

IDENTIFICATION AND PREDICTION DIABETES BY USING MACHINE LEARNING

Submitted in partial fulfillment of the requirements
for the award of
Bachelor of Engineering degree in information technology

by

TAMILINIYAN P (Reg.No.40120097)
SANTHOSH J (Reg.No.40120081)



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

APRIL 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

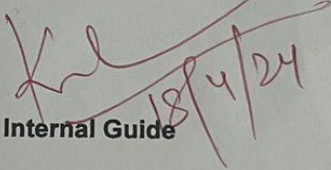
Accredited "A++" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF INFORMATION TECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the Bonafide work of **TAMILINIYAN P (40120097)** and **SANTHOSH J (40120081)** who carried out the Project entitled "**IDENTIFICATION AND PREDICTION OD DIABETES BY USING MACHINE LEARNING CONCEPTS**" under my supervision from January 2024 to April 2024.

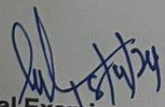

Internal Guide

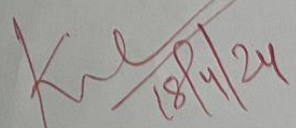
Dr. KAMATCHI K S, M.E., Ph.D.,


Head of the Department

Dr. R. SUBHASHINI, M.E., Ph.D.,

Submitted for Viva-voce Examination held on 18. 04. 2024


Internal Examiner

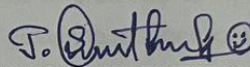

External Examiner

DECLARATION

I, **SANTHOSH J (40120081)** hereby declare that the Report entitled **"IDENTIFICATION AND PREDICTION OD DIABETES BY USING MACHINE LEARNING CONCEPTS** done by me under the guidance of **Dr. KAMATCHI KS, M.E., Ph.D.**, is submitted in partial fulfillment of the requirements for the award of Bachelor of Technology degree in **Information Technology**.

DATE: 18.04.2024

PLACE: Chennai



SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

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

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

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

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

ABSTRACT

Nowadays every industry is relying on machine learning technology in order to predict, analyze their growth, trend or an issue. The most prominent field is medical, where we can use dataset to predict the risk factor of any disease. This paper mainly concentrates on diabetes. Diabetes is a disorder where the body does not produce insulin or does not use it efficiently. A huge medical dataset is accessible in different data repositories, which we use for the analysis. We acquired data from University of California Irvine which has 520 instances with 17 attributes. The objective of this is to find whether a patient is likely to get affected with the diabetes or not, this is predicted based on symptoms possessed by the patient. We proposed a machine learning model using Logistic regression and random forest. Among the algorithms these two are the most interpretable. The data is pre-processed and normalized in such a way that it is ready for training. Further this data is split in testing and training into 8:1 proportion. Based upon model performance accuracy is evaluated using various parameters such as accuracy, recall and precision. Thus, dataset is the most important aspect in the architecture and this data is undergone many machine learning techniques and fitted into algorithms in order to achieve desired results.

TABLE OF CONTENTS

	ABSTRACT	v
	LIST OF FIGURES	viii
CHAPTER NO.	TITLE	PAGE NO.
1.	INTRODUCTION	1
	1.1 BUILDING A MACHINE LEARNING MACHINE	1
	1.2 METHODS	3
	1.2.1 SUPERVISED MACHINE LEARNING	4
	1.2.2 UNSUPERVISED MACHINE LEARNING	4
	1.2.3 SEMI-SUPERVISED MACHINE LEARNING	4
	1.2.4 REINFORCEMENT MACHINE LEARNING	5
	1.3 DEEP LEARNING	5
	1.4 REAL-WORLD ML USE CASES	5
2.	LITERATURE SURVEY	7
3	SYSTEM DESIGN AND METHODOLOGY	9
	3.1 EXISTING SYSTEM	9
	3.2 PROPOSED SYSTEM	9
	3.3 SYSTEM ARCHITECTURE	10
	3.3.1 DATA COLLECTION	10
	3.3.2 DATA PREPROCESSING	11
	3.3.2.1 DATA QUALITY ASSESSMENT	12
	3.3.2.2 BALANCED AND IMBALANCED DATASET	14
	3.3.2.3 FEATURE DISTRIBUTION ANALYSIS	16
	3.3.2.4 CORRELATION	19
	3.3.3 CLASSIFICATION ALGORITHMS	20
	3.3.4 MODEL EVALUATION TECHNIQUES	22
4	RESULTS AND DISCUSSION	26
5	CONCLUSION AND FUTURE WORK	31
	REFERENCES	32
	APPENDIX	33
	A. PLAGARISM REPORT	33
	B. PAPER	34
	C. SOURCE CODE	38

LIST OF FIGURES

FIGURE NO.	FIGURE NAME	PAGE NO.
3.1	SYSTEM ARCHITECTURE	10
3.2	CHECKING MISSING VALUES	12
3.3	DATASET AFTER BINARY CONVERSION	14
3.4	BALANCED DATASET (PIE CHART)	16
3.5	DISTRIBUTION OF GENDER	17
3.6	DISTRIBUTION OF POLYURIA	17
3.7	DISTRIBUTION OF POLYDIPSIA	17
3.8	DISTRIBUTION OF SUDDEN WEIGHT LOSS	17
3.9	DISTRIBUTION OF POLYPHAGIA	18
3.1.1	DISTRIBUTION OF GENTIAL THRUSH	18
3.1.2	DISTRIBUTION OF VISUAL BLURRING	18
3.1.3	DISTRIBUTION OF ITCHING	18
3.1.4	DISTRIBUTION OF IRRITABILITY	18
3.1.5	DISTRIBUTION OF DELAYED HEALING	18
3.1.6	DISTRIBUTION OF PARTIAL PARESIS	18
3.1.7	DISTRIBUTION OF MUSCLE STIFFNESS	18
3.1.8	DISTRIBUTION OF ALOPECIA	18
3.1.9	DISTRIBUTION OF OBESITY	18
3.2.1	CORRELATION OF DIABETES	20
3.2.2	LOGISTIC REGRESSION GRAPH REPRESENTATION	22
4	TRAIN AND TEST DATASET SPLIT	26
4.1	CONVERSION OF BINARY FORM OF AGE	26
4.2	SCORE AFTER PREDICTION	27
4.3	CONFUSION MATRIX OF LOGISTIC CLASSIFIER	27
4.4	RELATIVE FEATURE IMPORTANCE	28
4.5	SCORE BEFORE TUNING (RANDOM FOREST)	28
4.6	CONFUSION MATRIX OF RANDOM FOREST	28
4.7	TOP 10 SELECTED FEATURES	29
4.8	SCORE AFTER POST FEATURE SELECTION	29
4.9	RECEIVER OPERATING CHARACTERISTIC	30

CHAPTER 1

INTRODUCTION

Machine learning is a branch of artificial intelligence (AI) focused on building applications that learn from data and improve their accuracy over time without being programmed to do so. In data science, an algorithm is a sequence of statistical processing steps. In machine learning, algorithms are 'trained' to find patterns and features in massive amounts of data in order to make decisions and predictions based on new data. The better the algorithm, the more accurate the decisions and predictions will become as it processes more data.

1.1. BUILDING A MACHINE LEARNING MODEL:

There are four basic steps for building a machine learning application (or model). These are typically performed by data scientists working closely with the business professionals for whom the model is being developed.

Step 1: Select and prepare a training data set

Training data is a data set representative of the data the machine learning model will ingest to solve the problem it's designed to solve. In some cases, the training data is *labeled data*— 'tagged' to call out features and classifications the model will need to identify. Other data is *unlabeled*, and the model will need to extract those features and assign classifications on its own.

In either case, the training data needs to be properly prepared—randomized, de-duped, and checked for imbalances or biases that could impact the training. It should also be divided into two subsets: the *training subset*, which will be used to train the application, and the *evaluation subset*, used to test and refine it.

Step 2: Choose an algorithm to run on the training data set

Again, an algorithm is a set of statistical processing steps. The type of algorithm depends on the type (labeled or unlabeled) and amount of data in the training data set and on the type of problem to be solved.

Common types of machine learning algorithms for use with labeled data include the following:

- **Regression algorithms:** Linear and logistic regression are examples of regression algorithms used to understand relationships in data. Linear regression is used to predict the value of a dependent variable based on the value of an independent variable. Logistic regression can be used when the dependent variable is binary in nature: A or B. For example, a linear regression algorithm could be trained to predict a salesperson's annual sales (the dependent variable) based on its relationship to the salesperson's education or years of experience (the independent variables.) Another type of regression algorithm called a support vector machine is useful when dependent variables are more difficult to classify.
- **Decision trees:** Decision trees use classified data to make recommendations based on a set of decision rules. For example, a decision tree that recommends betting on a particular horse to win, place, or show could use data about the horse (e.g., age, winning percentage, pedigree) and apply rules to those factors to recommend an action or decision.
- **Instance-based algorithms:** A good example of an instance-based algorithm is K-Nearest Neighbor or k-nn. It uses classification to estimate how likely a data point is to be a member of one group or another based on its proximity to other data points.

Algorithms for use with unlabeled data include the following:

- **Clustering algorithms:** Think of clusters as groups. Clustering focuses on identifying groups of similar records and labeling the records according to the group to which they belong. This is done without prior knowledge about the groups and their characteristics. Types of clustering algorithms include the K-means, TwoStep, and Kohonen clustering.

- **Association algorithms:** Association algorithms find patterns and relationships in data and identify frequent ‘if-then’ relationships called *association rules*. These are similar to the rules used in data mining.
- **Neural networks:** A neural network is an algorithm that defines a layered network of calculations featuring an input layer, where data is ingested; at least one hidden layer, where calculations are performed make different conclusions about input; and an output layer, where each conclusion is assigned a probability. A deep neural network defines a network with multiple hidden layers, each of which successively refines the results of the previous layer. (For more, see the “Deep learning” section below.)

Step 3: Training the algorithm to create the model

Training the algorithm is an iterative process—it involves running variables through the algorithm, comparing the output with the results it should have produced, adjusting weights and biases within the algorithm that might yield a more accurate result, and running the variables again until the algorithm returns the correct result most of the time. The resulting trained, accurate algorithm is the machine learning model—an important distinction to note, because ‘algorithm’ and ‘model’ are incorrectly used interchangeably, even by machine learning maven.

Step 4: Using and improving the model

The final step is to use the model with new data and, in the best case, for it to improve in accuracy and effectiveness over time. Where the new data comes from will depend on the problem being solved. For example, a machine learning model designed to identify spam will ingest email messages, whereas a machine learning model that drives a robot vacuum cleaner will ingest data resulting from real-world interaction with moved furniture or new objects in the room.

1.2. METHODS:

1.2.1. SUPERVISED MACHINE LEARNING: Supervised machine learning trains itself on a labeled data set. That is, the data is labeled with information that the machine learning model is being built to determine and that may even be classified in ways the model is supposed to classify data. For example, a computer vision model designed to identify purebred German Shepherd dogs might be trained on a data set of various labeled dog images. Supervised machine learning requires less training data than other machine learning methods and makes training easier because the results of the model can be compared to actual labeled results. But, properly labeled data is expensive to prepare, and there's the danger of overfitting, or creating a model so closely tied and biased to the training data that it doesn't handle variations in new data accurately.

1.2.2. UNSUPERVISED MACHINE LEARNING:

Unsupervised machine learning ingests unlabeled data—lots and lots of it—and uses algorithms to extract meaningful features needed to label, sort, and classify the data in real-time, without human intervention. Unsupervised learning is less about automating decisions and predictions, and more about identifying patterns and relationships in data that humans would miss. Take spam detection, for example—people generate more email than a team of data scientists could ever hope to label or classify in their lifetimes. An unsupervised learning algorithm can analyze huge volumes of emails and uncover the features and patterns that indicate spam (and keep getting better at flagging spam over time).

1.2.3. UNSUPERVISED MACHINE LEARNING:

Unsupervised machine learning ingests unlabeled data—lots and lots of it—and uses algorithms to extract meaningful features needed to label, sort, and classify the data in real-time, without human intervention. Unsupervised learning is less about automating decisions and predictions, and more about identifying patterns and relationships in data that humans would miss. Take spam detection, for example—people generate more email than a team of data scientists could ever hope to label or classify in their lifetimes. An unsupervised learning algorithm can analyze huge volumes of emails and uncover the features and patterns that indicate spam (and keep getting better at flagging spam over time).

1.2.4. SEMI-SUPERVISED MACHINE LEARNING:

Semi-supervised learning offers a happy medium between supervised and unsupervised learning. During training, it uses a smaller labeled data set to guide classification and feature extraction from a larger, unlabeled data set. Semi-supervised learning can solve the problem of having not enough labeled data (or not being able to afford to label enough data) to train a supervised learning algorithm.

1.2.5. REINFORCEMENT MACHINE LEARNING:

Reinforcement machine learning is a behavioral machine learning model that is similar to supervised learning, but the algorithm isn't trained using sample data. This model learns as it goes by using trial and error. A sequence of successful outcomes will be reinforced to develop the best recommendation or policy for a given problem.

1.3. DEEP LEARNING:

Deep learning is a subset of machine learning (all deep learning is machine learning, but not all machine learning is deep learning). Deep learning algorithms define an artificial neural network that is designed to learn the way the human brain learns. Deep learning models require large amounts of data that pass through multiple layers of calculations, applying weights and biases in each successive layer to continually adjust and improve the outcomes. Deep learning models are typically unsupervised or semi-supervised. Reinforcement learning models can also be deep learning models. Certain types of deep learning models—including convolutional neural networks (CNNs) and recurrent neural networks (RNNs)—are driving progress in areas such as computer vision, natural language processing (including speech recognition), and self-driving cars.

1.4. REAL-WORLD ML USE CASES:

As noted at the outset, machine learning is everywhere. Here are just a few examples of machine learning you might encounter every day:

- **Digital assistants:** Apple Siri, Amazon Alexa, Google Assistant, and other digital assistants are powered by natural language processing (NLP), a machine learning application that enables computers to process text and voice data and 'understand' human language the way people do. Natural language processing also drives voice-driven applications like GPS and speech recognition (speech- to-text) software.
- **Recommendations:** Deep learning models drive 'people also liked' and 'just for you' recommendations offered by Amazon, Netflix, Spotify, and other retail, entertainment, travel, job search, and news services.

- **Contextual online advertising:** Machine learning and deep learning models can evaluate the content of a web page—not only the topic, but nuances like the author's opinion or attitude—and serve up advertisements tailored to the visitor's interests.
- **Chatbots:** Chatbots can use a combination of pattern recognition, natural language processing, and deep neural networks to interpret input text and provide suitable responses.
- **Fraud detection:** Machine learning regression and classification models have replaced rules-based fraud detection systems, which have a high number of false positives when flagging stolen credit card use and are rarely successful at detecting criminal use of stolen or compromised financial data.
- **Cybersecurity:** Machine learning can extract intelligence from incident reports, alerts, blog posts, and more to identify potential threats, advise security analysts, and accelerate response.
- **Medical image analysis:** The types and volume of digital medical imaging data have exploded, leading to more available information for supporting diagnoses but also more opportunity for human error in reading the data. Convolutional neural networks (CNNs), recurrent neural networks (RNNs), and other deep learning models have proven increasingly successful at extracting features and information from medical images to help support accurate diagnoses.
- **Self-driving cars:** Self-driving cars require a machine learning tour de force— they must continuously identify objects in the environment around the car, predict how they will change or move, and guide the car around the objects as well as toward the driver's destination. Virtually every form of machine learning and deep learning algorithm mentioned above plays some role in enabling a self-driving automobile.

CHAPTER 2

LITERATURE SURVEY

K. Vijay Kumar, and others [11] proposed random Forest algorithm for the prediction of diabetes to develop a system that can predict early diabetes for a more accurately-focused patient by using the machine-learning Random Forest algorithm. The model suggested provides the best results in diabetic prediction and the results demonstrated that the prediction system can efficiently, efficiently and most importantly, immediately predict diabetes disease.

The predictive onset of diabetes was presented to Nanos Nnamoko et al [13]. A supervised learning approach is used to combine their results with five widely used classifications. The results will be presented and compared to similar studies using the same dataset in the book. Diabetes beginning can be predicted with greater accuracy by using the proposed method. Roof N. The goal is to avoid diabetes by three numerous regulated methods of learning, namely SVM, logistic regression, and ANN, proposed by Joshi et al [11]. Predicting Diabetes Prediction Using Machine Learning Techniques. The project advocates an effective method for early diabetes detection.

Deeraj Shetty et al [15] suggested prediction of diabetic disease using data extraction, assembling a prediction method of intelligent diabetes that offers analysis to patients with diabetes via a database of diabetes. In this system, algorithms such as Bayesian and KNN are proposed to be applied to the database of patients with diabets and analysed by taking various diabetes attributes for diabetes prediction.

Accomplished by replacementing sing-values for mean. K-means used to remove data from the helpful vector machine, and genetic algorithms for the optimum identifying eatures (SVM). In each run, GA selected different attributes and the rating accuracy was recorded from the origingal set of attributes. The experiment was replicated 50 times to obtain a stable outcome and the findings were seen in the

table. The experimental findings indicate that the suggested model has an average accuracy of 98,79 per cent for Pima Indian diabetes in the UCI archive. In contrast to the SVM clustered K-means Data Preparation Scheme, stronger findings have also been obtained in the proposed approach (96.71 percent).

We are currently evaluating the same data packet split into the same training and prediction sets, both using a logistic regression and a linear perceptron model [5]. We expect to use all the threematics - ADAP, logistic regression and a linear perceptron to compare the ROC curves for this forecast. This paper is used by the amalgamated approach and obtained with greater accurate results in the compact formation of uncertainty matrix as a linear regression Data mining is a method that gathers valuable knowledge from a vast quantity of information that helps you to obtain more information. The use of data mining tools and methods thus leads to the diabetes prediction and thus decreases costs for care. In order to diagnose diabetes and discover effective approaches to also manage them, data mining methods are used in the field of medicinal data.

In this article [6], we used the neighbouring algorithm K-nearest to diagnose diabetes mellitus. The findings revealed that, as k value rises, accuracy and error rates have improved. We measured the correctness and error rate of $K = 3,5$. KNN is one of the most powerful and commonly diagnosed artificial intelligence algorithms. many classification methods have been used in the data mining process. More precise and effective results are obtainable via KNN. These methods were used to split the data into separate sets such that a relationship could be effectively defined with different attributes. Other data mining techniques we use three different classificational algorithms that are used in this experiment: ZeroR, OneR and Naiv. Different data mining techniques are used to help health professionals diagnose diabetes. This experiment reveals the quickest Naïve Bayes and the slowest ZeroR. A faulty data mining technique is used to identify efficiency comparisons. Each model shall be transformed into rules and these rules shall be incorporated. This study article deals

with machine learning, which has a sophisticated data structure and an extensive epidemiological and genetic diabetes risk base to revolutionise the estimation of diabetes risk. This paper addresses

Machine Learning. Diabetes diagnosis is the answer to recovery in the early stages. This dissertation outlines a machine-learning solution to diabetes prediction. The methodology can also help researchers build a precise and efficient method to help them make better decisions on the disease status of clinicians.

Diabetes in young people and ancient cultures was influenced by Amina Azaret al. in [7]. These are raised day by day and it is not curable. For early prediction, data mining is used. This paper in key purpose is offers the distinction and recommend best algorithm. You will use the PID datasets. In order to predict early diabetes diagnosis in highest precision and performance, decisions tree, Naive Bayes and K-Nearest neighbouring algorithms are contrasted and used. The WEKA is used for testing and validation using rapid miner. The decision tree is the perfect algorithm for predictions. It offers 75.65 percent of the precis

CHAPTER 3

SYSTEM DESIGN AND METHODOLOGY

3.1. EXISTING SYSTEM:

There are many machine learning approaches for the prediction of diabetes. The only important aspect that differ from model to model is the data. The data plays a major role in the prediction. Currently, the machine learning models can only predict whether a patient is diabetic or not after collecting required instances from the blood sample. This practice is not so effective because patient may prefer taking an instant test over this computerized diagnosis. The positive side is that the attained accuracy is similar to lab diagnosis. In practical these kinds of machine diagnosis can be replaced. Various Machine Learning Techniques provide efficient result to collect Knowledge by building various classification and ensemble models from collected dataset. Such collected data can be useful to predict diabetes. Various techniques of Machine Learning can capable to do prediction, however it's tough to choose best technique. Thus, for this purpose we apply popular classification and ensemble methods on dataset for prediction.

3.2. PROPOSED MODEL:

The dataset contains the sign and symptom data of newly diabetic or would-be diabetic patient. This consists of total 15 features and one target variable named class. The libraries such as pandas, numpy, matplotlib and sklearn are used. The data is pre-processed and normalized is such a way that it is ready for training. Further this data is split in testing and training into 8:1 proportion. Based upon model performance accuracy is evaluated using various parameters such as accuracy, recall and precision. The data is fitted into two algorithms called Logistic regression and random forest. Logistic regression is mainly based on the concept of probability which is used for classification problems and considered to be predictive analysis algorithm. Random forest is a dual- purpose algorithm where it is used for both classification and regression.

3.3. SYSTEM ARCHITECTURE: The system design consists of following steps:

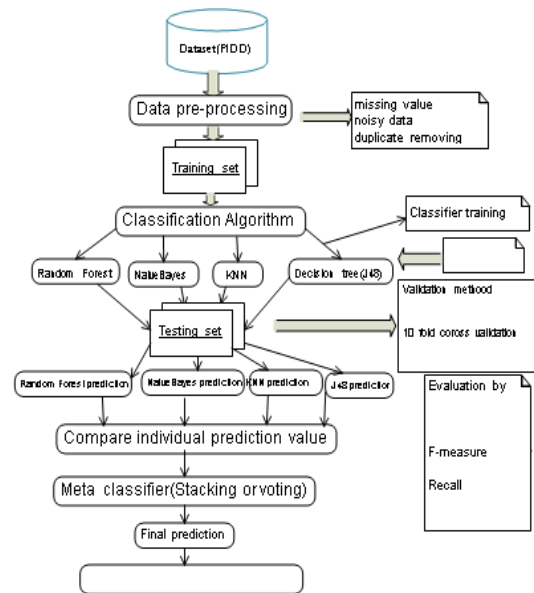


Fig 3.1. System Architecture

3.3.1. Data Collection:

Given the problem you want to solve, you will have to investigate and obtain data that you will use to feed your machine. The quality and quantity of information you get are very important since it will directly impact how well or badly your model will work. You may have the information in an existing database or you must create it from scratch. If it is a small project, you can create a spreadsheet that will later be easily exported as a CSV file.

Table 3.3.1.1: Features of dataset before Data Processing

It is also common to use the web scraping technique to automatically collect information from various sources such as APIs. The dataset contains the sign and symptom data of newly diabetic or would be diabetic patient. This consists of total 15 features and one target variable named class. Dataset is the most important aspect in the architecture. This data is undergone data preprocessing which is a data mining technique.

3.3.2. Data Preprocessing:

Data Preprocessing is a data mining technique which is used to transform the raw data in a useful and efficient format. Features can be in the following pattern or format:

- **Categorical:** Features whose values are taken from a defined set of values. For instance, days in a week: {Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday} is a category because its value is always taken from this set. Another example could be the Boolean set: {True, False}
- **Numerical:** Features whose values are continuous or integer-valued. They are represented by numbers and possess most of the properties of numbers. For instance, number of steps you walk in a day, or the speed at which you are driving your car at.

3.3.2.1. Now that we have gone over the basics, let us begin with the steps of Data Preprocessing. Remember, not all the steps are applicable for each problem, it is highly dependent on the data we are working with, so maybe only a few steps might be required with your dataset. Generally, they are: Data quality Assessment:

Because data is often taken from multiple sources which are normally not too reliable and that too in different formats, more than half our time is consumed in dealing with data quality issues when working on a machine learning problem. It is simply unrealistic to expect that the data will be perfect. There may be problems due to human error, limitations of measuring devices, or flaws in the data collection process. Let's go over a few of them and methods to deal with them:

1. **Missing Values:** It is very much usual to have missing values in your dataset. It may have happened during data collection, or maybe due to some data validation rule, but regardless missing values must be taken into consideration.
 - **Eliminate rows with missing data:**
Simple and sometimes effective strategy. Fails if many objects have missing values. If a feature has mostly missing values, then that feature itself can also be eliminated.
 - **Estimate missing values:**

2. **Inconsistent values:** We know that data can contain inconsistent values. Most probably we have already faced this issue at some point. For instance, the `_Address` field contains the `_Phone number`. It may be due to human error or maybe the information was misread while being scanned from a handwritten form. It is therefore always advised to perform data assessment like knowing what the data type of the features should be and whether it is the same for all the data objects.
3. **Duplicate values:** A dataset may include data objects which are duplicates of one another. It may happen when say the same person submits a form more than once. The term deduplication is often used to refer to the process of dealing with duplicates. In most cases, the duplicates are removed so as to not give that particular data object an advantage or *bias*, when running machine learning algorithms.

Table 3.3.: Features of dataset after Data Processing

3.3.2.2. Balanced and Imbalanced dataset:

Let's take a simple example if in our data set, we have positive values which are approximately same as negative values. Then we can say our dataset is in balance. If there is a very high difference between the positive values and negative values. Then we can say our dataset is an Imbalance Dataset.

3.3.2.2.1. Techniques to convert Imbalanced dataset to Balanced Dataset:

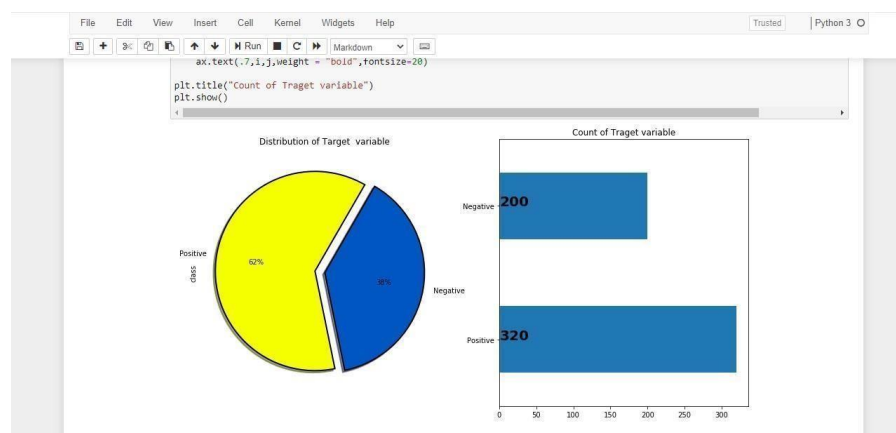
1. **Over-sampling:** This technique is used to modify the unequal data classes to create balanced datasets. When the quantity of data is insufficient, the oversampling method tries to balance by incrementing the size of rare samples. Over-sampling increases the number of minority class members in the training set. The advantage of over-sampling

is that no information from the original training set is lost, as all observations from the minority and majority classes are kept. On the other hand, it is prone to over fitting.

2. **Under-sampling:** Unlike oversampling, this technique balances the imbalance dataset by reducing the size of the class which is in abundance. There are various methods for classification problems such as cluster centroids and Tomek links. The cluster centroid methods replace the cluster of samples by the cluster centroid of a K-means algorithm and the Tomek link method removes unwanted overlap between classes until all minimally distanced nearest neighbors are of the same class.
3. **Feature Selection:** In order to tackle the imbalance problem, we calculate the one-sided metric such as correlation coefficient (CC) and odds ratios (OR) or two-sided metric evaluation such as information gain (IG) and chi-square (CHI) on both the positive class and negative class. Based on the scores, we then identify the significant features from each class and take the union of these features to obtain the final set of features. Then, we use this data to classify the problem. Identifying these features will help us generate a clear decision boundary with respect to each class. This helps the models to classify the data more accurately. This performs the function of intelligent subsampling and potentially helps reduce the imbalance problem.
4. **Feature Selection:** In order to tackle the imbalance problem, we calculate the one-sided metric such as correlation coefficient (CC) and odds ratios (OR) or two-sided metric evaluation such as information gain (IG) and chi-square (CHI) on both the positive class and negative class. Based on the scores, we then identify the significant features from each class and take the union of these features to obtain the final set of features. Then, we use this data to classify the problem. Identifying these features will help us generate a clear decision boundary with respect to each class. This helps the

models to classify the data more accurately. This performs the function of intelligent subsampling and potentially helps reduce the imbalance problem.

5. **Cost Sensitive Learning Technique:** The Cost-Sensitive Learning (CSL) takes the misclassification costs into consideration by minimizing the total cost. The goal of this technique is mainly to pursue a high accuracy of classifying examples into a set of known classes. It is playing as one of the important roles in the machine learning algorithms including the real-world data mining applications.
6. **Ensemble Learning Technique:** The ensemble-based method is another technique which is used to deal with imbalanced data sets, and the ensemble technique is combined the result or performance of several classifiers to improve the performance of single classifier. This method modifies the generalization ability of individual classifiers by assembling various classifiers. It mainly combines the outputs of multiple
7. base learners. There are various approaches in ensemble learning such as Bagging, Boosting, etc.



3.3.2.3. Feature Distribution Analysis:

The distribution of a feature over its range, with value on the horizontal axis and frequency on the vertical axis. For string and categorical encoded integers, the feature distribution is displayed as a bar chart with the highest, i.e., the label with the highest count, on the left side. For float, float tensor, not categorical encoded integer and image the feature distribution is displayed as a histogram. This is done by dividing the range into a number of bins and calculate how many data points fall within the boundaries of each bin. If you hover over a bin, you'll see the count and the range for that bin. The feature distribution helps in understanding what kind of feature you are dealing with, and what values you can expect this feature to have. You'll see if the values are centered or scattered. Why is the distribution important? Because deep learning models learn from data. Give them incorrect data and they will learn incorrectly. A deep learning model is only as good as the data you feed it. You can also see if the training and validation subsets differ if they look pretty much the same that's good. the data you feed it. You can also see if the training and validation subsets differ if they look pretty much the same that's good.

3.3.2.3.1. Normal Distribution: If a feature can be seen as a random variable, and enough data is used and the bins are narrow enough, the look of the distribution could be bell-shaped. This is called a normal distribution (or standard/Gaussian). The normal distribution is a very important probability distribution that arises in many situations. Generally speaking, when you have a large number of independent samples from a naturally occurring phenomenon, the data will follow a normal distribution. Groups of data might form lumps a little off to one side. Such lumps can be of special significance and worth having a further look at.

3.3.2.3.2. Erroneous Data: By looking at the distribution you can detect erroneous data such as outliers, these are single data points laying very far away from the rest of the data. If you have outliers or other erroneous data in the dataset you could try to normalize it or create a new subset to filter out the data. If this doesn't work you should remove the outliers outside the Platform and import the dataset again.

Out[10]:

	class	Negative	Positive
Gender			
Female		9.5	54.06
Male		90.5	45.94

Fig 3.5. Distribution of Gender

Out[12]:

	class	Negative	Positive
Polyuria			
No		92.5	24.06
Yes		7.5	75.94

Fig 3.6. Distribution of Polyuria

Out[14]:

	class	Negative	Positive
Polydipsia			
No		96	29.69
Yes		4	70.31

Out[15]:

	class	Negative	Positive
sudden weight loss			
No		85.5	41.25
Yes		14.5	58.75

Out[39]:

	class	Negative	Positive
Polyphagia			
No		76	40.94
Yes		24	59.06

Fig 3.9. Distribution of Polyphagia

Out[46]:

	class	Negative	Positive
visual blurring			
No		71	45.31
Yes		29	54.69

Fig 3.1.2. Distribution of Visual Blurring

Out[51]:

	class	Negative	Positive
Irritability			
No		92	65.62
Yes		8	34.38

Fig 3.1.4. Distribution of Irritability

Out[53]:

	class	Negative	Positive
partial paresis			
No		84	40
Yes		16	60

Fig 3.1.6. Distribution of Partial Paresis

Out[55]:

	class	Negative	Positive
Alopecia			
No		49.5	75.62
Yes		50.5	24.38

Fig 3.1.8 Distribution of Alopecia

Out[43]:

	class	Negative	Positive
Genital thrush			
No		83.5	74.06
Yes		16.5	25.94

Fig 3.1.1. Distribution of Genital thrush

Out[50]:

	class	Negative	Positive
Itching			
No		50.5	51.88
Yes		49.5	48.12

Fig 3.1.3. Distribution of Itching

Out[52]:

	class	Negative	Positive
delayed healing			
No		57	52.19
Yes		43	47.81

Fig 3.1.5. Distribution of Delayed Healing

Out[54]:

	class	Negative	Positive
muscle stiffness			
No		70	57.81
Yes		30	42.19

Fig 3.1.7. Distribution of Muscle Stiffness

Out[56]:

	class	Negative	Positive
Obesity			
No		86.5	80.94
Yes		13.5	19.06

Fig 3.1.9 Distribution of Obesity

3.3.2.4. Correlation:

Correlation is one of the most widely used — and widely misunderstood — statistical concepts. In this overview, we provide the definitions and intuition behind several types of correlation and illustrate how to calculate correlation using the Python pandas library. The term "correlation" refers to a mutual relationship or association between quantities. In almost any business, it is useful to express one quantity in terms of its relationship with others. For example, sales might increase when the marketing department spends more on TV advertisements, or a customer's average purchase amount on an e-commerce website might depend on a number of factors related to that customer. Often, correlation is the first step to understanding these relationships and subsequently building better business and statistical models.

3.3.2.4.1. Types of Correlation: Before we go into the details of how correlation is calculated, it is important to introduce the concept of *covariance*. Covariance is a statistical measure of association between two variables X and Y . First, each variable is centered by subtracting its mean. These centered scores are multiplied together to measure whether the increase in one variable associates with the increase in another. Finally, expected value (E) of the product of these centered scores is calculated as a summary of association. Intuitively, the product of centered scores can be thought of as the area of a rectangle with each point's distance from the mean describing a side of the rectangle:

$$\text{Cov}(X, Y) = E[(X - E[X])(Y - E[Y])]$$

If both variables tend to move in the same direction, we expect the "average" rectangle connecting each point (X_i, Y_i) to the means $(\bar{X}, \bar{Y})$ to have a large and positive diagonal vector, corresponding to a larger positive product in the equation above. If both variables tend to move in opposite directions, we expect the average rectangle to have a

the equation above. If the variables are unrelated, then the vectors should, on average, cancel out — and the total diagonal vector should have a magnitude near 0, corresponding to a product near 0 in the equation above.

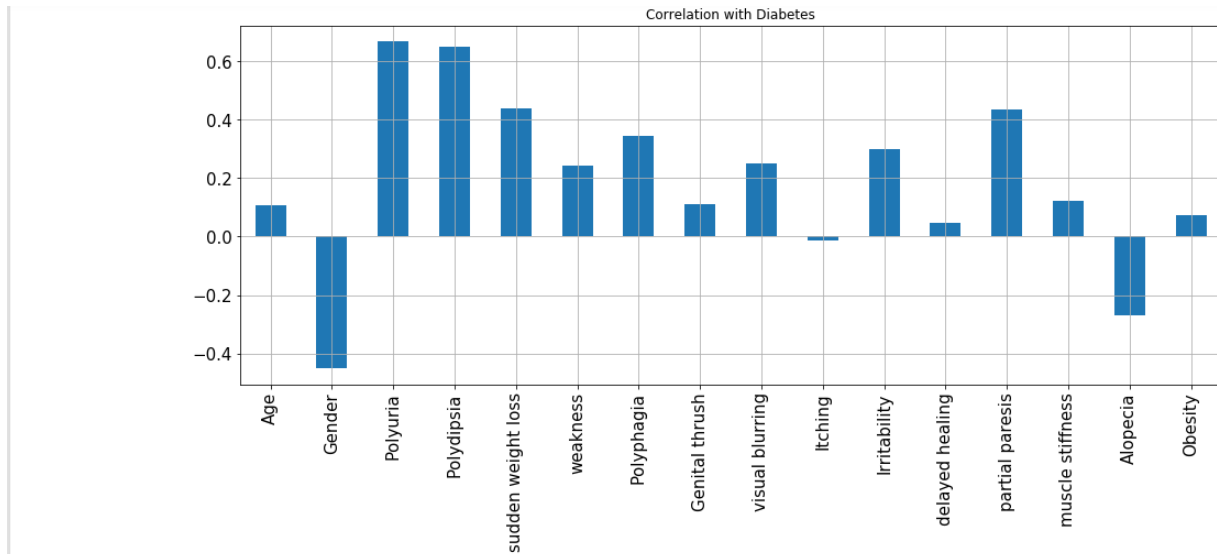


Fig 3.2.1. Correlation with diabetes

3.3.3. Classification Algorithms:

1. Linear Regression: It is used to estimate real values (cost of houses, number of calls, total sales etc.) based on continuous variable(s). Here, we establish relationship between independent and dependent variables by fitting a best line. This best fit line is known as regression line and represented by a linear equation $Y = a * X + b$. The best way to understand linear regression is to relive this experience of childhood. Let us say, you ask a child in fifth grade to arrange people in his class by increasing order of weight, without asking them their weights! What do you think the child will do? He /she would likely look (visually analyze) at the height and build of people and arrange them using a combination of these visible parameters. This is linear regression in real

life! The child has actually figured out that height and build would be correlated to the weight by a relationship, which looks like the equation above. Linear Regression is mainly of two types: Simple Linear Regression and Multiple Linear Regression. Simple Linear Regression is characterized by one independent variable. And, Multiple Linear Regression (as the name suggests) is characterized by multiple (more than 1) independent variables. While finding the best fit line, you can fit a polynomial or curvilinear regression. And these are known as polynomial or curvilinear regression.

2. **Logistic Regression:** Don't get confused by its name! It is a classification not a regression algorithm. It is used to estimate discrete values (Binary values like 0/1, yes/no, true/false) based on given set of independent variables. In simple words, it predicts the probability of occurrence of an event by fitting data to a logit function. Hence, it is also known as logit regression. Since, it predicts the probability, its output values lie between 0 and 1 (as expected). Let's say your friend gives you a puzzle to solve. There are only 2 outcome scenarios – either you solve it or you don't. Now imagine, that you are being given wide range of puzzles / quizzes in an attempt to understand which subjects you are good at. The outcome to this study would be something like this – if you are given a trigonometry based tenth grade problem, you are 70% likely to solve it. On the other hand, if it is grade fifth history question, the probability of getting an answer is only 30%. This is what Logistic Regression provides you.

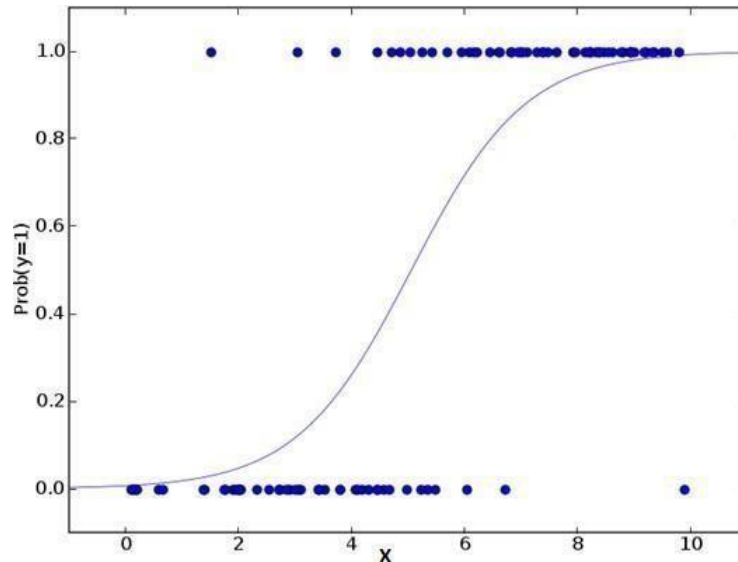


Fig 3.2.2. Logistic Regression Graph Representation

3. Random Forest:

Random Forest is a trademark term for an ensemble of decision trees. In Random Forest, we've collection of decision trees (so known as "Forest"). To classify a new object based on attributes, each tree gives a classification and we say the tree "votes" for that class. The forest chooses the classification having the most votes (over all the trees in the forest). Each tree is planted & grown as follows:

1. If the number of cases in the training set is N , then sample of N cases is taken at random but *with replacement*. This sample will be the training set for growing the tree.
2. If there are M input variables, a number $m \ll M$ is specified such that at each node, m variables are selected at random out of the M and the best split on these m is used to split the node. The value of m is held constant during the forest growing.
3. Each tree is grown to the largest extent possible. There is no pruning.

3.3.4. Model Evaluation Techniques:

Common Terms:

- **True positives (TP):** Predicted positive and are actually positive.

- **False positives (FP):** Predicted positive and are actually negative.
- **True negatives (TN):** Predicted negative and are actually negative.
- **False negatives (FN):** Predicted negative and are actually positive.

1. **Accuracy:** The most commonly used metric to judge a model and is actually not a clear indicator of the performance. The worse happens when classes are imbalanced.

$$\frac{TP + TN}{TP + FP + TN + FN}$$

Take for example a cancer detection model. The chances of actually having cancer are very low. Let's say out of 100, 90 of the patients don't have cancer and the remaining 10 actually have it. We don't want to miss on a patient who is having cancer but goes undetected (false negative). Detecting everyone as not having cancer gives an accuracy of 90% straight. The model did nothing here but just gave cancer free for all the 100 predictions.

2. **Precision:** Percentage of positive instances out of the *total predicted positive* instances. Here denominator is the model prediction done as positive from the whole given dataset. Take it as to find out *how much the model is right when it says it is right*.

$$\frac{TP}{TP + FP}$$

3. **Recall:** Percentage of positive instances out of the *total actual positive* instances. Therefore denominator ($TP + FN$) here is the *actual* number of positive instances present in the dataset. Take it as to find out *how much extra right ones, the model missed when it showed the right ones*.

$$\frac{TP}{TP + FN}$$

4. **Specificity:** Percentage of negative instances out of the total actual negative instances. Therefore denominator (TN + FP) here is the actual number of negative instances present in the dataset. It is similar to recall but the shift is on the negative instances. Like finding out how many healthy patients were not having cancer and were told they don't have cancer. Kind of a measure to see how separate the classes are.

$$\frac{TN}{TN + FP}$$

5. **F1 Score:** It is the harmonic mean of precision and recall. This takes the contribution of both, so higher the F1 score, the better. See that due to the product in the numerator if one goes low, the final F1 score goes down significantly. So, a model does well in F1

$$\frac{2}{\frac{1}{precision} + \frac{1}{recall}} = \frac{2 * precision * recall}{precision + recall}$$

score if the positive predicted are actually positives (precision) and doesn't miss out on positives and predicts them negative (recall).

One drawback is that both precision and recall are given equal importance due to which according to our application we may need one higher than the other and F1 score may not be the exact metric for it. Therefore, either weighted-F1 score or seeing the PR or ROC curve can help.

6. **ROC Curve:** ROC stands for receiver operating characteristic and the graph is plotted against TPR and FPR for various threshold values. As TPR increases FPR also increases. As you can see in the first figure, we have four categories and we want the threshold value that leads us closer to the top left corner. Comparing different predictors (here 3) on a given dataset also becomes easy as you can see in figure 2, one can choose the threshold according to the application at hand. ROC AUC is just the area under the curve, the higher its numerical value the better.

CHAPTER 4

RESULTS AND DISCUSSION

The primary approach starts with importing libraries such as pandas, numpy, matplotlib and sklearn. Every dataset needs to be examined initially in order to find whether any null values are present in the dataset or not and to find whether it is balanced. The dataset will be considered imbalanced when 90% of the single class attribute is positive and 10% is negative. In this case it is 60%-40% which is said to be a balanced data set. Thereby each and every symptom's distribution is analyzed and compared with the attribute. Initially the dataset is in the form of textual which needs to be converted into binary form; this is done using a method called `isna()`. Since we are importing pandas, finding correlation is an easy task because it is an inbuilt feature. Thereby the data now is ready for computation, which is split in two different proportions for training and testing.

```
In [24]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, stratify=y, random_state = 1234)

In [25]: ## checking distribution of target variable in train test split
print('Distribution of target variable in training set')
print(y_train.value_counts())

print('Distribution of target variable in test set')
print(y_test.value_counts())

Distribution of target variable in training set
1    256
0    160
Name: class, dtype: int64
Distribution of target variable in test set
1     64
0     40
Name: class, dtype: int64
```

Fig 4. Train and Test data split

Except the age every instance is converted into binary also the age is converted in data normalization stage using minmax scalar method.

Out[27]:

	Age	Gender	Polyuria	Polydipsia	sudden weight loss	weakness	Polyphagia	Genital thrush	visual blurring	Itching	Irritability	delayed healing	partial paresis	muscle stiffness	Alopecia
246	0.405405	1	0	0	0	1	0	0	0	1	0	1	0	0	1
274	0.554054	1	0	0	0	0	1	0	1	0	0	0	0	1	0
424	0.364865	1	1	1	1	1	0	1	0	0	0	0	0	0	0
316	0.527027	0	0	0	0	1	0	1	0	1	0	1	1	0	1
159	0.297297	0	1	1	1	1	1	0	1	1	1	1	1	1	0

Fig 4.1. Conversion of binary form of age

Thereby manipulation of data gets completed. The base machine learning model is logistic regression. We have used $\text{penalty}='l2'$ (ridge) which is regularization parameter and adds squared magnitudes of bids for minimizing the coefficients of lost function. After fitting this model, we cross validated using kfold model selection where 9-fold and 1-fold iteration takes place in order to improve the model capacity and found the confusion matrix of the classifier.

Out[30]:

	Model	Accuracy	Cross Val Accuracy	Precision	Recall	F1 Score	ROC
0	Logistic Regression	0.971154	0.918118	0.984127	0.96875	0.976378	0.971875

Fig 4.2. Score after prediction

Considering these results relative feature importance is visualized using bar graph.

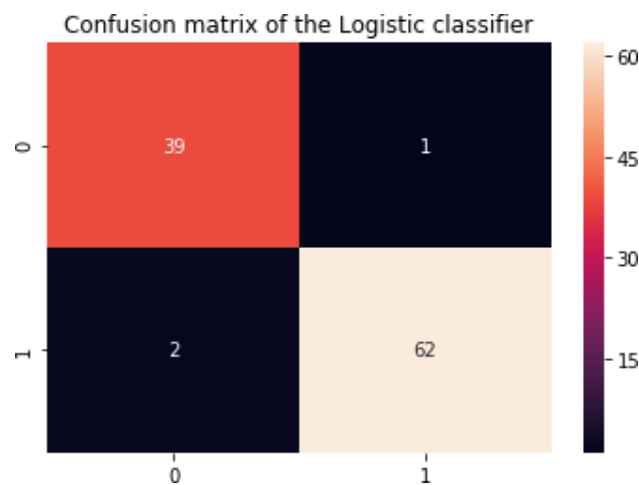


Fig 4.3. Confusion matrix of Logistic classifier

The second algorithm is random forest, the operations are done before tuning. We have used `_gini` as the criterion because it is faster when compared with another criterion. Similarly, cross validation is done and evaluation with respect to other parameters, thus accuracy is displayed.

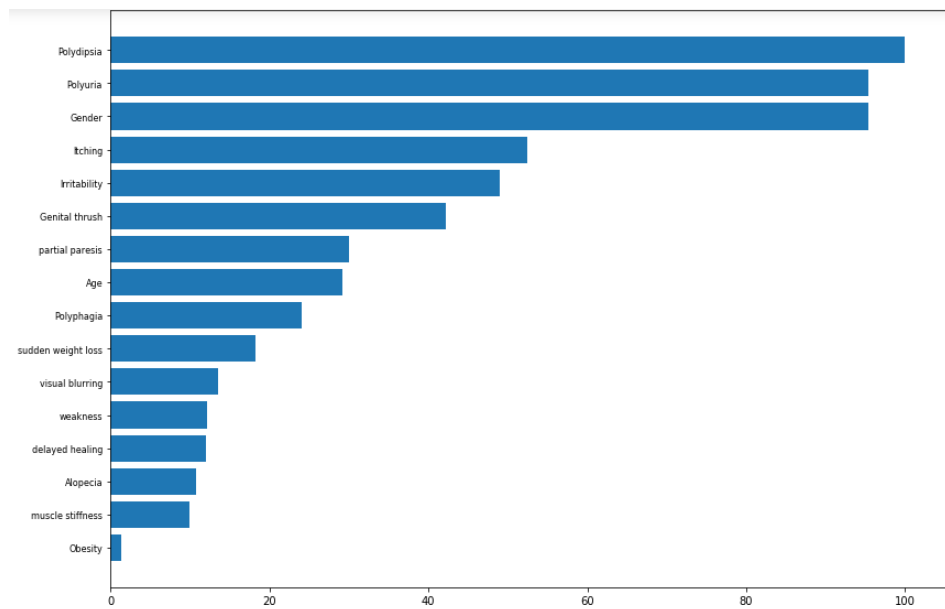


Fig 4.4. Relative feature importance

Since the accuracy is higher than the logistic regression, we have selected the top 10 features from the random forest. The entire process is again repeated in the similar pattern where split of data, cross validation and confusion matrix is performed.

Out[36]:

	Model	Accuracy	Cross Val Accuracy	Precision	Recall	F1 Score	ROC
0	Logistic Regression	0.971154	0.918118	0.984127	0.96875	0.976378	0.971875
1	Random Forest (Untuned)	0.980769	0.978223	1.000000	0.96875	0.984127	0.984375

Fig 4.5. Score before tuning (Random forest)

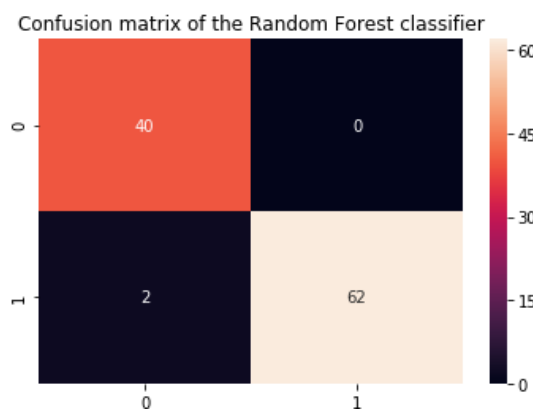


Fig 4.6. Confusion matrix of the Random forest classifier

Similarly, when the process repeated without tuning (random forest) we got highest accuracy. Since the accuracy has increased, top 10 features are selected and applied to the model.

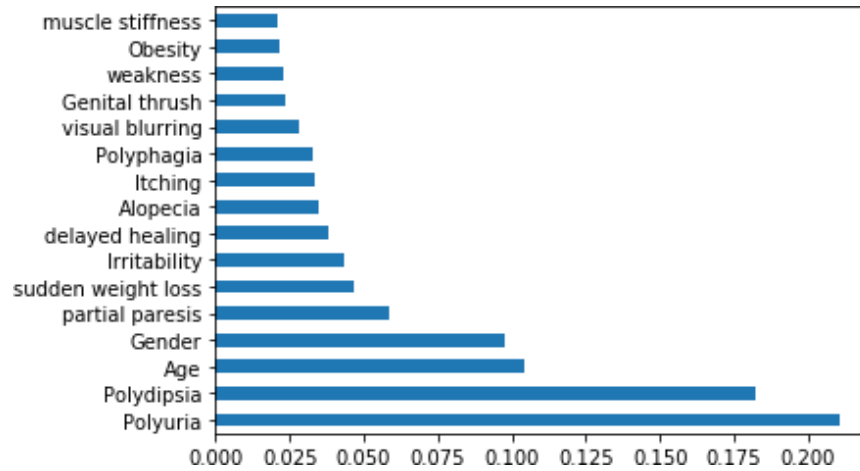


Fig 4.7. Top 10 selected features

The entire process is repeated after selection of features, considered to be post-selection evaluation. The accuracy got dropped to 2% in logistic regression and 1% random forest.

	Model	Accuracy	Cross Val Accuracy	Precision	Recall	F1 Score	ROC
0	Logistic Regression	0.971154	0.918118	0.984127	0.968750	0.976378	0.971875
1	Random Forest (Untuned)	0.980769	0.978223	1.000000	0.968750	0.984127	0.984375
2	Logistic Regression-Post FS	0.961538	0.898839	0.983871	0.953125	0.968254	0.964063
3	Random Forest- Post FS	0.980769	0.968641	1.000000	0.968750	0.984127	0.984375

Fig 4.8. Score after post feature selection

Finally, the ROC of both the models (logistic regression and random forest) are plotted on a graph with respect to specificity and sensitivity, where true positive rate is represented via Y-axis and false positive rate is represented via X-axis.

CHAPTER 5

CONCLUSION AND FUTURE WORK

CONCLUSION:

This computerized diagnosis at initial stage can help the doctors to guide and take required measures in order to stay out of risk. Since accuracy is not compromised at any stage of building the model, it is safe to use or follow. It is highly recommended to follow the instructions by the physician.

FUTURE WORK:

We focused only on Type 1 diabetes in this project. There are multiple types of diabetes which in further we wish to predict also, the consequences due to this disease are wider which can be prevented. This might require an accurate dataset and to find new patterns in it using data mining techniques.

REFERENCES

- [1] Alade, O.M., Sowunmi, O.Y., Misra, S., Maskeliūnas, R., Damaševičius, R.: A neural network based expert system for the diagnosis of diabetes mellitus. In: International Conference on Information Technology Science, pp. 14-22. Springer, Cham (Dec 2017)
- [2] Argentiero, P., Chin, R., Beaudet, P.: An automated approach to the design of decision tree classifiers. *IEEE Trans. Pattern Anal. Mach. Intell.* **1**, 51-57 (1982)
- [3] Breiman, L.: Random forests. *Mach. Learn.* **45**(1), 5-32 (2001)
- [4] Carrera, E.V., González, A., Carrera, R.: Automated detection of diabetic retinopathy using SVM. In: 2017 IEEE XXIV International Conference on Electronics, Electrical Engineering and Computing (INTERCON). IEEE (2017)
- [5] Cortes, C., Vapnik, V.: Support-vector networks. *Mach. Learn.* **20**(3), 273-297 (1995)
- [6] Khalil, R.M., Al-Jumaily, A.: Machine learning based prediction of depression among type 2 diabetic patients. In: 2017 12th International Conference on Intelligent Systems and Knowledge Engineering (ISKE), pp. 1-5. Nanjing (2017)
- [7] Kaveeshwar, S.A., Cornwall, J.: The current state of diabetes mellitus in India. *Australas. Med. J.* **7**(1), 45 (2014)
- [8] Lee, B.J., Kim, J.Y.: Identification of type 2 diabetes risk factors using phenotypes consisting of anthropometry and triglycerides based on machine learning. *IEEE J. Biomed. Health Inform.* **20**(1), 39-46 (2016)
- [9] Mohan, V., et al.: Epidemiology of type 2 diabetes: Indian scenario. *Indian J. Med. Res.* **125**(3), 217 (2007)
- [10] Sumangali, K., Geetika, B. S. R., Ambarkar, H.: A classifier-based approach for early detection of diabetes mellitus. In: 2016 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT). IEEE (2016)

APPENDIX

A.SOURCE CODE:

```
#!/usr/bin/env python
# coding: utf-8

# # Diabetes Prediction using Machine Learning
#
# Diabetes, is a group of metabolic disorders in which there are high blood sugar levels over a
# prolonged period. Symptoms of high blood sugar include frequent urination, increased
# thirst, and increased hunger. If left untreated, diabetes can cause many complications.
# Acute complications can include diabetic ketoacidosis, hyperosmolar hyperglycemic
# state, or death. Serious long-term complications include cardiovascular disease, stroke,
# chronic kidney disease, foot ulcers, and damage to the eyes.
#
# This dataset is originally from the National Institute of Diabetes and Digestive and Kidney
# Diseases. The objective of the dataset is to diagnostically predict whether or not a
# patient has diabetes, based on certain diagnostic measurements included in the
# dataset. Several constraints were placed on the selection of these instances from a
# larger database. In particular, all patients here are females at least 21 years old of Pima
# Indian heritage.
#
# ### **Objective**
#
# We will try to build a machine learning model to accurately predict whether or not the patients
# in the dataset have diabetes or not?
#
# ### **Details about the dataset:**
#
# The datasets consists of several medical predictor variables and one target variable,
# Outcome. Predictor variables includes the number of pregnancies the patient has had,
# their BMI, insulin level, age, and so on.
#
# - **Pregnancies**: Number of times pregnant
# - **Glucose**: Plasma glucose concentration a 2 hours in an oral glucose tolerance test
# - **BloodPressure**: Diastolic blood pressure (mm Hg)
```

```

# - **SkinThickness**: Triceps skin fold thickness (mm)
# - **Insulin**: 2-Hour serum insulin (mu U/ml)
# - **BMI**: Body mass index (weight in kg/(height in m)^2)
# - **DiabetesPedigreeFunction**: Diabetes pedigree function
# - **Age**: Age (years)
# - **Outcome**: Class variable (0 or 1)
#
# **Number of Observation Units: 768**
#
# **Variable Number: 9**
#
# ### **Machine Learning Workflow**
# ##### Steps
# 1. Data gathering
# 2. Data preparation
# 3. Exploratory Data Analysis
# 4. Data Preprocessing
# 5. Data Transformation
# 6. Model Building
# 7. Model Evaluation

# ## **1) Data Preparation and Exploratory Data Analysis**

# In[207]:

#Installation of required libraries
import numpy as np
import pandas as pd

```

```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import scale, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
import warnings
warnings.simplefilter(action = "ignore")
```

```
# In[208]:
```

```
#Reading the dataset
df = pd.read_csv("diabetes.csv")
```

```
# In[209]:
```

```
# The first 5 observation units of the data set were accessed.
df.head()
```

```
# In[210]:
```

The size of the data set was examined. It consists of 768 observation units and 9 variables.

```
df.shape
```

In[211]:

#Feature information

```
df.info()
```

In[212]:

Descriptive statistics of the data set accessed.

```
df.describe([0.10,0.25,0.50,0.75,0.90,0.95,0.99])
```

In[213]:

The distribution of the Outcome variable was examined.

```
df["Outcome"].value_counts()*100/len(df)
```

In[214]:

```
# The classes of the outcome variable were examined.
```

```
df.Outcome.value_counts()
```

```
# In[215]:
```

```
# The histogram of the Age variable was reached.
```

```
# If you look at the histogram of the age variable, we can see that the histogram is quite  
skewed towards left, meaning \n
```

```
# data is not distributed properly, so we need to do some preprocessing, such as  
stadardizing/normalising.
```

```
plt.figure(figsize = (12,10))
```

```
df["Age"].hist(edgecolor = "black");
```

```
plt.xlabel("Age");
```

```
plt.ylabel("Frequency");
```

```
plt.title("Age Histogram");
```

```
# In[216]:
```

```
print("Max Age: " + str(df["Age"].max()) + " Min Age: " + str(df["Age"].min()))
```

```
# In[217]:
```

```
# Histogram and density graphs of all variables were accessed.
```

Similar to previous chart, some density plots over here also shows signs of data skewdness, so we need to perform standardization

before training our model, so that there wont be any bias while predicting the data.

```
fig, ax = plt.subplots(4,2, figsize=(16,16))
sns.distplot(df.Age, bins = 20, ax=ax[0,0])
sns.distplot(df.Pregnancies, bins = 20, ax=ax[0,1])
sns.distplot(df.Glucose, bins = 20, ax=ax[1,0])
sns.distplot(df.BloodPressure, bins = 20, ax=ax[1,1])
sns.distplot(df.SkinThickness, bins = 20, ax=ax[2,0])
sns.distplot(df.Insulin, bins = 20, ax=ax[2,1])
sns.distplot(df.DiabetesPedigreeFunction, bins = 20, ax=ax[3,0])
sns.distplot(df.BMI, bins = 20, ax=ax[3,1])
```

In[218]:

Plotting a Par Plot to see the relationship between the variables.

```
sns.pairplot(df)
```

In[219]:

```
def group_mean(i):
    ab = df.groupby("Outcome").agg({"i":"mean"})
    bc = df.groupby("Outcome").agg({"Insulin": "max"})
    return ab, bc
cd = df.drop(columns=["Outcome"], axis = 1)
```

```
for i in cd.columns:  
    print(group_mean(i))
```

```
# In[220]:
```

```
df.groupby("Outcome").agg({"Pregnancies":"mean"})
```

```
# In[221]:
```

```
df.groupby("Outcome").agg({"Age":"mean"})
```

```
# In[222]:
```

```
df.groupby("Outcome").agg({"Age":"max"})
```

```
# In[223]:
```

```
df.groupby("Outcome").agg({"Insulin": "mean"})
```

```
# In[224]:
```

```
df.groupby("Outcome").agg({"Insulin": "max"})
```

```
# In[225]:
```

```
df.groupby("Outcome").agg({"Glucose": "mean"})
```

```
# In[226]:
```

```
df.groupby("Outcome").agg({"Glucose": "max"})
```

```
# In[227]:
```

```
df.groupby("Outcome").agg({"BMI": "mean"})
```

```
# In[228]:
```

```
# The distribution of the outcome variable in the data was examined and visualized.
```

```
f,ax=plt.subplots(1,2,figsize=(18,8))
```

```
df['Outcome'].value_counts().plot.pie(explode=[0,0.08],autopct='%1.1f%%',ax=ax[0],shadow=T
```

```

    rue)
ax[0].set_title('Target variable')
ax[0].set_ylabel("")
sns.countplot('Outcome',data=df,ax=ax[1])
ax[1].set_title('Outcome')
plt.show()

```

In[229]:

```

# Below Data Explains about the correlation diagram of the data set.(How well each variable is
  related to each other.)

# If the correlation value is >0, there is a positive correlation. While the value of one variable
  increases, the value of the other variable also increases.

# Correlation = 0 means no correlation.

# If the correlation is <0, there is a negative correlation. While one variable increases, the other
  variable decreases.

# When the correlations are examined, there are 2 variables that act as a positive correlation
  to the Outcome dependent variable.

# These variables are Glucose. As these increase, Outcome variable increases.

df.corr()

```

In[230]:

```

# Correlation matrix graph of the data set

f, ax = plt.subplots(figsize= [20,15])

sns.heatmap(df.corr(), annot=True, fmt=".2f", ax=ax, cmap = "magma" )

```

```
ax.set_title("Correlation Matrix", fontsize=30)
plt.show()
```

```
# In[231]:
```

```
df.corrwith(df.Outcome).plot(kind='bar', grid=True, figsize=(12, 8),
                             title="Correlation with target")
```

```
# ## **2) Data Preprocessing**
```

```
# ##### 2.1) Eliminating Missing Values
```

```
#
```

```
# As, We all were aware that, some values were shown as 0, instead they are null values if
    we look at them closely, so having null values in our data might lead to a serious problem
    while training our ML Model, since it might give wrong predictions. So Below we
    replace 0 value by NaN:
```

```
# In[232]:
```

```
df[['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']] =
    df[['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']].replace(0, np.NaN)
```

```
# In[233]:
```

```
df.head()
```

```
# In[234]:
```

```
# Now, we can look at where are missing values
```

```
df.isnull().sum()
```

```
# In[235]:
```

```
# Have been visualized using the missingno library for the visualization of missing observations.
```

```
# Plotting A Bar Plot, so as to visualize count of null values, in each variable.
```

```
import missingno as msno
```

```
msno.bar(df);
```

```
# In[236]:
```

```
# OK, Now there are alot of techniques on how to fill null values, such as, using Simple Impouter, mean(), median, mode, etc.
```

```
# But we are going to analyse each variable, calculate the mean of each variable, with respect to the target variable and then we imputer values, based on the mean for each specific taregt calss.
```

```
# The missing values will be filled with the median values of each variable.
```

```
def median_target(var):
```

```
temp = df[df[var].notnull()]
temp = temp[[var, 'Outcome']].groupby(['Outcome'])[var].median().reset_index()
return temp
```

```
# In[237]:
```

```
df['Insulin'].groupby(df['Outcome']).median()
```

```
# In[238]:
```

```
columns = df.columns
columns = columns.drop("Outcome")
for i in columns:
    print(median_target(i))
```

```
# In[239]:
```

```
df[df['Insulin'].notnull()]
```

```
# In[240]:
```

```
# The values to be given for incomplete observations are given the median value of people  
    who are not sick and the median values of people who are sick.
```

```
columns = df.columns
```

```
columns = columns.drop("Outcome")
```

```
for i in columns:
```

```
    median_target(i)
```

```
    df.loc[(df['Outcome'] == 0) & (df[i].isnull()), i] = median_target(i)[i][0]
```

```
    df.loc[(df['Outcome'] == 1) & (df[i].isnull()), i] = median_target(i)[i][1]
```

```
# In[241]:
```

```
df.head()
```

```
# In[242]:
```

```
# Missing values were filled.
```

```
df.isnull().sum()
```

```
# ##### 2.2) Outlier Analysis:
```

```
# Outliers are termed as extremely high or low values which are considered to be caused  
    due to noise/some kind of error.
```

```
# In[243]:
```

```
for i in df:
```

```
    print(i)
```

```
# In[244]:
```

```
df[df['Pregnancies'] > 10].any(axis = None)
```

```
# In[245]:
```

```
# In the data set, there were asked whether there were any outlier observations compared to  
    the 25% and 75% quarters.
```

```
# It was found to be an outlier observation.
```

```
for feature in df:
```

```
    Q1 = df[feature].quantile(0.25)
```

```
    Q3 = df[feature].quantile(0.75)
```

```
    IQR = Q3-Q1
```

```
    lower = Q1- 1.5*IQR
```

```
    upper = Q3 + 1.5*IQR
```

```
    if df[(df[feature] > upper)].any(axis=None):
```

```
        print(feature,"yes")
```

```
    else:
```

```
        print(feature, "no")
```

```
# In[246]:
```

```
# Visualizing the Insulin Variable in order to find out outliers, using boxplot.
```

```
plt.figure(figsize = (12,6))  
sns.boxplot(x = df["Insulin"]);  
plt.xlabel('Insulin')  
plt.ylabel(' ')  
plt.title('Outliers using Boxplot in Insulin Variable')
```

```
# In[247]:
```

```
# So in order to remove these outliers, we suppress them by replacing the outlier of that particular  
column with the Upper bound value,
```

```
# similarly it can be done with lower bound value.
```

```
Q1 = df.Insulin.quantile(0.25)  
Q3 = df.Insulin.quantile(0.75)  
IQR = Q3-Q1  
lower = Q1 - 1.5*IQR  
upper = Q3 + 1.5*IQR  
df.loc[df["Insulin"] > upper, "Insulin"] = upper
```

```
# In[248]:
```

```
plt.figure(figsize = (12,6))
sns.boxplot(x = df["Insulin"]);
plt.xlabel('Insulin')
plt.ylabel(' ')
plt.title('Boxplot After removing outliers.')
```

```
# In[249]:
```

```
df.shape
```

```
# ##### 2.3) Local Outlier Factor (LOF)
```

```
# In[250]:
```

```
# We determine outliers between all variables with the LOF method
# The anomaly score of each sample is called the Local Outlier Factor. It measures the local
  deviation of the density of a given sample with respect to its neighbors.
# It is local in that the anomaly score depends on how isolated the object is with respect to the
  surrounding neighborhood.

from sklearn.neighbors import LocalOutlierFactor
lof =LocalOutlierFactor(n_neighbors= 10)
lof.fit_predict(df)
```

```
# In[251]:
```

```
df_scores = lof.negative_outlier_factor_  
np.sort(df_scores)[0:30]
```

```
# In[252]:
```

```
#We choose the threshold value according to lof scores  
threshold = np.sort(df_scores)[7]  
threshold
```

```
# In[253]:
```

```
#We delete those that are higher than the threshold  
outlier = df_scores > threshold  
df = df[outlier]
```

```
# In[254]:
```

```
# The size of the data set was examined.  
df.shape
```

```

# ## **3) Feature Engineering/ Feature Transformation.**
#
# Creating new variables is important for models. But you need to create a logical new
    variable. For this data set, some new variables were created according to BMI, Insulin
    and glucose variables.

# In[255]:

# As we know BMI can be categorized into different categories based on their values, such as
    given below.

NewBMI = pd.Series(["Underweight", "Normal", "Overweight", "Obesity 1", "Obesity 2",
    "Obesity 3"], dtype = "category")

df["NewBMI"] = NewBMI

df.loc[df["BMI"] < 18.5, "NewBMI"] = NewBMI[0]
df.loc[(df["BMI"] > 18.5) & (df["BMI"] <= 24.9), "NewBMI"] = NewBMI[1]
df.loc[(df["BMI"] > 24.9) & (df["BMI"] <= 29.9), "NewBMI"] = NewBMI[2]
df.loc[(df["BMI"] > 29.9) & (df["BMI"] <= 34.9), "NewBMI"] = NewBMI[3]
df.loc[(df["BMI"] > 34.9) & (df["BMI"] <= 39.9), "NewBMI"] = NewBMI[4]
df.loc[df["BMI"] > 39.9, "NewBMI"] = NewBMI[5]

# In[256]:

df.head()

```

```
# In[257]:
```

```
# A categorical variable creation process is performed according to the insulin value.
```

```
def set_insulin(row):
```

```
    if row["Insulin"] >= 16 and row["Insulin"] <= 166:
```

```
        return "Normal"
```

```
    else:
```

```
        return "Abnormal"
```

```
# In[258]:
```

```
df['Insulin_Level'] = df.apply(set_insulin, axis = 1)
```

```
df.head()
```

```
# In[259]:
```

```
# Some intervals were determined according to the glucose variable and these were assigned categorical variables.
```

```
NewGlucose = pd.Series(["Low", "Normal", "Overweight", "High"], dtype = "category")
```

```
df["NewGlucose"] = NewGlucose
```

```
df.loc[df["Glucose"] <= 70, "NewGlucose"] = NewGlucose[0]
```

```
df.loc[(df["Glucose"] > 70) & (df["Glucose"] <= 99), "NewGlucose"] = NewGlucose[1]
```

```
df.loc[(df["Glucose"] > 99) & (df["Glucose"] <= 126), "NewGlucose"] = NewGlucose[2]
```

```
df.loc[df["Glucose"] > 126, "NewGlucose"] = NewGlucose[3]
```

```
# In[260]:
```

```
df.head()
```

```
# ##### 3.1) One Hot Encoding
```

```
#
```

```
# Categorical variables in the data set should be converted into numerical values, Inorder to  
    make the model understand the data. Inorde to do these we are using  
    pd.get_dummies() to generate dummy variable for each category present in the data  
    variable.
```

```
# In[261]:
```

```
# Here, by making One Hot Encoding transformation, categorical variables were converted into  
    numerical values. It is also protected from the Dummy variable trap.
```

```
df = pd.get_dummies(df, columns=["NewBMI","Insulin_Level", "NewGlucose"], drop_first =  
    True)
```

```
# In[262]:
```

```
df.head()
```

```
# In[263]:
```

```
categorical_df = df[['NewBMI_Obesity 1','NewBMI_Obesity 2', 'NewBMI_Obesity 3',  
                    'NewBMI_Overweight','NewBMI_Underweight',  
                    'Insulin_Level_Normal','NewGlucose_Low','NewGlucose_Normal',  
                    'NewGlucose_Overweight']]
```

```
# In[264]:
```

```
y = df["Outcome"]  
X = df.drop(["Outcome", 'NewBMI_Obesity 1','NewBMI_Obesity 2', 'NewBMI_Obesity 3',  
            'NewBMI_Overweight','NewBMI_Underweight',  
            'Insulin_Level_Normal','NewGlucose_Low','NewGlucose_Normal',  
            'NewGlucose_Overweight'], axis = 1)  
cols = X.columns  
index = X.index
```

```
# In[265]:
```

```
X.head()
```

```
# In[266]:
```

```
# The variables in the data set are an effective factor in increasing the performance of the  
models by standardization.
```

```
# There are multiple standardization methods. These are methods such as "Normalize",  
    "MinMax", "Robust" and "Scale".
```

```
transformer = StandardScaler().fit(X)
```

```
X = transformer.transform(X)
```

```
X = pd.DataFrame(X, columns = cols, index = index)
```

```
# In[267]:
```

```
X.head()
```

```
# In[268]:
```

```
# Combine the Categorical variables unto the scaled data.
```

```
X = pd.concat([X, categorical_df], axis = 1)
```

```
# In[269]:
```

```
X.head()
```

```
# In[270]:
```

```
y.head()
```

```
# In[271]:
```

```
df = pd.concat([X,y], axis = 1)
```

```
df.head()
```

```
# ## **4) Model Building and Model Evaluation**
```

```
# In[272]:
```

```
X = df.drop(columns = ['Outcome'], axis = 1)
```

```
y = df['Outcome']
```

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

```
# In[273]:
```

```
def print_score(clf, X_train, y_train, X_test, y_test, train=True):
```

```
    if train:
```

```
        pred = clf.predict(X_train)
```

```
        clf_report = pd.DataFrame(classification_report(y_train, pred, output_dict=True))
```

```
        print("Train Result:\n=====")
```

```

print(f"Accuracy Score: {accuracy_score(y_train, pred) * 100:.2f}%")
print("_____")
print(f"CLASSIFICATION REPORT:\n{clf_report}")
print("_____")
print(f"Confusion Matrix: \n {confusion_matrix(y_train, pred)}\n")

```

```

elif train==False:

```

```

    pred = clf.predict(X_test)
    clf_report = pd.DataFrame(classification_report(y_test, pred, output_dict=True))
    print("Test Result:\n=====")
    print(f"Accuracy Score: {accuracy_score(y_test, pred) * 100:.2f}%")
    print("_____")
    print(f"CLASSIFICATION REPORT:\n{clf_report}")
    print("_____")
    print(f"Confusion Matrix: \n {confusion_matrix(y_test, pred)}\n")

```

```

# ##### **1. Logistic Regression**

```

```

# In[274]:

```

```

lr_clf = LogisticRegression()
lr_clf.fit(X_train, y_train)

```

```

print_score(lr_clf, X_train, y_train, X_test, y_test, train=True)
print_score(lr_clf, X_train, y_train, X_test, y_test, train=False)

```

```
# ##### **2. Decision Tree Classifier**
```

```
# In[275]:
```

```
dt_clf = DecisionTreeClassifier()
```

```
dt_clf.fit(X_train, y_train)
```

```
print_score(dt_clf, X_train, y_train, X_test, y_test, train=True)
```

```
print_score(dt_clf, X_train, y_train, X_test, y_test, train=False)
```

```
# ##### **3. Random Forest Classifier**
```

```
# In[276]:
```

```
rf_clf = RandomForestClassifier()
```

```
rf_clf.fit(X_train, y_train)
```

```
print_score(rf_clf, X_train, y_train, X_test, y_test, train=True)
```

```
print_score(rf_clf, X_train, y_train, X_test, y_test, train=False)
```

```
# ##### **4. XGB(Gradient Boosting Classifier)**
```

```
# In[277]:
```

```

xgb_clf = GradientBoostingClassifier(random_state = 12345)
xgb_clf.fit(X_train, y_train)

print_score(xgb_clf, X_train, y_train, X_test, y_test, train=True)
print_score(xgb_clf, X_train, y_train, X_test, y_test, train=False)

# ### **Testing on a Single User.**

# In[278]:

new = X_test.iloc[0]
a = np.asarray(new)
a = a.reshape(1,-1)
p = xgb_clf.predict(a)

# In[279]:

if (p[0] == 1):
    print("Person has Diabetes and is at risk of dying")
else:
    print("Great! the result is negative and you don't have to worry")

# # **8) Conclusion**

#

```

The aim of this study was to create classification models for the diabetes data set and to predict whether a person is affected by Diabetes or not, with the help of machine learning and to obtain the best accuracy for our model which will help us to predict better results. The work flow is as follows:

#

1) Reading and Understanding the Diabetes Dataset.

#

2) With Exploratory Data Analysis; The data set's structural data were checked.

The types of variables in the dataset were examined. Size information of the dataset was accessed. The 0 values in the data set are missing values. Primarily these 0 values were replaced with NaN values. Descriptive statistics of the data set were examined.

#

3) Data Preprocessing section;

df for: The NaN values missing observations were filled with the median values of whether each variable was sick or not. The outliers were determined by LOF and dropped. The X variables were standardized with the robust method..

#

4) During Model Building;

At most, 4 algorithms were used and i would say, 4 of them have performed pretty well on out dataset, with better accuracy on both training and testing data.

Algorithms: Logistic Regression, Decision Tree Classifier, Random Forest Classifier, Gradient Boosting Classifier.

#

5) Result;

Out of 4, XGB has given us negligible accuracy score, given as Training - 98.85% and Testing - 89.47%.

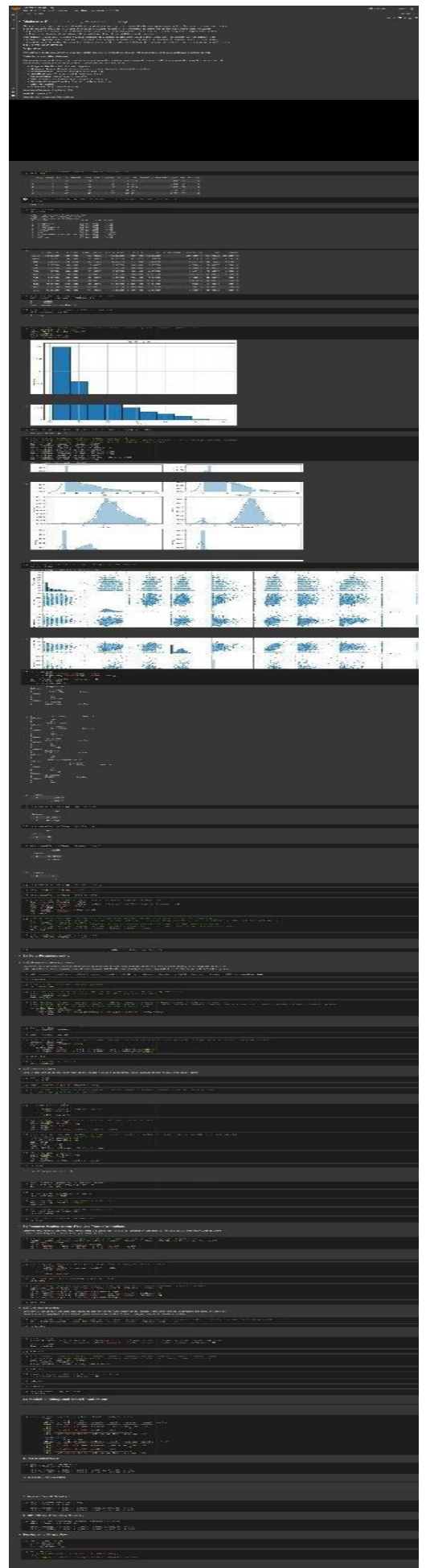
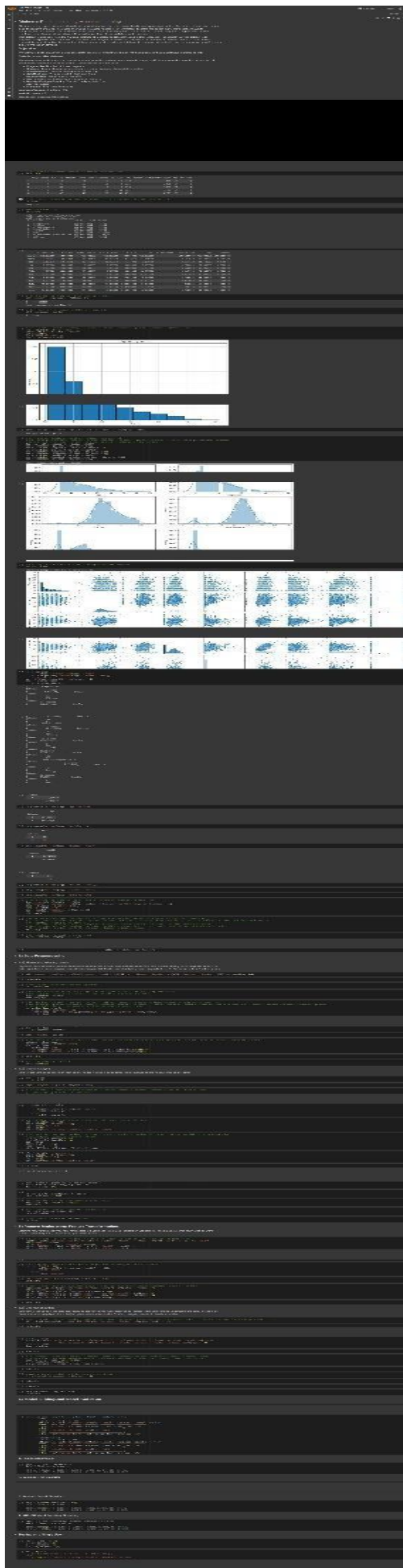
NoteL: Other algorithms also performed very well.

#

B.SCREENSHOTS:

```
Train Result:
=====
Accuracy Score: 85.86%
=====
CLASSIFICATION REPORT:
              0              1  accuracy  macro avg  weighted avg
precision    0.889447    0.800000  0.858553    0.844724    0.858258
recall       0.893939    0.792453  0.858553    0.843196    0.858553
f1-score     0.891688    0.796209  0.858553    0.843948    0.858396
support      396.000000    212.000000  608.000000    608.000000
=====
Confusion Matrix:
[[354  42]
 [ 44 168]]

Test Result:
=====
Accuracy Score: 86.84%
=====
CLASSIFICATION REPORT:
              0              1  accuracy  macro avg  weighted avg
precision    0.90099    0.803922  0.868421    0.852456    0.868421
recall       0.90099    0.803922  0.868421    0.852456    0.868421
f1-score     0.90099    0.803922  0.868421    0.852456    0.868421
support      101.000000    51.000000  152.000000    152.000000
=====
Confusion Matrix:
[[91 10]
 [10 41]]
```



Drive (A:) > TechieYan projects > MIX > Completed > Helping patients know whether they are diabetic or not using Machine Learning >

New

Name	^	Date modified	Type	Size
Results		07-07-2022 10:25	File folder	
abstract.txt		01-06-2022 15:41	Text Document	608 B
DIABETES .docx		07-07-2022 10:26	Microsoft Word Document	0 B
diabetes.csv		07-11-2020 20:36	Microsoft Excel Comma Separate...	23.31 KB
Diabetes.docx		21-05-2022 19:51	Microsoft Word Document	1.84 MB
Diabetes.pptx		05-04-2022 17:08	Microsoft PowerPoint Presentation	1.8 MB
main.ipynb		06-04-2022 15:32	IPYNB File	1.27 MB
main.py		05-04-2022 01:21	PY File	19.06 KB
main-checkpoint.ipynb		05-04-2022 01:21	IPYNB File	1.27 MB
Requirements.txt		07-07-2022 10:25	Text Document	66 B

Train Result:

Accuracy Score: 98.85%

CLASSIFICATION REPORT:

	0	1	accuracy	macro avg	weighted avg
precision	0.989924	0.985782	0.988487	0.987853	0.988480
recall	0.992424	0.981132	0.988487	0.986778	0.988487
f1-score	0.991173	0.983452	0.988487	0.987312	0.988480
support	396.000000	212.000000	0.988487	608.000000	608.000000

Confusion Matrix:

```
[[393  3]
 [  4 208]]
```

Test Result:

Accuracy Score: 89.47%

CLASSIFICATION REPORT:

	0	1	accuracy	macro avg	weighted avg
precision	0.912621	0.857143	0.894737	0.884882	0.894007
recall	0.930693	0.823529	0.894737	0.877111	0.894737
f1-score	0.921569	0.840000	0.894737	0.880784	0.894200
support	101.000000	51.000000	0.894737	152.000000	152.000000

Confusion Matrix:

```
[[94  7]
 [ 9 42]]
```

C.PUBLICATION:

**IDENTIFICATION AND PREDICTION OF DIABETES BY
USING MACHINE LEARNING CONCEPTS**

P.Tamiliniyan

U.G Scholar

Department of Information Technology

Sathyabama Institute of Science and Technology

Chennai, India

tamiliniyan658@gmail.com

J.Santhosh

U.G Scholar

Department of Information Technology

Sathyabama Institute of Science and Technology

Chennai, India

Jsanthosh29mar2@gmail.com

K.S Kamatchi

Associate professor

Department of Information Technology

Sathyabama Institute of Science and Technology

Chennai, India

Kamatchi.k.s.it@sathyabama.ac.in

Abstract:

Diabetes is a condition characterized by high levels of glucose in the bloodstream, leading to various complications if left untreated. It occurs when the body is incapable of utilizing the insulin it does create, or is incapable of forming enough of it.

When we eat carbohydrates, they're broken down into glucose during digestion. This glucose then enters the bloodstream, where it needs to be transported into cells to provide energy. Insulin, produced by the pancreas, helps facilitate this process by allowing glucose to enter cells.

3

There are different types of diabetes:

- **Type 1:** In this type, insulin-forming cells are mistakenly attacked and then destroyed by the immune system, which prevents the pancreas from producing insulin.
- **Type 2:** In this case, either the pancreas produces insufficient insulin to overcome the body's cells' resistance to insulin, or the cells develop resistance to it.
- **Gestational diabetes:** This type occurs during pregnancy when hormonal changes can interfere with insulin function.

Keywords: Artificial Intelligence, Machine Learning, Web Application, Django Framework, Obstructive Sleep Apnea Prediction, Python Programming Language, Graphical User Interface.

Introduction:

Diabetes is indeed a serious and chronic illness that cannot be cured once it develops. It poses significant health risks, including heart disease, kidney disease, eye problems, stroke, dental issues, foot problems, and nerve damage. Managing diabetes effectively is crucial to prevent or mitigate these complications. With the rising standards of living, diabetes has become increasingly prevalent in society, highlighting the importance of studying how to diagnose and analyze it accurately and quickly. In medicine, the diagnosis of diabetes typically involves assessing fasting blood glucose levels, glucose tolerance, and random blood glucose levels. For improved management and control, an early diagnosis is crucial. Based on people's daily physical examination data, machine learning techniques can help make preliminary decisions about diabetes, which can be a useful tool for medical professionals. However, selecting valid features and choosing the appropriate classifier are critical challenges in implementing machine learning methods for diabetes diagnosis.

3

There are several types of diabetes:

Type 1 Diabetes: This type is caused by the body immune system targeting and eliminating the pancreatic cells that create insulin.

Type 2 Diabetes: Type 2 diabetes is much more common & typically occurs in middle-aged or older individuals, often associated with obesity, high blood pressure, and other conditions. Lifestyle factors such as maintaining a healthy weight, regular exercise, and proper diet play significant roles in its prevention and management.

Gestational Diabetes: Gestational diabetes occurs at the time of pregnancy and affects how cells use sugar. While it can pose risks to both the mother and baby, it could be managed through exercise, healthy eating, and medication if necessary.

Overall, understanding the different types of diabetes and implementing appropriate diagnostic and management strategies are crucial steps in combating this widespread health issue.

Literature Review:

Various ML algorithms have been suggested for predicting & diagnosing diabetes, aiming for early detection and better management of the disease. One such algorithm is the Random Forest model, suggested by Vijay Kumar et al., which demonstrated high accuracy in predicting diabetes onset. Similarly, Nanos Nnameka et al. employed a supervised learning approach, combining results with five classification methods to predict diabetes onset more accurately. Joshi et al. proposed an approach using SVM (Support Vector Machines), LR (logistic regression), and ANN (Artificial Neural Networks) to prevent diabetes onset. Deeraj Shetty et al. suggested a prediction method utilizing Bayesian, K-nearest neighbor (KNN), and genetic algorithms for diabetes diagnosis, achieving high accuracy rates.

In another study, logistic regression and linear perceptron models were used to evaluate diabetes prediction. Data mining techniques, such as K-nearest neighbor (KNN), were also employed, showing improved accuracy with higher K values. Various classification

algorithms, including ZeroR, One, and Naive Bayes, were compared for diabetes diagnosis, with Naive Bayes demonstrating the fastest performance.

Furthermore, Azar et al. explored diabetes prediction in young people and ancient cultures using data mining approaches, like Naive Bayes, decision trees, and K-nearest neighbors. The decision tree algorithm was found to be highly efficient in predicting diabetes onset.

Overall, this research highlights the potential of ML algorithms in accurately predicting and diagnosing diabetes, offering valuable insights for healthcare professionals in managing the disease effectively and making informed decisions.

The predictive onset of diabetes was presented to Nanos Namiko et al [13]. A supervised learning approach is used to combine their results with five widely used classifications. The results will be presented and compared to similar studies using the same dataset in the book. Diabetes beginning can be predicted with greater accuracy by using the proposed method. Roof N. The goal is to avoid diabetes through three numerous regulated methods of learning, namely logistic regression, SVM, and ANN, proposed by Joshi et al [11]. Predicting Diabetes Prediction Using Machine Learning Techniques. The project advocates an effective method for early diabetes detection.

AIM AND SCOPE:

AIM: Aim and Scope of the Study

Diabetes is a complex metabolic disorder characterized by abnormal metabolism leading to elevated blood sugar levels, primarily due to issues with insulin secretion or action. As a chronic condition without a cure, diabetes poses significant health risks and is associated with both acute and chronic complications, including hyperglycemia-related emergencies and vascular damage.

Objective:

Our objective is to enhance the efficiency of existing diabetes classification models. In this analysis, we employ machine learning algorithms to classify diabetes, focusing on extensive experiments to identify the most effective feature selection methods and optimal attribute numbers.

Scope:

We propose a robust structure for the early detection of diabetes. This framework includes outlier rejection, handling missing values, K-fold validation, data standardization, and the implementation of various ML classifiers like k-nearest neighbors (k-NN), random forest (RF), and XGBoost (X). Additionally, we introduce the concept of weighted assembling of different machine learning models to further improve results.

The performance of these models is analyzed utilizing the Area Under the Curve (AUC) ROC as the performance metric, with the weights of the ensembled models estimated accordingly.

Symptoms of Diabetes:

- Weight loss
- Tired/Sleepiness
- Mood swings
- Increased thirst
- Frequent Urination
- frequent infections
- Confusion and difficulty concentrating
- Blurred vision

Causes of Diabetes:

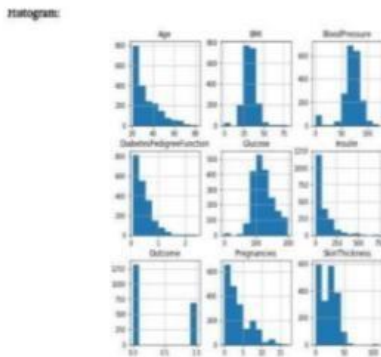
Diabetes is primarily caused by inborn errors in genetics. The disease is caused by at least two genes that have been mutated in chromosome 6, which is responsible for regulating how the body reacts to various antigens.

Infection with viruses can also play a role in the development of diabetes for both type 1 and type 2 patients. Research has indicated that contracting viruses like mumps, coxsackievirus, rubella,

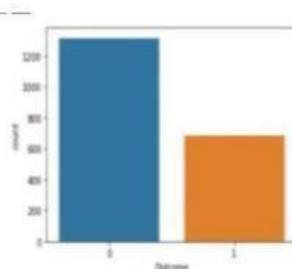
Diabetes risk is increased by the cytomegalovirus and hepatitis B virus.

It is evident that no single feature significantly correlates with our final value. Certain characteristics exhibit a negative association having the outcome value, while others show a positive correlation.

Histogram:



Let's examine the plotlines. It provides additional evidence for the necessity of scaling by displaying the distribution of each feature and label over various ranges. Next, each discrete bar you see indicates that this is a categorical variable in and of itself. Prior to using machine learning, these categorical variables must be addressed. We have two classes for our outcome labels: 0 for no disease and 1 for disease.

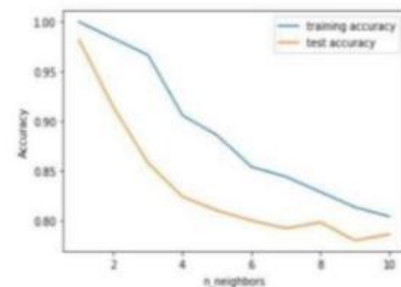


The data is skewed in favour of data points with an outcome value, as seen in the above graph. The number of people without diabetes is almost twice that of those with the disease.

k-Nearest Neighbors:

The k-NN algorithm is possibly the simplest ML algorithm. The only step in creating the model is to store the training data set. The algorithm finds the closest data points—its "nearest neighbors"—in training data set to identify a new data point.

Let's first see if we could validate the association among model complexity & accuracy:



The training along with the test set accuracy is plotted on y-axis against neighbor setting on x-axis in the above figure. If we choose the one nearest neighbour, the training set prediction is completely perfect. However, as more neighbours are considered, the training accuracy declines, showing which utilizing the single nearest neighbour indicates to an excessively complex model. Nine neighbours are thought to perform at their peak.

Training Accuracy	0.81
Testing Accuracy	0.78

Table-1

Logistic regression (LR):

It is among the most popular algorithms for classification.

RE-2022-205465-plag-report

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

/0

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

Instrumentation and Control Technologies (ICICT), 2022

Publication

9

"Computational Vision and Bio-Inspired
Computing", Springer Science and Business
Media LLC, 2020

Publication

<1 %

Exclude quotes ☒ On

Exclude matches ☐ Off

Exclude bibliography ☐ On

RE-2022-205465-plag-report

ORIGINALITY REPORT

8%

SIMILARITY INDEX

7%

INTERNET SOURCES

4%

PUBLICATIONS

2%

STUDENT PAPERS

PRIMARY SOURCES

1

ijsrcseit.com

Internet Source

2%

2

www.ijcse.com

Internet Source

1%

3

www.kanccmedicalplc.com

Internet Source

1%

4

www.coursehero.com

Internet Source

1%

5

ir.nctu.edu.tw

Internet Source

1%

6

www.coilyembrace.com

Internet Source

1%

7

dokumen.pub

Internet Source

<1%

8

Menaka V, Likitha V, Shravya M, R. Pari.
"Accurate Prediction of Type 1 and Type 2
Diabetes", 2022 Third International
Conference on Intelligent Computing

<1%