BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN

University of Applied Sciences

Fachbereich II Mathematik – Physik –Chemie

Bachelorarbeit

Von

Ranjit Sah

Zur Erlangung
des akademischen Grades
Bachelor of Science(B.Sc.)

im Studiengang
Mathematik

Thema: **Investigating tree models in the analysis of diabetes dataset**

Betreuer: Prof. Dr. Timothy Downie
Gutachter: Prof. Dr. Patrick Erdelt

Eingereicht: May 15, 2020

Declaration

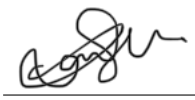
I assure that i wrote this thesis independently without outside help and that i only used the sources and resources indicated. Passages are taken literally or according to the meaning from other works that are marked with the sources.

Erklärung

Ich versichere, dass ich diese Abschlussarbeit ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

15.05.2020

Date



Signed

Abstract

Diabetes is a major health problem that is affecting more and more people over time. When it comes to diagnosing such diseases in humans, doctors make their diagnoses based on some suitable tests, and possibly without considering other factors related to the disease. Creating tools to analyze information about patients' current health can help physicians provide more information for diagnosis. Since there is still no cure for this disease, a person who has been diagnosed with diabetes need to control their blood sugar levels between some thresholds. This is extremely important because an uncontrolled blood sugar level can lead to serious health complications and can affect his lifestyle. Using tools that predict the subject's glucose level within a prediction horizon, the individual can take preventive measures to avoid exceeding normal thresholds. This work aims to investigate machine learning methods for such problems. These methods are already used in medicine and enable us to develop computational models that provide the medical team with more relevant data for diagnosing diabetes in humans and prevent patients from exceeding thresholds in controlling glucose levels. We have created various models for the problem of diabetes diagnosis, optimized and tested them using the PIMA data set. An important contribution of this work is the different methods that are introduced and analyzed. Firstly Dataset is loaded and then we applied the data cleaning process to get better results. After that, we Analyzed and predicted by creating a classifier and Regression model called Decision Tree classifier which gives around **85.28%**. We also predicted by random forest classifier model that gave around **90.48%**, which is the best model based on accuracy score. Then after that model was predicted by logistic regression which gave around **81.34%**, which gave a little less accuracy than tree classification model. Regarding several recent research papers and other sources, research was conducted by training and testing the selected algorithms on the same data sets. The experiments carried out in this research area deal with the classification of accuracy, sensitivity, and specificity using True Positive (TP) and false positive (FP) in a confusion matrix generated by the respective algorithms. The results obtained show that tree algorithms outperform logistics concerning accuracy, memory, precision, and f-score. The perfection of these algorithms in the classification task is explained in more detail by analyzing and visualizing. The results of the study show that a random forest performs better than the decision tree classifier and the logistics regression.

Kurzfassung

Diabetes ist ein großes Gesundheitsproblem, von dem im Laufe der Zeit immer mehr Menschen betroffen sind. Wenn es darum geht, solche Krankheiten beim Menschen zu diagnostizieren, stellen Ärzte ihre Diagnosen auf der Grundlage geeigneter Tests und möglicherweise ohne Berücksichtigung anderer krankheitsrelevanter Faktoren. Durch das Erstellen von Tools zur Analyse von Informationen über den aktuellen Gesundheitszustand von Patienten können Ärzte weitere Informationen für die Diagnose bereitstellen. Da es immer noch keine Heilung für diese Krankheit gibt, muss eine Person, bei der Diabetes diagnostiziert wurde, ihren Blutzuckerspiegel zwischen einigen Schwellenwerten kontrollieren. Dies ist äußerst wichtig, da ein unkontrollierter Blutzuckerspiegel zu schwerwiegenden gesundheitlichen Komplikationen führen und seinen Lebensstil beeinträchtigen kann. Mithilfe von Tools, die den Glukosespiegel des Probanden innerhalb eines Vorhersagehorizonts vorhersagen, kann die Person vorbeugende Maßnahmen ergreifen, um zu vermeiden, dass normale Schwellenwerte überschritten werden. Ziel dieser Arbeit ist es, Methoden des maschinellen Lernens für solche Probleme zu untersuchen. Diese Methoden werden bereits in der Medizin eingesetzt und ermöglichen es uns, Rechenmodelle zu entwickeln, die dem medizinischen Team relevantere Daten für die Diagnose von Diabetes beim Menschen liefern und verhindern, dass Patienten Schwellenwerte bei der Kontrolle des Glukosespiegels überschreiten. Wir haben verschiedene Modelle für das Problem der Diabetesdiagnose erstellt, diese optimiert und anhand des PIMA-Datensatzes getestet. Ein wichtiger Beitrag dieser Arbeit sind die verschiedenen Methoden, die eingeführt und analysiert werden. Zuerst wird der Datensatz geladen und dann haben wir den Datenbereinigungsprozess angewendet, um bessere Ergebnisse zu erzielen. Danach haben wir analysiert und vorhergesagt, indem wir einen Klassifikator und ein Regressionsmodell namens Decision Tree Classifier erstellt haben, die ungefähr **85,28%** ergeben. Wir haben auch das Random Forest Classifier Model vorhergesagt, das ungefähr **90.48%** ergab. Dies ist das beste Modell auf der Grundlage der Genauigkeitsbewertung. Danach wurde dieses Modell durch logistische Regression vorhergesagt, die ungefähr **81.34%** ergab, was eine etwas geringere Genauigkeit als das Baumklassifizierungsmodell ergab. Unter Bezugnahme auf mehrere neuere Forschungsarbeiten und andere Quellen wurde die Forschung durchgeführt, indem die ausgewählten Algorithmen an denselben Datensätzen trainiert und getestet wurden. Die in diesem Forschungsbereich durchgeführten Experimente befassen sich mit der Klassifizierung von Genauigkeit, Sensitivität und Spezifität unter Verwendung von True Positive (TP) und False Positive (FP) in einer von den jeweiligen Algorithmen erzeugten Verwirrungsmatrix. Die erhaltenen Ergebnisse zeigen, dass Baumalgorithmen die Logistik in Bezug auf Genauigkeit, Speicher, Präzision und F-Score übertreffen. Die Perfektion dieser Algorithmen in der Klassifizierungsaufgabe wird durch Analyse und Visualisierung näher erläutert. Die Ergebnisse der Studie zeigen, dass a random forest ist besser abschneidet als der Entscheidungsbaum klassifikator und die logistische regression.

Dedication

I would like to dedicate this work to god, my family, and many friends. A special feeling of gratitude towards my beloved parents, **Lal Babu Sah** and **Renu Rakhi** whose words of encouragement and push for tenacity resonate in my ears. I also dedicate this work to my Wife; **Nilam Tamang** who has encouraged me all the way and whose encouragement has made sure that I give it all it takes to finish that which I have started. I also dedicate this thesis to my many friends and family members of those who have supported me throughout the process. I will always be thankful for everything they have done. Thank you very much. My love for all of you can never be express. God bless you

Acknowledgements

First of all, I would like to thank God for the good health and well-being required to complete this thesis. I would like to thank all faculty members of Beuth University for their help and support. I also thank my parents for their constant encouragement, support, and attention. My special thanks go to my teacher Prof. Dr.Timothy Downie and Prof. Dr. Patrick Erdelt, who gave me the unique opportunity to carry out this wonderful project on this topic (examining tree models while analyzing diabetes data sets). It also helped me to do a lot of research and I learned so many new things that I am really grateful to them. Second, I would like to thank my parents and friends who have helped me a lot to complete this project within a limited time frame. I would also like to thank everyone who has contributed directly or indirectly to this work.

Table of contents

1. Introduction

Diabetes is one of the most common diseases that occur in older people worldwide. According to the International Diabetes Federation [2], 463 million people worldwide were diabetics in 2019. This number is expected to increase to 700 million people in the next 26 years. Diabetes is considered a chronic condition associated with an abnormal condition of the human body in which blood sugar levels are inconsistent with insulin due to a pancreatic dysfunction that produces little or no insulin and causes type 1 diabetes or cells and cause type 2 diabetes [20]. The main cause of diabetes is still unknown.

However, scientists believe that both genetic factors and the lifestyle of the environment play an important role in diabetes. Even if it is incurable, treatment and medication can treat it. Diabetics run the risk of developing secondary health problems such as heart disease and nerve damage. The early detection and treatment of diabetes can thus prevent complications and reduce the risk of serious health problems.

1.1 Related Works

I took the diabetes datasets from the kaggle website which is originally taken from the National Institute of Diabetes and Digestive and Kidney Diseases for a case study, there are many benefits to Kaggle's website services. It provides this data for free to the user and invites programmers to take part in the challenge of the diabetes dataset in which the participant can look for an algorithm that can diagnose whether a patient has diabetes or not based on the dataset and produce a reasonable comparison with the next release of the dataset. There are many related jobs available based on this dataset. Also, due to the increasing size and complexity of the Data Many Machine Learning and Deep Learning (DL) was introduced to predict diagnostically whether a patient has diabetes or not [33]. Current studies that have used ML gave remarkable results. The rate of accuracy produced by these methods varied. This has encouraged researchers to try to improve accuracy by creating models with classifiers that have not been used or combined with different classifiers. The majority of diabetes prediction studies used this public Pima Indian record from the UCI repository. Some surveys have been published [1], but they differ from this study. This paper provides a systematic overview of the services of Different machines learning classifiers derived from different papers over the past Years.

Many previous studies have used prediction to address this disease and to develop systems and tools that help predict diabetes. In this study, we will use different common types of machine learning algorithms like Decision trees, Random Forest, and Logistic regression. which is the most common algorithm for Classification and Regression [20]. The Artificial Neural Network (ANN) is another type of machine learning technique. It's commonly known for its high performance and accuracy [1, 33]

1.2 Motivation and goals

In the analysis of big data, it involves processing large amounts of diverse data and discovering the correlation and trends of the data for useful information [29]. There is always a challenge in processing this type of data set, as the amount of data to be processed is very high and the engineer needs to use computational resources effectively to avoid large analysis times. The present project revolves around the big data analysis of a selected dataset, i.e. the diabetes dataset. It reads various information from the data set and performs the necessary data analysis to obtain certain useful interpretations. Most previous studies on the Diabetes dataset revolved around prediction algorithms [34].

In this study, we attempt to introduce other data analysis methods such as decision tree models, logistic regression models, and random forest models of diabetes data to gain some insight into the Medical field that may be useful to doctors that help predict diabetes. The primary objectives of the project can be divided into two parts. The first is to understand the diabetes dataset and study several independent variables which include the number of pregnancies the patient has had, their BMI, insulin level, age, and so one and one target(dependent) variable called Outcome(0 or 1). In this study, relevant features are obtained based on certain selection criteria. In the second part of the study, a subsequent Exploratory Data Analysis(EDA) of the data is made based on the classification. Interpretations were made from the studies. The scope of the thesis is dedicated to big data research and approaches regarding the selected dataset. I used the reference dataset to perform a true interpretation of the data based on the analysis. The study introduces various functions for reading and analyzing the data, and a sensitivity study of details on various parameters used in these functions is performed.

1.3 Organization of this Thesis

Generally, one component of the study organization is to provide a map that guides the reader through reading and understanding the dissertation. Organization of the thesis is categories in different ways as follows:

- **Chapter 2:**The background of the present study is discussed
- **Chapter 3:** Discusses the Decision Tree methods that are used in this study
- **Chapter 4:**Discusses the results and interpretations based on Random forest.
- **Chapter 5:**Discusses the results and interpretations based on Logistic Regression.
- **Chapter 6:**Comparison of the Models
- **Chapter 7:**Discusses the summary of the present study.

2. Background

2.1 Theory

The decision tree is a flow-like tree structure in which an internal node represents a feature (or attribute), the branch represents a decision rule, and each leaf node represents the result. The top node in a decision tree is called the root node. Learn to partition based on the attribute value. Partition the tree recursively and call recursive partitions. This chart-like structure helps us to make decisions. Decision trees can be drawn by hand or created with a special graphics program or software. Informal decision trees are useful for focusing on the discussion when a group has to make a decision [17]. Decision trees are easy to understand and interpret.

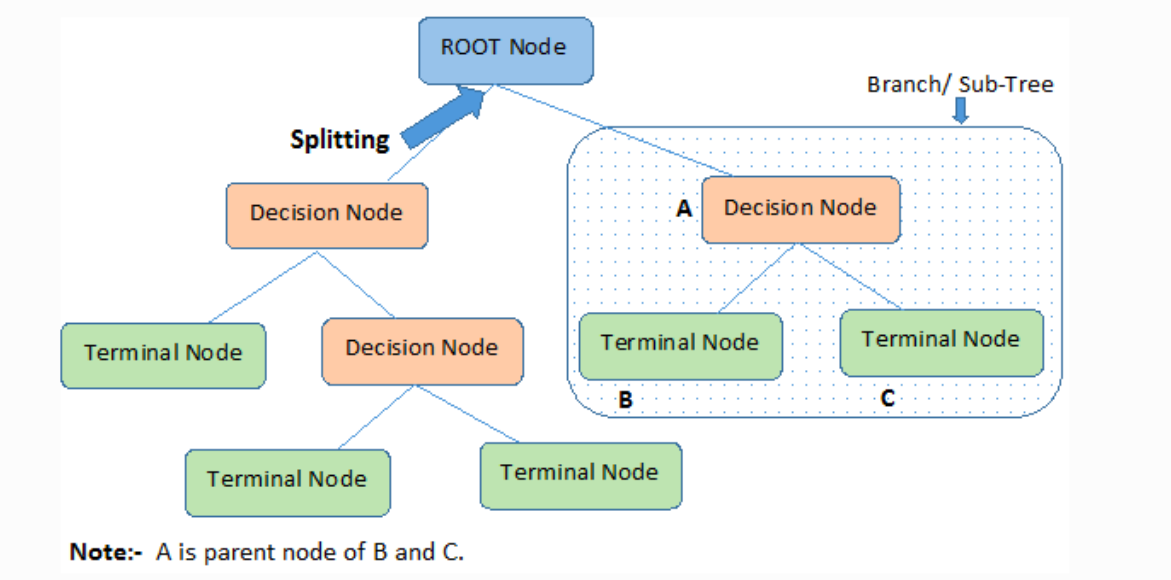


Figure 2.1: Decision tree classifier [17]

A decision tree [17] could be a form of white box ML algorithm. According to Dr. Ryohei Fujimaki, founder and CEO of data automation platform DotData, White box models are models that can clearly explain how they behave, how they make predictions, and what influencing variables they have. Two key elements make up a white box model: the characteristics must be understandable and the ML process must be transparent. These models include linear and decision/regression, tree models. It shares internal decision-making logic, which is not available within the recording equipment under algorithms such as Neural Network. Our training time is faster compared to the neural network algorithm. The time complexity of decision trees could be a function of the number of records and number of attributes within the given data. The choice tree could be a parametric or nonparametric method, which does not depend on probability distribution assumptions. The parametric term refers to the relationship between the number of parameters in the model and the data. When the number of parameters is fixed, the model is parametric. If the number of parameters increases with the

data, the model is not parametric. A decision tree is not parametric. However, if we limit its size for regularization, the number of parameters is also limited and can be considered fixed. Therefore, it is not so clear for decision trees. Decision trees can handle high-dimensional data with good accuracy. [11]. A decision tree can also be used to create automated predictive models that have applications for machine learning, data mining, and statistics. This method, known as decision tree learning, takes into account observations on an element to predict the value of that element. In these decision trees, the nodes represent data instead of decisions. This type of tree is also called a classification tree Each branch contains a set of attributes or classification rules that are associated with a particular class tag that is at the end of the branch. These rules also called decision rules [17, 22].

2.1.1 Terminology

- **Root Node:** In a decision tree, the root node is the first node or the main node that represents the entire population or sample. Generally, nodes can have primary and secondary nodes, but since the root is the first node, it only has secondary nodes. A root node is like any node because it is part of a data structure that consists of one or more fields with links to other nodes and contains a data field. It just turns out to be the first knot. In this regard, each node can be a root node concerning itself and its child nodes if this section of the tree is selected objectively.

- **Parent and child node:** A node, which is divided into subnodes, is called the parent node of subnodes while the subnodes are the child of a parent node. In another way, the parent node and all other nodes associated with it are referred to as child nodes, so we can understand it so that the entire top node belongs to a parent node and the entire bottom node derived from an upper node is a child node, Creating a node for the node is a child node [22].

- **Branch / Sub-tree:** A subsection of the entire tree or a tree which is a child of a node is called a branch or sub-tree. This branch or subtree is formed by splitting the tree.

- **Terminal/Leaf Node** Leaf node is also known as terminal node where decision trees are terminated or Ends. when we reach the end of the tree right that is we cannot further segregate it down to any other level. simply a node in a decision tree that has no child nodes.

- **Splitting:**Splitting is a process of dividing a node into two or more subnodes. The main goal of learning the decision tree is to find small and precise models. To heuristically get the smallest trees, several ways are adopted to determine the variable selection metric in which division is performed for a node. Different algorithms implement different metrics to decide which variable best divides the data set. we have to test all the attributes and choose the one that creates the purest nodes to be split at every step. Therefore, the Gini index (see page **7**)is used as the division criterion, the task of dividing the rules is to find the division that leads to the maximum information gain or Gini gain.

- **Pruning:** Pruning is the method of performing decision tree by removing the branches that use features with little importance. we can tell the opposite process of splitting. In this way, we reduce the complexity of the tree and thus increase its predictive power by reducing the overfitting. The simplest method of pruning we have discussed in section 3.4 Optimizing decision tree Performance. This change will persist unless accuracy is compromised. It is also referred to as reduced debugging. More sophisticated cleanup methods can be used, e.g. Clean up cost complexity using a learning parameter (alpha) to weigh whether nodes can be removed based on the size of the subtree. This is also called shortening the weakest links.

2.2 Algorithum

The decision tree algorithm belongs to the family of supervised learning algorithms. Unlike other supervised learning algorithms, the decision tree algorithm can be be used to solve both regression and classification problems [17]. The goal of using a decision tree is to create a training model that you can use to predict the class or value of the target variable by learning simple decision rules inferred from previous data (training data).

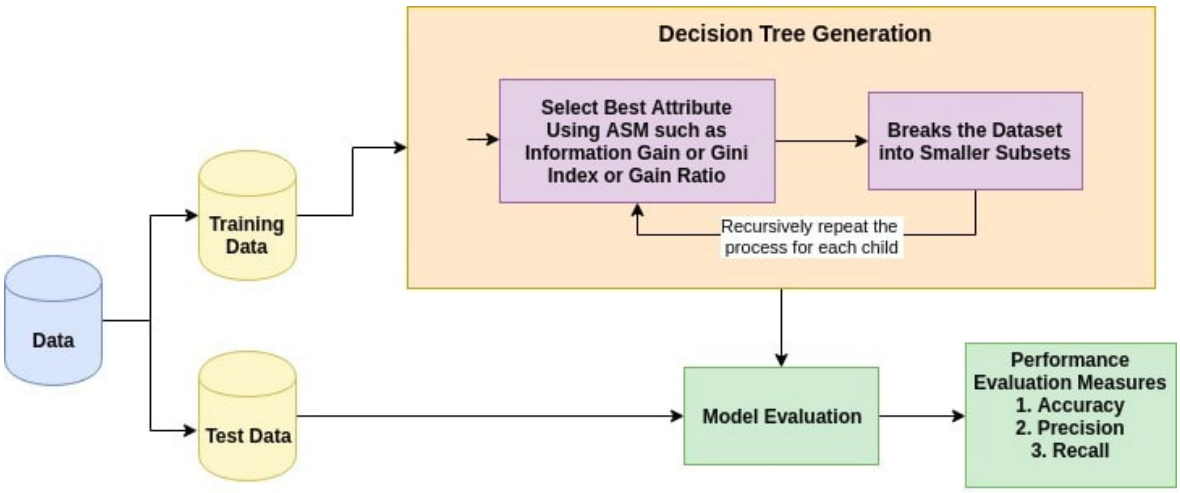


Figure 2.2: Decision Tree Algorithm [17]

In decision tree, to provide a class label for a record we start from the root of the tree. We compare the values of the root attribute with the record attribute. At the base of comparison, we follow the branch corresponding to that value and jump to the next node [17].

2.2.1 Attribute Selection Measures

The measure of attribute selection is the most important part of the Analysis field. It could be a heuristic to pick out the distribution criterion with which the information is split within the absolute best way. it's also called a split rule because it helps us determine breakpoints for tuples during a particular node. Attribute selection measures classifies each attribute explaining the required record. The most effective scoring attribute is chosen because of the divided attribute [28]. In the case of a continuous-valued attribute, split points for branches also need to define. The most popular selection measures are Information Gain, Gain Ratio, and Gini Index.
The best way to understand the importance of feature selection is to work with a set of data that contains a variety of features. This type of data set is often called a high-dimension data set. With this high dimensional, many problems arise, such as high dimensionality will significantly increase the training time of your machine learning model, it can make our model very complicated which in turn may lead to overfitting. A set of high-dimension features often contains several features that are redundant, which means that these features are nothing more than extensions of the other essential features. These redundant functions also do not contribute effectively to model training. Therefore, it is clear that the most important and relevant characteristics for a data set must be extracted to achieve the most effective predictive modeling performance [24].

2.2.2 Information Gain

Information gain is a statistical property that measures how well a given attribute separates the training examples according to their target classification [27]. In the figure below, we can see that an attribute with low information gain (right) splits the data relatively evenly and as a result doesn't bring us any closer to a decision. Whereas an attribute with high information gain (left) splits the data into groups with an uneven number of positives and negatives and as a result helps in separating the two from each other [31]. Now, we can say

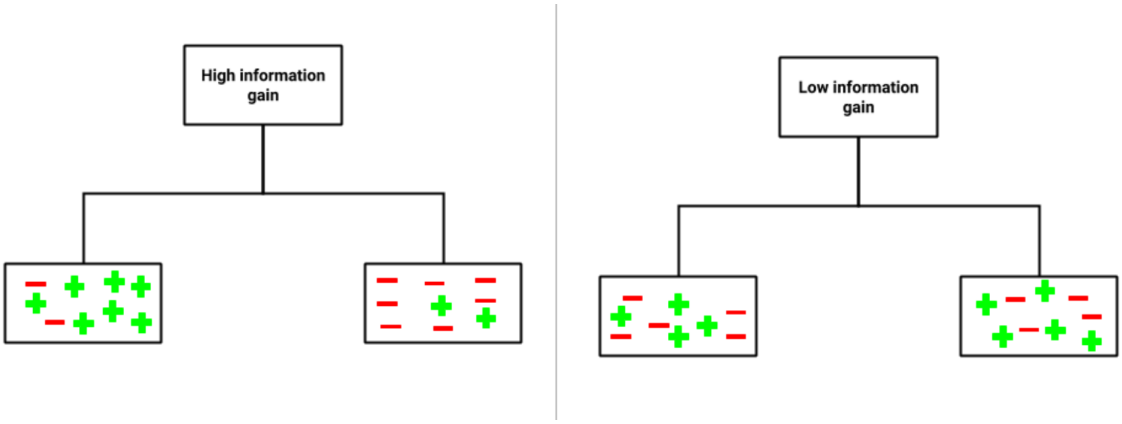


Figure 2.3: high and low information gain [31]

that less impure node requires less information to describe it. And, the more impure node requires more information.

Let node N represents or hold the tuple of partition D. The attribute with the highest information gain is chosen as the splitting attribute for the node N. The expected information needed to classify a tuple in D is given by,

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i) \tag{2.1}$$

Where p_i is the probability that an arbitrary tuple in D belongs to class C_i and is estimated by $|C_{i,D}|$ / |D|. Info(D) is the average amount of information needed to identify the class label of a tuple in D. Info(D) is also known as the entropy of D. The expected information required to classify a tuple from D, based on the partitioning by attribute A is calculated by

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \cdot Info(D) \tag{2.2}$$

Information gain is defined as the difference between the original information requirement (i.e. based on the classes) and the new requirement (i.e. obtained after partitioning on A)

$$Gain(A) = Info(D) - Info_A(D) \tag{2.3}$$

2.2.3 Gain Ratio

The information gain is biased for the attribute with many results. This means that the attribute with a large number of different values is preferred. Assume that an attribute with a unique identifier such as customer_ID has no information due to the pure partition (D). This maximizes information gain and creates useless partitions. The gain ratio addresses the problem of distortion by normalizing the collection of information through divided information.

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \cdot \log_2(\frac{|D_j|}{|D|}) \tag{2.4}$$

Where $|D_j|/|D|$ acts as the weight of the jth partition. v is the number of discrete values in attribute A. The gain ratio can be defined as

$$GainRatio(D) = \frac{Gain(A)}{SplitInfo_A(D)} \tag{2.5}$$

The attribute with the highest gain ratio is chosen as the splitting attribute [23].

2.2.4 Gini index

The Gini Index considers a binary split for each attribute. The Gini Index measures the impurity of D, a data partition or set of training tuples as,

$$Gini(D) \ = 1 - \sum_{j=1}^{m} p_I^2 \tag{2.6}$$

Where P_i is the probability that a tuple in D belongs to Class Ci and is estimated by $|C_{i,D}|$ / |D|. When considering a binary split, we compute a weighted sum of the impurity of each resulting partition. For example, if a binary split on A partitions D into D1 and D2, the gini index of D given that partitioning is

$$GiniA(D) = \frac{|D_1|}{|D|} \cdot \ GiniA(D_1) + \frac{|D_2|}{|D|} \cdot GiniA(D_2) \tag{2.7}$$

For each attribute, each of the possible binary split is considered. For a discrete valued attribute, the subset that gives the minimum gini index for that attribute is as its splitting attribute [8, 28].

2.3 Data Description

The data sets consist of several medical predictor (independent) variables and one target (dependent) variable, Outcome. Independent variables include 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 2 hours as part of an oral glucose tolerance test.
- **Blood pressure:** Diastolic blood pressure (mm Hg))
- **Skin Thickness:** Triceps skin fold thickness (mm)
- **Insulin:** 2-Hour serum insulin (mu U/ml)
- **BMI:** Body mass index (weight in kg /height in m^2)
- **Diabetes Pedigree Function:** Diabetes family tree function
- **Age:** Age (years)
- **Outcome:** Class variable (0 or 1)

2.3.1 Pima indians Diabities Dataset

The diabetes data set originally comes from the National Institute for Diabetes and Digestive and Kidney Diseases. The goal of the data set is to predict diagnostically whether a patient has diabetes or not based on certain diagnostic measurements included in the data set. Various restrictions have been imposed on the selection of these instances from a larger database. In particular, all patients here are women who are at least 21 years old and who are Pima Indians.

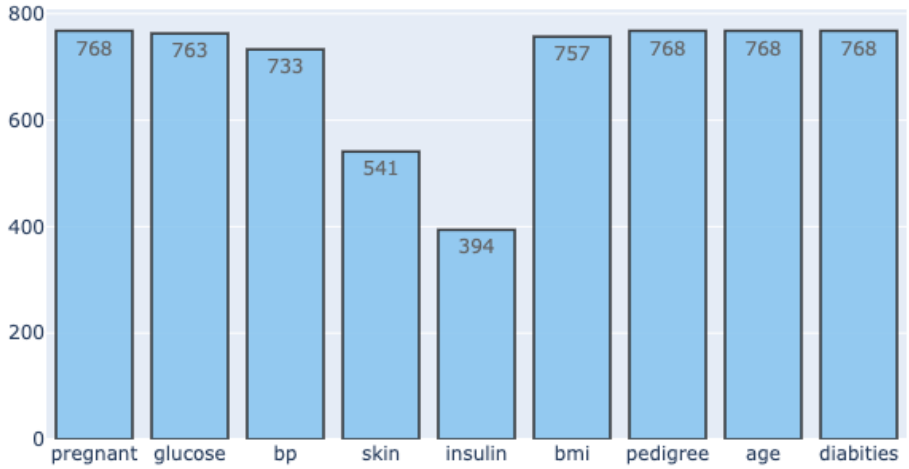


Figure 2.4: Decision Tree Algorithm

2.3.2 Loading and Reading Dataset

As data Analysts, we often work with lots of data. The data we want to load can be saved in different ways. The most common formats are CSV files, Excel files, or databases. The data can also be available in all web services. To work with the data, we have to present it in a tabular structure. All tables are arranged in a table with rows and columns. The data we want to load and read are explored using Pandas and Numpy Library. Python Code for the following Table is attached to the Appendix section.

Nr	Pregnancies	Glucose	BP	Thickness	Insulin	BMI	Pedigree	Age	Diabetes
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
5	5	116	74	0	0	25.6	0.201	30	0
6	3	78	50	32	88	31.0	0.248	26	1
7	10	115	0	0	0	35.3	0.134	29	0
8	2	197	70	45	543	30.5	0.158	53	1
9	8	125	96	0	0	0.0	0.232	54	1
10	4	110	92	0	0	37.6	0.191	30	0
11	10	168	74	0	0	38.0	0.537	34	1
12	10	139	80	0	0	27.1	1.441	57	0
13	1	189	60	23	846	30.1	0.398	59	1
14	5	166	72	19	175	25.8	0.587	51	1
15	7	100	0	0	0	30.0	0.484	32	1
16	0	118	84	47	230	45.8	0.551	31	1
17	7	107	74	0	0	29.6	0.254	31	1
18	1	103	30	38	83	43.3	0.183	33	0
19	1	115	70	30	96	34.6	0.529	32	1
20	3	126	88	41	235	39.3	0.704	27	0
21	8	99	84	0	0	35.4	0.388	50	0
22	7	196	90	0	0	39.8	0.451	41	1
23	9	119	80	35	0	29.0	0.263	29	1
24	11	143	94	33	146	36.6	0.254	51	1
25	10	125	70	26	115	31.1	0.205	41	1
26	7	147	76	0	0	39.4	0.257	43	1
27	1	97	66	15	140	23.2	0.487	22	0
28	13	145	82	19	110	22.2	0.245	57	0
29	5	117	92	0	0	34.1	0.337	38	0
30	5	109	75	26	0	36.0	0.546	60	0
31	3	158	76	36	245	31.6	0.851	28	1
32	3	88	58	11	54	24.8	0.267	22	0
33	6	92	92	0	0	19.9	0.188	28	0
34	10	122	78	31	0	27.6	0.512	45	0

Table 2.1: overview Diabities Dataset

2.3.3 Target Variable

The target or objective variable in the context of machine learning is the variable that is or should be the output. For example, it could be a binary 0 or 1. In statistics, it is also called the dependent variable. The above table 2.1 shows that the data is unbalanced. The following diagram shows that the number of non-diabetic is 268(34.9%) the number of diabetic patients is 500(65.1%).

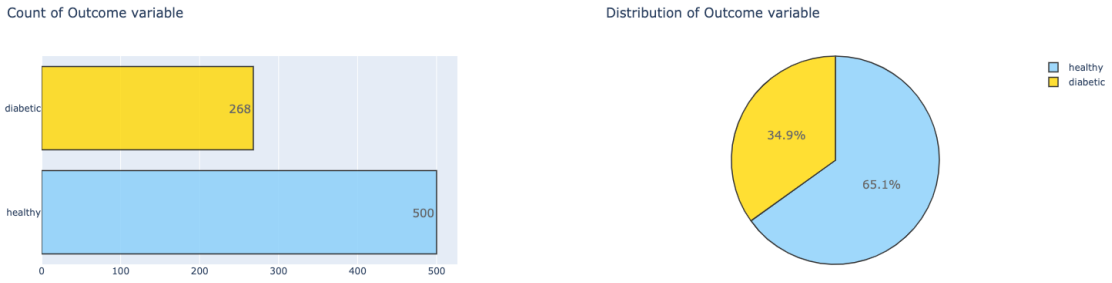


Figure 2.5: Target Variable(count & %)

2.3.4 Frequency Table

In statistics, a frequency distribution is a list, table, or graph that shows the frequency of different results in a sample. Each entry in the table contains the frequency or number of occurrences of values within a specific group or a specific range. We can overview the sample of each variable in a table form. we can also obtain statically information such as Mean, standard deviation, minimum value, maximum value and also quantile like 25%,50%,75% and so on 2.2.

	Variables	count	mean	std	min	25%.	50%.	75%.	max
1	pregnant	768	3.85	3.37	0	1	3	6	17
2	glucose	763	121.69	30.54	44	99	117	141	199
3	BP	733	72.41	12.38	24	64	72	80	122
4	skin	541	29.15	10.48	7	22	29	36	99
5	insulin	394	155.55	118.78	14	76.25	125	190	846
6	BMI	757	32.46	6.92	18.20	27.50	32.30	36.60	67.10
7	pedigree	768	0.47	0.33	0.08	0.24	0.37	0.63	2.42
8	age	768	33.24	11.76	21	24	29	41	81
9	diabetes	768	0.35	0.48	0	0	0	1	1

Table 2.2: Description of Dataset

2.4 Data preparation

Data preparation is a process of inspecting, cleaning, transforming and modeling data to discover useful information, suggesting conclusions and supporting decision-making.

2.4.1 Data Cleaning

There are many different tasks in data processing, e.g. data cleaning, data integration, and data transformation. The task that deals with such data problems are called data cleaning. Common data problems are missing values, outliers, inconsistent column names, etc. A large amount of data is created every day. With machine learning, this data can be learned and predictions made to make it valuable. A major problem, however, is that real data is almost never accessed in a clean manner, and poor data quality can seriously affect the effectiveness of learning algorithms. As a result, unprocessed data must be preprocessed before we can continue with the training or run machine learning models (see illustration). This process is as important and inevitable as data preprocessing, but also lengthy and problematic.

2.4.2 Missing Values

To manage missing values effectively, the first step is to understand the data and try to find out why the data is missing.We saw on **data.head()** that some features contain 0, it doesn't make sense here. The missing value were replaced as 0 by **NaN** in the table 2.1 on page 9.

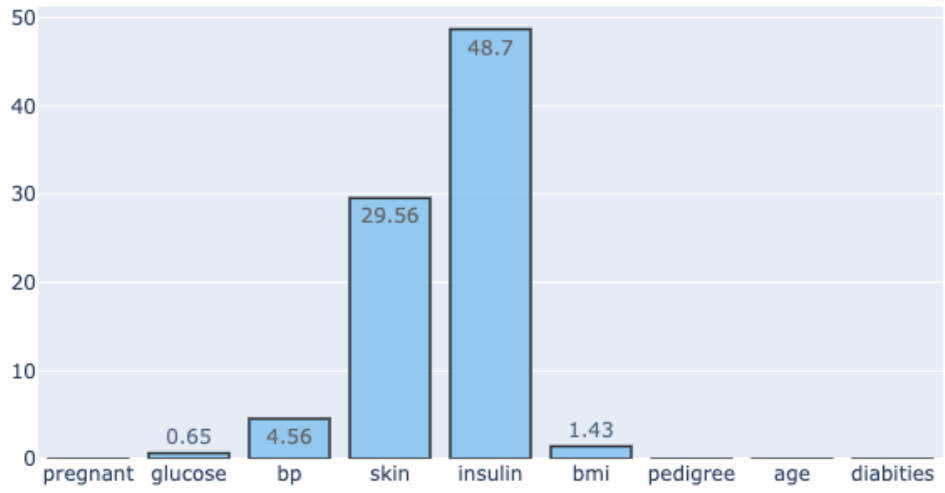


Figure 2.6: Missing Value in percentage

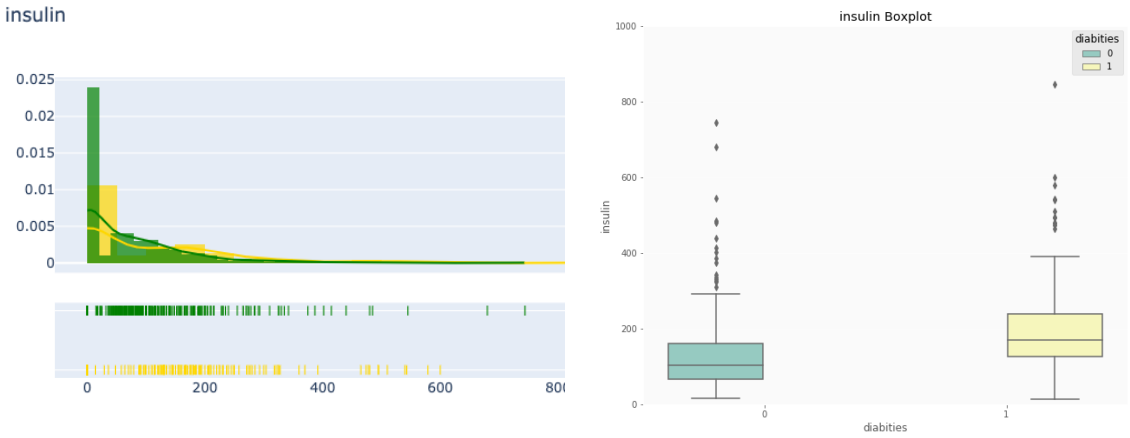
- Insulin = 48.7%- 374
- SkinThickness = 29.56% - 227
- BloodPressure = 4.56% - 35
- BMI = 1.43% - 11
- Glucose = 0.65% - 5

2.4.3 Replace Missing Values

After determining the mechanism of the missing data, the next step is to determine the appropriate method to replace the missing value. We can replace missing values under a variable with statistical information such as mean, median, or mode. However, the variability of the data is also underestimated, as the actual missing value can be far from the mean, median, or mode. In addition, estimates of covariance and correlation in the data can be weakened due to ignorance of the relationship and dependency between variables. Insulin, Skin Thickness, Blood pressure, BMI, and Glucose are replaced at first by using 0 as Nan and then Nan as a median of each feature.

2.4.3.1 Insulin

Insulin is critical for glucose to succeed in and be utilized by several important target tissues throughout the body. These include the liver muscle and fat insulin is critical to stay blood sugar levels stable within the body. Therefore, insulin helps cells absorb glucose, which is employed to come up with energy. When the body has enough energy, insulin prompts the liver to soak up glucose and store it as glycogen. The liver can store up to five of its mass as glycogen. Some cells within the body can absorb blood glucose without insulin, but most cells need insulin to be present. [6]. According to Fig 2.6, Insulin has the highest missing Values i.e 48.7%. The missing values of Insulin have been replaced by 102.5 healthy patients and 169.5 for diabetes patients.



(a) plot distribution for Insulin

(b) Boxplot for Insulin

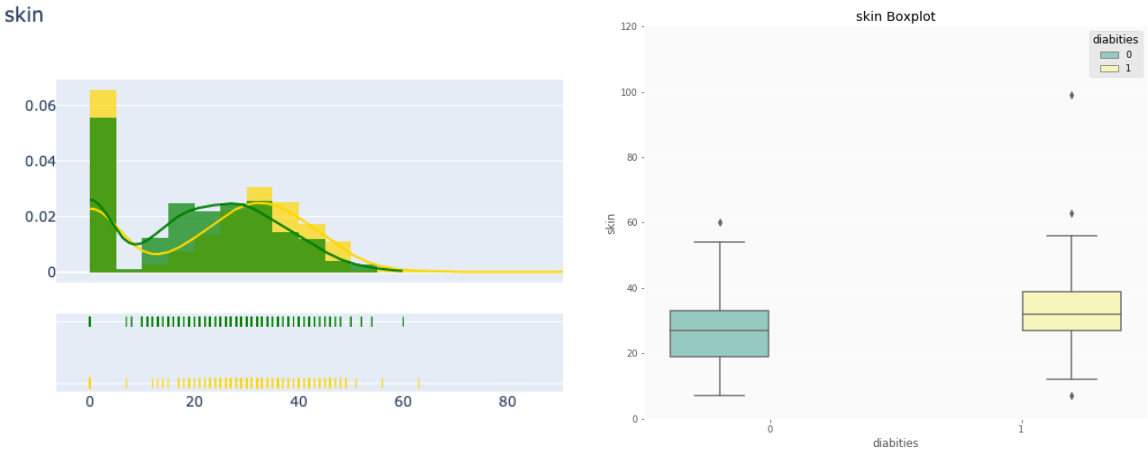
Figure 2.7: Graphic overview for Insulin variable

In the following subsections, we discuss a comparison of a box plot and the probability density function (PDF) which is essentially a smooth version of the histogram. In fig(a) y-axis is given in terms of density, and the histogram is normalized by default to have the same scale as the density graph. we can also see that there are two bell curve also called kernel density estimate(KDE), which lies on the Horizontal axis with green and yellow for healthy and diabetes persons respectively. We can see that from a rug plot that lies on a horizontal axis the insulin value from 0-200 overall is more distributed. We can easily read that insulin from 0-19 has covered the highest density for a healthy person. Also, insulin from 0-49 has covered the highest density for diabetes person. The reason for showing a statistical distribution is more common than looking at a boxplot. In other words, it could help us to understand a box diagram. In the above Fig (b) we can see two boxplot diagram. A box plot with green color on the left side is for the healthy person, which doesn't have diabetes. another side on right with yellow color is for a diabetic person. A box plot is a standardized method for displaying

data distribution based on a summary with five numbers which has "minimum"=14, first quartile (Q1)=76.25, median=125, third quartile (Q3)=190 and "maximum"=846. It can give us information about our outliers and their values. we can also determine if our data is symmetrical, how closely it is grouped, and whether our data is distorted.

2.4.3.2 Skin

Diabetes can affect any part of the body, including the skin. Such problems are sometimes the primary sign that someone has diabetes. Fortunately, most skin disorders may be easily prevented or treated if detected early. It has been suggested that thick skin on the back of the hands and fingers is a common finding in diabetes mellitus and that affected people have a higher prevalence of diabetic microvascular retinal diseases [14]. According to Fig 2.6 Skin has second-highest missing Values i.e 29.56%. The missing values of Insulin have been replaced by 27.0 healthy patients and 32.0 for diabetes patients.



(a) plot distribution for skin

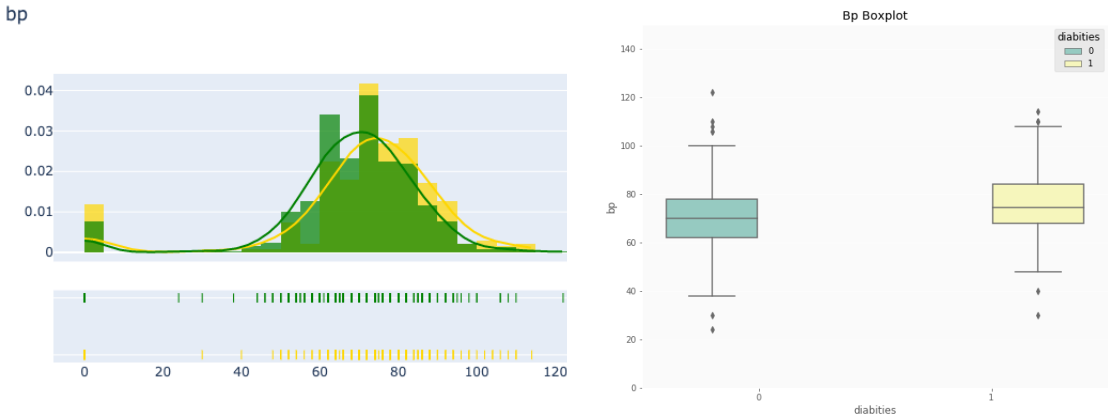
(b) Boxplot for skin

Figure 2.8: Graphic overview for skin variable

The picture 2.8 above is a comparison of a box plot(fig b) and the probability density function(fig a) for a healthy and diabetes person comparison to the skin variable. We can easily read that the skin from 0-4 has covered the highest density i.e 0.06 for diabetes person. Also, the insulin from 0-4 has covered the highest density i.e 0.05 for a healthy person. According to our frequency table 2.2, we can read all the information for Boxplot. In the above Fig (b) we can see two box plot diagrams. A box plot with yellow color on the right side is for diabetes person, which has "minimum"=7, first quartile (Q1)=22, median=29, third quartile (Q3)=36 and "maximum"=99.

2.4.3.3 Bloodpressure

Diabetes increases the risk of heart disease, stroke, kidney disease, and other health problems. High blood pressure also increases this risk. If a patient suffers from diabetes and high blood pressure together, this further increases his/her risk of health problems. If a patient has diabetes, our doctor should make sure that blood pressure is very well controlled. This means that they will probably want his/her blood pressure to be below 130 over 80. According to Fig 2.6 Blood pressure has the third-highest missing Values i.e 4.56%. The imputed medians were calculated separately for diabetes(74.5) and non-diabetes(70) groups. The missing values of BP have been replaced by 70.0 healthy patients and 74.5 for diabetes patients.



(a) plot distribution for BP

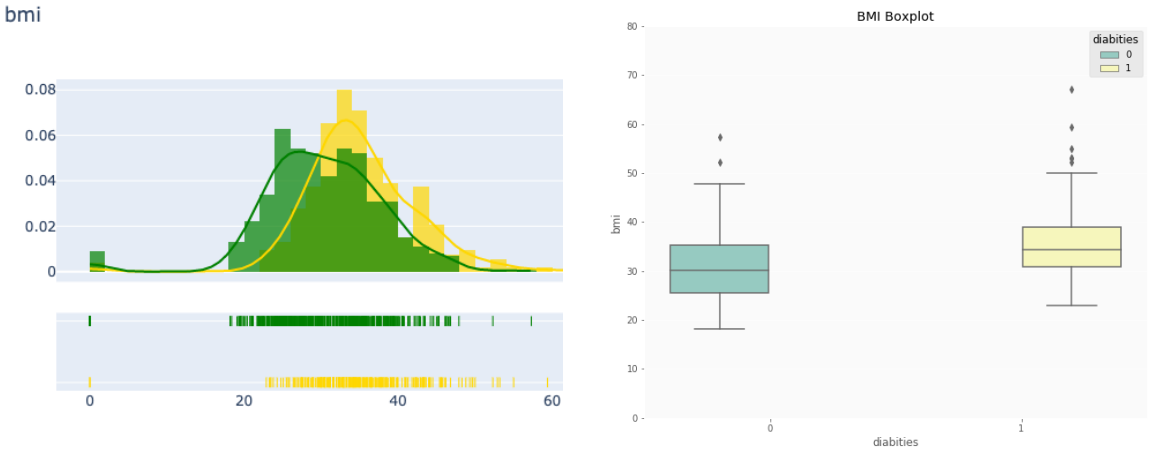
(b) Boxplot for Blood pressure

Figure 2.9: Graphic overview for Blood pressure variable

The picture 2.9 is a comparison of a box plot(fig b) and the probability density function(fig a) for a healthy and diabetes person comparison to the blood pressure variable. We can easily read that the Blood pressure from 70-74 has covered the highest density i.e 0.03 for a healthy person. Also, the Blood pressure from 0-74 has cover the highest density i.e 0.04 for diabetes person. From range 40-100 has more data distribution. According to our frequency table 2.2, we can read all the information for the boxplot. In the above Fig (b) we can see two box plot diagrams. A box plot with yellow color on the right side is for diabetes person, which has "minimum"=24, first quartile (Q1)=64, median=72, third quartile (Q3)=80 and "maximum"=122.

2.4.3.4 BMI

A higher amount of body fat increases the likelihood of developing diabetes. if a patient has diabetes, extra weight means his/her is at higher risk of complications like heart attacks or strokes. When talking about diabetes and weight, our doctor will likely point to our Body Mass Index (BMI), which is a measure of how much body fat we have. According to the American Diabetes Association (ADA), it is an important measure of diabetes and weight management, but it has its limits. According to Fig 2.6, BodyMassindex has the fourth-highest missing Values i.e 1.43%. The missing values of body mass index have been replaced by 30.1 healthy patients and 34.3 for diabetes patients.



(a) plot distribution for BMI (b) Boxplot for BMI

Figure 2.10: Graphic overview for BMI variable

The above picture 2.10 is a comparison of a box plot(fig b) and the probability density function(fig a) for a healthy and diabetes person comparison to the Body mass index variable. We can easily read that the value from 24-25 has covered the highest density i.e 0.063 for a healthy person. Also, the BMI from 32-33 has covered the highest density i.e 0.08 for diabetes person. From range 20-50 has more data distribution. According to our frequency table 2.2, we can read all the information for Boxplot. In the above Fig (b) we can see two box plot diagrams. A box plot with yellow color on the right side is for diabetes person, which has "minimum"=18.20, first quartile (Q1)=27.5, median=32.30, third quartile (Q3)=36.60 and "maximum"=67.10.

2.4.3.5 Glucose

Glucose is a natural sugar that supplies energy to cells. The presence of glucose in the blood stimulates the pancreas to secrete insulin. Insulin facilitates the transport of glucose from the blood to the cells in which it is used. If not enough insulin is excreted, the blood glucose level remains high. A constantly high level of blood sugar, caused by insufficient insulin, is diabetes mellitus [9]. According to Fig 2.6, Glucose has minimum missing Values i.e 0.65%. The missing values of glucose has been replaced by 107.0 healthy patient and 140.0 for diabetes patient.

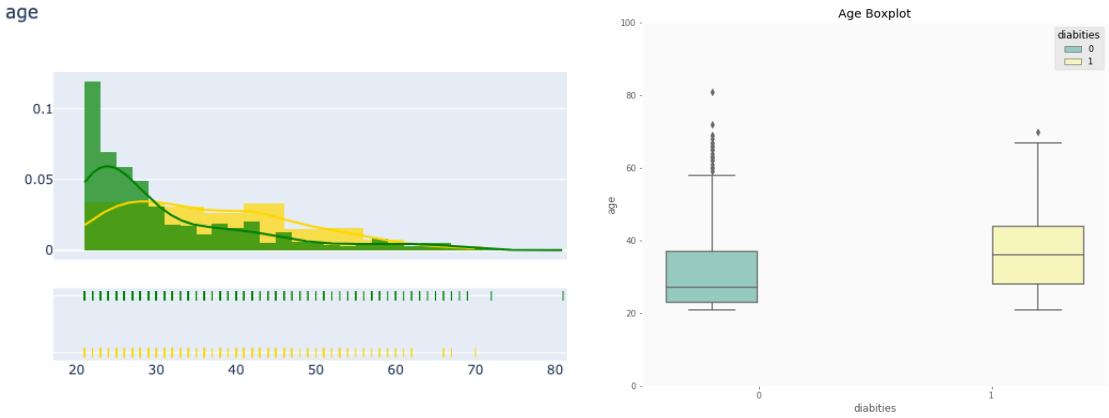
(a) plot distribution for Glucose (b) Boxplot for Glucose

Figure 2.11: Graphic overview for Glucose variable

The picture 2.11 is a comparison of a box plot(fig b) and the probability density function(fig a) for a healthy and diabetes person comparison to the glucose variable. We can easily read that the glucose from 105-109 has covered the highest density i.e 0.019 for a healthy person. Also, the glucose from 120-129 has covered the highest density i.e 0.013 for diabetes person. From range 50-150 more data is distributed. According to the frequency table 2.3.4, we can read all the information for the boxplot. In the above Fig (b) we can see two box plot diagrams. A box plot with yellow color on the right side is for diabetes person, which has "minimum"=44, first quartile (Q1)=99, median=117, third quartile (Q3)=141 and "maximum"=199.

2.4.3.6 Age

According to Fig 2.6 Age, Pedigree and Pregnant have no missing values. Aging can make diabetes difficult to control. One reason for this is that insulin resistance increases with age and glucose tolerance decreases, but there are also other indirect reasons. As people's attention decreases with age, they may be less able to focus on tasks related to the treatment of diabetes, such as carbohydrate counting, meal planning, blood sugar control and determining the right insulin doses.



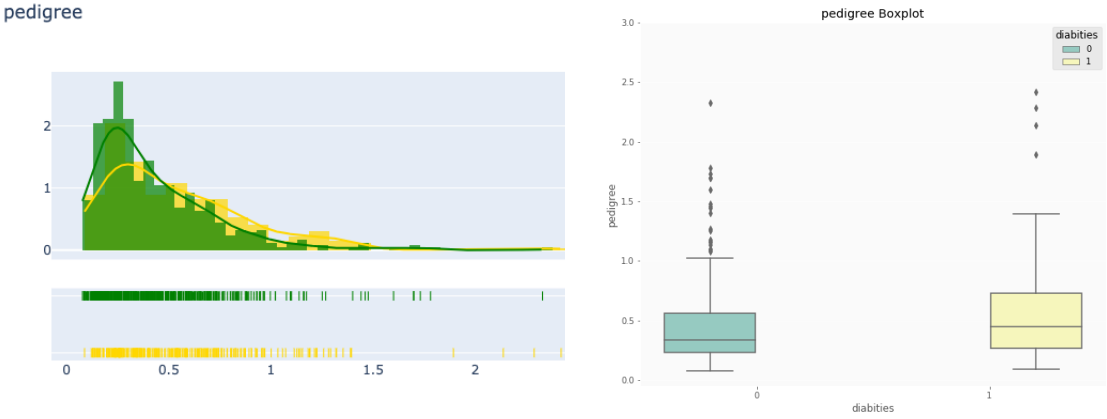
(a) plot distribution for Age (b) Boxplot for Age

Figure 2.12: Graphic overview for Age variable

Similarly, the above picture 2.12 is a comparison of a box plot(fig b) and distribution plot(fig a) for a healthy and diabetes person comparison to the Age variable. The age between 21.22 has a more density value of i.e 0.11 for a healthy person. similarly, the age between 30-55 has a density of nearly 0.025 for diabetes person. From the range, 30-60 age has more data distribution and they are mostly diabetes persons. According to our frequency table 2.2, we can read all the information for Boxplot. In the above Fig (b) we can see two box plot diagrams. A box plot with yellow color on the right side is for diabetes person, which has "minimum"=21, first quartile (Q1)=24, median=29, third quartile (Q3)=41 and "maximum"=81.

2.4.3.7 Pedigree function

Similarly, fig 2.13 is a comparison of a box plot(fig b) and the probability density function(fig a) for a healthy and diabetes person comparison to the pedigree function variable. The pedigree function value from 0.22-0.27 has the highest density of i.e 2.72 for a healthy person. Also, the range from 0.32-0.38 has the highest density value i.e 1.33 for diabetes person. Overall from range from 0.07-0.78 more data is distributed. According to our frequency table 2.2, we can read all the information for Boxplot. In the above Fig (b) we can see two box plot diagrams. A box plot with yellow color on the right side is for diabetes person, which has "minimum"=0.08, first quartile (Q1)=0.24, median=0.37, third quartile (Q3)=0.63 and "maximum"=2.42.

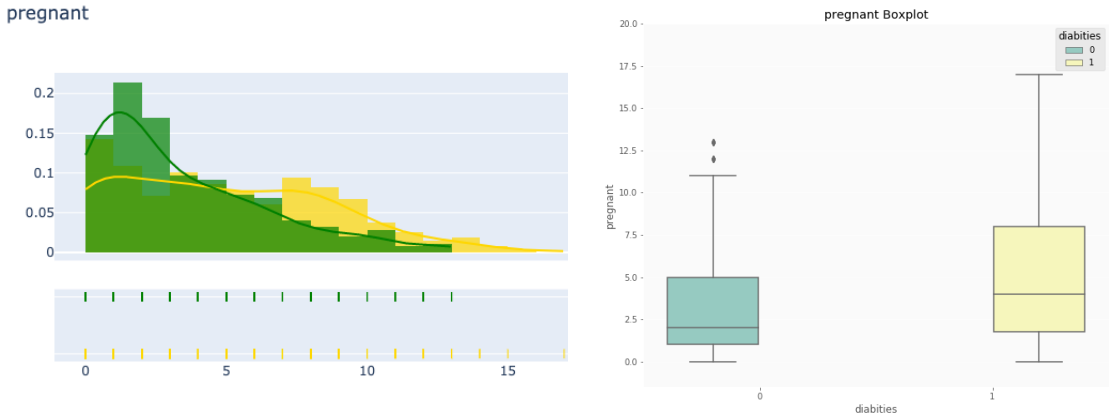


(a) plot distribution for pedigree (b) Boxplot for pedigree

Figure 2.13: Graphic overview for pedigree variable

2.4.3.8 Pregnant

Finally, the picture 2.14 is a comparison of a box plot(fig b) and the probability density function(fig a) for a healthy and diabetes person comparison to the pregnant variable. According to our frequency table 2.3.4, we can read all the information for boxplot. In the above fig (b) we can see two box plot diagrams. A box plot with yellow color on the right side is for diabetes person, which has "minimum"=0, first quartile (Q1)=1, median=3, third quartile (Q3)=6 and "maximum"=17.



(a) plot distribution for Age

(b) Boxplot for pregnant

Figure 2.14: Graphic overview for pregnant variable

2.4.4 Prepared Data

Data preparation creates higher quality data for analysis and other data management related tasks by eradicating errors and normalizing raw data before it is processed. It is important, but it takes a long time and may require certain skills. However, with an intelligent data preparation tool, the process is now faster and more accessible for a greater number of users.

Nr	Pregnancies	Glucose	BP	Thickness	Insulin	BMI	Pedigree	Age	Diabetes
0	6	148	72	35	169.5	33.6	0.627	50	1
1	1	85	66	29	102.5	26.6	0.351	31	0
2	8	183	64	32	169.5	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
5	5	116	74	27	102.5	25.6	0.201	30	0
6	3	78	50	32	88	31	0.248	26	1
7	10	115	70	27	102.5	35.3	0.134	29	0
8	2	197	70	45	543	30.5	0.158	53	1
9	8	125	96	32	169.5	34.3	0.232	54	1
10	4	110	92	27	102.5	37.6	0.191	30	0
11	10	168	74	32	169.5	38	0.537	34	1
12	10	139	80	27	102.5	27.1	1.441	57	0
13	1	189	60	23	846	30.1	0.398	59	1
14	5	166	72	19	175	25.8	0.587	51	1
15	7	100	74	32	169.5	30	0.484	32	1

Table 2.3: overview after cleaning the diabetes dataset

2.4.5 Correlation matrix

A correlation matrix is a table that shows the correlation coefficients between variables. Each cell in the table shows the correlation between the two variables. A correlation matrix is used to summarize data. Typically, a correlation matrix is "square", with the same variables shown in the rows and columns. I've shown the table as an example below. The line of 1.00s going from the top left to the bottom right is the main diagonal, which shows that each variable always perfectly correlates with itself. This matrix is symmetrical, with the same correlation shown above the main diagonal being a mirror image of those below the main diagonal [7].

	pregnant	glucose	BP	skin	insulin	BMI	pedigree	Age	Diabetes
pregnant	1	0.13	0.21	0.09	0.06	0.02	-0.03	0.54	0.22
glucose	0.13	1	0.22	0.23	0.49	0.24	0.14	0.27	0.50
bp	0.21	0.22	1	0.20	0.07	0.29	-0.00	0.33	0.17
skin	0.09	0.23	0.20	1	0.20	0.57	0.11	0.13	0.30
insulin	0.06	0.49	0.07	0.20	1	0.24	0.15	0.12	0.38
bmi	0.02	0.24	0.29	0.57	0.24	1	0.15	0.03	0.32
pedigree	-0.03	0.14	-0.00	0.11	0.15	0.15	1	0.03	0.17
age	0.54	0.27	0.33	0.13	0.12	0.03	0.03	1	0.24
diabetes	0.22	0.50	0.17	0.30	0.38	0.32	0.17	0.24	1

From the above table we can see that insulin(0.49), BMI(0.57), age(0.54) and glucose(0.50) are correlated with the target variable. we can also see in fig 2.6 as a graphical representation called a Heat map.

2.4.5.1 Heat Map

A heat map is a graphical representation of data in which the individual values contained in an array are shown as colors (see figure). Finally, I made a heat map to show how the features are related to each other or the target variable. Correlation can be positive (increase in one value of feature increases the value of the target variable) or negative (increase in one value of feature decreases the value of the target variable)

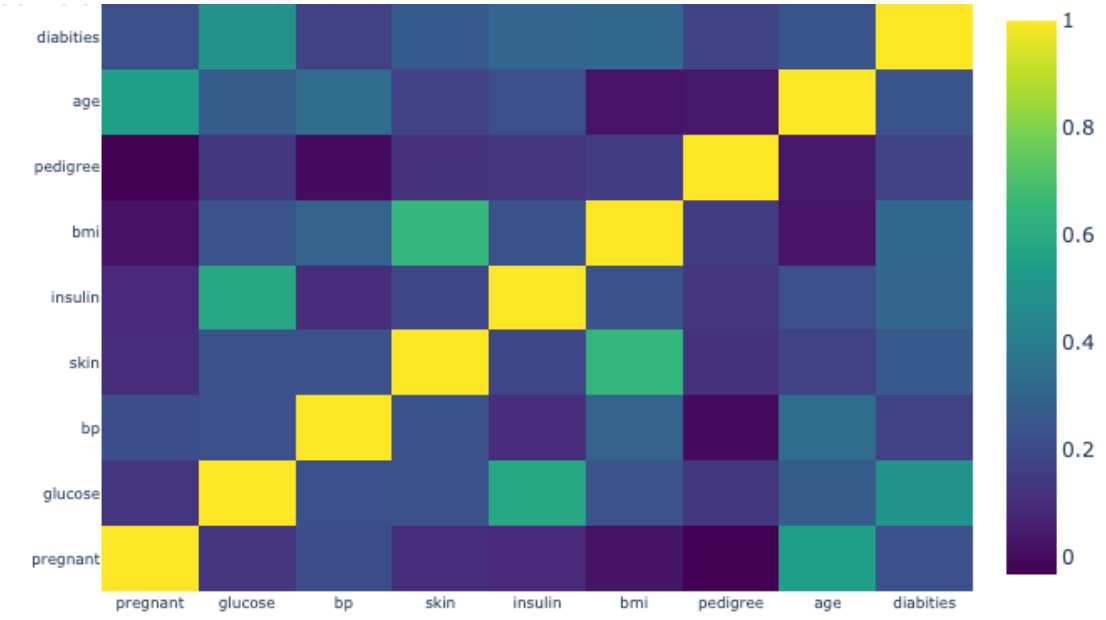


Figure 2.15: Heat Map

From the above heat map, we can see that non-diagonal is in yellow. It means that they are themself highly correlated to each other like age and age, BMI and BMI, and so on. Their correlated value is 1. Another way we can see that green color close to the non-diagonal i.e yellow colors. we can read from the figure that BMI and skin are correlated having value 0.56 that means close to yellow color. Insulin and glucose are also correlated having value 0.49 to each other. we can read the correlated values from above and under non-diagonal that gives the same value. On the right side, we can see that a horizontal line with a different color. It starts at 0 and ends to 1. we can also read the value from this line using color.

2.4.6 K-Fold Cross-Validation

Evaluating a machine learning model can be quite challenging. Generally, we divide the data set into training and test sets and use the training set to train the model and testing set to test the model. We then evaluate the performance of the model using an error metric to determine the accuracy of the model by using K-Fold Cross-Validation [19].

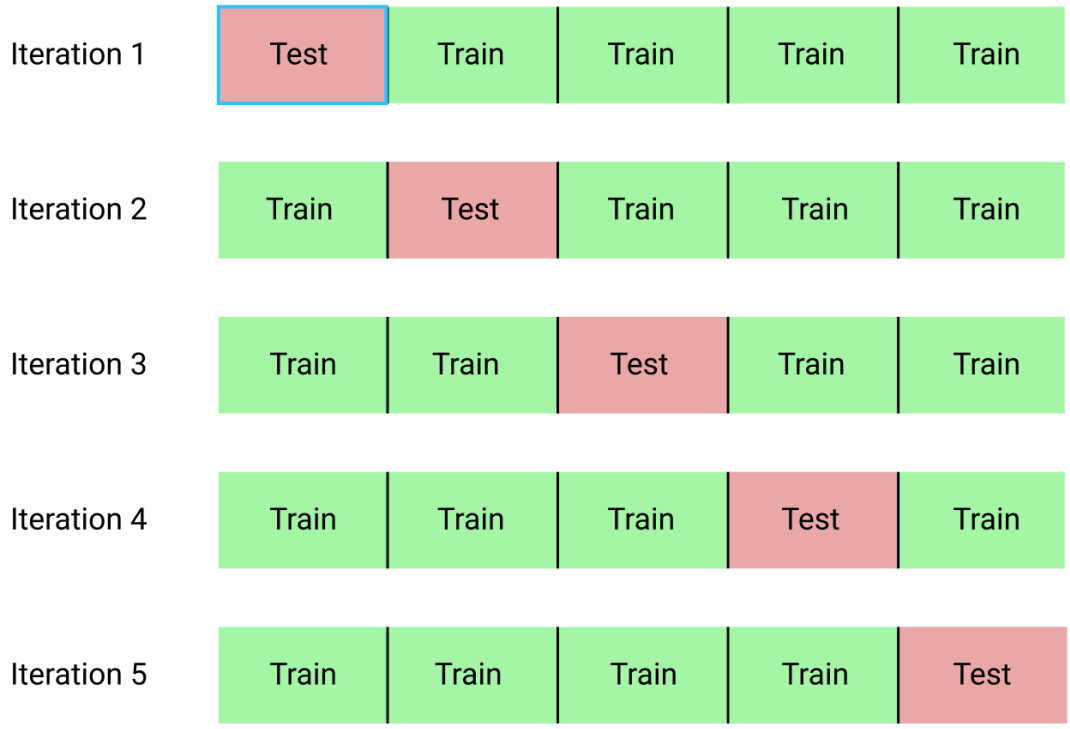


Figure 2.16: K-Fold Cross Validation [30]

The procedure has a single parameter called k that refers to the number of groups into which a particular sample of data is divided. As such, the procedure is often called k-fold cross-validation. When a specific value for k is chosen, it may be used in place of k in the reference to the model, such as k=10 becoming 10-fold cross-validation. Suppose in the above fig we divided the dataset into 5 fold. Then k=5, the process will run 5 times, each time with a different 5 test data set. Let's discuss how the accuracy of the model is evaluated by using the k-Fold Cross-Validation stepwise [30].

- Taking the group as a test data set
- Taking the remaining groups as a training data set
- Fitting a model on the training set and evaluate it on the test set
- Retaining the evaluation score and discard the model
- At the end of the above process, we will Summarize the skill of the model by using the sample of model evaluation scores.

3. Classification Trees Model

3.1 Decision Tree Classification

To understand Classification Let us assume some practical examples. Suppose a person working in a Bank as a marketing manager, he/she wish a collection of consumers who are most likely to buy a product. This can be how he/she will be able to save the marketing budget by finding our audience. As a loan manager, we would like to spot risky loan applications to attain a lower loan default rate. This process of classifying customers into a bunch of potential and non-potential customers or safe or risky loan applications is thought of as a classification problem. Classification could be a two-step process, learning step, and prediction step. within the learning step, the model is developed supported given training data. within the prediction step, the model is employed to predict the response for given data. Decision Tree is one of the easiest and popular classification algorithms to understand and interpret. It can be utilized for both classification and regression kind of problem [16].

3.2 Train/Test Split

Train and Test Split are those two quite important concepts in data science and data analysis. When we use a statistical and machine learning model, we usually split our data into two subsets: training data and testing data, and fit our model on the train data, to make predictions on the test data. [4].

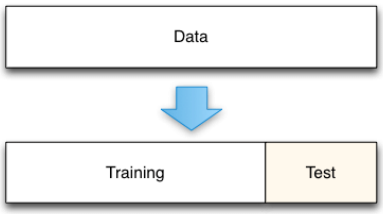


Figure 3.1: Train and Test Split

The training set contains a known output and the model learns on this data in order to be generalized to other data later on. We have the test data set (or subset) in order to test our model's prediction on this subset. In this above figure, the data set split by using the **train_test_split()** function. We need to pass three parameters features, target, and test_set size. The **test_size=0.3** inside the function indicates the percentage of the data that should be held over for testing. It's usually around 80% to 20% and 70% to 30%. I used 70% to 30% i.e 70% dataset for training and 30% dataset for testing.

3.3 Building Decision Tree Model in Scikit-learn

Scikit-learn is an open-source Python module integrating a wide range of state-of-the-art machine learning algorithms for medium-scale supervised and unsupervised problems [26]. It features various classification, regression and clustering algorithms including Support Vector Machine, Random Forest, and K-Neighbors, as well as numerical and scientific Python libraries such as NumPy and SciPy. In this Thesis, we will apply machine learning with the help of the scikit-learn library, which was created to make machine learning in Python easier and more robust. Now Let's create a Decision Tree model using Scikit-learn [24, 26].

3.3.1 Evaluating the Model

Model evaluation is an integral part of the model development process. It helps to find the best model that represents our data and how well the chosen model will work in the future. Let's estimate, how accurately the classifier or model can predict. Accuracy can be computed by comparing actual test set values and predicted values. A classification rate of **84.84%** was obtained from testing data, which can be considered good accuracy. The following table below is a confusion matrix to determine how many observations were correctly or incorrectly classified. It compares the observed and predicted outcome values and shows the number of correct and incorrect predictions.

n=231	Predicted: No	Predicted: Yes
Actual: No	126	20
Actual: Yes	15	70

Table 3.1: confusion matrix

The diagonal elements of the confusion matrix indicate correct predictions, while the off-diagonals represent incorrect predictions. So, the correct classification rate is the sum of the number on the diagonal divided by the sample size in the test data. that is (126 + 70)/231 = **84.84%** . Similarly, incorrect classification rate is (20 + 15)/231 = **15.16%** .The following classification table helps us to evaluate model in brief. For more detail we will explain in section 3.4.2.

	precision	recall	f1-score	support
0	0.893617	0.863014	0.878049	146.000000
1	0.777778	0.823529	0.800000	85.000000
accuracy	0.848485	0.848485	0.848485	0.848485
macro avg	0.835697	0.843272	0.839024	231.000000
weighted avg	0.850992	0.848485	0.849330	231.000000

Table 3.2: classification report

We can improve this accuracy by tuning the parameters in the Decision Tree Algorithm.

3.3.2 Visualizing Decision Trees

For the visualization of the Decision Tree Model, we can use Scikitlearn's **export_graph viz** function to display the tree within a Jupyter notebook. The following large decision tree is obtained by **"gini"** criterion.

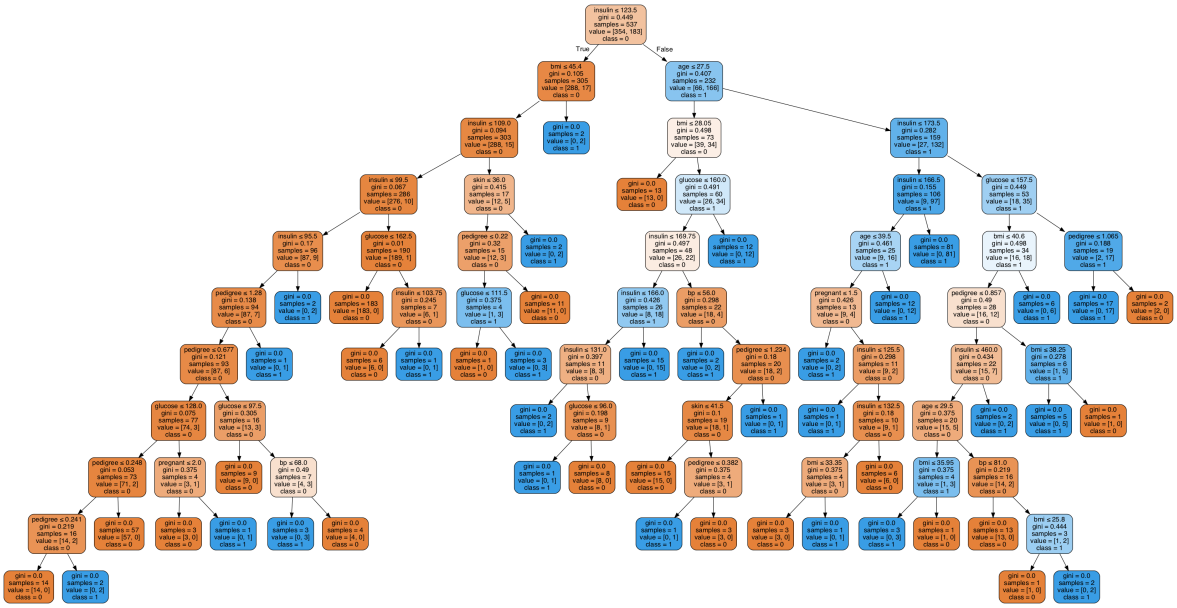


Figure 3.2: Visualizing Decision Trees

In the decision tree chart, each internal node has a decision rule that splits the data. we can say a node is pure when all of its records belong to the same class, such nodes known as the leaf node. Here, the resultant tree is unpruned. This unpruned tree is unexplainable and not easy to understand. In the next section, let's optimize it by pruning [24, 26].

3.4 Optimizing Decision Tree Performance

In Scikit-learn, optimization of decision tree classifier performed by only pre-pruning. The maximum depth of the tree can be used as a control variable for pre-pruning. In the following example, we can plot a decision tree on the same data with **max_depth=2**. Other than pre-pruning parameters, we can also try other attribute selection measure such as entropy. The following points are some parameter which helps to optimize Decision Tree performance.

- **criterion: optional ("gini", "entropy", default="gini")** or Choose attribute selection measure: This parameter allows us to use the different-different attribute selection measure. Supported criteria are "gini" for the Gini index and "entropy" for the information gain.

- **splitter: ("best", "random", default="best")** or Choose attribute selection measure: This parameter allows us to choose the split strategy. Supported strategies are "best" to choose the best split and "random" to choose the best random split.

- **max_depth:(int, default=None or Maximum Depth of a Tree)** or the maximum depth of the tree: If None, then nodes are expanded until all the leaves contain less than **min_samples_split** samples.The higher value of maximum depth causes overfitting, and a lower value causes under fitting [26].

3.4.1 Visualizing Decision Tree Model

Finally, we have developed our Decision Tree model by selection criterion **"entropy"**. This model is less complex, explainable, and easy to understand than the previous decision tree model plot [24]. Insulin is the first variable to separate the sample into two subgroups. Insulin

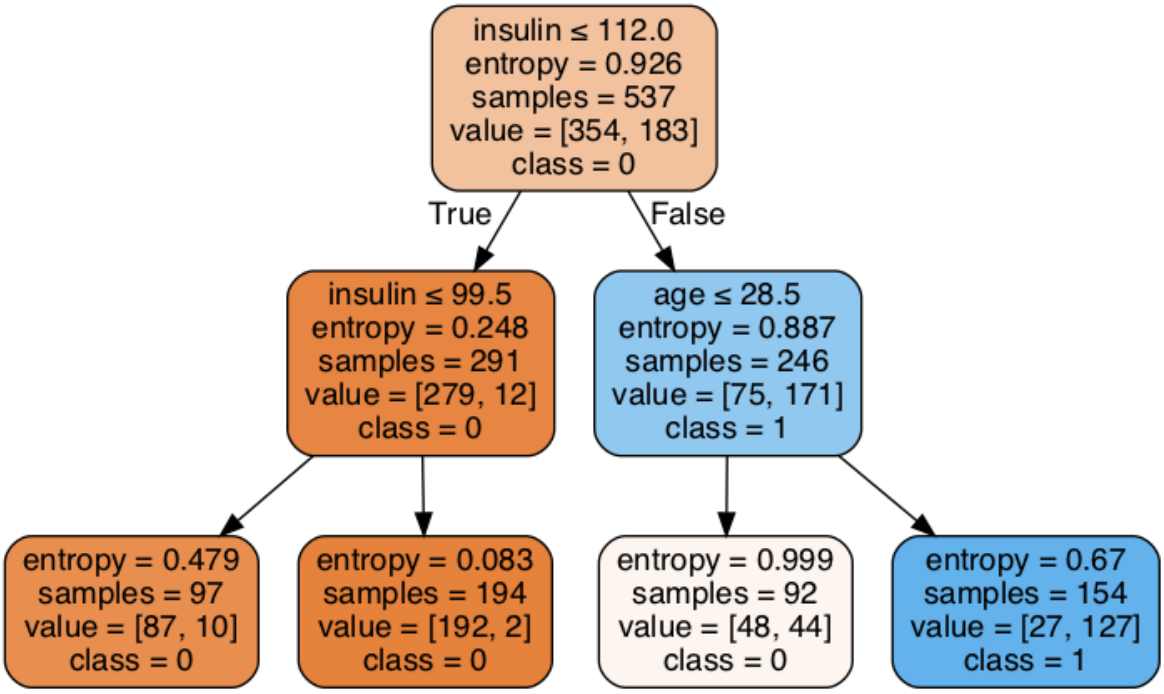


Figure 3.3: Final Decision Tree Model

greater than 112 is more likely to have diabetes compared to a healthy person. The interesting part is the second level subdivision variable i.e age. The patient age having more than 28.5 are more likely to have diabetes class. According to this model analysis, I believe patients having insulin less or equal to 112 are healthy.

3.4.2 Evaluating the final Model

A classification rate of **85.28%** was obtained from testing data, which can be considered good accuracy than the previous model. we can see in the following table.

n=231	Predicted: No	Predicted: Yes
Actual: No	136	10
Actual: Yes	24	61

Table 3.3: confusion matrix

The correct classification rate is $(136 + 61)/231 =$ **85.28%** and the incorrect classification rate is $(10 + 24)/231 =$ **14.72%**.
A Classification report is used to measure the quality of predictions from a classification algorithm. The report shows the main classification metrics precision, recall, f1-score, and support on a per-class basis. The metrics are calculated by using true and false positives, true and false negatives. Positive and negative in this case are generic names for the predicted classes. There are four ways to check if the predictions are right or wrong:

- **TP / True Positive:** when a case was positive and predicted positive
- **TN / True Negative:**when a case was negative and predicted negative

	precision	recall	f1-score	support
0	0.850000	0.931507	0.888889	146.000000
1	0.859155	0.717647	0.782051	85.000000
accuracy	0.852814	0.852814	0.852814	0.852814
macro avg	0.854577	0.824577	0.835470	231.000000
weighted avg	0.853369	0.852814	0.849576	231.000000

Table 3.4: classification report

- **FP / False Positive:**when a case was negative but predicted positive
- **FN / False Negative:** when a case was positive but predicted negative

n=231	Predicted: No	Predicted: Yes
Actual: No	**TN=136**	**FP=10**
Actula: Yes	**FN=24**	**TP=61**

Table 3.5: confusion matrix

More additional measures are required to evaluate a classifier.

- **Precision:** It is the number of positive predictions divided by the total number of positive class values predicted. It is also called the Positive Predictive Value. Precision for **1** is the number of True Positives divided by the number of True Positives and False Negative predicted.i.e precision for **1=TP / (TP+FP) =61/(61+10)=0.8591**. Also, precision for **0=TN / (TN+FN) =136/(136+24)=0.850**.
- **Recall:** Recall is the ability of a classifier to find all positive instances. It is the number of positive predictions divided by the total number of positive class actual valued. It is also called the Positive Predictive Value. Recall for **1** is the number of True Positives divided by the number of True Positives and False Negative of actual value.i.e Recall for **1=TP / (TP+FN) =61/(61+24)=0.7176**. Also, Recall for **0=TN / (TN+FN) =136/(136+10)=0.9315**.
- **F1 score:** F1 is a function of Precision and Recall. Looking at **Wikipedia**, In statistical analysis of binary classification, the F1 score (also F-score or F-measure) is a measure of a test's accuracy. It considers both the precision and the recall of the test to compute the score. The formula is as follows:

  $$\textbf{F1 score}=2\frac{\textbf{precision.Recall}}{\textbf{precision+Recall}}$$

  F1 score for 1 $=2\frac{0.859*0.7176}{0.859+0.7176}=$**0.7820**

  F1 score for 0 $=2\frac{0.85*0.9315}{0.85+0.9315}=$**0.888**
- **Support:** The support is the number of samples of the true response that lie in that class. The support number of samples that lie in that class 0 and 1 are 146 and 85 respectively.
- **Macro avg:** The second last line gives a macro average of precision, recall and f1-score. so for precision the macro avg is $(0.8500 + 0.8591)/2 =$ **0.8545**.
- **Weighted avg:** The last line gives a weighted average of precision, recall, and f1-score where the weights are the support values. so for precision the avg is $(0.850 * 146 + 0.8591 * 85)/231 =$ **0.8533**. The total is just for total support which is 231 here.

3.5 AUC-ROC Analysis

The ROC curve that stands for Receiver Operating Characteristic. This curve is a plot of the true positive rate against the false-positive rate. It shows the trade-off between sensitivity and specificity. It is a curve that is used to assess the accuracy of a continuous measurement for predicting a binary outcome. It generally shows the performance of a classification model at all classification thresholds.

Figure 3.4: ROC curve

Finally, we demonstrate how ROC curves can be plotted using Python. However this ROC curve is only a point i.e (x,y)=(FPR,TPR), where FPR - false positive rate and TPR - true positive rate. The true positive rate is calculated as the number of **TP** divided by the sum of the number of **TP** and the number of **FN**. Also, false positive rate is calculated as the number of **FP** divided by the sum of the number of **FP** and the number of **TN**. It's started with point(0,0) i.e (FPR,TPR) and ends with point(1,1). Similarly, each False Positive(FPR) on x-axis and True positive(TPR) forms a green curve. The diagonal line is the baseline, that can be obtained with a random classifier. The area covered by the curve is the area between the orange line (ROC) and the axis. This area covered is AUC. The bigger the area covered, the better the machine learning models are at distinguishing the given classes. The AUC value lies between 0.5 to 1 where 0.5 denotes a bad classifier and 1 denotes an excellent classifier or ideal. The score we got for decision tree from **roc_auc_score** is **89%** which is considered excellent for this model [5].

4. Random Forest Model

Random forest is a supervised learning algorithm. The "forest" that is built an ensemble of decision trees, which are mostly trained using the “bagging” method. The general idea of the filling method is that a combination of learning models increases the overall result.it is a flexible, easy-to-use machine learning algorithm that usually delivers excellent results even without hyper-parameter tuning. It is also one of the most commonly used algorithms because it is simple and can be used for classification and regression tasks. In this section, I will explain how the random forest algorithm works [18].

4.1 Random Forest Classification

The random forest algorithm creates and merges multiple decision trees to get a more accurate and stable prediction. One of the great advantages of a random forest is that it can be used for both classification and regression problems that the majority of current machine tools Form learning systems. I’m going to talk about random overall structures in classification because classification is sometimes seen as a building block for machine learning. Here’s what a random forest with two trees would look like:

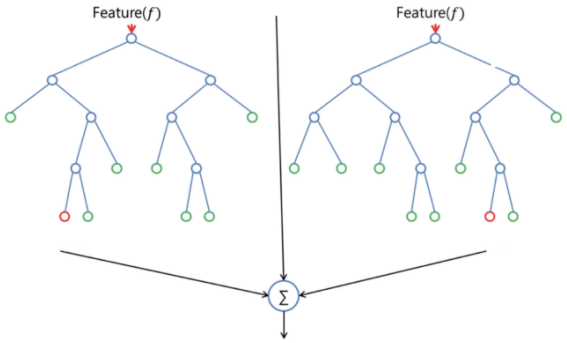


Figure 4.1: Random Forest [18]

Random forest has almost the same hyperparameters as a decision tree or a classifier for filling. As already mentioned, we can also use Random forest to perform regression tasks using random forest regression. Random forest adds randomness to the model as the trees grow. Instead of looking for the most important feature while splitting a node, it looks for the best feature among a random subset of features. This leads to a great variety, which generally leads to a better model. Therefore, in random forest only a random subset of the features is taken into account by the node splitting algorithm. we can even make trees more random by using random thresholds for each feature instead of looking for the best possible thresholds (like a normal decision tree) [18, 32].

4.2 Classification and Regression

Logistic Regression (LR) and Decision Tree (DT) both fit a model to the data, and both can be interpreted easily; however, both have pros and cons. Based on the nature of our data choose the appropriate algorithm. Of course, at the initial level, we apply both algorithms. Then, we choose which model gives the best result. But have we ever thought of why a particular model is performing best on your data? Classification and regression are two major prediction problems which are usually dealt with data mining and machine learning. The main difference between them is that the output variable in the regression is numerical (or continuous) while that for classification is categorical (or discrete).

4.3 Advantage and Disadvantage

As previously mentioned, the random forest has the advantage that it can be used for both regression and classification tasks and that the relative importance assigned to the input functions is easy to see. A random Forest is also considered a very practical and easy-to-use algorithm because its standard hyperparameters often give a good prediction result [12]. The number of hyperparameters is also not that high and they are easy to understand. One of the biggest problems with machine learning is overfitting, but most of the time, this won't happen so easily to a random structure. This is because if there are enough trees in the forest, the classifier will not overfit the model. The main limitation of random forests is that a large number of trees can make the real-time prediction algorithm slow and ineffective. In general, these algorithms are quick to train, but they are slow to generate predictions once they are trained. A more accurate prediction requires more trees, which leads to a slower model. The random forest algorithm is fast enough in most real-world applications, but there may be situations where run time performance is important and other approaches are preferred. And of course, Random Forest is a modeling prediction tool and not a descriptive tool.34. That is, if we are looking for a description of the relationships in your data, other approaches are preferred [12, 18]. One important disadvantage is that the easy to interpret tree diagrams are no longer available.

4.4 Its important uses

The random forest algorithm is used in many different areas, for example in the areas of banking, stock exchange, medicine, and e-commerce. In banking, for example, this is used to identify customers who use the bank's services more often than others and who repay their debts on time. In this domain, it is also used to identify fraud customers who want to defraud the bank. In finance, it is used to determine the behavior of stock in the future. In the healthcare field, it is used to identify the correct combination of ingredients in medicine and to analyze a patient's medical history to identify diseases. And finally, e-commerce uses the random forest to determine whether a customer likes the product or not.

4.5 Building a Random Forest Model in Scikit-learn

A random forest is a meta estimator that fits several decision trees classifiers on various sub-samples of the data set and use averaging to improve predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement if **'bootstrap=True' (default)** [10]. There are some parameter for **sklearn.ensemble.RandomForestClassifier** which also helps to optimize our Model [3].

- **n_estimators integer, optional, (default=100):** The number of trees in the forest. The default value of n_estimators changed from 10 to 100.
- **criterion:string, optional, (default="gini"):** The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain. Note: this parameter is tree-specific.
- **max_depth:integer or None, optional, (default=None):** The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.
- **min_samples_leaf:int, float, optional (default=1):** The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least min_samples_leaf training samples in each of the left and right branches. This may have the effect of smoothing the model, especially in regression.

The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples. The default values for the parameters controlling the size of the trees (e.g. **max_depth**, **min_samples_leaf**, etc.) lead to fully grown and unpruned trees which can potentially be very large on some data sets. To reduce memory consumption, the complexity and size of the trees should be controlled by setting those parameter values. The features are always randomly permuted at each split. Therefore, the best-found split may vary, even with the same training data, **max_features=n_features** and **bootstrap=False**, if the improvement of the criterion is identical for several splits enumerated during the search of the best split. To obtain a deterministic behavior during fitting, **random_state** has to be fixed [3].
The random forest algorithm is to train early in the model development process to see how it works and it is difficult to create a "bad" random forest because it is so simple. This algorithm is also a good choice if we need to develop a model in a short time. It also offers a pretty good indicator of the importance it assigns to our functions. Of course, we can probably always find a better performing model, like a neural network, but this development usually takes much longer. Also, they can deal with many different feature types such as binary, categorical, and numerical. Overall, Random Forest is a (mostly) quick, easy, and flexible tool, though it has its limitations [35]. Finally, we create a Random forest model which gives 90.48% which a better accuracy score compare to the decision tree model.

4.6 Model Evaluation

A confusion matrix helps us to evaluate the performance of a classification model [24]. Diagonal values represent accurate predictions, while non-diagonal elements are inaccurate predictions. In the output, **140** and **69** are correct predictions, and **6** and **16** are incorrect predictions. Well, we got a classification rate of **90.48%** using the test data , considered as good accuracy(see in Appendix section).

	precision	recall	f1-score	support
0	0.90	0.96	0.93	146
1	0.92	0.81	0.86	85
accuracy	0.90	0.90	0.90	0.90
macro avg	0.91	0.89	0.89	231
weighted avg	0.91	0.90	0.90	231

Table 4.1: classification report

4.7 AUC-ROC Analysis

The following ROC curve is a plot between sensitivity and specificity. It is a curve that is used to assess the accuracy of a continuous measurement for predicting a binary outcome. It generally shows the performance of a classification model at all classification thresholds.

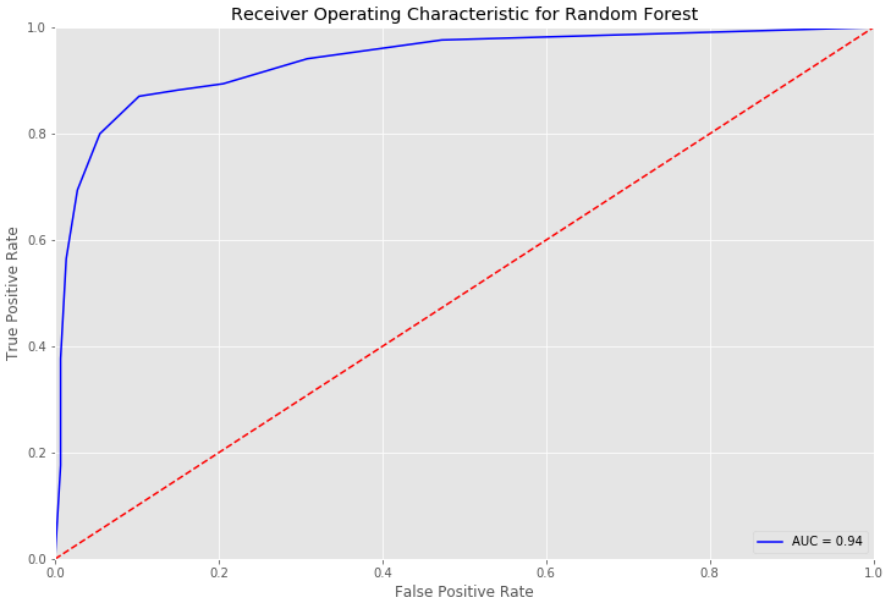


Figure 4.2: ROC curve

Finally, we demonstrate how ROC curves for random forest can be plotted using Python. Its looks not so smooth as decision tree. The area covered by the blue curve is more than decision tree. The bigger the area covered, the better the machine learning models are at distinguishing the given classes. The AUC value lies between 0.5 to 1 where 0.5 denotes a bad classifier and 1 denotes an excellent classifier or ideal. The score we got from **roc_auc_score** is **94%** which is considered excellent model than decision tree.

5. Logistic Regression Model

As part of this bachelor thesis, Logistic regression models were created for the target variable (0 and 1). These target is a binary variable. Logistic regression analysis (binary) is used to verify whether there is a connection between a dependent binary variable and one or more independent variables. Unlike the simple regression analysis and the multiple regression analysis, the dependent variable is binary. That means it has only two forms that is 0('No', ' false') and 1('yes','True'). Binary logistic regression analysis examines the relationship between the probability of the dependent variable taking the value 1 and the independent variables. This means that the value of the dependent variable is not predicted, but the probability that the dependent variable assumes the value 1. The question of logistic regression analysis is often shortened as follows: "Do independent variables influence the probability that the dependent variable takes the value 1? How strong is its influence?"

5.1 Model Assumptions

The objective of logistic regression analysis is to model and analyze the conditional probability as a function of the variables.

$$\pi_i = P(y_i = 1) = P(y_i = 1|x_{i1}, ..., x_{ip}) = E(Y_i = 1|x_{i1}, ..., x_{ip}) \tag{5.1}$$

The model can be represented in the following forms:

$$\pi_i = \frac{\exp(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}}{1 + \exp(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}} \tag{5.2}$$

$$\log(\frac{\pi_i}{1 - \pi_i}) = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} \tag{5.3}$$

$$\frac{\pi_i}{1 - \pi_i} = \log(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}) \tag{5.4}$$

The conditional expected value $E(y_i|x_i)$ in (5.2) is modeled by π_i. The formation from logistic equation 5.2 gives the forms in equations (5.3) and (5.4), where $\frac{\pi}{1-\pi}$ is odd Ratio i.e the possibilities ("probabilities") and $\log(\frac{\pi_i}{1-\pi_i})$ is called the log-odds or logit.

5.2 Parameter estimation

The parameters in the logistic regression model are estimated using the maximum likelihood method. The basic intuition behind using maximum likelihood to fit a logistic regression model is as follows: we seek estimates for β_0 and β_1 such that the predicted probability $p(\hat{x_i})$ of default for each individual, using (5.2), corresponds as closely as possible to the individual's observed default status. In other words, we try to find β_0 and β_1 such that plugging these estimates into the model for p(X), given in (5.2), yields a number close to one for all individuals

who defaulted, and a number close to zero for all individuals who did not. This intuition can be formalized using a mathematical equation called a likelihood function: [15]:

$$l(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i':y'_i=0} (1 - p(x'_i)) \tag{5.5}$$

The principle of maximum likelihood states: Choose the parameter for the realizations as an estimate of the parameter for which the likelihood is maximum. The maximum likelihood estimates can be determined by obtaining the partial derivatives setting them equal to zero and and solve the resulting system of equations according to the vector parameter β. By setting the scoring function to zero, this results in Maximum Likelihood(ML) equation, which is usually solved iterative using Fisher's scoring algorithm. This method provides the maximum likelihood of estimator $\hat{\beta}$ for β.

5.3 Logit Model in Scikit-learn

Logistic regression is one of the simplest and most commonly used machine learning algorithms for classifying into two classes. Logistic regression describes and estimates the connection between a dependent binary variable and independent variables. Let's build the diabetes prediction model using scikit-learn Module from python code which i had attached through this Thesis in Appendix section. we are going to predict diabetes using logistic regression classifier. We will start with the basic logistic regression model with only one covariate, 'age', predicting 'diabetes'.The lines of code below fits the univariate logistic regression model and prints the result summary.

Dep. Variable:	diabetes	**No. Observations:**	768
Model:	GLM	**Df Residuals:**	766
Model Family:	Binomial	**Df Model:**	1
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-475.36
Date:	Wed, 18 Mar 2020	**Deviance:**	950.72
Time:	16:22:14	**Pearson chi2:**	761.
No. Iterations:	4		

Table 5.1: GLM with single variable

	coef	**std err**	**z**	**P> |z|**	**[0.025**	**0.975]**
Intercept	-2.0475	0.239	-8.572	0.000	-2.516	-1.579
age	0.0420	0.007	6.380	0.000	0.029	0.055

Table 5.2: Results summary with single variable

Using the **P>|z|** result above, we can conclude that the variable 'age' is an important predictor for 'diabetes', as the value is less than 0.05. Now, similarly we can also fit all variables in a logistic regression model. Below we fit a logistic regression for 'diabetes' using all the variables.

Dep. Variable:	diabetes	**No. Observations:**	768
Model:	GLM	**Df Residuals:**	759
Model Family:	Binomial	**Df Model:**	8
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-344.33
Date:	Sun, 29 Mar 2020	**Deviance:**	688.67
Time:	20:59:55	**Pearson chi2:**	1.59e+03
No. Iterations:	5		

Table 5.3: GLM with all variables

	coef	**std err**	**z**	**P> |z|**	**[0.025**	**0.975]**
Intercept	-9.2850	0.832	-11.162	0.000	-10.915	-7.655
pregnant	0.1174	0.032	3.651	0.000	0.054	0.180
glucose	0.0304	0.004	7.827	0.000	0.023	0.038
skin	0.0378	0.014	2.736	0.006	0.011	0.065
bp	-0.0047	0.009	-0.531	0.595	-0.022	0.013
insulin	0.0054	0.001	3.606	0.000	0.002	0.008
bmi	0.0591	0.018	3.287	0.001	0.024	0.094
pedigree	0.7873	0.301	2.617	0.009	0.198	1.377
age	0.0131	0.010	1.355	0.175	-0.006	0.032

Table 5.4: Generalized Linear Model Regression Results with multivariables

The above output shows that adding all variables to the model leads to a big shift in the age parameter (its p-value increased to over the significance level of 0.05). This can happen in statistical models while adding or removing other variables from a model. Looking at the p-values, the variables **'age'** and **'bp'** seem to be insignificant predictors. All the other variables have their p-values smaller than 0.05 and therefore, they are significant.

Now let's remove some variable which is not significant i.e we remove age and bp. Finally we create our predictive Model to find the probability of patient having diabetes. Below we fit a logistic regression for 'diabetes' using only significant variables.

Dep. Variable:	diabetes	**No. Observations:**	768
Model:	GLM	**Df Residuals:**	761
Model Family:	Binomial	**Df Model:**	6
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-345.26
Date:	Sun, 29 Mar 2020	**Deviance:**	690.52
Time:	21:10:53	**Pearson chi2:**	1.65e+03
No. Iterations:	5		

Table 5.5: GLM predictive modell

	coef	std err	z	P> |z|	[0.025	0.975]
Intercept	-9.2211	0.724	-12.742	0.000	-10.639	-7.803
pregnant	0.1374	0.028	4.956	0.000	0.083	0.192
glucose	0.0311	0.004	8.247	0.000	0.024	0.039
skin	0.0383	0.014	2.787	0.005	0.011	0.065
insulin	0.0053	0.001	3.590	0.000	0.002	0.008
bmi	0.0550	0.017	3.160	0.002	0.021	0.089
pedigree	0.8062	0.299	2.694	0.007	0.220	1.393

Table 5.6: summary of predictive model

From above summary of logistic regression now we can predict weather the patient have diabetes or not.The following Model helps us to predict the diabetes.

$$\pi_i = \frac{\exp(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4)}{1 + \exp(\exp(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4)}$$

$$E(Y_i|x_i) = \frac{\exp(-9.2211+0.1374*pregnant+0.0311*glucose+0.0383*skin+0.0053*insulin+0.0550*bmi+0.8062*pedigree)}{1+\exp(-9.2211+0.1374*pregnant+0.0311*glucose+0.0383*skin+0.0053*insulin+0.0550*bmi+0.8062*pedigree)}$$

Nr	pregnant	glucose	skin	insulin	bmi	pedigree	diabetes	$E(Y_i\|x_i)$
0	6	148.0	35.0	169.5	33.6	0.627	1	0.69
1	1	85.0	29.0	102.5	26.6	0.351	0	0.62
2	8	183.0	32.0	169.5	23.3	0.672	1	0.82
3	1	89.0	23.0	94.0	28.1	0.167	0	0.03
4	0	137.0	35.0	168.0	43.1	2.288	1	0.81

Table 5.7: predective Diabities

Let's predict from above data set table.we will predict Nr 0 and 3.According to our predictive model we got the following results.

$$E(Y_0|x_0) = \frac{\exp(-9.2211+0.1374*6+0.0311*148+0.0383*35+0.0053*169.5+0.0550*33.6+0.8062*0.627)}{1+\exp(-9.2211+0.1374*6+0.0311*148+0.0383*35+0.0053*169.5+0.0550*33.6+0.8062*0.627)} = \mathbf{0.69}$$

$$E(Y_3|x_3) = \frac{\exp(-9.2211+0.1374*1+0.0311*89+0.0383*23+0.0053*94+0.0550*28.1+0.8062*0.167)}{1+\exp(-9.2211+0.1374*1+0.0311*89+0.0383*23+0.0053*94+0.0550*28.1+0.8062*0.167)} = \mathbf{0.03}$$

5.4 Interpretation of the Odds,odds- Ratio,log-odds-Ratio

Let's begin with probability. Probabilities range between 0 and 1.we can consider the total number of patients which have diabetes or not is 768. The total number of patient with diabetes is 268. Let's say that the probability of patient having diabetes is 0.35, thus $\mathbf{p = 0.35}$,Then the probability of having not diabetes is $\mathbf{q = 1 - p = 0.65}$
Odds are determined from probabilities and range between 0 and infinity. The following equation helps us to compute an odds and odds ratio.

$$\mathbf{Odds(A)} = \frac{\mathbf{P(A)}}{\mathbf{1\text{-}P(A)}} \tag{5.6}$$

Odds are defined as the ratio of the probability of diabetes and the probability of not having diabetes. The odds of diabetes are: $\mathbf{odds(diabetes) = p/(1\text{-}p)}$ **or** $\mathbf{p/q = 0.35/0.65 = 0.53}$ that is, the odds of patient having not diabetes are 768 to 500. The odds of patient having not diabetes are: $\mathbf{odds(healthy) = q/p = 0.65/0.35 = 1.85}$ The odds of diabetes

and the odds of healthy patient are just reciprocals of one another, i.e., 0.35/0.65 = 0.53 and 0.65/0.35 = 1.85. The formula to compute its odd Ratio is given by:

$$\mathbf{OR(A,B)} = \frac{\mathbf{Odds(A)}}{\mathbf{Odds(B)}} \tag{5.7}$$

From equation 5.7 we can calculate the Odds Ratio for diabetes with respect to healthy person which is **0.28**.

$$\mathbf{OR(diabetes,healthy)} = \frac{\mathbf{Odds(diabetes)}}{\mathbf{Odds(healthy)}} = \frac{\mathbf{0.53}}{\mathbf{1.85}} = \mathbf{0.28}$$

Thus, the odds of patient having diabetes are 0.28 times as large as the odds for a healthy patient. Odds ratios are fairly easy to interpret. If the odds ratio is greater than 1, we can assume that there is an association between characteristic A and characteristic B in such a way that the presence of characteristic A increases the probability of the presence of characteristic B. Similarly, if the odds ratio is less than 1, the presence of feature A lowers the probability of the presence of feature B. [13].
There is a direct relationship between the coefficients generated by logit and the odds ratios produced by logistics.A logit is defined as the log base e log of the odds. The range is negative infinity to positive infinity.

$$\mathbf{logit(P)} = \mathbf{log(odds)} = \mathbf{log(p/q)}$$

The following table show that odds-Ratio for each term.

Term	odds Ratio
pregnant	1.147
glucose	1.031
skin	1.039
insulin	1.005
bmi	1.056
pedigree	2.239

Table 5.8: odds Ratio

From table 5.6, the intercept= -9.221 which corresponds to the log odds for base line i.e β_0. The coefficient for pregnant= 0.1374 which corresponds to the log of odds ratio between pregnant and baseline. The odds ratio can be calculated by exponentiation of the coefficient values i.e e^{β}.

5.5 Model Evaluation

A confusion matrix helps us to evaluate the performance of a classification model [24]. we can also visualize the performance of an algorithm. The fundamental of a confusion matrix is the number of correct and incorrect predictions are summed up class-wise. Here, we can see the confusion matrix in the form of the array object. The dimension of this matrix is 2*2 because this model is a binary classification. we have two classes 0 and 1. Diagonal values represent accurate predictions, while non-diagonal elements are inaccurate predictions. In the output, **136** and **52** are correct predictions, and **10** and **33** are incorrect predictions. Well, we got a classification rate of **81.39%** using the test data , considered as good accuracy.

n=231	Predicted: No	Predicted: Yes
Actual: No	**TN=136**	**FP=10**
Actula: Yes	**FN=33**	**TP=52**

Table 5.9: confusion matrix

	precision	recall	f1-score	support
0	0.804734	0.931507	0.863492	146.000000
1	0.838710	0.611765	0.707483	85.000000
accuracy	0.813853	0.813853	0.813853	0.813853
macro avg	0.821722	0.771636	0.785488	231.000000
weighted avg	0.817236	0.813853	0.806086	231.000000

Table 5.10: classification report

5.6 AUC-ROC Analysis

Interpreting the ROC curve is easier than another curve. The following ROC curve is for logistic regression which is in orange color. It gives more accuracy when we want our classifiers to be placed as close to the top left corner as possible. This indicates a low false-positive rate (high specificity) and a high true positive rate (high sensitivity) points out.

Figure 5.1: ROC curve

The area covered by this curve is the area between the orange line (ROC) and the axis. This area covered by this curve is smoother than the other two models. We can Analyse that this curve covered less are comparatively to decision tree and random forest model ROC curve. .The score we got from **roc_auc_score** is **90%** which is considered excellent for this model.

6. Comparison between the Models

6.1 Decision Tree Vs Random Forest

In this section of my thesis, I would like to explain the difference between random forests and decision trees and what are the advantages of random forests compared to decision trees. Both a decision tree and a random forest are similar ways to build a machine learning model. A choice tree is created during a whole data set using all the variables of interest, while a random forest randomly selects observations/rows and certain features/variables to form multiple decision trees so averaging the results. After a large number of trees have been created using this method, each tree "votes" or selects the class, and the class that receives the most simple majority votes is the "winner" or the predicted class. Of course, there are some more detailed differences, but this is the main conceptual difference. If a decision tree model is used for a specific training data set, the accuracy improves with more and more divisions. we can easily overstep the data and not know when you crossed the line unless you use **cross – validation**. The advantage of a simple decision tree is that the model is easy to interpret. we know which variable and what value of this variable is used to divide the data and predict the result. It is a forest that we can build and control. we can specify the number of trees we want to display in our forest (**n_estimators**), and we can also specify the maximum number of features we want to use on each tree. Accuracy increases with the number of trees but becomes constant at a certain point. In contrast to the decision tree, a strongly biased model is not created and the variance is reduced [25].

When we use decision tree [25]:

- When we want our predictive model to be simple and explainable then a decision tree is easy to use it.
- When we want non parametric model.
- When we don't want to worry about feature selection.

When we use random forest [25]:

- When you don't bother much about interpreting the model but want better accuracy.
- Random forest will reduce variance part of error rather than the bias part, so on a given training data set decision tree may be more accurate than random forest. But on an unexpected validation data set, Random forest always wins in terms of accuracy.

6.2 Decision Tree Vs Logistic Regression

Logistic regression and decision trees both model solve the classification Problems and both can be interpreted easily. However, both have advantages and disadvantages. Based on the nature of our data we always choose the appropriate algorithm. Then, we choose which model gives the best result. But have we ever thought of why a particular model is performing best

on our data? Categorical data works well with Decision Trees, while continuous data work well with logistic regression. If our data is categorical, then Logistic Regression cannot handle pure categorical data (string format). Rather, we need to convert it into numerical data. Logistic regression does not handle missing values; we need to impute those values by mean, mode, and median. If there are many missing values, then imputing those may not be a good idea, since we are changing the distribution of data by imputing mean everywhere. Decision trees work with missing values. If a tree is made up of a large number of nodes, it may take considerable mental effort to capture all the divisions that lead to a particular prediction.while logistic regression model is simply a list of coefficients [21, 25].

Decision Trees and Logistic Regression algorithms use different logic to create decision limits for the classification. Depending on how the data is distributed, one of these algorithms provides better classification results than the other. Logistic regression fits a single line to divide the space into two and perform better than a decision tree if the data is distributed in such a way that it can be linearly classified. Although a single linear limit can sometimes be a limiting factor for logistic regression. If the data is distributed non-linearly or for higher-dimensional data, the decision tree algorithm proposes a logistic regression because it can better divide the room into smaller subspaces. However, there is some point that helps which is better Model [21]:

- Logistic Regression performs faster.
- It can be turned into an online algorithm.
- Logistic Regression is less complex.
- Logistic Regression is easier to identify.

Use logistic regression over decision trees, when:

- Their performance is equal (Occam's Razor).
- weigh up to the increased complexity of the implementation of the model.
- The user/owner of the model demands a logical and easy to follow explanation for its predictions (regulatory institutions and some project managers).

6.3 The Best Model

Selecting the perfect machine learning model is a great challenge. Each model will have different performance characteristics. We develop multiple models and now we have to decide which model performs well by comparing to the other model. We have to pick the best in both competitive and real-world applications. Using resampling methods like cross-validation, we can get an estimate for how accurate each model may be on unseen data. We need to be able to use these estimates to choose one or two best models from the suite of models that we have created.

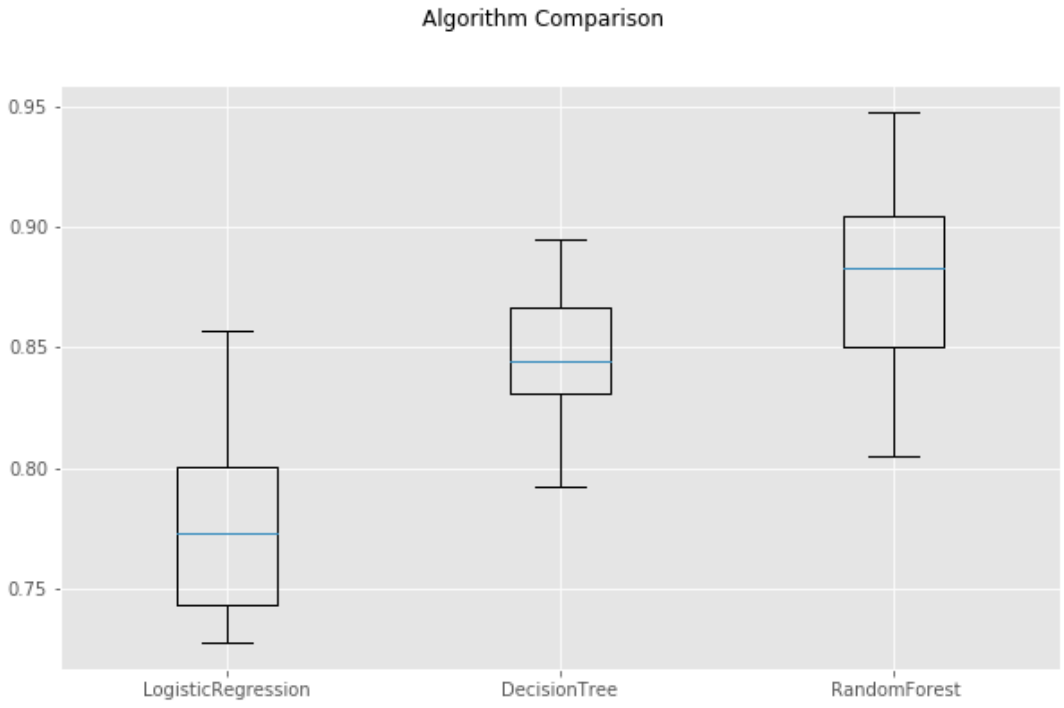


Figure 6.1: Comparison between the Model

In this above fig we compare three different algorithms. The 10-fold cross-validation procedure is used to evaluate each algorithm, importantly configured with the same random seed to ensure that the same splits to the whole data are performed and that each algorithm is evaluated in precisely the same way. From these results, it would suggest that a random forest analysis is perhaps worthy of further study on this problem. Each algorithm is given a name, useful for summarizing results afterward. The following table shows the mean accuracy and the standard deviation accuracy for each Model using whole data.

Model	Accuracy	Std
Logistics Regression	0.778640	0.04128
Decision Tree	0.847710	0.027020
Random Forest	0.868558	0.031300

Table 6.1: Accuracy Table

7. Conclusion

Machine learning has the great ability to revolutionize diabetes risk prediction using advanced calculation methods and the availability of a large amount of epidemiological and genetic data on diabetes. Diabetes is one of the deadliest diseases in the world. It is not only a disease but also a creator of different kinds of diseases like heart attack, blindness, kidney diseases, etc [36]. Early detection of diabetes is the key to treatment. This work has described a machine learning approach to predict diabetes levels. The technique can also help researchers develop an accurate and effective tool that reaches the medical table to help them make a better decision about the condition of the disease. Diabetes is a disease that can cause many complications. It is worth studying how this disease can be accurately predicted and diagnosed using machine learning.

Based on my different machine learning model, we have found that the accuracy of using a Decision tree is around 85% and the results of using the Random forest give better results than Decision tree and logistic regression. The result, using only insulin and Age worked better, especially in the Decision tree and also fitting a random forest to just insulin and age. This means that insulin and Age are the most important index for prediction, but only the use of both features cannot achieve the best result. So if we want to accurately predict, we need more features. If we compare the results of three classifications, we can also see that there is not a great difference between random forests, decision trees, and logistic regression, but random forests are better than the other classifiers in some methods. The best result for the classification problem is 0.89. This may indicate that machine learning can be used to predict diabetes. However, it is very important to find suitable attributes, classifiers, and data mining methods. Based on the data, we cannot predict the type of diabetes. Therefore, in the future, we would like to predict the type of diabetes and examine the proportion of each indicator, which may improve the accuracy of the prediction of diabetes.

Bibliography

Akm Ashiquzzaman, Abdul Kawsar Tushar, Md Rashedul Islam, Dongkoo Shon, Kichang Im, Jeong-Ho Park, Dong-Sun Lim, and Jongmyon Kim. Reduction of overfitting in diabetes prediction using deep learning neural network. In *IT Convergence and Security 2017*, pages 35–43. Springer, 2018.

Diabetes Atlas. International diabetes federation. *IDF Diabetes Atlas, 7th edn. Brussels, Belgium: International Diabetes Federation*, 2015.

Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

Adi Bronshtein. Train/test split and cross validation in python. *Understanding Machine Learning*, 2017.

J Brownlee. How and when to use roc curves and precision-recall curves for classification in python. *Machine Learning Mastery*, 2018.

Erol Cerasi, Suad Efendić, and Rolf Luft. Dose-response relation between plasma-insulin and blood-glucose levels during oral glucose loads in prediabetic and diabetic subjects. *The Lancet*, 301(7807):794–797, 1973.

Robert Cudeck. Analysis of correlation matrices using covariance structure models. *Psychological Bulletin*, 105(2):317, 1989.

Jianhua Dai, Wentao Wang, Haowei Tian, and Liang Liu. Attribute selection based on a new conditional entropy for incomplete decision systems. *Knowledge-Based Systems*, 39:207–213, 2013.

Taner Damci, Zeynep Oşar, Sule Beyhan, Hasan Ilkova, Mücahit Ozyazar, Ugur Gorpe, and Nazif Bagriacik. Does instantaneous blood glucose affect vibration perception threshold measurement using biothesiometer? *Diabetes research and clinical practice*, 46(1):19–22, 1999.

Scikit Learn Developers. Random forest classifier, 2018.

Thanh-Nghi Do, Philippe Lenca, Stéphane Lallich, and Nguyen-Khang Pham. Classifying very-high-dimensional data with random forests of oblique decision trees. In *Advances in knowledge discovery and management*, pages 39–55. Springer, 2010.

Ronny Hänsch and Olaf Hellwich. Random forests. In *Photogrammetrie und Fernerkundung*, pages 603–643. Springer, 2017.

Wanja A. Hemmerich. Statistikguru, 2016.

AC Huntley and Jr RM Walter. Quantitative determination of skin thickness in diabetes mellitus: relationship to disease parameters. *Journal of medicine*, 21(5):257–264, 1990.

Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

Shahzad Khan. Ethem alpaydin. introduction to machine learning (adaptive computation and machine learning series), 2004.

Rabbert Klein. Black holes and their relation to hiding eggs. *Theoretical Easter Physics*, 2010. (to appear).

Will Koehrsen. Hyperparameter tuning the random forest in python. *Towards Data Science*, 2018.

Krishni. K-fold cross validation. *medium.com*, 2018.

Souad Larabi-Marie-Sainte, Linah Aburahmah, Rana Almohaini, and Tanzila Saba. Current techniques for diabetes prediction: Review and case study. *Applied Sciences*, 9(21):4604, 2019.

William J Long, John L Griffith, Harry P Selker, and Ralph B D'agostino. A comparison of logistic regression to decision-tree induction in a medical domain. *Computers and Biomedical Research*, 26(1):74–97, 1993.

Bernard ME Moret. Decision trees and diagrams. *ACM Computing Surveys (CSUR)*, 14(4):593–623, 1982.

Binoy B Nair, VP Mohandas, and NR Sakthivel. A genetic algorithm optimized decision tree-svm based stock market trend prediction system. *International journal on computer science and engineering*, 2(9):2981–2988, 2010.

Avinash Navlani. Decision tree classification in python. *Data Camp*, 2018.

Zach Pavan Ebbadi. Difference between random forests and decision tree. *International journal of advanced research in computer and communication engineering*, 2017.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

Devesh Poojari. *Machine Learning Basics: Decision Tree From Scratch*, 2019.

Dr. K. Nirmala R. Aruna devi. Construction of decision tree : Attribute selection measures. *Machine Learning*, 2(1):1–5, 2013.

Erhard Rahm and Hong Hai Do. Data cleaning: Problems and current approaches. *IEEE Data Eng. Bull.*, 23(4):3–13, 2000.

R Shaikh. Cross validation explained: Evaluating estimator performance. *Towards Data Science*, 2018.

Andrew Hershy shubham. machine-learning-algorithms-ml-decision-tree-tutorial. *https://www.hackerearth.com/zh/practice/machine-learning/machine-learning-algorithms/ml-decision-tree/tutorial/*, 2018.

Nishant Singh. Classification and regression by randomforest,discussion in 'big data and analytics'. *simplelearn community*, 2(3):18–22, 2018.

G Swapna, Soman Kp, and R Vinayakumar. Automated detection of diabetes using cnn and cnn-lstm network and heart rate signals. *Procedia computer science*, 132:1253–1262, 2018.

Dhanya Therese Jose. Big data analytics-case study-yelp dataset. Master's thesis, University of Stavanger, Norway, 2017.

Ritu Tiwari. Facts about random forest - and why they matter. *Machine learning*, 2019.

Amani Yahyaoui, Akhtar Jamil, Jawad Rasheed, and Mirsat Yesiltepe. A decision support system for diabetes prediction using machine learning and deep learning techniques. In *2019 1st International Informatics and Software Engineering Conference (UBMYK)*, pages 1–4. IEEE, 2019.

List of figures

List of Tables

Appendices

EDA and Prediction

May 15, 2020

1 Load libraries

[1]:
```
# Python libraries
# Classic,data manipulation and linear algebra
import pandas as pd
import numpy as np

#plots
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import plotly.offline as py
import plotly.figure_factory as ff
import plotly.graph_objs as go

# Data processing, metrics and modeling
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
```

2 Read the data

[2]:
```
col_names =␣
 ↪['pregnant','glucose','bp','skin','insulin','bmi','pedigree','age','diabetes']
data=pd.read_csv("/Users/ranjitsah/Documents/Bachelorarbeit/Data Analysis/
 ↪diabetes.csv",header=0,names=col_names)
```

[3]:
```
data.shape
```

[3]:
```
(768, 9)
```

[4]:
```
display(data.info(),data.head())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
```

```
pregnant    768 non-null int64
glucose     768 non-null int64
bp          768 non-null int64
skin        768 non-null int64
insulin     768 non-null int64
bmi         768 non-null float64
pedigree    768 non-null float64
age         768 non-null int64
diabetes    768 non-null int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

None

	pregnant	glucose	bp	skin	insulin	bmi	pedigree	age	diabetes
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

[5]:
```
data.nunique()
```

[5]:
```
pregnant     17
glucose     136
bp           47
skin         51
insulin     186
bmi         248
pedigree    517
age          52
diabetes      2
dtype: int64
```

[6]:
```
data.describe()
```

[6]:

	pregnant	glucose	bp	skin	insulin	bmi	\
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	

	pedigree	age	diabetes

```
count  768.000000  768.000000  768.000000
mean     0.471876   33.240885    0.348958
std      0.331329   11.760232    0.476951
min      0.078000   21.000000    0.000000
25%      0.243750   24.000000    0.000000
50%      0.372500   29.000000    0.000000
75%      0.626250   41.000000    1.000000
max      2.420000   81.000000    1.000000
```

[7]:
```
data['insulin'].max()
```

[7]:
```
846
```

3 Target variable

[8]:
```
# 2 datasets
#D for diabities person
# H for healthy person
D = data[(data['diabetes'] != 0)]
H = data[(data['diabetes'] == 0)]

#------------COUNT-----------------------
def target_count():
    trace = go.Bar( x = data['diabetes'].value_counts().values.tolist(),
                    y = ['healthy','diabetes' ],
                    orientation = 'h',
                    text=data['diabetes'].value_counts().values.tolist(),
                    textfont=dict(size=15),
                    textposition = 'auto',
                    opacity = 0.8,marker=dict(
                    color=['lightskyblue', 'gold'],
                    line=dict(color='#000000',width=1.5)))

    layout = dict(title =  'Count of Outcome variable')

    fig = dict(data = [trace], layout=layout)
    py.iplot(fig)

#------------PERCENTAGE-------------------
def target_percent():
    trace = go.Pie(labels = ['healthy','diabetes'], values = data['diabetes'].
 ↪value_counts(),
                   textfont=dict(size=15), opacity = 0.8,
                   marker=dict(colors=['lightskyblue', 'gold'],
                               line=dict(color='#000000', width=1.5)))
```

```
    layout = dict(title =  'Distribution of Outcome variable')

    fig = dict(data = [trace], layout=layout)
    py.iplot(fig)
```

[ ]:
```
target_count()
target_percent()
```

4 Data Preparation(Missing values)

[9]:
```
zeroCols = ['glucose', 'bp', 'skin', 'insulin','bmi'] # Columns with incorrect␣
 ↪Zero values
df2 = data.copy() # create a copy of the original dataframe
df2[zeroCols] = df2[zeroCols].replace(0,np.NaN) #Replace 0s with NaNs
df2.head()
```

[9]:

	pregnant	glucose	bp	skin	insulin	bmi	pedigree	age	diabetes
0	6	148.0	72.0	35.0	NaN	33.6	0.627	50	1
1	1	85.0	66.0	29.0	NaN	26.6	0.351	31	0
2	8	183.0	64.0	NaN	NaN	23.3	0.672	32	1
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21	0
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33	1

[10]:
```
df2.isnull().sum()
```

[10]:
```
pregnant      0
glucose       5
bp           35
skin        227
insulin     374
bmi          11
pedigree      0
age           0
diabetes      0
dtype: int64
```

[268]:
```
dataset=df2.copy()
```

[11]:
```
# Define missing plot to detect all missing values in dataset
def missing_plot(dataset, key) :
    null_feat = pd.DataFrame(dataset.isnull().sum(), columns = ['Count'])

    trace = go.Bar(x = null_feat.index, y = null_feat  ['Count'] ,opacity = 0.8,␣
 ↪text = null_feat ['Count'],  textposition = 'auto',marker=dict(color =␣
 ↪'#7EC0EE',
```

```
            line=dict(color='#000000',width=1.5)))


    fig = dict(data = [trace])
    py.iplot(fig)
```

[ ]:
```
missing_plot(dataset, 'diabities')
```

5 Replacing Missing Values and EDA

[12]:
```
# function To Fill Missing values
def median_target(var):
    temp = df2[df2[var].notnull()]
    temp = temp[[var, 'diabetes']].groupby(['diabetes'])[[var]].median().
 ↪reset_index()
    return temp
```

6 Insulin

[13]:
```
def plot_distribution(data_select, size_bin) :
    # 2 datasets
    tmp1 = D[data_select]
    tmp2 = H[data_select]
    hist_data = [tmp1, tmp2]

    group_labels = ['diabetic', 'healthy']
    colors = ['#FFD700', 'green']

    fig = ff.create_distplot(hist_data, group_labels, colors = colors, show_hist␣
 ↪= True, bin_size = size_bin, curve_type='kde')

    fig['layout'].update(title = data_select)

    py.iplot(fig, filename = 'Density plot')
```

[ ]:
```
plot_distribution('insulin',0)
```

[14]:
```
median_target('insulin')
```

[14]:

	diabetes	insulin
0	0	102.5
1	1	169.5

[15]:
```
df2.loc[(df2['diabetes'] == 0 ) & (df2['insulin'].isnull()), 'insulin'] = 102.5␣
↪#+ np.random.normal(0,1,236)
df2.loc[(df2['diabetes'] == 1 ) & (df2['insulin'].isnull()), 'insulin'] = 169.5␣
↪#+ np.random.normal(0,1,138)
#(236,138)
```

7 Glucose

[16]:
```
plot_distribution('glucose',0)
```

[17]:
```
median_target('glucose')
```

[17]:

	diabetes	glucose
0	0	107.0
1	1	140.0

[18]:
```
df2.loc[(df2['diabetes'] == 0 ) & (df2['glucose'].isnull()), 'glucose'] = 107.0
df2.loc[(df2['diabetes'] == 1 ) & (df2['glucose'].isnull()), 'glucose'] = 140.0
```

8 Skin

[19]:
```
plot_distribution('skin',0)
```

[20]:
```
median_target('skin')
```

[20]:

	diabetes	skin
0	0	27.0
1	1	32.0

[21]:
```
df2.loc[(df2['diabetes'] == 0 ) & (df2['skin'].isnull()), 'skin'] = 27.0 #+ np.
↪random.normal(0,1,139)
df2.loc[(df2['diabetes'] == 1 ) & (df2['skin'].isnull()), 'skin'] = 32.0 #+ np.
↪random.normal(0,1,88)

#(139,88)
```

9 BP

[22]:
```
plot_distribution('bp',0)
```

[23]:
```
median_target('bp')
```

[23]:
	diabetes	bp
0	0	70.0
1	1	74.5

[24]:
```
df2.loc[(df2['diabetes'] == 0 ) & (df2['bp'].isnull()), 'bp'] = 70.0 #+ np.
 ↪random.normal(0,1,19)
df2.loc[(df2['diabetes'] == 1 ) & (df2['bp'].isnull()), 'bp'] = 74.0 #+ np.
 ↪random.normal(0,1,16)
#(19,16)
```

10 BMI

[25]:
```
plot_distribution('bmi',0)
```

[26]:
```
median_target('bmi')
```

[26]:
	diabetes	bmi
0	0	30.1
1	1	34.3

[27]:
```
df2.loc[(data['diabetes'] == 0 ) & (df2['bmi'].isnull()), 'bmi'] = 30.1
df2.loc[(data['diabetes'] == 1 ) & (df2['bmi'].isnull()), 'bmi'] = 34.3
```

11 Age,pregnant and pedigree

[28]:
```
plot_distribution('age', 0)
plot_distribution('pregnant', 0)
plot_distribution('pedigree', 0)
```

12 Boxplot

[ ]:
```
plt.style.use('ggplot') # Using ggplot2 style visuals

f, ax = plt.subplots(figsize=(10, 8))

ax.set_facecolor('#fafafa')
ax.set(ylim=(.5, 950))
plt.ylabel('Bp')
plt.title("Bp Boxplot")
ax = sns.boxplot(x='diabetes',y='insulin',hue='diabetes',data=dataset,
 palette="Set3")
```

[29]:
```
def correlation_plot():
    #correlation
    correlation = df2.corr()
    #tick labels
    matrix_cols = correlation.columns.tolist()
    #convert to array
    corr_array  = np.array(correlation)
    trace = go.Heatmap(z = corr_array,
                       x = matrix_cols,
                       y = matrix_cols,
                       colorscale='Viridis',
                       colorbar   = dict() ,
                      )
    layout = go.Layout(dict(title = 'Correlation Matrix for variables',
                            #autosize = False,
                            #height  = 1400,
                            #width   = 1600,
                            margin  = dict(r = 0 ,l = 100,
                                           t = 0,b = 100,
                                          ),
                            yaxis   = dict(tickfont = dict(size = 9)),
                            xaxis   = dict(tickfont = dict(size = 9)),
                           )
                      )
    fig = go.Figure(data = [trace],layout = layout)
    py.iplot(fig)
```

[ ]:
```
correlation_plot()
```

13 creating Decision Tree Model

[30]:
```
array=df2.values
feature_cols=['pregnant','glucose','bp','skin','insulin','bmi','pedigree','age']
```

[31]:
```
#split dataset in features and target variable
X=array[:,0:8]
y=array[:,8]
```

[32]:
```
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=1) # 70% training and 30% test
```

[33]:
```
# Create Decision Tree classifer object
clf = DecisionTreeClassifier(random_state=1)

# Train Decision Tree Classifer
```

```
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)
```

[34]:
```
from sklearn.metrics import accuracy_score

# Model Accuracy, how often is the classifier correct?
print(round(accuracy_score(y_test,y_pred)*100,2))
```

```
84.42
```

[35]:
```
from sklearn.metrics  import confusion_matrix,classification_report
confusion_matrix = confusion_matrix(y_test,y_pred)
print(confusion_matrix)
```

```
[[125  21]
 [ 15  70]]
```

[36]:
```
from sklearn.metrics  import confusion_matrix,classification_report
confusion_matrix = classification_report(y_test,y_pred) #output_dict=True
print(confusion_matrix)
```

```
              precision    recall  f1-score   support

         0.0       0.89      0.86      0.87       146
         1.0       0.77      0.82      0.80        85

    accuracy                           0.84       231
   macro avg       0.83      0.84      0.83       231
weighted avg       0.85      0.84      0.85       231
```

[37]:
```
print(confusion_matrix)
```

```
              precision    recall  f1-score   support

         0.0       0.89      0.86      0.87       146
         1.0       0.77      0.82      0.80        85

    accuracy                           0.84       231
   macro avg       0.83      0.84      0.83       231
weighted avg       0.85      0.84      0.85       231
```

[40]:
```
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
```

[41]:
```
from sklearn.tree import export_graphviz
from sklearn.externals.six import StringIO
from IPython.display import Image
import pydotplus
```

[ ]:
```
dot_data = StringIO()
export_graphviz(clf, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True,feature_names =␣
 ↪feature_cols,class_names=['0','1'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('diabetes.png')
Image(graph.create_png())
```

14 optimizing Decision Tree performance

[42]:
```
clf = DecisionTreeClassifier(criterion='entropy',max_depth=2)
clf = clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
```

[43]:
```
# Model Accuracy, how often is the classifier correct?
print(round(metrics.accuracy_score(y_test,y_pred)*100,2))
```

```
85.28
```

[44]:
```
from sklearn.metrics  import confusion_matrix,classification_report
confusion_matrix = confusion_matrix(y_test,y_pred)
print(confusion_matrix)
```

```
[[136  10]
 [ 24  61]]
```

[45]:
```
from sklearn.metrics  import confusion_matrix,classification_report
confusion_matrix = classification_report(y_test,y_pred) #output_dict=True
```

[46]:
```
print(confusion_matrix)
```

```
              precision    recall  f1-score   support

         0.0       0.85      0.93      0.89       146
         1.0       0.86      0.72      0.78        85

    accuracy                           0.85       231
   macro avg       0.85      0.82      0.84       231
weighted avg       0.85      0.85      0.85       231
```

[ ]:
```
from sklearn.externals.six import StringIO
from sklearn.externals.six import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus
dot_data = StringIO()
export_graphviz(clf, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True, feature_names =␣
↪feature_cols,class_names=['0','1'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('diabetes_final.png')
Image(graph.create_png())
```

15 Roc- AUC Analysis for Decision Tree

[47]:
```
import sklearn.metrics as metrics
# calculate the fpr and tpr for all thresholds of the classification
probs = clf.predict_proba(X_test)
preds = probs[:,1]
fpr, tpr, threshold = metrics.roc_curve(y_test, preds)
roc_auc = metrics.auc(fpr, tpr)
```

[ ]:
```
plt.figure(figsize=(12,7))
plt.title('Receiver Operating Characteristic for Decision Tree')
plt.plot(fpr, tpr, 'green', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.savefig('/Users/ranjitsah/Downloads/roc_auc_dctree.png')
```

16 Comparing with Randomforest model

[48]:
```
rclf = RandomForestClassifier(random_state=5)
```

[49]:
```
rclf = rclf.fit(X_train,y_train)
y_pred = rclf.predict(X_test)
# Model Accuracy, how often is the classifier correct?
print(round(metrics.accuracy_score(y_test,y_pred)*100,2))
```

```
90.48
```

[50]:
```
from sklearn.metrics  import confusion_matrix,classification_report
confusion_matrix = confusion_matrix(y_test,y_pred)
print(confusion_matrix)
```

```
[[135  11]
 [ 11  74]]
```

[51]:
```
confusion_matrix=classification_report(y_test,y_pred)
```

[52]:
```
print(confusion_matrix)
```

```
              precision    recall  f1-score   support

         0.0       0.92      0.92      0.92       146
         1.0       0.87      0.87      0.87        85

    accuracy                           0.90       231
   macro avg       0.90      0.90      0.90       231
weighted avg       0.90      0.90      0.90       231
```

17 Roc- AUC Analysis for Random forest

[53]:
```
import sklearn.metrics as metrics
# calculate the fpr and tpr for all thresholds of the classification
probs = rclf.predict_proba(X_test)
preds = probs[:,1]
fpr, tpr, threshold = metrics.roc_curve(y_test, preds)
roc_auc = metrics.auc(fpr, tpr)
```

[ ]:
```
# method I: plt
plt.figure(figsize=(12,8))
plt.title('Receiver Operating Characteristic for Random Forest')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.savefig('/Users/ranjitsah/Downloads/roc_rf.png')
```

18 comparing with LogisticRegression

[54]:
```
## Importing stats models for running logistic regression
import statsmodels.api as sm
from statsmodels.formula.api import logit
```

[55]:
```
## Defining the model and assigning Y (Dependent) and Age (Independent Variables)
model_lin = sm.GLM.from_formula("diabetes ~ age",family=sm.families.Binomial(),␣
 ↪data=df2)
## Fitting the model and publishing the results
result_lin = model_lin.fit()
print(result_lin.summary())
```

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                diabetes   No. Observations:                  768
Model:                            GLM   Df Residuals:                      766
Model Family:                Binomial   Df Model:                            1
Link Function:                  logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -475.36
Date:                Fri, 15 May 2020   Deviance:                       950.72
Time:                        08:17:37   Pearson chi2:                     761.
No. Iterations:                     4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -2.0475      0.239     -8.572      0.000      -2.516      -1.579
age            0.0420      0.007      6.380      0.000       0.029       0.055
==============================================================================
```

[56]:
```
## Defining the model and assigning Y (Dependent) and X (Independent Variables)
model_lin = sm.GLM.from_formula("diabetes ~␣
 ↪pregnant+glucose+skin+bp+insulin+bmi+pedigree+age",family=sm.families.
 ↪Binomial(), data=df2)
## Fitting the model and publishing the results
result_lin = model_lin.fit()
print(result_lin.summary())
```

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                diabetes   No. Observations:                  768
Model:                            GLM   Df Residuals:                      759
Model Family:                Binomial   Df Model:                            8
Link Function:                  logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -344.33
Date:                Fri, 15 May 2020   Deviance:                       688.67
```

```
Time:                        08:17:38   Pearson chi2:                 1.59e+03
No. Iterations:                     5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -9.2850      0.832    -11.162      0.000     -10.915      -7.655
pregnant       0.1174      0.032      3.651      0.000       0.054       0.180
glucose        0.0304      0.004      7.827      0.000       0.023       0.038
skin           0.0378      0.014      2.736      0.006       0.011       0.065
bp            -0.0047      0.009     -0.531      0.595      -0.022       0.013
insulin        0.0054      0.001      3.606      0.000       0.002       0.008
bmi            0.0591      0.018      3.287      0.001       0.024       0.094
pedigree       0.7873      0.301      2.617      0.009       0.198       1.377
age            0.0131      0.010      1.355      0.175      -0.006       0.032
==============================================================================
```

19 Inference from the Logistic Regression¶

-To identify which variables influence the diabetes, we will look at the p-value of each variable. We expect the p-value to be less than 0.05(alpha risk)

-When p-value<0.05, we can say the variable influences the outcome Hence we will eliminate Age and bp, then we will re run the model.

[57]:
```
## Defining the model and assigning Y (Dependent) and Age (Independent Variables)
model_lin = sm.GLM.from_formula("diabetes ~␣
 ↪pregnant+glucose+skin+insulin+bmi+pedigree",family=sm.families.Binomial(),␣
 ↪data=df2)
## Fitting the model and publishing the results
result_lin = model_lin.fit()
print(result_lin.summary())
```

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:               diabetes   No. Observations:                  768
Model:                            GLM   Df Residuals:                      761
Model Family:                Binomial   Df Model:                            6
Link Function:                  logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -345.26
Date:                Fri, 15 May 2020   Deviance:                       690.52
Time:                        08:17:38   Pearson chi2:                 1.65e+03
No. Iterations:                     5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -9.2211      0.724    -12.742      0.000     -10.639      -7.803
```

```
pregnant       0.1374     0.028      4.956      0.000      0.083      0.192
glucose        0.0311     0.004      8.247      0.000      0.024      0.039
skin           0.0383     0.014      2.787      0.005      0.011      0.065
insulin        0.0053     0.001      3.590      0.000      0.002      0.008
bmi            0.0550     0.017      3.160      0.002      0.021      0.089
pedigree       0.8062     0.299      2.694      0.007      0.220      1.393
==============================================================================
```

20 Logit Model with significant variables

[58]:
```
cols=["pregnant", "glucose","skin","insulin","bmi","pedigree"]
X=df2[cols]
y=df2.diabetes
```

[59]:
```
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=1) # 70% training and 30% test
```

[60]:
```
from sklearn.linear_model import LogisticRegression
logreg=LogisticRegression()
logreg=logreg.fit(X_train,y_train)
```

[61]:
```
y_pred=logreg.predict(X_test)
```

[62]:
```
# Model Accuracy, how often is the classifier correct?
print(round(accuracy_score(y_test,y_pred)*100,2))
```

```
81.39
```

[63]:
```
from sklearn.metrics  import confusion_matrix,classification_report
confusion_matrix = confusion_matrix(y_test,y_pred)
print(confusion_matrix)
```

```
[[136  10]
 [ 33  52]]
```

[64]:
```
confusion_matrix=classification_report(y_test,y_pred)
```

[65]:
```
print(confusion_matrix)
```

```
              precision    recall  f1-score   support

           0       0.80      0.93      0.86       146
           1       0.84      0.61      0.71        85

    accuracy                           0.81       231
   macro avg       0.82      0.77      0.79       231
```

```
weighted avg       0.82      0.81      0.81       231
```

21 ROC-AUC Analysis for Logistics Regression

[66]:
```
import sklearn.metrics as metrics
# calculate the fpr and tpr for all thresholds of the classification
probs = logreg.predict_proba(X_test)
preds = probs[:,1]
fpr, tpr, threshold = metrics.roc_curve(y_test, preds)
roc_auc = metrics.auc(fpr, tpr)
```

[ ]:
```
plt.figure(figsize=(12,7))
plt.title('Receiver Operating Characteristic for  Logistic Regression')
plt.plot(fpr, tpr, 'orange', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.savefig('/Users/ranjitsah/Downloads/roc_lgr.png')
```

22 Comparison between the Models

[67]:
```
array=df2.values
#split dataset in features and target variable
X=array[:,0:8]
y=array[:,8]
```

[68]:
```
# prepare models
from sklearn import model_selection
models = []
models.append(('LogisticRegression',LogisticRegression()))
models.append(('DecisionTree', DecisionTreeClassifier()))
models.append((' RandomForest', RandomForestClassifier()))
```

[69]:
```
# evaluate each model in turn
results = []
names = []
scoring = 'accuracy'
for name, model in models:
        kfold = model_selection.KFold(n_splits=10, random_state=7)
        cv_results = model_selection.cross_val_score(model, X,y, cv=kfold,␣
 ↪scoring=scoring)
```

```
        results.append(cv_results)
        names.append(name)
        msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
        print(msg)
```

```
LogisticRegression: 0.778640 (0.041128)
DecisionTree: 0.845130 (0.037581)
 RandomForest: 0.865875 (0.039446)
```

[ ]:
```
# boxplot algorithm comparison
fig = plt.figure(figsize=(10,6))
fig.suptitle('Algorithm Comparison')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
```