default: how often is correct?)

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$$

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. The question that this metric answer is of all passengers that labeled as survived, how many actually survived? High precision relates to the low false positive rate. We have got 0.788 precision which is pretty good.

Recall: The proportion of positive observed values correctly predicted. (The proportion of actual defaulters that the model will correctly predict)

$$\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$$

Recall(Sensitivity) - Recall is the ratio of correctly predicted positive observations to the all observations in actual class - yes.

F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it's better to look at both Precision and Recall.

General Formula:

$$\text{F- Measure} = 2\text{TP} / (2\text{TP} + \text{FP} + \text{FN})$$

F1-Score Formula:

$$\text{F1 Score} = 2 * (\text{Recall} * \text{Precision}) / (\text{Recall} + \text{Precision})$$

Algorithm Explanation

In machine learning and statistics, classification is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observation. This data set may simply be bi-class (like identifying whether the person is male or female or that the mail is spam or non-spam) or it may be multi-class too. Some examples of classification problems are: speech recognition, handwriting recognition, bio metric identification, document classification etc. In Supervised Learning, algorithms learn from labeled data. After understanding the data, the algorithm determines which label should be given to new data based on pattern and associating the patterns to the unlabeled new data.

Used Python Packages:

sklearn:

- In python, sklearn is a machine learning package which include a lot of ML algorithms.
- Here, we are using some of its modules like `train_test_split`, `DecisionTreeClassifier` or `Logistic Regression` and `accuracy_score`.

NumPy:

- It is a numeric python module which provides fast maths functions for calculations.
- It is used to read data in numpy arrays and for manipulation purpose.

Pandas:

- Used to read and write different files.
- Data manipulation can be done easily with data frames.

Matplotlib:

- Data visualization is a useful way to help with identify the patterns from given dataset.
- Data manipulation can be done easily with data frames.

Logistic Regression

It is a statistical method for analysing a data set in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes). The goal of logistic regression is to find the best fitting model to describe the relationship between the dichotomous characteristic of interest (dependent variable = response or outcome variable) and a set of independent (predictor or explanatory) variables. Logistic regression is a Machine Learning classification algorithm that is used to predict the probability of a categorical dependent variable. In logistic regression, the dependent variable is a binary variable that contains data coded as 1 (yes, success, etc.) or 0 (no, failure, etc.).

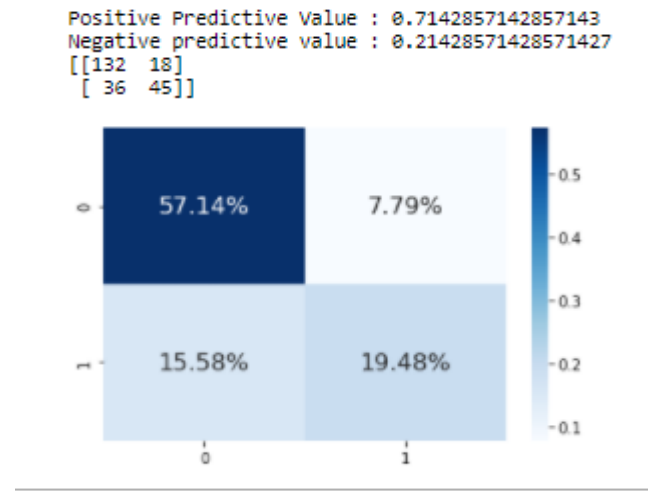


Fig 4.4 Positive and Negative predictive value of Logistic Regression

In other words, the logistic regression model predicts $P(Y=1)$ as a function of X . Logistic regression Assumptions:

- o Logistic regression predicts the output of a categorical dependent variable. Q 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.

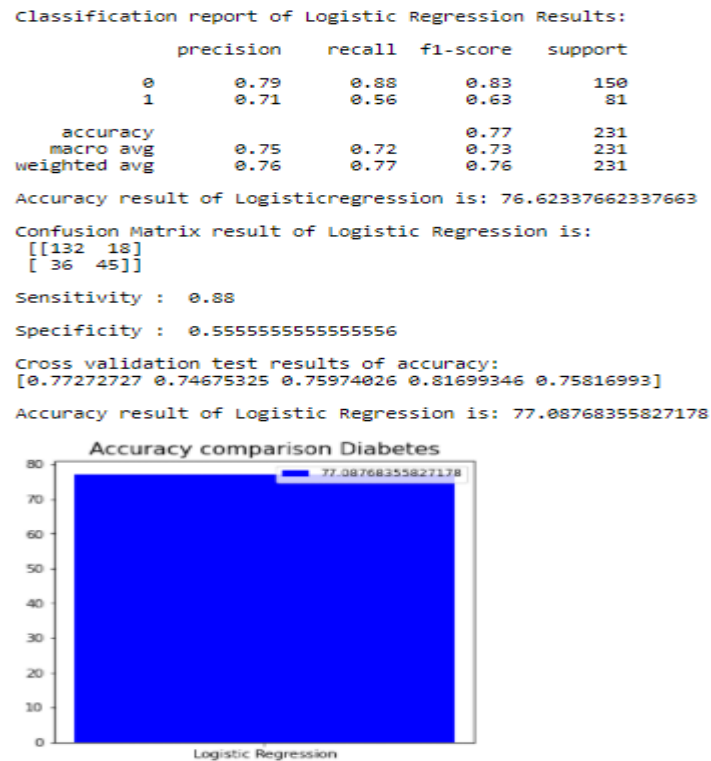


Fig 4.5 Accuracy result of logistic regression

- 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.

- o Logistic Regression can be used to classify the observations using different types of data and can easily determine the most effective variables used for the classification. The below image is showing the logistic function:

Logistic Function (Sigmoid Function):

- o The sigmoid function is a mathematical function used to map the predicted values to probabilities.
- o It maps any real value into another value within a range of 0 and 1.
- o The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the "S" form. The S-form curve is called the Sigmoid function or the logistic function.

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

- o Binary or Binomial: In such a kind of classification, a dependent variable will have only two possible types either 1 and 0. For example, these variables may represent success or failure, yes or no, win or loss etc.
- o Multinomial: In such a kind of classification, dependent variable can have 3 or more possible unordered types or the types having no quantitative significance. For example, these variables may represent "Type A" or "Type B" or "Type C".
- o Ordinal; In such a kind of classification, dependent variable can have 3 or more possible ordered types or the types having a quantitative significance. For example, these variables may represent "poor" or "good", "very good", "Excellent" and each category can have the scores like 0,1,2,3.

Decision Tree

In general, Decision tree analysis is a predictive modelling tool that can be applied across many areas. Decision trees can be constructed by an algorithmic approach that can split the dataset in different ways based on different conditions. Decision trees are the most powerful algorithms that fall under the category of supervised algorithms. They can be used for both classification and regression tasks. The two main entities of a tree are decision nodes, where the data is split and leaves, where we get outcome. The example of a binary tree for predicting whether a person is fit or unfit providing various information like age, eating habits and exercise habits, is given below –

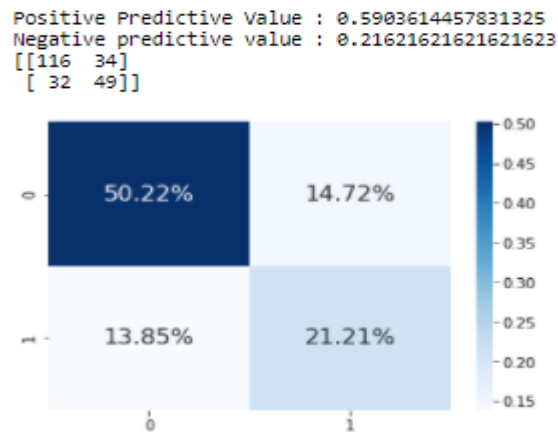


Fig 4.6 Positive and Negative Predictive value of Decision tree

In the above decision tree, the question are decision nodes and final outcomes are leaves. We have the following two types of decision trees.

- Classification decision trees – In this kind of decision trees, the decision variable is categorical. The above decision tree is an example of classification decision tree.
- Regression decision trees – In this kind of decision trees, the decision variable is continuous.

Implementing Decision Tree Algorithm

Gini Index

Important Terminology related to Decision Tree

1. **Root Node:** It represents the entire population or sample and this further gets divided into two or more homogeneous sets.
2. **Splitting:** It is a process of dividing a node into two or more sub-nodes.
3. **Decision Node:** When a sub-node splits into further sub-nodes, then it is called the decision node.
4. **Leaf / Terminal Node:** Nodes do not split is called Leaf or Terminal node.
5. **Pruning:** When we remove sub-nodes of a decision node, this process is called pruning. You can say the opposite process of splitting.
6. **Branch / Sub-Tree:** A subsection of the entire tree is called branch or sub-tree.
7. **Parent and Child Node:** A node, which is divided into sub-nodes is called a parent node of sub-nodes whereas sub-nodes are the child of a parent node.

Decision trees classify the examples by sorting them down the tree from the root to some leaf/terminal node, with the leaf/terminal node providing the classification of the example.

Each node in the tree acts as a test case for some attribute, and each edge descending from the node corresponds to the possible answers to the test case. This process is recursive in nature and is repeated for every subtree rooted at the new node.

```

Classification report DecisionTree classifier Results:

              precision    recall  f1-score   support

     0               0.78        0.77        0.78        150
     1               0.59        0.60        0.60         81

 accuracy               0.69
macro avg               0.69        0.69        0.69        231
weighted avg            0.72        0.71        0.72        231

Accuracy result of DecisionTree is: 71.42857142857143
Confusion Matrix result of DecisionTree Classifier is:
[[116  34]
 [ 32  49]]

Sensitivity :  0.7733333333333333
Specificity :  0.6049382716049383

Cross validation test results of accuracy:
[0.67532468 0.65584416 0.68831169 0.81045752 0.7254902 ]

Accuracy result of DecisionTree Classifier is: 71.1085646379764

```

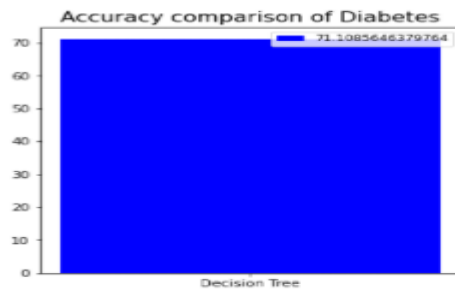


Fig 4.7 Accuracy result of Decision Tree

Entropy is a measure of the randomness in the information being processed. The higher the entropy, the harder it is to draw any conclusions from that information. Flipping a coin is an example of an action that provides information that is random.

From the above graph, it is quite evident that the entropy $H(X)$ is zero when the probability is either 0 or 1. The Entropy is maximum when the probability is 0.5 because it projects perfect randomness in the data and there is no chance if perfectly determining the outcome.

Mathematically Entropy for 1 attribute is represented as:

$$E(S) = \sum_{i=1}^c -p_i \log_2 p_i$$

Where $S \rightarrow$ Current state, and $P_i \rightarrow$ Probability of an event i of state S or Percentage of class i in a node of state S .

Information Gain

Information gain or IG is a statistical property that measures how well a given attribute separates the training examples according to their target classification. Constructing a decision tree is all about finding an attribute that returns the highest information gain and the smallest entropy.

$$\text{Information Gain} = \text{Entropy}(\text{before}) - \sum_{j=1}^K \text{Entropy}(j, \text{after})$$

Information gain is a decrease in entropy. It computes the difference between entropy before split and average entropy after split of the dataset based on given attribute values. ID3 (Iterative Dichotomiser) decision tree algorithm uses information gain.

Mathematically, IG is represented as:

$$\text{Information Gain}(T, X) = \text{Entropy}(T) - \text{Entropy}(T, X)$$

In a much simpler way, we can conclude that:

$$\text{Information Gain} = \text{Entropy}(\text{before}) - \sum_{j=1}^K \text{Entropy}(j, \text{after})$$

Information Gain

Where “before” is the dataset before the split, K is the number of subsets generated by the split, and (j, after) is subset j after the split.

Split Creation

A split is basically including an attribute in the dataset and a value. We can create a split in dataset with the help of following three parts –

- Part 1: Calculating Gini Score – We have just discussed this part in the previous section.
- Part 2: Splitting a dataset – It may be defined as separating a dataset into two lists of rows having index of an attribute and a split value of that attribute. After getting the two groups - right and left, from the dataset, we can calculate

the value of split by using Gini score calculated in first part. Split value will decide in which group the attribute will reside.

- Part 3: Evaluating all splits – Next part after finding Gini score and splitting dataset is the evaluation of all splits. For this purpose, first, we must check every value associated with each attribute as a candidate split. Then we need to find the best possible split by evaluating the cost of the split. The best split will be used as a node in the decision tree.

Random forest Algorithm

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

The below diagram explains the working of the Random Forest algorithm:

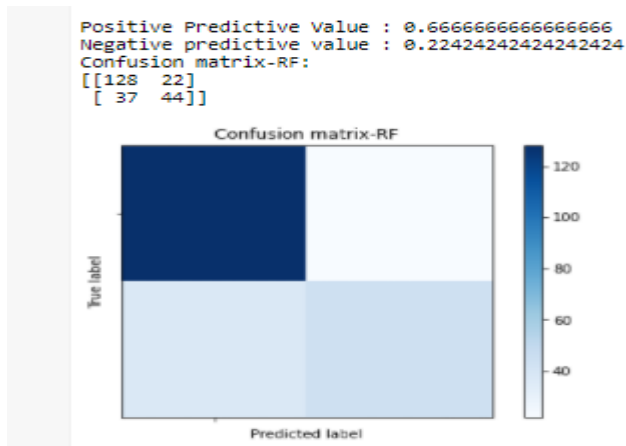


Fig 4.8 Positive and Negative Predictive value of Random Forest

Why use Random Forest?

Below are some points that explain why we should use the Random Forest algorithm:

- o It takes less training time as compared to other algorithms.
- o It predicts output with high accuracy, even for the large dataset it runs efficiently.
- o It can also maintain accuracy when a large proportion of data is missing.

How does Random Forest algorithm work?

Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.

The Working process can be explained in the below steps and diagram:

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

Step-2: Build the decision trees associated with the selected data points (Subsets).

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

Step-4: Repeat Step 1 & 2.

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

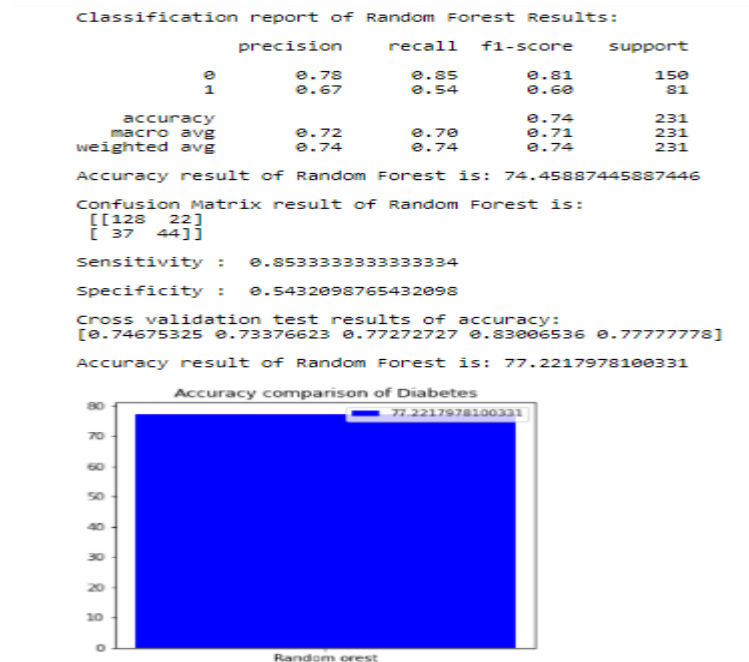


Fig 4.9 Accuracy result of Random Forest

The working of the algorithm can be better understood by the below example:
 Example: Suppose there is a dataset that contains multiple fruit images. So, this dataset is given to the Random forest classifier. The dataset is divided into subsets and given to each decision tree. During the training phase, each decision tree produces a prediction result, and when a new data point occurs, then based on the majority of results, the Random Forest classifier predicts the final decision. Consider the below image:

Implementation in Scikit-learn

For each decision tree, Scikit-learn calculates a nodes importance using Gini Importance, assuming only two child nodes (binary tree):

$$ni_j = w_j C_j - w_{left(j)} C_{left(j)} - w_{right(j)} C_{right(j)}$$

- $ni_{sub(j)}$ = the importance of node j
- $w_{sub(j)}$ = weighted number of samples reaching node j
- $C_{sub(j)}$ = the impurity value of node j
- $left(j)$ = child node from left split on node j
- $right(j)$ = child node from right split on node j

$sub()$ is being used as subscript isn't available in Medium

See method `compute_feature_importances` in `_tree.pyx`

The importance for each feature on a decision tree is then calculated as:

$$fi_i = \frac{\sum_{j: \text{node } j \text{ splits on feature } i} ni_j}{\sum_{k \in \text{all nodes}} ni_k}$$

- $fi_{sub(i)}$ = the importance of feature i
- $ni_{sub(j)}$ = the importance of node j

These can then be normalized to a value between 0 and 1 by dividing by the sum of all feature importance values:

$$normfi_i = \frac{fi_i}{\sum_{j \in \text{all features}} fi_j}$$

The final feature importance, at the Random Forest level, is it's average over all the trees. The sum of the feature's importance value on each trees is calculated and divided by the total number of trees:

$$RFfi_i = \frac{\sum_{j \in \text{all trees}} normfi_{ij}}{T}$$

- $RFfi_{sub(i)}$ = the importance of feature i calculated from all trees in the Random Forest model
- $normfi_{sub(ij)}$ = the normalized feature importance for i in tree j
- T = total number of trees

See method `feature_importances_` in `forest.py`

Notation was inspired by this [StackExchange](#) thread which I found incredible useful for this post.

Implementation in Spark

For each decision tree, Spark calculates a feature's importance by summing the gain, scaled by the number of samples passing through the node:

$$fi_i = \sum_{j: \text{nodes } j \text{ splits on feature } i} s_j C_j$$

- $f_{i \text{ sub}(i)}$ = the importance of feature i
- $s_{\text{sub}(j)}$ = number of samples reaching node j
- $C_{\text{sub}(j)}$ = the impurity value of node j

See method `computeFeatureImportance` in `treeModels.scala`

To calculate the final feature importance at the Random Forest level, first the feature importance for each tree is normalized in relation to the tree:

$$\text{normfi}_{i_t} = \frac{f_{i_t}}{\sum_{j \in \text{all features}} f_{i_j}}$$

- $\text{normfi}_{\text{sub}(i)}$ = the normalized importance of feature i
- $f_{i \text{ sub}(i)}$ = the importance of feature i

Then feature importance values from each tree are summed normalized:

$$\text{RFfi}_{i_t} = \frac{\sum_j \text{normfi}_{i_{tj}}}{\sum_{j \in \text{all features}, k \in \text{all trees}} \text{normfi}_{i_{jk}}}$$

- $\text{RFfi}_{\text{sub}(i)}$ = the importance of feature i calculated from all trees in the Random Forest model
- $\text{normfi}_{\text{sub}(ij)}$ = the normalized feature importance for i in tree j

Support Vector Machines

Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier. The objective of applying SVMs is to find the best line in two dimensions or the best hyperplane in more than two dimensions in order to help us separate our space into classes. The hyperplane (line) is found through the maximum margin, i.e., the maximum distance between data points of both classes. Don't you think the definition and idea of SVM look a bit abstract? No worries, let me explain in details.

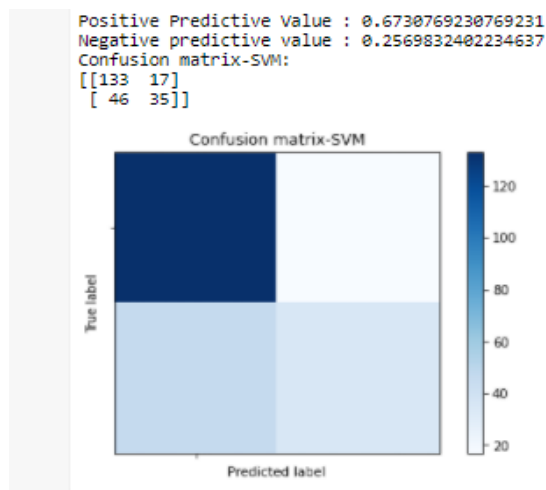


Fig 4.10 Positive and Negative Predictive value of SVM

Support Vector, Hyperplane, and Margin

The vector points closest to the hyperplane are known as the support vector points because only these two points are contributing to the result of the algorithm, and other points are not. If a data point is not a support vector, removing it has no effect on the model. On the other hand, deleting the support vectors will then change the position of the hyperplane.

The dimension of the hyperplane depends upon the number of features. If the number of input features is 2, then the hyperplane is just a line. If the number of input features is 3, then the hyperplane becomes a

two-dimensional plane. It becomes difficult to imagine when the number of features exceeds 3.

The distance of the vectors from the hyperplane is called the margin, which is a separation of a line to the closest class points. We would like to choose a hyperplane that maximises the margin between classes. The graph below shows what good margin and bad margin are.

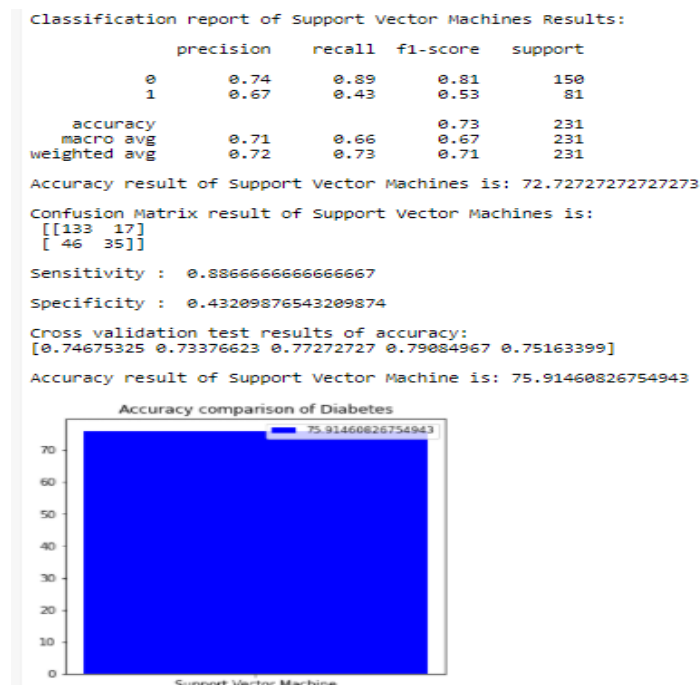


fig 4.11 Accuracy result of SVM

o Hard Margin

If the training data is linearly separable, we can select two parallel hyperplanes that separate the two classes of data, so that the distance between them is as large as possible.

o Soft Margin

As most of the real-world data are not fully linearly separable, we will allow some margin violation to occur, which is called soft margin classification. It is better to have a large margin, even though some constraints are

violated. Margin violation means choosing a hyperplane, which can allow some data points to stay in either the incorrect side of the hyperplane and between the margin and the correct side of the hyperplane.

In order to find the maximal margin, we need to maximize the margin between the data points and the hyperplane. In the following session, I will share the mathematical concepts behind this algorithm.

o Linear Algebra Revisited

Before we move on, let's review some concepts in Linear Algebra.

A unit vector $\hat{\mathbf{x}}$ is a vector with length 1 and it can be calculated by:

$$\hat{\mathbf{x}} = \frac{\mathbf{x}}{\|\mathbf{x}\|}$$

o

Maximising the Margin

You probably learned that an equation of a line is $\mathbf{y}=\mathbf{ax+b}$. However, you will often find that the equation of a hyperplane is defined by:

Note that

$$y = ax + b$$

is the same thing as

$$y - ax - b = 0$$

Given two vectors $\mathbf{w} \begin{pmatrix} -b \\ -a \\ 1 \end{pmatrix}$ and $\mathbf{x} \begin{pmatrix} 1 \\ x \\ y \end{pmatrix}$

$$\mathbf{w}^T \mathbf{x} = -b \times (1) + (-a) \times x + 1 \times y$$

$$\mathbf{w}^T \mathbf{x} = y - ax - b$$

The two equations are just two different ways of expressing the same thing.

For Support Vector Classifier (SVC), we use $\mathbf{w}^T \mathbf{x} + b$ where $\mathbf{w}$ is the weight vector, and b is the bias.

$$\mathbf{w}^T \mathbf{x} + b = 0$$

You can see that the name of the variables in the hyperplane equation are w and x , which means they are vectors! A vector has magnitude (size) and direction, which works perfectly well in 3 or more dimensions. Therefore, the application of “vector” is used in the SVMs algorithm.

$$(\mathbf{x}^+ - \mathbf{x}^-) \cdot \hat{\mathbf{w}} = (\mathbf{x}^+ - \mathbf{x}^-) \cdot \frac{\mathbf{w}}{\|\mathbf{w}\|} = \mathbf{x}^+ \cdot \frac{\mathbf{w}}{\|\mathbf{w}\|} - \mathbf{x}^- \cdot \frac{\mathbf{w}}{\|\mathbf{w}\|}$$

The equation of calculating the Margin.

Cost Function and Gradient Updates

Maximizing-Margin is equivalent to Minimizing Loss.

In the SVM algorithm, we are looking to maximize the margin between the data points and the hyperplane. The loss function that helps maximize the margin is hinge loss.

$$L(w) = \sum_{i=1} \underbrace{\max(0, 1 - y_i[w^T x_i + b])}_{\text{Loss function}} + \underbrace{\lambda \|w\|_2^2}_{\text{regularization}}$$

$\lambda=1/C$ (C is always used for regularization coefficient).

The function of the first term, hinge loss, is to penalize misclassifications. It measures the error due to misclassification (or data points being closer to the classification boundary than the margin). The second term is the regularization term, which is a technique to avoid overfitting by penalizing large coefficients in the solution vector. The λ (lambda) is the regularization coefficient, and its major role is to determine the trade-off between increasing the margin size and ensuring that the x_i lies on the correct side of the margin.

“Hinge” describes the fact that the error is 0 if the data point is classified correctly (and is not too close to the decision boundary).

When the true class is -1 (as in your example), the hinge loss looks like this in the graph.

We need to minimise the above loss function to find the max-margin classifier.

We can derive the formula for the margin from the hinge-loss. If a data point is on the margin of the classifier, the hinge-loss is exactly zero. Hence, on the margin, we have:

$$1 = [y_i + \mathbf{x}_i^T \mathbf{w}] \|\mathbf{w}\|$$

$$0 = ([y_i + \mathbf{x}_i^T \mathbf{w}] \|\mathbf{w}\| - 1) \lambda$$

Note that y_i is either +1 or -1.

Therefore, we have:

$$\mathbf{x}^+ \cdot \frac{\mathbf{w}}{\|\mathbf{w}\|} - \mathbf{x}^- \cdot \frac{\mathbf{w}}{\|\mathbf{w}\|} = \frac{1 - b}{\|\mathbf{w}\|} - \frac{-b - 1}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|}$$

Our objective function is then:

$$\max \frac{2}{\|\mathbf{w}\|} \rightarrow \max \frac{1}{\|\mathbf{w}\|} \rightarrow \min \|\mathbf{w}\| \rightarrow \min \frac{1}{2} \|\mathbf{w}\|^2$$

To minimize such an objection function, we should then use Lagrange Multiplier.

- o Classifying non-linear data

What about data points are not linearly separable?

Non-linear separate.

SVM has a technique called the kernel trick. These are functions that take low dimensional input space and transform it into a higher-dimensional space, i.e., it converts not separable problem to separable problem. It is mostly useful in non-linear separation problems. This is shown as follows:

4.1.1 DEPLOYMENT USING FLASK

Flask is a micro web framework written in Python. It is classified as a micro-framework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.

However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.

Flask was created by Armin Ronacher of Pocoo, an international group of Python enthusiasts formed in 2004. According to Ronacher, the idea was originally an April Fool's joke that was popular enough to make into a serious application. The name is a play on the earlier Bottle framework.

Flask is based on Werkzeug, Jinja2 and inspired by Sinatra Ruby framework, available under BSD licence. It was developed at pocoo by Armin Ronacher. Although Flask is rather young compared to most Python frameworks, it holds a great promise and has already gained popularity among Python web developers. Let's take a closer look into Flask, so-called "micro" framework for Python.

3.2 REQUIREMENTS AND TECHNOLOGIES USED:

Requirements are the basic constraints that are required to develop a system. Requirements are collected while designing the system. The following are the requirements that are to be discussed.

1. Functional requirements
2. Non-Functional requirements
3. Environment requirements
 - A. Hardware requirements
 - B. software requirements

3.2.1 Functional requirements:

The software requirements specification is a technical specification of requirements for the software product. It is the first step in the requirements analysis process. It lists requirements of a particular software system. The following details to follow the special libraries like sk-learn, pandas, numpy, matplotlib and seaborn.

3.2.2 Non-Functional Requirements:

Process of functional steps,

1. Problem define
2. Preparing data
3. Evaluating algorithms
4. Improving results
5. Prediction the result

3.2.3 Environmental Requirements:

1. Software Requirements:

Operating System : Windows

Tool : Anaconda with Jupyter Notebook

2. Hardware requirements:

Processor : Pentium IV/III

Hard disk : minimum 80 GB

RAM : minimum 2 GB

Jupyter Notebook App: The Jupyter Notebook App is a server-client application that allows editing and running notebook documents via a web browser. The Jupyter Notebook App can be executed on a local desktop requiring no internet access (as described in this document) or can be installed on a remote server and accessed through the internet. In addition to displaying/editing/running notebook documents, the Jupyter Notebook App

has a “Dashboard” (Notebook Dashboard), a “control panel” showing local files and allowing to open notebook documents or shutting down their kernels.

Notebook Dashboard: The *Notebook Dashboard* is the component which is shown first when you launch Jupyter Notebook App. The *Notebook Dashboard* is mainly used to open notebook documents, and to manage the running kernels (visualize and shutdown). The *Notebook Dashboard* has other features similar to a file manager, namely navigating folders and renaming/deleting files.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

The analytical process started from data cleaning and processing, missing value, exploratory analysis and finally model building and evaluation. The best accuracy on public test set is higher accuracy score is will be find out. This application can help to find the human diabetes problems.

Comparing Algorithm with prediction in the form of best accuracy result. 4 different algorithms are compared: Logistic Regression, Decision Tree, Random Forest, Support Vector Machine. Therefore Logistic Regression is the best model for this prediction since it has an accuracy score of 0.71.

5.2 Application

The predicting the Diabetes disease problem, ML algorithms prediction model is effective because of the following reasons: It provides better results in classification problem.

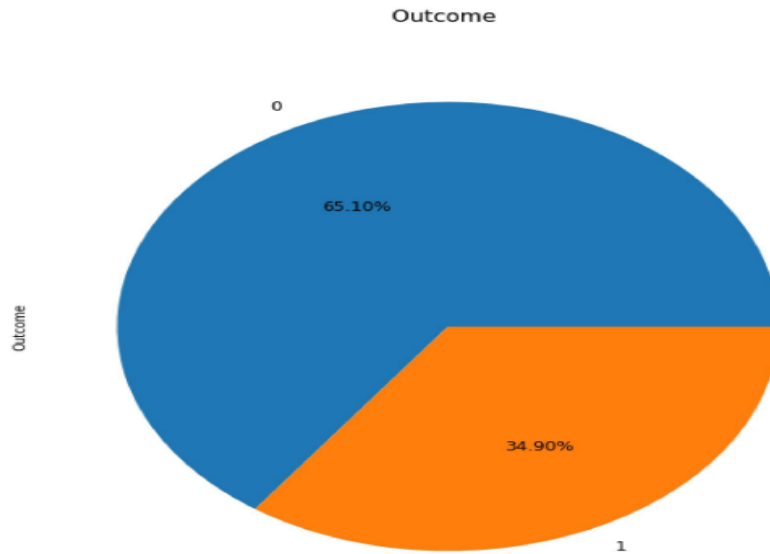


Fig 5.1 Pie Chart of Positive and Negative cases in Percentage

5.3 Future Work

Human diabetes problems connect with AI model.

To automate this process by show the prediction result in web application or desktop application.

To optimize the work to implement in Artificial Intelligence environment.

REFERENCES

- [1] Gauri D. Kalyankar, Shivananda R. Poojara and Nagaraj V. Dharwadkar," Predictive Analysis of Diabetic Patient Data Using Machine Learning and Hadoop", International Conference On I-SMAC, 978-1-5090-3243-3, 2017.
- [2] Ayush Anand and Divya Shakti," Prediction of Diabetes Based on Personal Lifestyle Indicators", 1st International Conference on Next Generation Computing Technologies, 978-1-4673-6809-4, September 2015.
- [3] B. Nithya and Dr. V. Ilango," Predictive Analytics in Health Care Using Machine Learning Tools and Techniques", International Conference on Intelligent Computing and Control Systems, 978-1-5386-2745-7, 2017.
- [4] Dr Saravana kumar N M, Eswari T, Sampath P and Lavanya S," Predictive Methodology for Diabetic Data Analysis in Big Data", 2nd International Symposium on Big Data and Cloud Computing, 2015.
- [5] Aiswarya Iyer, S. Jeyalatha and Ronak Sumbaly "Diagnosis of Diabetes Using Classification Mining Techniques", International Journal of Data Mining & Knowledge Management Process (IJDMP), 5 (1) (January 2015)
- [6] P. Suresh Kumar and S. Pranavi "Performance Analysis of Machine Learning Algorithms on Diabetes Dataset using Big Data Analytics", International Conference on Infocom Technologies and Unmanned Systems, 978-1-5386-0514-1, Dec. 18-20, 2017.
- [7] Mani Butwall, Shraddha Kumar "A Data Mining Approach for the Diagnosis of Diabetes Mellitus

using Random Forest Classifier”, International Journal of Computer Applications, 120 (Number 8, 2015)

[8] K. Rajesh, V. Sangeetha “Application of Data Mining Methods and Techniques for Diabetes Diagnosis” International Journal of Engineering and Innovative Technology (IJEIT), 2 (3) (September 2012)

[9] Humar Kahramanli, Novruz Allahverdi “Design of a Hybrid System for the Diabetes and Heart Disease” Expert Systems with Applications: An International Journal, 35 (1-2) (July, 2008).

[10] B.M. Patil, R.C. Joshi and Durga Toshniwal, “Association Rule for Classification of Type-2 Diabetic Patients”, ICMLC ‘10 Proceedings of the 2010 Second International Conference on Machine Learning and Computing, February 09 - 11, 2010.

[11] Dost Muhammad Khan¹, Nawaz Mohamudally² “An Integration of K-means and Decision Tree (ID3) towards a more Efficient Data Mining Algorithm ” Journal Of Computing, 3 (12) (December 2011).

APPENDIX

A. SAMPLE CODE

Module 1: Data validation and pre-processing technique

#import library packages

import pandas as pd

import numpy as np

#Load given dataset

data = pd.read_csv("diabetes.csv")

import warnings

warnings.filterwarnings('ignore')

Before drop the given dataset

data.head()

data.tail()

shape

data.shape

#size

```

data.size
data.info()
data.describe()
After drop the given dataset
# duplicated values
data.duplicated()
data.duplicated().sum()
#Checking null values
data.isnull()
# sum of null values
data.isnull().sum()
#drop the duplicated values
data = data.dropna()
# features of the dataset
data.columns
# unique valures of individual column
data.Pregnancies.unique()
data.BloodPressure.unique()
data.SkinThickness.unique()
data.BMI.unique()
data.Age.unique()
data.Outcome.unique()
#no of not affected peoples
data[data["Outcome"] == 0 ]
#number of not affected peoples
data[data["Outcome"] == 0 ].shape
#number of not affected peoples
data[data["Outcome"] == 1 ].shape
# Age details
print("Maximum Age :",data["Age"].max())

```

```

print("Minimum Age :",data["Age"].min())
print("Mean of Age :", "%.2f" % data["Age"].mean())
# BMI details
print("Maximum BMI :",data["BMI"].max())
print("Minimum BMI :",data["BMI"].min())
print("Mean of BMI :", "%.2f" % data["BMI"].mean())
# BMI details
print("Maximum BloodPressure :",data["BloodPressure"].max())
print("Minimum BloodPressure :",data["BloodPressure"].min())
print("Mean of BloodPressure :", "%.2f" % data["BloodPressure"].mean())
# Glucose details
print("Maximum Glucose :",data["Glucose"].max())
print("Minimum Glucose :",data["Glucose"].min())
print("Mean of Glucose :", "%.2f" % data["Glucose"].mean())
pd.Categorical(data["Pregnancies"]).describe()
data.corr()
df=data.corr()
df["Age"].sort_values(ascending=False)
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
var_mod = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
           'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome']

for i in var_mod:
    data[i] = le.fit_transform(data[i].astype(int))
df.head()

```

Module 2: Exploration data analysis of visualization and training a model by given attributes

#import library packages

```

import pandas as p
import matplotlib.pyplot as plt

import seaborn as s
import numpy as n

data = p.read_csv("diabetes.csv")

import warnings
warnings.filterwarnings('ignore')

data.head()

data.shape

#diabetes possitive reports

data.columns

#missing values

data.isnull().sum()


data["Outcome"].value_counts()

data.shape

#missing values by heatmap

plt.figure(figsize=(8,6))

s.heatmap(data.isnull(),annot = True)

plt.show()

# diadbetes pesion with blood pressure

plt.figure(figsize=(16,4))

plt.subplot(1,2,1)

plt.hist(data['BMI'])

plt.title("BMI")

plt.xlabel("BMI VRange")

plt.ylabel("count")

plt.subplot(1,2,2)

plt.hist(data['BloodPressure'])

```

```
plt.xlabel('BloodPressure')
plt.title('BloodPressure Range')
plt.ylabel("Count")
plt.show()
```

```
plt.figure(figsize=(16,6))
s.countplot(x="Age",data=data)
plt.show()
```

result based on age

```
plt.figure(figsize=(16,6))
s.countplot(x='Age',hue='Outcome',data=data)
plt.show()
plt.figure(figsize=(16,4))
```

```
plt.scatter(data['BMI'],data['Glucose'])
plt.xlabel("Age")
plt.ylabel("Glucose")
plt.show()
plt.figure(figsize=(16,4))
```

```
plt.scatter(data['BloodPressure'],data['Glucose'])
plt.xlabel('BloodPressure')
plt.ylabel("Glucose")
plt.show()
```

```
boxplot = data.boxplot(column = ['Glucose'], by = 'Outcome',figsize = (8,5), fontsize = 15)
plt.show()
```

```
boxplot = data.boxplot(column = ['BloodPressure'], by = 'Outcome',figsize = (8,5), fontsize = 15)
plt.show()
```

```
fig, ax = plt.subplots(figsize=(14,10))
s.heatmap(data.corr(),annot = True, fmt='0.2%',cmap = 'Reds',ax=ax)
s.boxplot(data['Glucose'])
plt.show()
```

```
s.boxplot(data["BloodPressure"])
plt.show()
data.columns
```

#Propagation by variable

```
def PropByVar(df, variable):
    dataframe_pie = df[variable].value_counts()
    ax = dataframe_pie.plot.pie(figsize=(10,10), autopct='%1.2f%%', fontsize = 12)
    ax.set_title(variable + '\n', fontsize = 15)
    return n.round(dataframe_pie/df.shape[0]*100,2)
```

```
PropByVar(data, 'Outcome')
data.head()
p.plotting.scatter_matrix(data, alpha=0.05, figsize=(18,10), diagonal='hist')
axis = 'off'
```

```
plt.show()
data.columns
```

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
var_mod = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
           'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome']
```

```
for i in var_mod:
    data[i] = le.fit_transform(data[i].astype(int))
```

```
data.head()
```

Module 3 : Performance measurements of Logistic regression

```
#import library packages
```

```
import pandas as p
```

```
import matplotlib.pyplot as plt
```

```
import numpy as n
```

```
import seaborn as sns
```

```
#Load given dataset
```

```
data = p.read_csv("diabetes.csv")
```

```
import warnings
```

```
warnings.filterwarnings('ignore')
```

```
data.head(5)
```

```
data.tail()
```

```
data["Outcome"].value_counts()
```

```
data.isnull().sum()
```

```
data.duplicated().sum()
```

```
df = data.dropna()
```

```
df.columns
```

```
data.shape
```

```
data.corr().describe()
```

```
data.corr()
```

```
data.head()
```

```
#labelencoding
```

```
from sklearn.preprocessing import LabelEncoder
```

```
var_mod=["Outcome"]
```

```
le = LabelEncoder()
```

```
for i in var_mod:
```

```

data[i] = le.fit_transform(data[i]).astype(int)

#According to the cross-validated MCC scores, the random forest is the best-performing model, so now let's evaluate its performance on the test set.

from sklearn.metrics import confusion_matrix, classification_report, matthews_corrcoef,
cohen_kappa_score, accuracy_score, average_precision_score, roc_auc_score

X = data.drop(labels='Outcome', axis=1)

#Response variable

y = data.loc[:, 'Outcome']

#We'll use a test size of 30%. We also stratify the split on the response variable, which is very important to do because there are so few fraudulent transactions.

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1,
stratify=y)

Logistic Regression :

from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score

logR= LogisticRegression()

logR.fit(X_train,y_train)

predictLR = logR.predict(X_test)

print("")
print('Classification report of Logistic Regression Results:')
print("")
print(classification_report(y_test,predictLR))
x = (accuracy_score(y_test,predictLR)*100)

print('Accuracy result of Logisticregression is:', x)

```

```

print("")

cm2=confusion_matrix(y_test,predictLR)

print('Confusion Matrix result of Logistic Regression is:\n',cm2)

print("")

sensitivity2 = cm2[0,0]/(cm2[0,0]+cm2[0,1])

print('Sensitivity : ', sensitivity2 )

print("")

specificity2 = cm2[1,1]/(cm2[1,0]+cm2[1,1])

print('Specificity : ', specificity2)

print("")

accuracy = cross_val_score(logR, X, y, scoring='accuracy')

print('Cross validation test results of accuracy:')

print(accuracy)

#get the mean of each fold

print("")

print("Accuracy result of Logistic Regression is:",accuracy.mean() * 100)

LR=accuracy.mean() * 100

def graph():

    import matplotlib.pyplot as plt

    data=[LR]

    alg="Logistic Regression"

    plt.figure(figsize=(5,5))

    b=plt.bar(alg,data,color=("b"))

    plt.title("Accuracy comparison Diabetes",fontsize=15)

    plt.legend(b,data,fontsize=9)

graph()

```

```

TN = cm2[1][0]
FN = cm2[0][0]
TP = cm2[1][1]
FP = cm2[0][1]

print("True Positive :",TP)
print("True Negative :",TN)
print("False Positive :",FP)
print("False Negative :",FN)
print("")

TPR = TP/(TP+FN)
TNR = TN/(TN+FP)
FPR = FP/(FP+TN)
FNR = FN/(TP+FN)

print("True Positive Rate :",TPR)
print("True Negative Rate :",TNR)
print("False Positive Rate :",FPR)
print("False Negative Rate :",FNR)
print("")

PPV = TP/(TP+FP)
NPV = TN/(TN+FN)

print("Positive Predictive Value :",PPV)
print("Negative predictive value :",NPV)


cm2=confusion_matrix(y_test, predictLR)
print(cm2)

sns.heatmap(cm2/n.sum(cm2), annot=True, cmap = 'Blues', annot_kws={"size": 16},
fmt='.2%',)

plt.show()

import joblib

```

```
joblib.dump(logR, 'LR.pkl')
```

Module 4 : Performance measurements of DecisionTree:

#import library packages

```
import pandas as p
```

```
import matplotlib.pyplot as plt
```

```
import numpy as n
```

```
import seaborn as sns
```

#Load given dataset

```
data = p.read_csv("diabetes.csv")
```

```
import warnings
```

```
warnings.filterwarnings('ignore')
```

```
data.head(5)
```

```
data.isnull().sum()
```

```
data.duplicated().sum()
```

```
df=data.dropna()
```

```
df.columns
```

```
data.shape
```

```
data.corr()
```

```
data.head()
```

```
data.head()
```

#According to the cross-validated MCC scores, the random forest is the best-performing model, so now let's evaluate its performance on the test set.

```
from sklearn.metrics import confusion_matrix, classification_report, matthews_corrcoef,  
cohen_kappa_score, accuracy_score, average_precision_score, roc_auc_score
```

```
X = data.drop(labels='Outcome', axis=1)
```

#Response variable

```
y = data.loc[:, 'Outcome']
```

#We'll use a test size of 30%. We also stratify the split on the response variable, which is very important to do because there are so few fraudulent transactions.

```

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1,
stratify=y)
Decision Tree Classifier :
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import cross_val_score

DT=DecisionTreeClassifier()

DT.fit(X_train,y_train)

predictDT = DT.predict(X_test)

print("")
print('Classification report DecisionTree classifier Results:')
print("")
print(classification_report(y_test,predictDT))

print("")
x = (accuracy_score(y_test,predictDT)*100)

print('Accuracy result of DecisionTree is:', x)
print("")

cm2=confusion_matrix(y_test,predictDT)
print('Confusion Matrix result of DecissionTree Classifier is:\n',cm2)
print("")
sensitivity2 = cm2[0,0]/(cm2[0,0]+cm2[0,1])
print('Sensitivity : ', sensitivity2 )

```

```

print("")
specificity2 = cm2[1,1]/(cm2[1,0]+cm2[1,1])
print('Specificity : ', specificity2)
print("")

accuracy = cross_val_score(DT, X, y, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)
#get the mean of each fold
print("")
print("Accuracy result of DecisionTree Classifier is:",accuracy.mean() * 100)
dt=accuracy.mean() * 100

def graph():
    import matplotlib.pyplot as plt
    data=[dt]
    alg="Decision Tree"
    plt.figure(figsize=(5,5))
    b=plt.bar(alg,data,color=("b"))
    plt.title("Accuracy comparison of Diabetes",fontsize=15)
    plt.legend(b,data,fontsize=9)

graph()
TN = cm2[1][0]
FN = cm2[0][0]
TP = cm2[1][1]
FP = cm2[0][1]
print("True Positive :",TP)
print("True Negative :",TN)
print("False Positive :",FP)

```

```

print("False Negative :",FN)

print("")

TPR = TP/(TP+FN)

TNR = TN/(TN+FP)

FPR = FP/(FP+TN)

FNR = FN/(TP+FN)

print("True Positive Rate :",TPR)

print("True Negative Rate :",TNR)

print("False Positive Rate :",FPR)

print("False Negative Rate :",FNR)

print("")

PPV = TP/(TP+FP)

NPV = TN/(TN+FN)

print("Positive Predictive Value :",PPV)

print("Negative predictive value :",NPV)


cm2=confusion_matrix(y_test, predictDT)

print(cm2)

sns.heatmap(cm2/n.sum(cm2), annot=True, cmap = 'Blues', annot_kws={"size": 16},
fmt='.2%',)

plt.show()

```

Module 5 : Performance measurements of Random Forest algorithms

#import library packages

```

import pandas as p

import matplotlib.pyplot as plt

import numpy as n

import seaborn as sns

```

#Load given dataset

```

data = p.read_csv("diabetes.csv")

import warnings
warnings.filterwarnings('ignore')

data.head(5)

data.isnull().sum()

data.duplicated().sum()

df.columns

data.shape

data.corr()

data.corr().describe()

data.head()

#According to the cross-validated MCC scores, the random forest is the best-performing model, so now let's evaluate its performance on the test set.

from sklearn.metrics import confusion_matrix, classification_report, matthews_corrcoef,
cohen_kappa_score, accuracy_score, average_precision_score, roc_auc_score

X = data.drop(labels='Outcome', axis=1)

#Response variable

y = data.loc[:, 'Outcome']

#We'll use a test size of 30%. We also stratify the split on the response variable, which is very important to do because there are so few fraudulent transactions.

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1,
stratify=y)

Random Forest:

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score, confusion_matrix

from sklearn.model_selection import cross_val_score

rfc = RandomForestClassifier()

```

```

rfc.fit(X_train,y_train)

predictR = rfc.predict(X_test)

print("")
print('Classification report of Random Forest Results:')
print("")

print(classification_report(y_test,predictR))
x = (accuracy_score(y_test,predictR)*100)

print('Accuracy result of Random Forest is:', x)
print("")
cm1=confusion_matrix(y_test,predictR)
print('Confusion Matrix result of Random Forest is:\n',cm1)
print("")
sensitivity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])
print('Sensitivity : ', sensitivity1 )
print("")
specificity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])
print('Specificity : ', specificity1)
print("")

accuracy = cross_val_score(rfc, X, y, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)
#get the mean of each fold
print("")
print("Accuracy result of Random Forest is:",accuracy.mean() * 100)
RFC=accuracy.mean() * 100

```

```

def graph():
    import matplotlib.pyplot as plt
    data=[RFC]
    alg="Random orest"
    plt.figure(figsize=(5,5))
    b=plt.bar(alg,data,color=("b"))
    plt.title("Accuracy comparison of Diabetes")
    plt.legend(b,data,fontsize=9)

```

```

graph()
TN = cm1[1][0]
FN = cm1[0][0]
TP = cm1[1][1]
FP = cm1[0][1]
print("True Positive :",TP)
print("True Negative :",TN)
print("False Positive :",FP)
print("False Negative :",FN)
print("")
TPR = TP/(TP+FN)
TNR = TN/(TN+FP)
FPR = FP/(FP+TN)
FNR = FN/(TP+FN)
print("True Positive Rate :",TPR)
print("True Negative Rate :",TNR)
print("False Positive Rate :",FPR)
print("False Negative Rate :",FNR)
print("")

```

```

PPV = TP/(TP+FP)
NPV = TN/(TN+FN)

print("Positive Predictive Value :",PPV)
print("Negative predictive value :",NPV)

def plot_confusion_matrix(cm2, title='Confusion matrix-RF', cmap=plt.cm.Blues):
    target_names=[]
    plt.imshow(cm2, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = n.arange(len(target_names))
    plt.xticks(tick_marks, target_names, rotation=45)
    plt.yticks(tick_marks, target_names)
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

cm2=confusion_matrix(y_test, predictR)
print('Confusion matrix-RF:')
print(cm2)
plot_confusion_matrix(cm2)

```

Module 6 : Performance measurements of Support Vector Machines algorithms

#import library packages

```

import pandas as p
import matplotlib.pyplot as plt
import numpy as n
import seaborn as s

```

#Load given dataset

```

data = p.read_csv("diabetes.csv")

import warnings
warnings.filterwarnings('ignore')

data.head(5)

data.isnull().sum()

data.duplicated().sum()

df=data.dropna()

df.columns

data.shape

data.corr()

data.corr().describe()

data.head()

#According to the cross-validated MCC scores, the random forest is the best-performing model, so now let's evaluate its performance on the test set.

from sklearn.metrics import confusion_matrix, classification_report, matthews_corrcoef,
cohen_kappa_score, accuracy_score, average_precision_score, roc_auc_score

X = data.drop(labels='Outcome', axis=1)

#Response variable

y = data.loc[:, 'Outcome']

#We'll use a test size of 30%. We also stratify the split on the response variable, which is very important to do because there are so few fraudulent transactions.

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1,
stratify=y)

```

Support Vector Machines :

```

from sklearn.svm import SVC

from sklearn.metrics import accuracy_score, confusion_matrix

from sklearn.model_selection import cross_val_score

```

```
s = SVC()
```

```

s.fit(X_train,y_train)
predicts = s.predict(X_test)

print("")
print('Classification report of Support Vector Machines Results:')
print("")

print(classification_report(y_test,predicts))
x = (accuracy_score(y_test,predicts)*100)

print('Accuracy result of Support Vector Machines is:', x)
print("")
cm2=confusion_matrix(y_test,predicts)
print('Confusion Matrix result of Support Vector Machines is:\n',cm2)
print("")
sensitivity1 = cm2[0,0]/(cm2[0,0]+cm2[0,1])
print('Sensitivity : ', sensitivity1 )
print("")
specificity1 = cm2[1,1]/(cm2[1,0]+cm2[1,1])
print('Specificity : ', specificity1)
print("")

accuracy = cross_val_score(s, X, y, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)
#get the mean of each fold
print("")
print("Accuracy result of Support Vector Machine is:",accuracy.mean() * 100)
S=accuracy.mean() * 100

```

```

def graph():
    import matplotlib.pyplot as plt
    data=[S]
    alg="Support Vector Machine"
    plt.figure(figsize=(5,5))
    b=plt.bar(alg,data,color=("b"))
    plt.title("Accuracy comparison of Diabetes")
    plt.legend(b,data,fontsize=9)
graph()
TN = cm2[1][0]
FN = cm2[0][0]
TP = cm2[1][1]
FP = cm2[0][1]
print("True Positive :",TP)
print("True Negative :",TN)
print("False Positive :",FP)
print("False Negative :",FN)
print("")
TPR = TP/(TP+FN)
TNR = TN/(TN+FP)
FPR = FP/(FP+TN)
FNR = FN/(TP+FN)
print("True Positive Rate :",TPR)
print("True Negative Rate :",TNR)
print("False Positive Rate :",FPR)
print("False Negative Rate :",FNR)
print("")
PPV = TP/(TP+FP)
NPV = TN/(TN+FN)
print("Positive Predictive Value :",PPV)

```

```

print("Negative predictive value :",NPV)

def plot_confusion_matrix(cm2, title='Confusion matrix-SVM', cmap=plt.cm.Blues):
    target_names=[]
    plt.imshow(cm2, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(target_names))
    plt.xticks(tick_marks, target_names, rotation=45)
    plt.yticks(tick_marks, target_names)
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
cm2=confusion_matrix(y_test, predicts)
print('Confusion matrix-SVM:')
print(cm2)
plot_confusion_matrix(cm2)

```

Flask deploy

```

import numpy as np
from flask import Flask, request, jsonify, render_template
import pickle
import joblib
app = Flask(__name__)
model = joblib.load('LR.pkl')

@app.route('/')
def home():
    return render_template('index.html')

```

```

@app.route('/predict',methods=['POST'])
def predict():
    """
    For rendering results on HTML GUI
    """
    int_features = [(x) for x in request.form.values()]
    final_features = [np.array(int_features)]
    print(final_features)
    prediction = model.predict(final_features)

    output = prediction[0]
    print(output)
    if output == 0:
        output = 'Not Affected'
    elif output == 1:
        output = 'Affected'

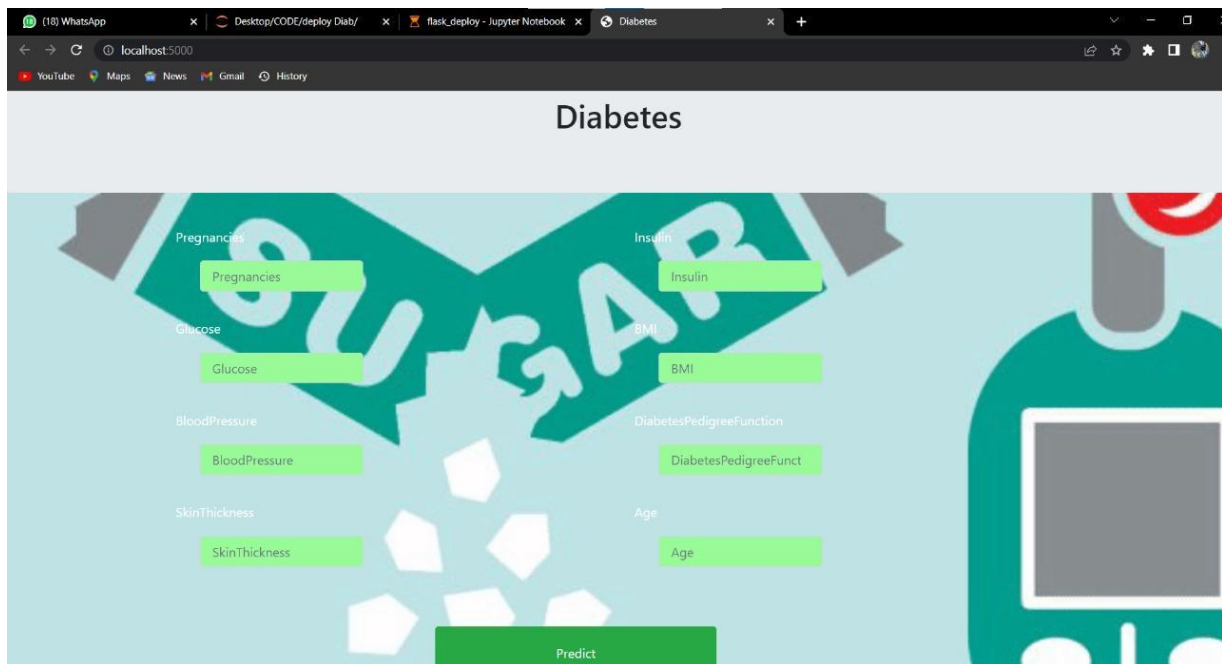
    return render_template('index.html', prediction_text='Diabetes:
{}'.format(output))

if __name__ == "__main__":
    app.run(host="localhost", port=5006)

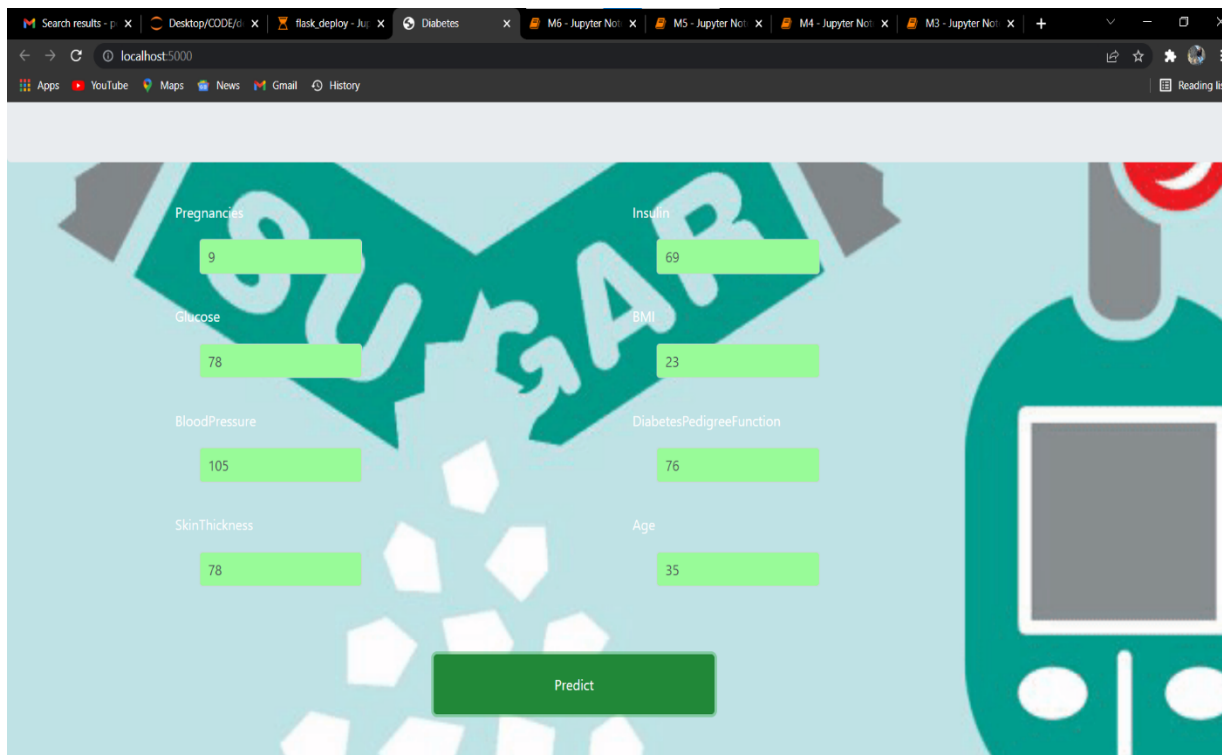
```

B. SCREEN SHORT

1. Output screen



2. After entering the values



3. Predicted result

Search results x Desktop/CODE x flask_deploy - Jupyter x Diabetes x M6 - Jupyter No x M5 - Jupyter No x M4 - Jupyter No x M3 - Jupyter No x +

localhost:5000/predict

Apps YouTube Maps News Gmail History

Pregnancies

Glucose

BloodPressure

SkinThickness

Insulin

BMI

DiabetesPedigreeFunction

Age

Predict

Diabetes: Not Affected

