

EARLY-STAGE DIABETES PREDICTION USING MACHINE LEARNING

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

by

JAWAHAR K (Roll. No. 17SCS0175)
GUNASEKHAR J (Roll. No. 17SCS0172)



**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 2021



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with "A" grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

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 **J A W A H A R K (Roll. No. 17SCS0175)** and **G U N A S E K H A R J (Roll.No. 17SCS0172)** who carried out the project entitled **"EARLY-STAGE DIABETES PREDICTION USING MACHINE LEARNING"** under my supervision from **November 2020** to **April 2021**.

Internal Guide

Dr. A. Siva Sangari, M.E., Ph.D.,

Head of the Department

Submitted for Viva voce Examination held on _____

Internal Examiner

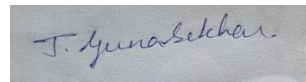
External Examiner

DECLARATION

I **GUNASEKHAR J** hereby declare that the Project Report entitled “**EARLY-STAGE DIABETES PREDICTION USING MACHINE LEARNING**” is done by me and my partner **JAWAHAR K** under the guidance of **Dr. A. SIVA SANGARI, M.E., Ph.D.**, (Internal Guide) is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE: CHENNAI

A rectangular box containing a handwritten signature in blue ink that reads "J. Gunasekhar."

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of Sathyabama Institute 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, **Dr. S. Vigneswari, M.E., Ph.D., and Dr. L. Lakshmanan, 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. A. Siva Sangari, M.E., Ph.D.**, for her 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

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.	AIM AND SCOPE	7
	2.1 AIM OF PROJECT	7
	2.1.1 OBJECTIVES	7
	2.2 SCOPE	7
	2.3 BENEFITS IN HEALTH CARE INDUSTRY	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 EVALUTION 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. 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.

1.2.4. 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

AIM AND SCOPE

2.1. AIM OF PROJECT:

The primary aim of this project is to predict whether a person could be diabetic or not at early stage.

2.1.1. OBJECTIVES:

The proposed model is built using the latest dataset. This dataset is cleaned and pre-processed into a balanced dataset. Based on correlation among the instances distribution of class is analyzed. Thereby machine learning model is built using classification and regression algorithms. Finally, the performance of each model is evaluated with respect to certain parameters.

2.2. SCOPE:

The proposed model can be used as computerized diagnosis in hospitals. Physicians can use this application to take precautionary measures and suggest the patients to take appropriate guidelines in order to avoid the risk factor of affecting diabetes for long term.

2.3. BENEFITS IN HEALTH CARE INDUSTRY:

- **Nutrition coaching:** A nutrition-focused approach to helping diabetics and others living with chronic diseases manage their health. The startup claims that its app uses AI to recommend meal options that comply with the diet restrictions of the user.
- **Diabetes AI Applications to Complement Physician Care:** Diabetes management, to a large degree, involves pattern recognition thus positioning it well for applications of AI. For example, key factors such as blood glucose, weight and blood pressure must be consistently measured and monitored to inform patient care. By automating routine processes, clinicians can devote less time to data

entry and more time to patient interaction. Patients may also gain an increased sense of control over their diabetes by having coaching tools and support at their disposal. The real potential for AI is the ability to scale up these innovations while still personalizing the user experience. Additionally, a better continuity of care may be achieved between face-to-face visits with physician and patient. Challenges for market penetration include data privacy, patient consent and the current limited data on clinical results. Strategic partnership with hospitals and healthcare systems in general will be necessary to conduct the research required to validate company claims regarding products. An important theme that emerged from our research is that these applications are not supplementing but rather complementing care delivered by clinicians. This is an important consideration going forward as more applications emerge in an effort to reduce the rate and negative impacts of diabetes.

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:

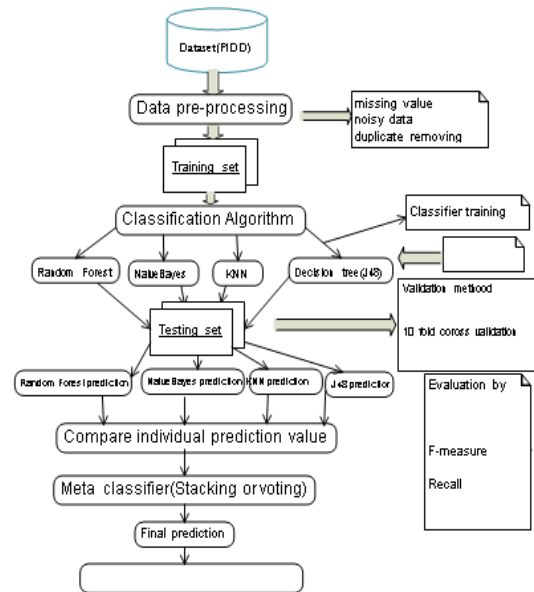


Fig 3.1. System Architecture

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

Feature Type	Feature Name	Range
Demographic	Sex	Male and Female
	Age	25-70
	Class	Positive and Negative
Diagnosis	Polyuria	Yes / No
	Polydipsia	Yes / No
	Polyphagia	Yes / No

	Genital Thrush	Yes / No
	Itching	Yes / No
	Irritability	Yes / No
	Obesity	Yes / No
Lab Investigation	Visual Blurring	Yes / No
	Sudden Weight Loss	Yes / No
	Delayed Healing	Yes / No
	Alopecia	Yes / No
	Partial Paresis	Yes / No

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 techninue 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.

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:

3.3.2.1. 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:

If only a reasonable percentage of values are missing, then we can also run simple interpolation methods to fill in those values. However, most common method of dealing with missing values is by filling them in with the mean, median or mode value of the respective feature.

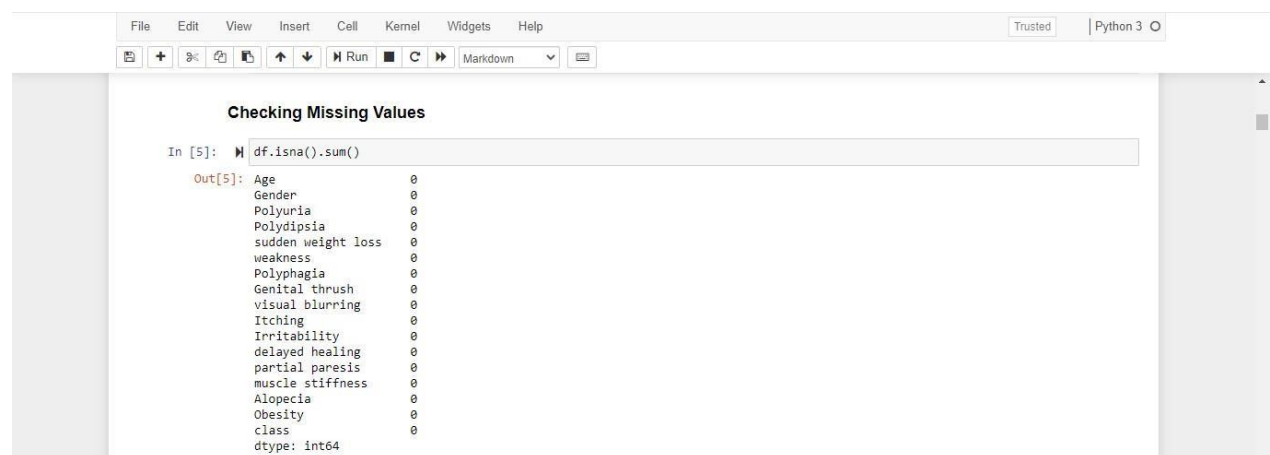


Fig 3.2. Checking Missing Values

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

Feature Type	Feature Name	Range
Demographic	Sex	0 / 1 (0 for no and 1 for yes)
	Age	0 / 1 (0 for no and 1 for yes)
	Class	0 / 1 (0 for no and 1 for yes)
Diagnosis	Polyuria	0 / 1 (0 for no and 1 for yes)
	Polydipsia	0 / 1 (0 for no and 1 for yes)
	Polyphagia	0 / 1 (0 for no and 1 for yes)
	Genital Thrush	0 / 1 (0 for no and 1 for yes)
	Itching	0 / 1 (0 for no and 1 for yes)
	Irritability	0 / 1 (0 for no and 1 for yes)
	Obesity	0 / 1 (0 for no and 1 for yes)
Lab Investigation	Visual Blurring	0 / 1 (0 for no and 1 for yes)
	Sudden Weight Loss	0 / 1 (0 for no and 1 for yes)
	Delayed Healing	0 / 1 (0 for no and 1 for yes)

	Alopecia	0 / 1 (0 for no and 1 for yes)
	Partial Paresis	0 / 1 (0 for no and 1 for yes)

Out[21]:

	Age	Gender	Polyuria	Polydipsia	sudden weight loss	weakness	Polyphagia	Genital thrush	visual blurring	Itching	Irritability	delayed healing	partial paresis	muscle stiffness	Alopecia	Obesity
0	40	1	0	1	0	1	0	0	0	1	0	1	0	1	1	1
1	58	1	0	0	0	1	0	0	1	0	0	0	1	0	1	0
2	41	1	1	0	0	1	1	0	0	1	0	1	0	1	1	0
3	45	1	0	0	1	1	1	1	0	1	0	1	0	0	0	0
4	60	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1

Fig 3.3. Dataset after Binary conversion.

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 Imbalanced Dataset.

- **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 imbalanced 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. **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.
5. **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

base learners. There are various approaches in ensemble learning such as Bagging, Boosting, etc.

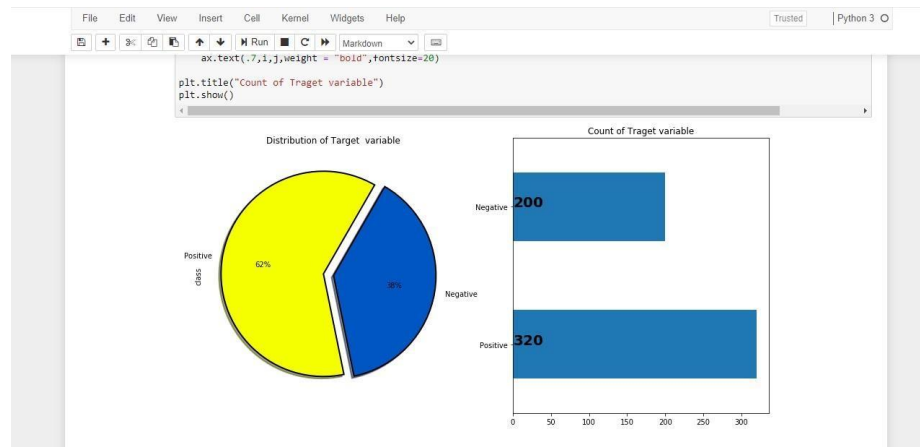


Fig 3.4. Balanced Dataset (pie chart)

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.

- **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.
- **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

Fig 3.7. Distribution of Polydipsia

Out[15]:

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

Fig 3.8. Distribution of Sudden weight loss

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.

- **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 diagonal vector that is large and negative, corresponding to a larger negative product in

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.

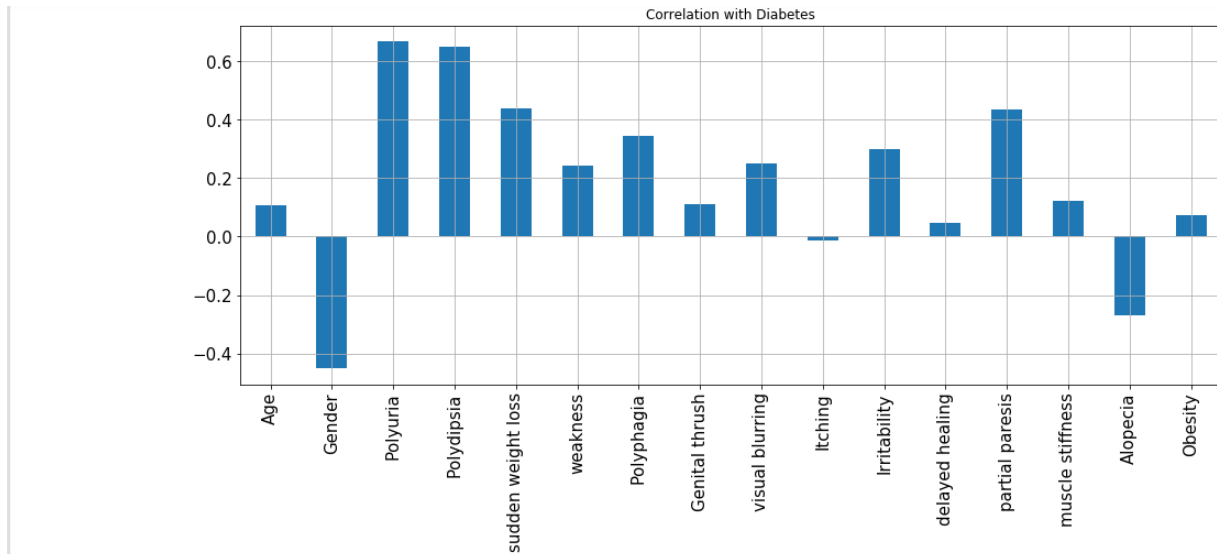


Fig 3.2.1. Correlation with diabetes

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.

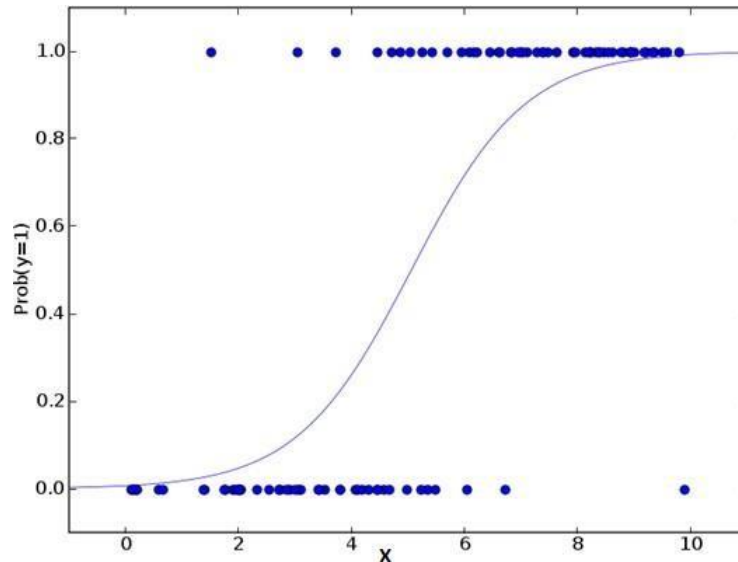


Fig 3.2.2. Logistic Regression Graph Representation

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.

$$\text{True Positive Rate (TPR)} = \text{RECALL} = \frac{TP}{TP+FN}$$

$$\text{False Positive Rate (FPR)} = 1 - \text{Specificity} = \frac{FP}{TN+FP}$$

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 is now ready for computation, which is split into 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 the data normalization stage using the 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 `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.

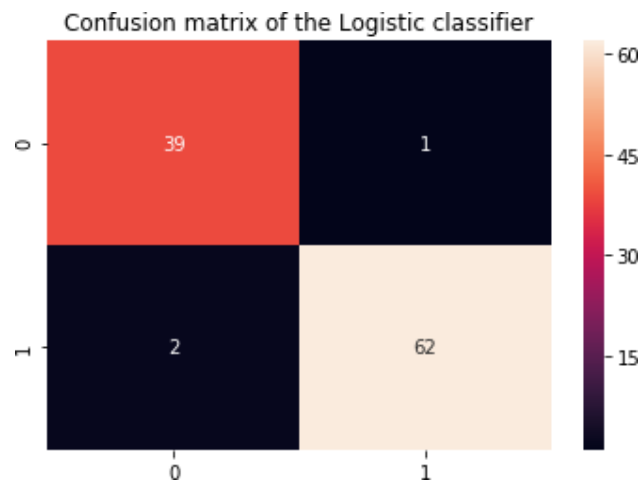


Fig 4.3. Confusion matrix of Logistic classifier

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.

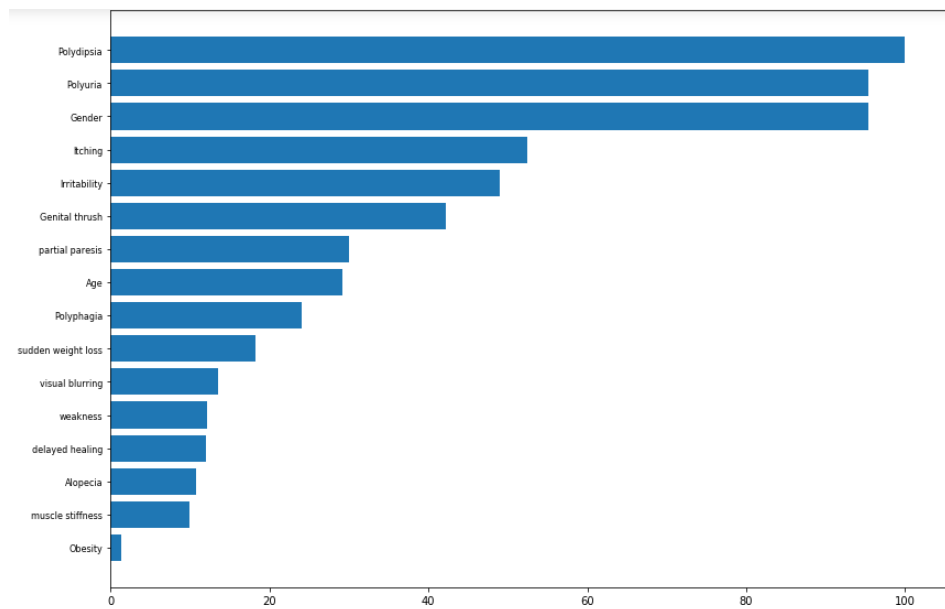


Fig 4.4. Relative feature importance

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)

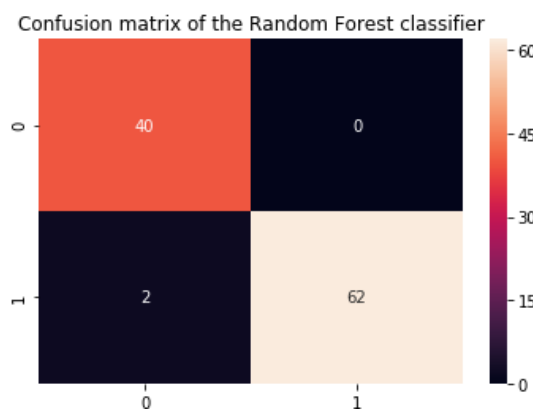


Fig 4.6. Confusion matrix of the Random forest classifier

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.

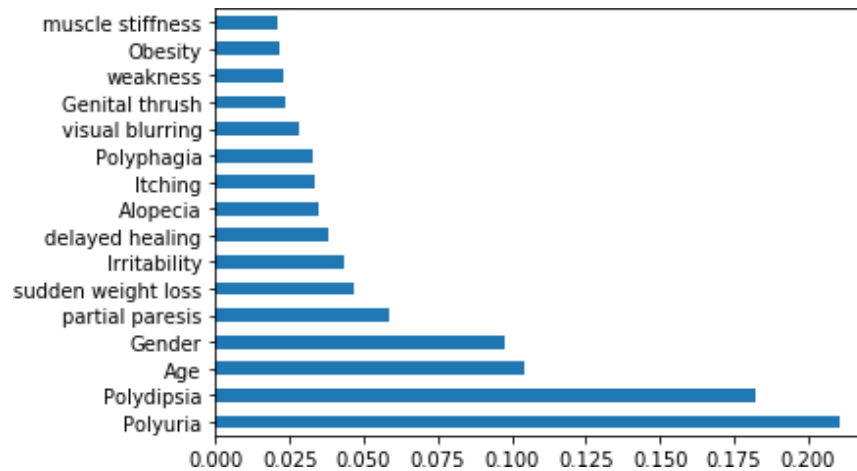


Fig 4.7. Top 10 selected features

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.

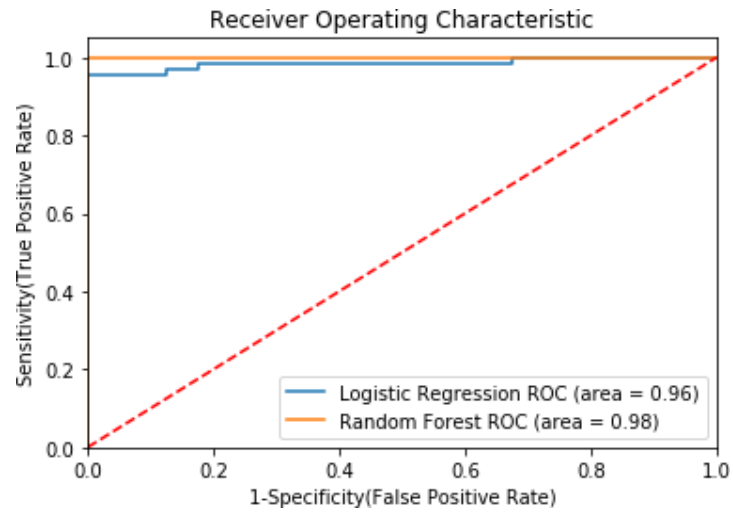


Fig 4.9. Receiver Operating Characteristic

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:

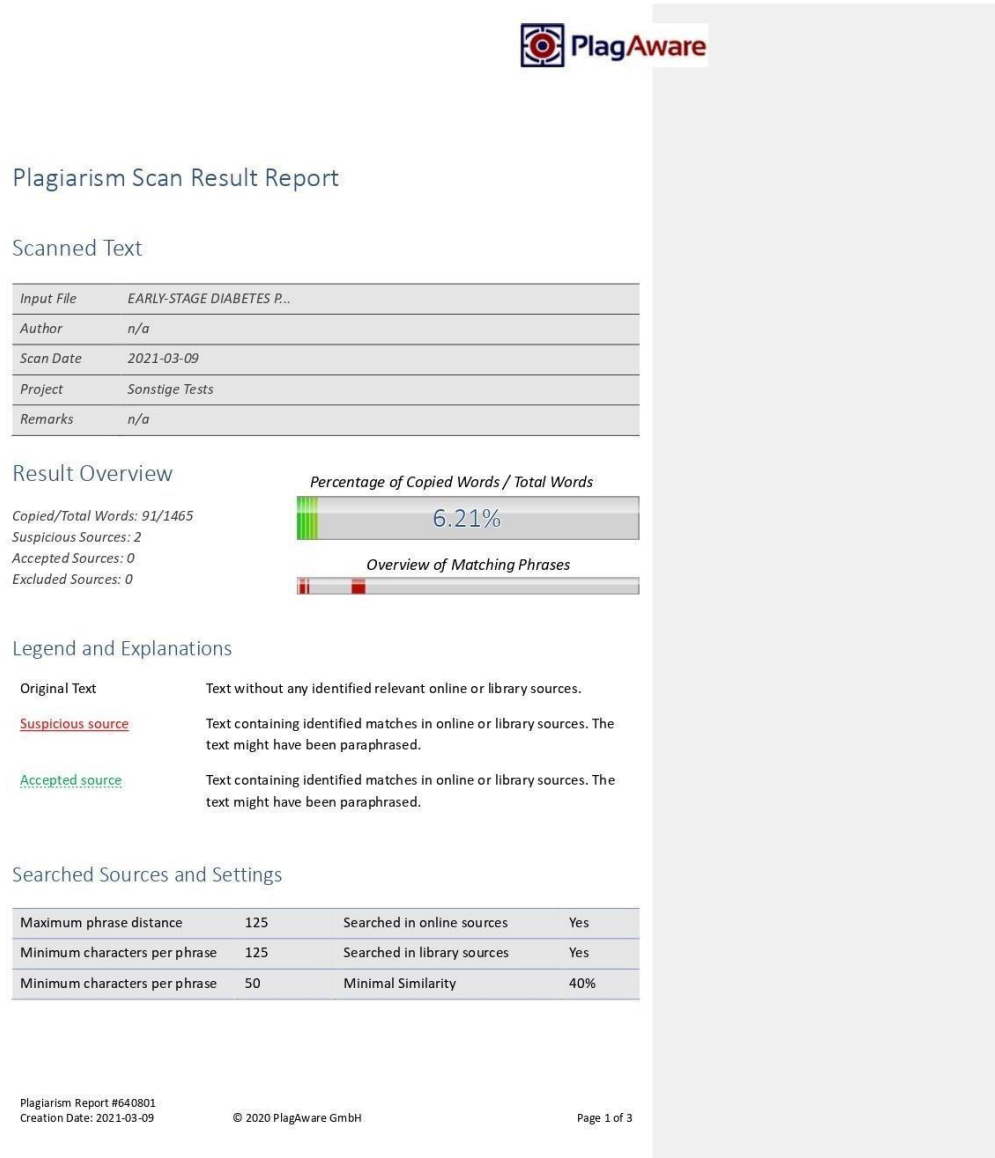
We focused only on Type 1 diabetes in this project. There are multiple types of diabetes which in further we wish to predict also, the consequences due to this disease are wider which can be prevented. This might require an accurate dataset and to find new patterns in it using data mining techniques.

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. Plagiarism Report:



B. PAPER

EARLY-STAGE DIABETES PREDICTION USING MACHINE LEARNING

¹A.Sivasangari
Dept of IT, Sathyabama Institute
of Science and Technology
Chennai, India
sivasangarikavya@gmail.com

²Jawahar.K
Dept of CSE, Sathyabama
Institute of Science and
Technology, Chennai, India.
kandrajawahar6@gmail.com

³Gunasekhar.J
Dept of CSE, Sathyabama
Institute of Science and
Technology Chennai, India.
Jallurisekhar1@gmail.com

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. 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.

Keywords: Machine learning, Logistic regression, Random forest.

I. INTRODUCTION

Machine learning is a field of study in artificial intelligence where a system can acquire knowledge and experience from the given data and make appropriate decisions without human involvement. There are 3 different kinds of methods tangled in machine learning: (firstly) Supervised, (secondly) Unsupervised and Reinforcement learning. In the early days of AI utmost of the scientists were concerned in having machines absorb from data. Probabilistic perceptive is also employed, especially in computerized medical diagnosis. Resurging curiosity in machine learning is due to the

identical factors that have thru data mining and Bayesian analysis further widespread than ever. Things like collective volumes and varieties of available data, computational processing that is inexpensive and more influential, and affordable data storage. All of these effects mean it's possible to rapidly and automatically yield models that can analyze bigger, more compound data and send quicker. This technology is used in every sector such as Government (to public safety), Retail (to recommend products), Financial services (to prevent fraud), Oil and Gas (to find new energy sources), Health care (to improve diagnosis and treatment). There are many factors responsible for diabetes some of them are as follows: Firstly, combination of genetic vulnerability and environment factor can root diabetes. Secondly, overweight may lead to basis of diabetes in the long run. Low levels of high-density lipoprotein are also the cause of stirring the risk. Complications that arise due to diabetes are as follows: damage in the nerves, damage in the kidneys, damage in eyes and Alzheimer's disease. 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.

II. MOTIVATION

According to official report from the World Health Organization, India has highest amount of diabetes. Thereby in order to help to find or predict at early stage will lower the chances of complications and can be cautions. Its always a

challenging task to anyone who is dealing with medical data.

III. RELATED WORK

Alade, O.M., Sowunmi, O.Y., Misra, S., Maskeliūnas, R., Damaševičius, R had proposed [1] a four-layer artificial neural network using two methods: backpropagation and Bayesian method. Thereby a computer-assisted diagnosis is proposed [4] by Carrera et al based on the digital processing of retinal images to detect diabetes. Lee et al found and stated in his paper [8] 10-fold cross-validation method and found the relation between diabetes type II and phenotype HW. Khalil et al found a new probabilistic neural network (PNN) [6].

IV. 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 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. 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.

V. IMPLEMENTATION

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.

	CLASS	NEGATIVE	POSITIVE
GENDER	FEMALE	9.5	54.06
	MALE	90.5	45.94
POLYURIA	NO	92.5	24.06
	YES	7.5	75.94
ITCHING	NO	50.5	51.88
	YES	49.5	48.12
GENITAL THRUSS	NO	83.5	74.06
	YES	16.5	25.94
VISUAL BLURRING	NO	71	45.31
	YES	29	54.69
DELAYED HEALING	NO	57	52.19
	YES	43	47.81
PARTIAL PARESIS	NO	84	40
	YES	16	60

Fig1. Distribution of symptoms

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 into two different proportions for training and testing. Expect the age every instance is converted into binary also the age is converted in the data normalization stage using `minmaxscalar` method. Thereby manipulation of data gets completed. The base machine learning model is logistic regression. We have used `penalty='l2'` (ridge) which is a regularization parameter and adds squared magnitudes of coefficients for minimizing the loss 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. Considering these results, relative feature importance is visualized using a bar graph. 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. 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.

VI. MODEL EVALUATION

Here, we have used following quality measures to check the performance of the model:

1. Accuracy= $(TP+TN)/(TP+TN+FP+FN)$
2. Recall= $TP/(TP+FN)$
3. Precision= $TP/(TP+FP)$
4. F₁measure= $2*(precision*recall)/(precision+recall)$

were, true positive (TP), false positive (FP), true negative (TN), false negative (FN).

MODEL	Logistic Regression
ACCURACY	0.971154
PRECISION	0.984127
RECALL	0.96875
F1 SCORE	0.976378

Fig2. Base model evaluation

Similarly, when the process repeated without tuning (random forest) we got highest accuracy.

MODEL	Random forest
ACCURACY	0.980769
PRECISION	1.000000
RECALL	0.96875
F1 SCORE	0.984127

Fig3: Random forest evaluation (before tuning)

Since the accuracy has increased, top 10 features are selected and applied to the model (Fig4). 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 (Fig5).

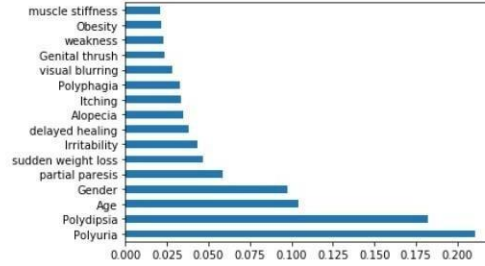


Fig4: Top 10 features plotted

	Model	Accuracy	Cross Val Accuracy	Precision	Recall	F1 Score	ROC
0	Logistic Regression	0.971154	0.918113	0.984127	0.958750	0.979378	0.971875
1	Random Forest (Untuned)	0.980769	0.978223	1.000000	0.958750	0.984127	0.984375
2	Logistic Regression-Post FS	0.961538	0.898639	0.983871	0.953125	0.968254	0.964063
3	Random Forest-Post FS	0.980769	0.968641	1.000000	0.958750	0.984127	0.984375

Fig5: Final chart of evaluation

VII. RESULTS

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.

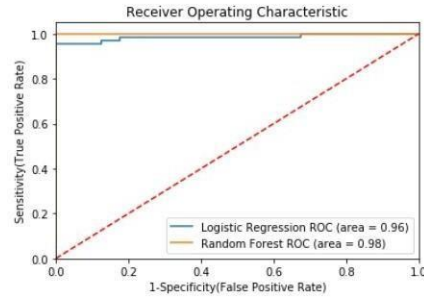


Fig6: Final ROC

VIII. 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.

IX. 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)

C. Source Code:

```
import warnings

warnings.filterwarnings('ignore')

import numpy as np
import pandas as pd

from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score, f1_score, precision_score, confusion_matrix,
recall_score, roc_auc_score

from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier

from sklearn.svm import SVC

import matplotlib.pyplot as plt

%matplotlib inline

from IPython.display import Image

df = pd.read_csv('diabetes_data_upload.csv')

df.head()

df.isna().sum()

df.info()

import seaborn as sns

sns.countplot(df['class'], data=df)

# plotting to create pie chart and bar plot as subplots

plt.figure(figsize=(14, 7))

plt.subplot(121)

df["class"].value_counts().plot.pie(autopct = "%1.0f%%", colors =
sns.color_palette("prism", 7), startangle = 60, labels=["Positive", "Negative"],
wedgeprops={"linewidth":2, "edgecolor":"k"}, explode=[.1, 0], shadow = True)

plt.title("Distribution of Target variable")

plt.subplot(122)

ax = df["class"].value_counts().plot(kind="barh")

for i, j in enumerate(df["class"].value_counts().values):
```

```

ax.text(.7,i,j,weight = "bold",fontsize=20)

plt.title("Count of Traget variable")

plt.show()

import seaborn as sns

sns.countplot(df['Gender'],hue=df['class'], data=df)

plot_criteria= ['Gender', 'class']

cm = sns.light_palette("red", as_cmap=True)

(round(pd.crosstab(df[plot_criteria[0]], df[plot_criteria[1]], normalize='columns') *
100,2)).style.background_gradient(cmap = cm)

Image(filename='polyuria.jpg')

plot_criteria= ['Polyuria', 'class']

cm = sns.light_palette("red", as_cmap=True)

(round(pd.crosstab(df[plot_criteria[0]], df[plot_criteria[1]], normalize='columns') *
100,2)).style.background_gradient(cmap = cm)

Image(filename='polydipsia.jpg')

plot_criteria= ['Polydipsia', 'class']a

cm = sns.light_palette("red", as_cmap=True)

(round(pd.crosstab(df[plot_criteria[0]], df[plot_criteria[1]], normalize='columns') *
100,2)).style.background_gradient(cmap = cm)

plot_criteria= ['sudden weight loss', 'class']

cm = sns.light_palette("red", as_cmap=True)

(round(pd.crosstab(df[plot_criteria[0]], df[plot_criteria[1]], normalize='columns') *
100,2)).style.background_gradient(cmap = cm)

plot_criteria= ['weakness', 'class']

cm = sns.light_palette("red", as_cmap=True)

(round(pd.crosstab(df[plot_criteria[0]], df[plot_criteria[1]], normalize='columns') *
100,2)).style.background_gradient(cmap = cm)

Image(filename='polyphagia.png')

plot_criteria= ['Polyphagia', 'class']

cm = sns.light_palette("red", as_cmap=True)

(round(pd.crosstab(df[plot_criteria[0]], df[plot_criteria[1]], normalize='columns') *
100,2)).style.background_gradient(cmap = cm)

plot_criteria= ['Genital thrush', 'class']

cm = sns.light_palette("red", as_cmap=True)

```

```

(round(pd.crosstab(df[plot_criteria[0]], df[plot_criteria[1]], normalize='columns') *
100,2)).style.background_gradient(cmap = cm)

plot_criteria= ['visual blurring', 'class']

cm = sns.light_palette("red", as_cmap=True)

(round(pd.crosstab(df[plot_criteria[0]], df[plot_criteria[1]], normalize='columns') *
100,2)).style.background_gradient(cmap = cm)

plot_criteria= ['Itching', 'class']

cm = sns.light_palette("red", as_cmap=True)

(round(pd.crosstab(df[plot_criteria[0]], df[plot_criteria[1]], normalize='columns') *
100,2)).style.background_gradient(cmap = cm)

plot_criteria= ['Irritability', 'class']

cm = sns.light_palette("red", as_cmap=True)

(round(pd.crosstab(df[plot_criteria[0]], df[plot_criteria[1]], normalize='columns') *
100,2)).style.background_gradient(cmap = cm)

plot_criteria= ['delayed healing', 'class']

cm = sns.light_palette("red", as_cmap=True)

(round(pd.crosstab(df[plot_criteria[0]], df[plot_criteria[1]], normalize='columns') *
100,2)).style.background_gradient(cmap = cm)

Image(filename='paresis.png')a

plot_criteria= ['partial paresis', 'class']

cm = sns.light_palette("red", as_cmap=True)

(round(pd.crosstab(df[plot_criteria[0]], df[plot_criteria[1]], normalize='columns') *
100,2)).style.background_gradient(cmap = cm)

plot_criteria= ['muscle stiffness', 'class']

cm = sns.light_palette("red", as_cmap=True)

(round(pd.crosstab(df[plot_criteria[0]], df[plot_criteria[1]], normalize='columns') *
100,2)).style.background_gradient(cmap = cm)

Image(filename='alopecia.png')

plot_criteria= ['Alopecia', 'class']

cm = sns.light_palette("red", as_cmap=True)

(round(pd.crosstab(df[plot_criteria[0]], df[plot_criteria[1]], normalize='columns') *
100,2)).style.background_gradient(cmap = cm)

plot_criteria= ['Obesity', 'class']

cm = sns.light_palette("red", as_cmap=True)

```

```

(round(pd.crosstab(df[plot_criteria[0]], df[plot_criteria[1]], normalize='columns') *
100,2)).style.background_gradient(cmap = cm)

df['class'] = df['class'].apply(lambda x: 0 if x=='Negative' else 1)

X= df.drop(['class'],axis=1)

y=df['class']

objList = X.select_dtypes(include = "object").columns
print (objList)

#Label Encoding for object to numeric conversion
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

for feat in objList:

    X[feat] = le.fit_transform(X[feat].astype(str))

print (X.info())

X.head()

X.corrwith(y)

#Correlation with Response Variable class

X.corrwith(y).plot.bar(

    figsize = (16, 6), title = "Correlation with Diabetes", fontsize = 15,

    rot = 90, grid = True)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,stratify=y, random_state = 1234)

## checking distribution of traget variable in train test split

print('Distribution of traget variable in training set')

print(y_train.value_counts())

print('Distribution of traget variable in test set')

print(y_test.value_counts())

minmax = MinMaxScaler()

X_train[['Age']] = minmax.fit_transform(X_train[['Age']])

X_test[['Age']] = minmax.transform(X_test[['Age']])

X_train.head()

logi = LogisticRegression(random_state = 0, penalty = 'l2')

logi.fit(X_train, y_train)

from sklearn import model_selection

```

```

kfold = model_selection.KFold(n_splits=10, random_state=7)
scoring = 'accuracy'
acc_logi = cross_val_score(estimator = logi, X = X_train, y = y_train, cv = kfold,scoring=scoring)
acc_logi.mean()
y_predict_logi = logi.predict(X_test)
acc= accuracy_score(y_test, y_predict_logi)
roc=roc_auc_score(y_test, y_predict_logi)
prec = precision_score(y_test, y_predict_logi)
rec = recall_score(y_test, y_predict_logi)
f1 = f1_score(y_test, y_predict_logi)
results = pd.DataFrame([['Logistic Regression',acc, acc_logi.mean(),prec,rec, f1,roc]],
                        columns = ['Model', 'Accuracy','Cross Val Accuracy', 'Precision', 'Recall', 'F1 Score','ROC'])
results
cm_logi = confusion_matrix(y_test, y_predict_logi)
plt.title('Confusion matrix of the Logistic classifier')
sns.heatmap(cm_logi,annot=True,fmt="d")
plt.show()
TP = cm_logi[1,1] # true positive
TN = cm_logi[0,0] # true negatives
FP = cm_logi[0,1] # false positives
FN = cm_logi[1,0] # false negatives
# Let us calculate specificity
TN / float(TN+FP)
feature_importance = abs(logi.coef_[0])
feature_importance = 100.0 * (feature_importance / feature_importance.max())
sorted_idx = np.argsort(feature_importance)
pos = np.arange(sorted_idx.shape[0]) + .3
featfig = plt.figure(figsize=(12,8))
featax = featfig.add_subplot(1, 1, 1)
featax.barh(pos, feature_importance[sorted_idx], align='center')
featax.set_yticks(pos)

```

```

featax.set_yticklabels(np.array(X_train.columns)[sorted_idx], fontsize=8)
featax.set_xlabel('Relative Feature Importance')
plt.tight_layout()
plt.show()

rf = RandomForestClassifier(criterion='gini',n_estimators=100)
rf.fit(X_train,y_train)

kfold = model_selection.KFold(n_splits=10, random_state=7)
scoring = 'accuracy'

acc_rf = cross_val_score(estimator = rf, X = X_train, y = y_train, cv = kfold,scoring=scoring)
acc_rf.mean()

y_predict_r = rf.predict(X_test)
roc=roc_auc_score(y_test, y_predict_r)
acc = accuracy_score(y_test, y_predict_r)
prec = precision_score(y_test, y_predict_r)
rec = recall_score(y_test, y_predict_r)
f1 = f1_score(y_test, y_predict_r)

model_results = pd.DataFrame([[ 'Random Forest (Untuned)',acc, acc_rf.mean(),prec,rec, f1,roc]],
                             columns = ['Model', 'Accuracy','Cross Val Accuracy', 'Precision', 'Recall', 'F1 Score','ROC'])

results = results.append(model_results, ignore_index = True)

results

cm_rf = confusion_matrix(y_test, y_predict_r)
plt.title('Confusion matrix of the Random Forest classifier')
sns.heatmap(cm_rf,annot=True,fmt="d")
plt.show()

TP = cm_rf[1,1] # true positive
TN = cm_rf[0,0] # true negatives
FP = cm_rf[0,1] # false positives
FN = cm_rf[1,0] # false negatives
# Let us calculate specificity
TN / float(TN+FP)

feat_importances = pd.Series(rf.feature_importances_, index=X_train.columns)
feat_importances.nlargest(20).plot(kind='barh')

```

```

X.columns
X.head()

X_fs = X[['Polyuria', 'Polydipsia', 'Age', 'Gender', 'partial paresis', 'sudden weight loss', 'Irritability',
'delayed healing', 'Alopecia', 'Itching']]

X_train, X_test, y_train, y_test = train_test_split(X_fs, y, test_size = 0.2, stratify=y, random_state =
1234)

minmax = MinMaxScaler()

X_train[['Age']] = minmax.fit_transform(X_train[['Age']])
X_test[['Age']] = minmax.transform(X_test[['Age']])

logi = LogisticRegression(random_state = 0, penalty = 'l2')

logi.fit(X_train, y_train)

kfold = model_selection.KFold(n_splits=10, random_state=7)

scoring = 'accuracy'

acc_logi = cross_val_score(estimator = logi, X = X_train, y = y_train, cv = kfold, scoring=scoring)

acc_logi.mean()

y_predict_logi = logi.predict(X_test)

acc= accuracy_score(y_test, y_predict_logi)

roc=roc_auc_score(y_test, y_predict_logi)

prec = precision_score(y_test, y_predict_logi)

rec = recall_score(y_test, y_predict_logi)

f1 = f1_score(y_test, y_predict_logi)

model_results = pd.DataFrame([['Logistic Regression-Post FS', acc, acc_logi.mean(), prec, rec, f1, roc]],
                             columns = ['Model', 'Accuracy', 'Cross Val Accuracy', 'Precision', 'Recall', 'F1 Score', 'ROC'])

results = results.append(model_results, ignore_index = True)

results

rf = RandomForestClassifier(criterion='gini', n_estimators=100)

rf.fit(X_train, y_train)

kfold = model_selection.KFold(n_splits=10, random_state=7)

scoring = 'accuracy'

acc_rf = cross_val_score(estimator = rf, X = X_train, y = y_train, cv = kfold, scoring=scoring)

acc_rf.mean()

y_predict_r = rf.predict(X_test)

```

```

roc=roc_auc_score(y_test, y_predict_r)
acc = accuracy_score(y_test, y_predict_r)
prec = precision_score(y_test, y_predict_r)
rec = recall_score(y_test, y_predict_r)
f1 = f1_score(y_test, y_predict_r)
model_results = pd.DataFrame([[ 'Random Forest- Post FS',acc, acc_rf.mean(),prec,rec, f1,roc]],
                             columns = ['Model', 'Accuracy','Cross Val Accuracy', 'Precision', 'Recall', 'F1 Score','ROC'])
results = results.append(model_results, ignore_index = True)
results
cm_rf = confusion_matrix(y_test, y_predict_r)
plt.title('Confusion matrix of the Random Forest classifier')
sns.heatmap(cm_rf,annot=True,fmt="d")
plt.show()
TP = cm_rf[1,1] # true positive
TN = cm_rf[0,0] # true negatives
FP = cm_rf[0,1] # false positives
FN = cm_rf[1,0] # false negatives
# Let us calculate specificity
TN / float(TN+FP)
feat_importances = pd.Series(rf.feature_importances_, index=X_train.columns)
feat_importances.nlargest(20).plot(kind='barh')
from sklearn import metrics
import matplotlib.pyplot as plt

plt.figure()

# Add the models to the list that you want to view on the ROC plot
models = [
{
    'label': 'Logistic Regression',
    'model': LogisticRegression(random_state = 0, penalty = 'l2'),
},

```

```
{
    'label': 'Random Forest',
    'model': RandomForestClassifier(n_estimators = 100,criterion='gini', random_state = 47),
},

]
```

Below for loop iterates through your models list

for m in models:

 model = m['model'] # select the model

 model.fit(X_train, y_train) # train the model

 y_pred=model.predict(X_test) # predict the test data

Compute False postive rate, and True positive rate

 fpr, tpr, thresholds = metrics.roc_curve(y_test, model.predict_proba(X_test)[:,-1])

Calculate Area under the curve to display on the plot

 auc = metrics.roc_auc_score(y_test,model.predict(X_test))

Now, plot the computed values

 plt.plot(fpr, tpr, label='%s ROC (area = %0.2f)' % (m['label'], auc))

Custom settings for the plot

plt.plot([0, 1], [0, 1], 'r--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('1-Specificity(False Positive Rate)')

plt.ylabel('Sensitivity(True Positive Rate)')

plt.title('Receiver Operating Characteristic')

plt.legend(loc="lower right")

plt.show()