**Contents**

1. Abstract
2. Introduction
3. Problem Definition and Algorithm
   1. Task Definition
   2. Diabetes Dataset
   3. Algorithm Definition
4. Experimental Setup and Evaluation
   1. Components to be Installed
   2. Methodology
   3. System Overview
5. Related Work
6. Conclusion
7. Future Work

**1. Abstract**

Nowadays, diabetes is one of deadliest diseases in the world. It is not only a disease but also a creator of other diseases like heart attack, blindness, kidney problems, etc. The growth of the diabetic patients is increasing day-by-day due to several reasons such as toxic or chemical contents mix with the food, auto immune reaction, obesity, bad diet, change in lifestyles, eating habit, environment pollution, stress etc. Hence, diagnosing the diabetes is very essential to save the human life from diabetes.

But with the rise of Machine Learning approaches we have the ability to find a solution to this issue. Machine Learning is defined as an application of artificial intelligence where available information is used through algorithms to process or assist the processing of statistical data. It is the set of rules that a computer develops in order to solve problems. The basic idea is that the machine learning computer will find out the patterns in data. And predict the outcomes. In this project we are going to develop a system which has the ability to predict whether the patient has diabetes or not. Furthermore, predicting the disease early leads to treating the patients before it becomes critical. In this study, the main focus is to investigate different types of machine learning classification algorithms and show their comparative analysis. To implement this project scikit-learn(python package) tools are used.

(Keywords: Classification, K-Nearest-Neighbor, Logistic Regression, Naïve Bayes, Decision Tree, Machine Learning)

**2. Introduction**

Now days from healthcare industries large amount of data is generating. This data contains lots of useful information that can be used to take efficient medical decisions. Hence it is necessary to store and process such data to extract knowledge from it and by using that take efficient decisions. There are various techniques such as data mining, machine learning, statistical analysis etc can be used to analyze medical data. By using these recent techniques early diagnosis of various diseases can be done so that patients can be provided with better cure and care. In current situations the numbers of people suffering from Diabetes are increasing per day. Diabetic Mellitus (DM) belongs to the family of Non Communicable Diseases (NCD). It is one of the dangerous diseases as it has long term impacts on patient’s body. Diabetic Mellitus (DM) is categorized into three types. First is Type 1 DM which is called as Insulin - Dependent Diabetes Mellitus (IDDM). Type 1 DM is caused when patient’s body is unable to produce insulin and requires injecting insulin externally to the patient. The second type of DM is Type 2 DM which is called as Non-Insulin-Dependent Diabetes Mellitus (NIDDM). This type of diabetes is caused when patient’s body cell are not able to use insulin properly. Third type of DM is gestational diabetes which is caused in pregnant women due to the development of high blood sugar without the knowledge of Pre-diagnosis of Diabetes. This type of DM is very dangerous and can lead to Type 2 DM. As numbers of patients of this serious disease are increasing day by day, the size of diabetic data set is also increasing. With the utilization of machine learning algorithm, it can be possible to analyze the diabetes data and accordingly provide early diagnosis and better treatment. Machine learning is the set of different methods that can be used to find patterns from the dataset and then use those patterns to predict future conditions or to make efficient decisions under some conditions. Machine learning introduces different algorithms that can be used to make understanding of the current situations to machine and then based upon that machines can take decisions. Machine learning can make decisions independently at its own. Machine learning is categorized into to two types, supervised learning and unsupervised learning.

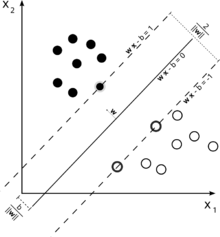
1. **Supervised Learning:** In supervised learning the input and output both is known. This type of machine learning is known as supervised learning because it learns from the training data set and creates knowledge base and applies it on the testing data set to make appropriate decisions.

2. **Unsupervised Learning:** In Unsupervised learning we are aware about only input. Hence the very important task of unsupervised learning algorithm is to build up class labels automatically. Unsupervised learning algorithm finds relationship between the data and then same characterized data are grouped together which is known as cluster. Cluster analysis is the second name of unsupervised learning.

Machine learning (ML) is the scientific study of algorithms and statistical models that computer systems use to progressively improve their performance on a specific task. Machine learning algorithms build a mathematical model of sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to perform the task.[1][2]:2 Machine learning algorithms are used in the applications of email filtering, detection of network intruders, and computer vision, where it is infeasible to develop an algorithm of specific instructions for performing the task. Machine learning is closely related to computational statistics, which focuses on making predictions using computers. The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. Data mining is a field of study within machine learning, and focuses on exploratory data analysis through unsupervised learning. In its application across business problems, machine learning is also referred to as predictive analytics.

The name *machine learning* was coined in 1959 by Arthur Samuel. Tom M. Mitchell provided a widely quoted, more formal definition of the algorithms studied in the machine learning field: "A computer program is said to learn from experience *E* with respect to some class of tasks *T* and performance measure *P* if its performance at tasks in *T*, as measured by *P*, improves with experience *E*." This definition of the tasks in which machine learning is concerned offers a fundamentally operational definition rather than defining the field in cognitive terms. This follows Alan Turing's proposal in his paper "Computing Machinery and Intelligence", in which the question "Can machines think?" is replaced with the question "Can machines do what we (as thinking entities) can do?". In Turing's proposal the various characteristics that could be possessed by a *thinking machine* and the various implications in constructing one are exposed.

**Machine learning tasks**



*A support vector machine is a supervised learning model that divides the data into regions separated by a linear boundary. Here, the linear boundary divides the black circles from the white.*

Machine learning tasks are classified into several broad categories. In supervised learning, the algorithm builds a mathematical model of a set of data that contains both the inputs and the desired outputs. For example, if the task were determining whether an image contained a certain object, the training data for a supervised learning algorithm would include images with and without that object (the input), and each image would have a label (the output) designating whether it contained the object. In special cases, the input may be only partially available, or restricted to special feedback. Semi-supervised learning algorithms develop mathematical models from incomplete training data, where a portion of the sample inputs are missing the desired output.

Classification algorithms and regression algorithms are types of supervised learning. Classification algorithms are used when the outputs are restricted to a limited set of values. For a classification algorithm that filters emails, the input would be an incoming email, and the output would be the name of the folder in which to file the email. For an algorithm that identifies spam emails, the output would be the prediction of either "spam" or "not spam", represented by the Boolean values one and zero. Regression algorithms are named for their continuous outputs, meaning they may have any value within a range. Examples of a continuous value are the temperature, length, or price of an object.

In unsupervised learning, the algorithm builds a mathematical model of a set of data which contains only inputs and no desired outputs. Unsupervised learning algorithms are used to find structure in the data, like grouping or clustering of data points. Unsupervised learning can discover patterns in the data, and can group the inputs into categories, as in feature learning. Dimensionality reduction is the process of reducing the number of "features", or inputs, in a set of data.

Active learning algorithms access the desired outputs (training labels) for a limited set of inputs based on a budget, and optimize the choice of inputs for which it will acquire training labels. When used interactively, these can be presented to a human user for labeling. Reinforcement learning algorithms are given feedback in the form of positive or negative reinforcement in a dynamic environment, and are used in autonomous vehicles or in learning to play a game against a human opponent. Other specialized algorithms in machine learning include topic modeling, where the computer program is given a set of natural language documents and finds other documents that cover similar topics. Machine learning algorithms can be used to find the unobservable probability density function in density estimation problems. Meta learning algorithms learn their own inductive bias based on previous experience. In developmental robotics, robot learning algorithms generate their own sequences of learning experiences, also known as a curriculum, to cumulatively acquire new skills through self-guided exploration and social interaction with humans. These robots use guidance mechanisms such as active learning, maturation, motor synergies, and imitation.

Supervised learning algorithms build a mathematical model of a set of data that contains both the inputs and the desired outputs. The data is known as training data, and consists of a set of training examples. Each training example has one or more inputs and a desired output, also known as a supervisory signal. In the case of semi-supervised learning algorithms, some of the training examples are missing the desired output. In the mathematical model, each training example is represented by an array or vector, and the training data by a matrix. Through iterative optimization of an objective function, supervised learning algorithms learn a function that can be used to predict the output associated with new inputs.[19] An optimal function will allow the algorithm to correctly determine the output for inputs that were not a part of the training data. An algorithm that improves the accuracy of its outputs or predictions over time is said to have learned to perform that task.

Supervised learning algorithms include classification and regression. Classification algorithms are used when the outputs are restricted to a limited set of values, and regression algorithms are used when the outputs may have any numerical value within a range. Similarity learning is an area of supervised machine learning closely related to regression and classification, but the goal is to learn from examples using a similarity function that measures how similar or related two objects are. It has applications in ranking, recommendation systems, visual identity tracking, face verification, and speaker verification.

Unsupervised learning algorithms take a set of data that contains only inputs, and find structure in the data, like grouping or clustering of data points. The algorithms therefore learn from test data that has not been labeled, classified or categorized. Instead of responding to feedback, unsupervised learning algorithms identify commonalities in the data and react based on the presence or absence of such commonalities in each new piece of data. A central application of unsupervised learning is in the field of density estimation in statistics,[21]though unsupervised learning encompasses other domains involving summarizing and explaining data features.

Cluster analysis is the assignment of a set of observations into subsets (called *clusters*) so that observations within the same cluster are similar according to one or more predesignated criteria, while observations drawn from different clusters are dissimilar. Different clustering techniques make different assumptions on the structure of the data, often defined by some *similarity metric* and evaluated, for example, by *internal compactness*, or the similarity between members of the same cluster, and *separation*, the difference between clusters. Other methods are based on *estimated density* and *graph connectivity*.

Reinforcement learning is an area of machine learning concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward. Due to its generality, the field is studied in many other disciplines, such as game theory, control theory, operations research, information theory, simulation-based optimization, multi-agent systems, swarm intelligence, statistics and genetic algorithms.[22][23] In machine learning, the environment is typically represented as a Markov Decision Process (MDP). Many reinforcement learning algorithms use dynamic programming techniques. Reinforcement learning algorithms do not assume knowledge of an exact mathematical model of the MDP, and are used when exact models are infeasible.Reinforcement learning algorithms are used in autonomous vehicles or in learning to play a game against a human opponent.

**Supervised Learning**

**Supervised learning** is the machine learning task of learning a function that maps an input to an output based on example input-output pairs. It infers a function from *labeled training data* consisting of a set of *training examples*.[2] In supervised learning, each example is a *pair* consisting of an input object (typically a vector) and a desired output value (also called the *supervisory signal*). A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way (see inductive bias).

The parallel task in human and animal psychology is often referred to as concept learning.

**Steps**

In order to solve a given problem of supervised learning, one has to perform the following steps:

1. Determine the type of training examples. Before doing anything else, the user should decide what kind of data is to be used as a training set. In case of handwriting analysis, for example, this might be a single handwritten character, an entire handwritten word, or an entire line of handwriting.
2. Gather a training set. The training set needs to be representative of the real-world use of the function. Thus, a set of input objects is gathered and corresponding outputs are also gathered, either from human experts or from measurements.
3. Determine the input feature representation of the learned function. The accuracy of the learned function depends strongly on how the input object is represented. Typically, the input object is transformed into a feature vector, which contains a number of features that are descriptive of the object. The number of features should not be too large, because of the curse of dimensionality; but should contain enough information to accurately predict the output.
4. Determine the structure of the learned function and corresponding learning algorithm. For example, the engineer may choose to use support vector machines or decision trees.
5. Complete the design. Run the learning algorithm on the gathered training set. Some supervised learning algorithms require the user to determine certain control parameters. These parameters may be adjusted by optimizing performance on a subset (called a *validation* set) of the training set, or via cross-validation.

6.Evaluate the accuracy of the learned function. After parameter adjustment and learning, the performance of the resulting function should be measured on a test set that is separate from the training set.

**Algorithm choice**

A wide range of supervised learning algorithms are available, each with its strengths and weaknesses. There is no single learning algorithm that works best on all supervised learning problems (see the No free lunch theorem).

There are four major issues to consider in supervised learning:

### Bias-variance tradeoff

A first issue is the tradeoff between *bias* and *variance*. Imagine that we have

Available several different, but equally good, training data sets. A learning algorithm is biased for a particular input {\displaystyle x} if, when trained on each of these data sets, it is systematically incorrect when predicting the correct output for{\displaystyle x}. A learning algorithm has high variance for a particular input {\displaystyle x}if it predicts different output values when trained on different training sets. The prediction error of a learned classifier is related to the sum of the bias and the variance of the learning algorithm. Generally, there is a tradeoff between bias and variance. A learning algorithm with low bias must be "flexible" so that it can fit the data well. But if the learning algorithm is too flexible, it will fit each training data set differently, and hence have high variance. A key aspect of many supervised learning methods is that they are able to adjust this tradeoff between bias and variance (either automatically or by providing a bias/variance parameter that the user can adjust).

### Function complexity and amount of training data

The second issue is the amount of training data available relative to the complexity of the "true" function (classifier or regression function). If the true function is simple, then an "inflexible" learning algorithm with high bias and low variance will be able to learn it from a small amount of data. But if the true function is highly complex (e.g., because it involves complex interactions among many different input features and behaves differently in different parts of the input space), then the function will only be learnable from a very large amount of training data and using a "flexible" learning algorithm with low bias and high variance.

### Dimensionality of the input space

A third issue is the dimensionality of the input space. If the input feature vectors have very high dimension, the learning problem can be difficult even if the true function only depends on a small number of those features. This is because the many "extra" dimensions can confuse the learning algorithm and cause it to have high variance. Hence, high input dimensionality typically requires tuning the classifier to have low variance and high bias. In practice, if the engineer can manually remove irrelevant features from the input data, this is likely to improve the accuracy of the learned function. In addition, there are many algorithms for feature selection that seek to identify the relevant features and discard the irrelevant ones. This is an instance of the more general strategy of dimensionality reduction, which seeks to map the input data into a lower-dimensional space prior to running the supervised learning algorithm.

### Noise in the output values

A fourth issue is the degree of noise in the desired output values (the supervisory target variables). If the desired output values are often incorrect (because of human error or sensor errors), then the learning algorithm should not attempt to find a function that exactly matches the training examples. Attempting to fit the data too carefully leads to over fitting. You can over fit even when there are no measurement errors (stochastic noise) if the function you are trying to learn is too complex for your learning model. In such a situation, the part of the target function that cannot be modeled "corrupts" your training data - this phenomenon has been called deterministic noise. When either type of noise is present, it is better to go with a higher bias, lower variance estimator.

In practice, there are several approaches to alleviate noise in the output values such as early stopping to prevent over fitting as well as detecting and removing the noisy training examples prior to training the supervised learning algorithm. There are several algorithms that identify noisy training examples and removing the suspected noisy training examples prior to training has decreased generalization error with statistical significance.

### Other factors to consider (important)

Other factors to consider when choosing and applying a learning algorithm include the following:

* Heterogeneity of the data. If the feature vectors include features of many different kinds (discrete, discrete ordered, counts, continuous values), some algorithms are easier to apply than others. Many algorithms, including Support Vector Machines, linear regression, logistic regression, neural networks, and nearest neighbor methods, require that the input features be numerical and scaled to similar ranges (e.g., to the [-1,1] interval). Methods that employ a distance function, such as nearest neighbor methods and support vector machines with Gaussian kernels, are particularly sensitive to this. An advantage of decision trees is that they easily handle heterogeneous data.
* Redundancy in the data. If the input features contain redundant information (e.g., highly correlated features), some learning algorithms (e.g., linear regression, logistic regression, and distance based methods) will perform poorly because of numerical instabilities. These problems can often be solved by imposing some form of regularization.
* Presence of interactions and non-linearity. If each of the features makes an independent contribution to the output, then algorithms based on linear functions (e.g., linear regression, logistic regression, Support Vector Machines, naive Bayes) and distance functions (e.g., nearest neighbor methods, support vector machines with Gaussian kernels) generally perform well. However, if there are complex interactions among features, then algorithms such as decision trees and neural networks work better, because they are specifically designed to discover these interactions. Linear methods can also be applied, but the engineer must manually specify the interactions when using them.

When considering a new application, the engineer can compare multiple learning algorithms and experimentally determine which one works best on the problem at hand (see cross validation). Tuning the performance of a learning algorithm can be very time-consuming. Given fixed resources, it is often better to spend more time collecting additional training data and more informative features than it is to spend extra time tuning the learning algorithms.

### Algorithms

The most widely used learning algorithms are:

* Support Vector Machines
* linear regression
* logistic regression
* naive Bayes
* linear discriminant analysis
* decision trees
* k-nearest neighbor algorithm
* Neural Networks (Multilayer perceptron)

**Unsupervised Learning**

**Unsupervised learning** is a branch of machine learning that learns from test data that has not been labeled, classified or categorized. Instead of responding to feedback, unsupervised learning identifies commonalities in the data and reacts based on the presence or absence of such commonalities in each new piece of data. Alternatives include supervised learning and reinforcement learning.

A central application of unsupervised learning is in the field of density estimation in statistics, though unsupervised learning encompasses many other domains involving summarizing and explaining data features. It could be contrasted with supervised learning by saying that whereas supervised learning intends to infer a conditional probability distribution  px(x|y) {\textstyle p\_{X}(x\,|\,y)}pp conditioned on the label {\textstyle y}xxof y input data; unsupervised learning intends to infer an a priori probability distribution {\textstyle p\_{X}(x)}px(x).

**Approaches**

Compared to supervised learning where training data is labeled with the appropriate classifications, models using unsupervised learning must learn relationships between elements in a data set and classify the raw data without "help." This hunt for relationships can take many different algorithmic forms, but all models have the same goal of mimicking human logic by searching for indirect hidden structures, patterns or features to analyze new data.

Some of the most common algorithms used in unsupervised learning include:

* ***Clustering***
  + hierarchical clustering,
  + k-means
  + mixture models
  + DBSCAN
  + OPTICS algorithm
* ***Anomaly detection***
  + Local Outlier Factor
* ***Neural Networks***
  + Auto encoders
  + Deep Belief Nets
  + Hebbian Learning
  + Generative Adversarial Networks
  + Self-organizing map
* ***Approaches for learning latent variable models*** such as
  + Expectation–maximization algorithm (EM)
  + Method of moments
  + Blind signal separation techniques

**Reinforcement learning** (**RL**)

**Reinforcement learning** (**RL**) is an area of machine learning concerned with how software agents ought to take *actions* in an *environment* so as to maximize some notion of cumulative *reward*. The problem, due to its generality, is studied in many other disciplines, such as game theory, control theory, operations research, information theory, simulation-based optimization, multi-agent systems, swarm intelligence, statistics and genetic algorithms. In the operations research and control literature, reinforcement learning is called *approximate dynamic programming,* or *neuro-dynamic programming.*The problems of interest in reinforcement learning have also been studied in the theory of optimal control, which is concerned mostly with the existence and characterization of optimal solutions, and algorithms for their exact computation, and less with learning or approximation, particularly in the absence of a mathematical model of the environment. In economics and game theory, reinforcement learning may be used to explain how equilibrium may arise under bounded rationality. In machine learning, the environment is typically formulated as a Markov Decision Process (MDP), as many reinforcement learning algorithms for this context utilize dynamic programming techniques. The main difference between the classical dynamic programming methods and reinforcement learning algorithms is that the latter do not assume knowledge of an exact mathematical model of the MDP and they target large MDPs where exact methods become infeasible.

Reinforcement learning is considered as one of three machine learning paradigms, alongside supervised learning and unsupervised learning. It differs from supervised learning in that correct input/output pairs need not be presented, and sub-optimal actions need not be explicitly corrected. Instead the focus is on performance, which involves finding a balance between exploration (of uncharted territory) and exploitation (of current knowledge).The exploration vs. exploitation trade-off has been most thoroughly studied through the multi-armed bandit problem and in finite MDPs.

**Future of Machine Learning**

While machine learning algorithms have been around for decades, they've attained new popularity as artificial intelligence (AI) has grown in prominence. Deep learning models in particular power today's most advanced AI applications.

In this work five machine learning algorithms; KNN, Logistic regression, Naïve Bayes, Decission Tree and Support Vector Machine algorithms are used for the analysis of diabetes data.

**3. Problem definition and algorithm**

**3.1 Task definition**

Due to rising cost of health care, it is useful to assist patients to control diabetes by themselves. In many instances, early information related to diabetes might help in avoidance, curing and appropriate treatment of the disease. Many computer programs or systems were developed and are being developed by emulating human intelligence that could be used to assist the users or patients in managing diabetes. We assessed different systems such as artificial intelligence systems, mobile phone applications and specially designed devices for the prediction and diagnosis of diabetes. The focus of this project is to investigate for a model to predict and diagnose diabetes in the long run. Most of the models have been developed to diagnose diabetes and predict the blood sugar level for a short term. However, there are rarely any systems developed to predict the onset of diabetes in the long run. In the next section, a brief review on all related systems is done.

Applying machine learning methods in the diabetes mellitus research is a key approach to using large volumes of available data for extracting knowledge. The severe social impact of the disease renders diabetes mellitus as one of the most prioritized in medical science research, which inevitably generates huge amounts of data. Therefore, without any doubt machine learning approach is of the most concern when it comes to detection and diagnosis or any other such medical/clinical aspects to the disease.

**3.2 Diabetes data set**

To perform the experimentation Pima Indians Diabetes Dataset has been used. This dataset is located at UCI Machine Learning Repository. The dataset contains 8 attributes and 1 class variable. This data set contains total 768 diabetic and non-diabetic women records whose age is above 21 years. All attributes in data set are numeric. For this work the dataset is considered as complete data set having no missing value in it.

The 8 attributes that are defined here are as follows:

1. Number of Times Pregnancy occurred
2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test
3. Diastolic blood pressure (mm Hg)
4. Triceps skin fold thickness (mm)
5. 2-Hour serum insulin (mu U/ml)
6. Body mass index (weight in kg/(height in m)^2)
7. Diabetes pedigree function
8. Age (years)
9. Class variable (0 or 1)
10. 0-It indicates False Diabetic Test
11. II. 1-It indicates True Diabetic Test

More than 70% Pima Indian population is suffering from the diabetes.

**3.3 Algorithm definition**

**Classification algorithms:**

In machine learning and statistics, **classification** is the problem of identifying to which of a set of categories (sub-populations) a new observation belongs, on the basis of a training set of data containing observations (or instances) whose category membership is known. Examples are assigning a given email to the "spam" or "non-spam" class, and assigning a diagnosis to a given patient based on observed characteristics of the patient (sex, blood pressure, presence or absence of certain symptoms, etc.). Classification is an example of pattern recognition.

In the terminology of machine learning, classification is considered an instance of supervised learning, i.e., learning where a training set of correctly identified observations is available. The corresponding unsupervised procedure is known as clustering, and involves grouping data into categories based on some measure of inherent similarity or distance.

Often, the individual observations are analyzed into a set of quantifiable properties, known variously as explanatory variables or *features*. These properties may variously be categorical (e.g. "A", "B", "AB" or "O", for blood type), ordinal (e.g. "large", "medium" or "small"), integer-valued (e.g. the number of occurrences of a particular word in an email) or real-valued (e.g. a measurement of blood pressure). Other classifiers work by comparing observations to previous observations by means of a similarity or distance function.

An algorithm that implements classification, especially in a concrete implementation, is known as a **classifier**. The term "classifier" sometimes also refers to the mathematical function, implemented by a classification algorithm, that maps input data to a category.

Terminology across fields is quite varied. In statistics, where classification is often done with logistic regression or a similar procedure, the properties of observations are termed explanatory variables (or independent variables, regressors, etc.), and the categories to be predicted are known as outcomes, which are considered to be possible values of the dependent variable. In machine learning, the observations are often known as *instances*, the explanatory variables are termed *features* (grouped into a feature vector), and the possible categories to be predicted are *classes*. Other fields may use different terminology: e.g. in community ecology, the term "classification" normally refers to cluster analysis, i.e., a type of unsupervised learning, rather than the supervised learning described in this article.

In unsupervised learning, classifiers form the backbone of cluster analysis and in supervised or semi-supervised learning, classifiers are how the system characterizes and evaluates unlabeled data. In all cases though, classifiers have a specific set of dynamic rules, which includes an interpretation procedure to handle vague or unknown values, all tailored to the type of inputs being examined.

Since no single form of classification is appropriate for all data sets, a large toolkit of classification algorithms have been developed. The most commonly used include:

* Linear classifiers
* Logistic regression
  + Naive Bayes classifier
  + Perceptron
  + Support vector machines
* Least squares support vector machines
* Quadratic classifiers
* Kernel estimation
* k-nearest neighbor
* Boosting (meta-algorithm)
* Decision trees
* Random forests
* Neural networks
* Learning vector quantization

**K-Nearest Neighbor(KNN):**

KNN is known as a instance-based learning, or also popular as a lazy learning, where the role is only estimated locally and all calculation is postponed until classification. The k-NN algorithm is the most simplest among all machine learning algorithms. It studies all existing cases and classifies into new cases based on common factor. (e.g., distance functions). Features for which it is very popular is its simplicity of interpretation and short calculation time even with such ease, it can provide greatly feasible results. The neighbors are taken from a set of objects for which the object property value (for k-NN regression) or the classes (for kNN classification) are recognized. This can be supposed of as the training set for the algorithm, though no explicit training step is essential. Significance of k is constantly a positive integer. Here neighbors are selected from set of objects so constantly accurate classification is recognized. Categorizing objects based on the following exercise data in the feature space.

In pattern recognition, the ***k*-nearest neighbors algorithm** (***k*-NN**) is a non-parametric method used for classification and regression. In both cases, the input consists of the *k* closest training examples in the feature space. The output depends on whether *k*-NN is used for classification or regression:

* In *k-NN classification*, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its *k* nearest neighbors (*k* is a positive integer, typically small). If *k* = 1, then the object is simply assigned to the class of that single nearest neighbor.
* In *k-NN regression*, the output is the property value for the object. This value is the average of the values of its *k* nearest neighbors.

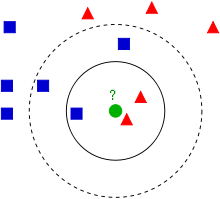
*k*-NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification. The *k*-NN algorithm is among the simplest of all machine learning algorithms.

Both for classification and regression, a useful technique can be used to assign weight to the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. For example, a common weighting scheme consists in giving each neighbor a weight of 1/*d*, where *d* is the distance to the neighbor.

The neighbors are taken from a set of objects for which the class (for *k*-NN classification) or the object property value (for *k*-NN regression) is known. This can be thought of as the training set for the algorithm, though no explicit training step is required.

A peculiarity of the *k*-NN algorithm is that it is sensitive to the local structure of the data.

**EXAMPLE**



*Example of k-NN classification. The test sample (green circle) should be classified either to the first class of blue squares or to the second class of red triangles. If k = 3 (solid line circle) it is assigned to the second class because there are 2 triangles and only 1 square inside the inner circle. If k = 5 (dashed line circle) it is assigned to the first class (3 squares vs. 2 triangles inside the outer circle).*

The training examples are vectors in a multidimensional feature space, each with a class label. The training phase of the algorithm consists only of storing the feature vectors and class labels of the training samples.

In the classification phase, *k* is a user-defined constant, and an unlabeled vector (a query or test point) is classified by assigning the label which is most frequent among the *k* training samples nearest to that query point.

A commonly used distance metric for continuous variables is Euclidean distance. For discrete variables, such as for text classification, another metric can be used, such as the **overlap metric** (or Hamming distance). In the context of gene expression microarray data, for example, *k*-NN has also been employed with correlation coefficients such as Pearson and Spearman. Often, the classification accuracy of *k*-NN can be improved significantly if the distance metric is learned with specialized algorithms such as Large Margin Nearest Neighbor or Neighborhood components analysis.

A drawback of the basic "majority voting" classification occurs when the class distribution is skewed. That is, examples of a more frequent class tend to dominate the prediction of the new example, because they tend to be common among the *k* nearest neighbors due to their large number.One way to overcome this problem is to weight the classification, taking into account the distance from the test point to each of its *k* nearest neighbors. The class (or value, in regression problems) of each of the *k* nearest points is multiplied by a weight proportional to the inverse of the distance from that point to the test point. Another way to overcome skew is by abstraction in data representation. For example, in a self-organizing map (SOM), each node is a representative (a center) of a cluster of similar points, regardless of their density in the original training data. *K*-NN can then be applied to the SOM.

The best choice of *k* depends upon the data; generally, larger values of *k* reduces effect of the noise on the classification,but make boundaries between classes less distinct. A good *k* can be selected by various heuristic techniques (see hyper parameter optimization). The special case where the class is predicted to be the class of the closest training sample (i.e. when *k* = 1) is called the nearest neighbor algorithm.

The accuracy of the *k*-NN algorithm can be severely degraded by the presence of noisy or irrelevant features, or if the feature scales are not consistent with their importance. Much research effort has been put into selecting or scaling features to improve classification. A particularly popularapproach is the use of evolutionary algorithms to optimize feature scaling. Another popular approach is to scale features by the mutual information of the training data with the training classes.

***In binary (two class) classification problems, it is helpful to choose k to be an odd number as this avoids tied votes. One popular way of choosing the empirically optimal k in this setting is via bootstrap method.***

## How does KNN work?

In the classification setting, the K-nearest neighbor algorithm essentially boils down to forming a majority vote between the K most similar instances to a given “unseen” observation. Similarity is defined according to a distance metric between two data points. A popular choice is the Euclidean distance given by

d(x,x′)=(x1−x′1)2+(x2−x′2)2+…+(xn−x′n)2−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−√d(x,x′)=(x1−x1′)2+(x2−x2′)2+…+(xn−xn′)2

but other measures can be more suitable for a given setting and include the Manhattan, Chebyshev and Hamming distance.

More formally, given a positive integer K, an unseen observation xx and a similarity metric dd, KNN classifier performs the following two steps:

* It runs through the whole dataset computing dd between xx and each training observation. We’ll call the K points in the training data that are closest to xx the set AA. Note that K is usually odd to prevent tie situations.
* It then estimates the conditional probability for each class, that is, the fraction of points in AA with that given class label. (Note I(x)I(x) is the indicator function which evaluates to 11 when the argument xx is true and 00otherwise)

P(y=j|X=x)=1K∑i∈AI(y(i)=j)P(y=j|X=x)=1K∑i∈AI(y(i)=j)

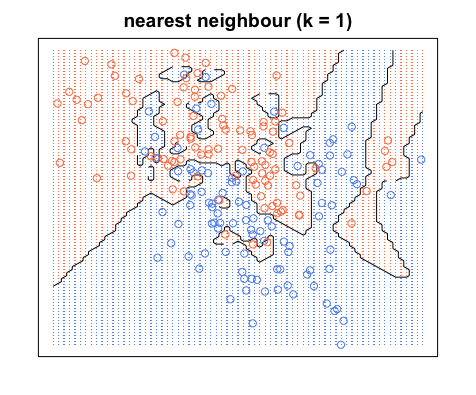
Finally, our input xx gets assigned to the class with the largest probability.

***KNN searches the memorized training observations for the K instances that most closely resemble the new instance and assigns to it the their most common class.***

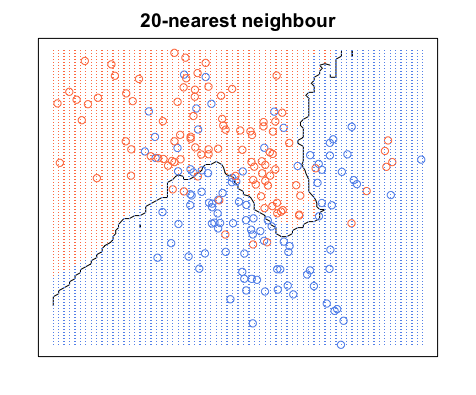
An alternate way of understanding KNN is by thinking about it as calculating a decision boundary (i.e. boundaries for more than 2 classes) which is then used to classify new points.

Like most machine learning algorithms, the K in KNN is a hyper parameter that we, as a designer, must pick in order to get the best possible fit for the data set. Intuitively, we can think of K as controlling the shape of the decision boundary.

When K is small, we are restraining the region of a given prediction and forcing our classifier to be “more blind” to the overall distribution. A small value for K provides the most flexible fit, which will have low bias but high variance. Graphically, our decision boundary will be more jagged.



On the other hand, a higher K averages more voters in each prediction and hence is more resilient to outliers. Larger values of K will have smoother decision boundaries which mean lower variance but increased bias.



## Pros and Cons of KNN

## Pros

One of the most attractive features of the K-nearest neighbor algorithm is that is simple to understand and easy to implement. With zero to little training time, it can be a useful tool for off-the-bat analysis of some data set you are planning to run more complex algorithms on. Furthermore, KNN works just as easily with multiclass data sets whereas other algorithms are hardcoded for the binary setting. Finally, as we mentioned earlier, the non-parametric nature of KNN gives it an edge in certain settings where the data may be highly “unusual”.

#### Cons

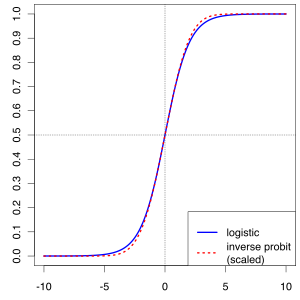
One of the obvious drawbacks of the KNN algorithm is the computationally expensive testing phase which is impractical in industry settings. Note the rigid dichotomy between KNN and the more sophisticated Neural Network which has a lengthy training phase albeit a **very fast** testing phase. Furthermore, KNN can suffer from skewed class distributions. For example, if a certain class is very frequent in the training set, it will tend to dominate the majority voting of the new example (large number = more common). Finally, the accuracy of KNN can be severely degraded with high-dimension data because there is little difference between the nearest and farthest neighbor.

**Improvements**

With that being said, there are many ways in which the KNN algorithm can be improved.

* A simple and effective way to remedy skewed class distributions is by implementing **weighed voting**. The class of each of the K neighbors is multiplied by a weight proportional to the inverse of the distance from that point to the given test point. This ensures that nearer neighbors contribute more to the final vote than the more distant ones.
* **Changing the distance metric** for different applications may help improve the accuracy of the algorithm. (i.e. Hamming distance for text classification)
* **Rescaling your data** makes the distance metric more meaningful. For instance, given 2 features height and weight, an observation such as x=[180,70]x=[180,70] will clearly skew the distance metric in favor of height. One way of fixing this is by column-wise subtracting the mean and dividing by the standard deviation. Scikit-learn’s normalize() method can come in handy.
* **Dimensionality reduction** techniques like PCA should be executed prior to applying KNN and help make the distance metric more meaningful.
* **Approximate Nearest Neighbor** techniques such as using k-d trees to store the training observations can be leveraged to decrease testing time. Note however that these methods tend to perform poorly in high dimensions (20+). Try using **locality sensitive hashing (LHS)** for higher dimensions.

**Logistic regression:**



In statistics, the logistic model (or logit model) is a widely used statistical model that, in its basic form, uses a logistic function to model a binary dependent variable; many more complex extensions exist. In regression analysis, logistic regression (or logit regression) is estimating the parameters of a logistic model; it is a form of binomial regression. Mathematically, a binary logistic model has a dependent variable with two possible values, such as pass/fail, win/lose, alive/dead or healthy/sick; these are represented by an indicator variable, where the two values are labeled "0" and "1". In the logistic model, the log-odds (the logarithm of the odds) for the value labeled "1" is a linear combination of one or more independent variables ("predictors"); the independent variables can each be a binary variable (two classes, coded by an indicator variable) or a continuous variable (any real value).

The corresponding probability of the value labeled "1" can vary between 0 (certainly the value "0") and 1 (certainly the value "1"), hence the labeling; the function that converts log-odds to probability is the logistic function, hence the name. The unit of measurement for the log-odds scale is called a *logit*, from *logistic unit*, hence the alternative names. Analogous models with a different sigmoid function instead of the logistic function can also be used, such as the probit model; the defining characteristic of the logistic model is that increasing one of the independent variables multiplicatively scales the odds of the given outcome at a *constant* rate, with each dependent variable having its own parameter; for a binary independent variable this generalizes the odds ratio.

Logistic regression was developed by statistician David Cox in 1958. The binary logistic regression model has extensions to more than two levels of the dependent variable: categorical outputs with more than two values are modeled by multinomial logistic regression, and if the multiple categories are ordered, by ordinal logistic regression, for example the proportional odds ordinal logistic model. The model itself simply models probability of output in terms of input, and does not perform statistical classification (it is not a classifier), though it can be used to make a classifier, for instance by choosing a cutoff value and classifying inputs with probability greater than the cutoff as one class, below the cutoff as the other; this is a common way to make a binary classifier. The coefficients are generally not computed by a closed-form expression, unlike linear least squares.

**Applications**

Logistic regression is used in various fields, including machine learning, most medical fields, and social sciences. For example, the Trauma and Injury Severity Score (TRISS), which is widely used to predict mortality in injured patients, was originally developed by Boyd et al. using logistic regression. Many other medical scales used to assess severity of a patient have been developed using logistic regression. Logistic regression may be used to predict the risk of developing a given disease (e.g. diabetes; coronary heart disease), based on observed characteristics of the patient (age, sex, body mass index, results of various blood tests, etc.).Another example might be to predict whether an Indian voter will vote BJP or Trinamool Congress or Left Front or Congress, based on age, income, sex, race, state of residence, votes in previous elections, etc.The technique can also be used in engineering, especially for predicting the probability of failure of a given process, system or product. It is also used in marketing applications such as prediction of a customer's propensity to purchase a product or halt a subscription, etc.In economics it can be used to predict the likelihood of a person's choosing to be in the labor force, and a business application would be to predict the likelihood of a homeowner defaulting on a mortgage. Conditional random fields, an extension of logistic regression to sequential data, are used in natural language processing.

Logistic regression measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic function, which is the cumulative logistic distribution. Thus, it treats the same set of problems as probit regression using similar techniques, with the latter using a cumulative normal distribution curve instead. Equivalently, in the latent variable interpretations of these two methods, the first assumes a standard logistic distribution of errors and the second a standard normal distribution of errors.

Logistic regression can be seen as a special case of the generalized linear model and thus analogous to linear regression. The model of logistic regression, however, is based on quite different assumptions (about the relationship between dependent and independent variables) from those of linear regression. In particular the key differences between these two models can be seen in the following two features of logistic regression. First, the conditional distribution {\displaystyle y\mid x}is a Bernoulli distribution rather than a Gaussian distribution, because the dependent variable is binary. Second, the predicted values are probabilities and are therefore restricted to (0, 1) through the logistic distribution function because logistic regression predicts the **probability** of particular outcomes rather than the outcomes themselves.

Logistic regression is an alternative to Fisher's 1936 method, linear discriminant analysis. If the assumptions of linear discriminant analysis hold, the conditioning can be reversed to produce logistic regression. The converse is not true, however, because logistic regression does not require the multivariate normal assumption of discriminant analysis.

Logistic Regression was used in the biological sciences in early twentieth century. It was then used in many social science applications. Logistic Regression is used when the dependent variable(target) is categorical.

For example,

* To predict whether an email is spam (1) or (0)
* Whether the tumor is malignant (1) or not (0)

Consider a scenario where we need to classify whether an email is spam or not. If we use linear regression for this problem, there is a need for setting up a threshold based on which classification can be done. Say if the actual class is malignant, predicted continuous value 0.4 and the threshold value is 0.5, the data point will be classified as not malignant which can lead to serious consequence in real time.

From this example, it can be inferred that linear regression is not suitable for classification problem. Linear regression is unbounded, and this brings logistic regression into picture. Their value strictly ranges from 0 to 1.

**Simple Logistic Regression**

**Model**

Output = 0 or 1

Hypothesis => Z = WX + B

hΘ(x) = sigmoid (Z)

**Sigmoid Function**

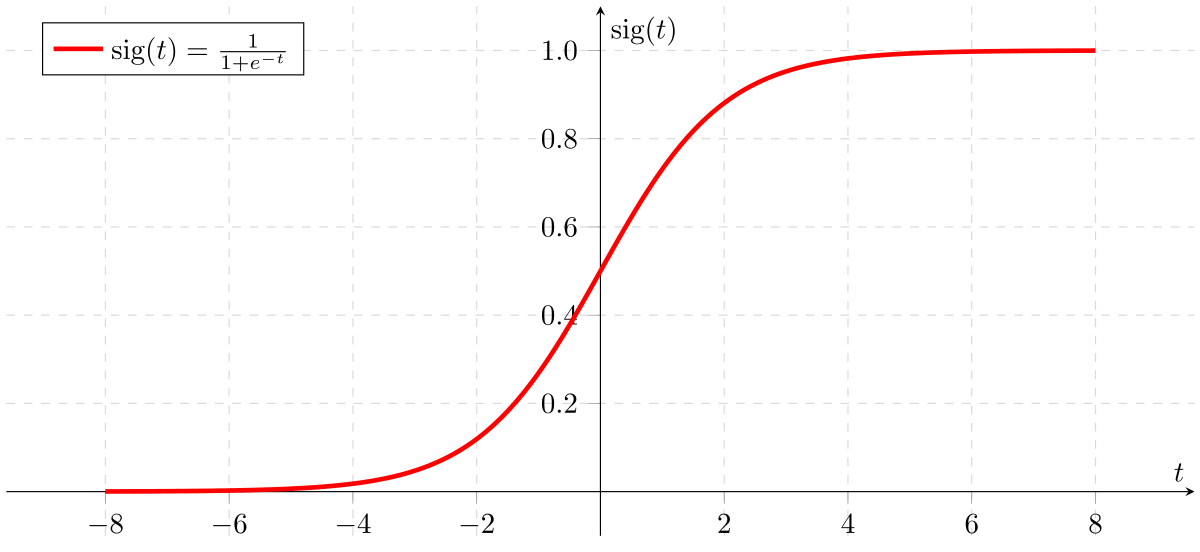


Figure 2: Sigmoid Activation Function

If ‘Z’ goes to infinity, Y(predicted) will become 1 and if ‘Z’ goes to negative infinity, Y(predicted) will become 0.

**Analysis of the hypothesis**

The output from the hypothesis is the estimated probability. This is used to infer how confident can predicted value be actual value when given an input X. Consider the below example,

X = [x0 x1] = [1 IP-Address]

Based on the x1 value, let’s say we obtained the estimated probability to be 0.8. This tells that there is 80% chance that an email will be spam.

Mathematically this can be written as,

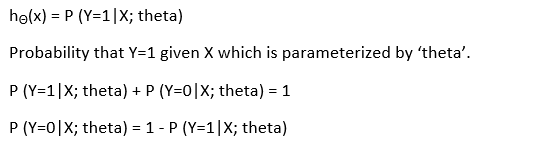


Figure 3: Mathematical Representation

This justifies the name ‘logistic regression’. Data is fit into linear regression model, which then be acted upon by a logistic function predicting the target categorical dependent variable.

**Types of Logistic Regression**

1. Binary Logistic Regression

The categorical response has only two 2 possible outcomes. Example: Spam or Not

2. Multinomial Logistic Regression

Three or more categories without ordering. Example: Predicting which food is preferred more (Veg, Non-Veg, Vegan)

3. Ordinal Logistic Regression

Three or more categories with ordering. Example: Movie rating from 1 to 5

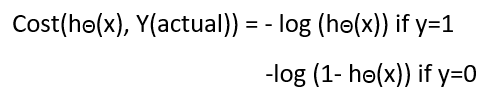
**Decision Boundary**

To predict which class a data belongs, a threshold can be set. Based upon this threshold, the obtained estimated probability is classified into classes.

Say, if predicted\_value ≥ 0.5, then classify email as spam else as not spam.

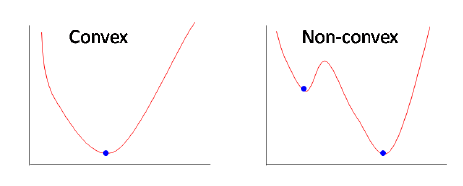
Decision boundary can be linear or non-linear. Polynomial order can be increased to get complex decision boundary.

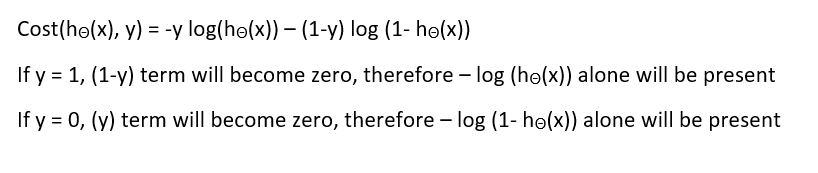
**Cost Function**



Why cost function which has been used for linear can not be used for logistic?

Linear regression uses mean squared error as its cost function. If this is used for logistic regression, then it will be a non-convex function of parameters (theta). Gradient descent will converge into global minimum only if the function is convex.



**Simplified cost function**

**Why this cost function?**

This negative function is because when we train, we need to maximize the probability by minimizing loss function.

Decreasing the cost will increase the maximum likelihood assuming that samples are drawn from an identically independent distribution.

**Pros and Cons:**

**Pros:**

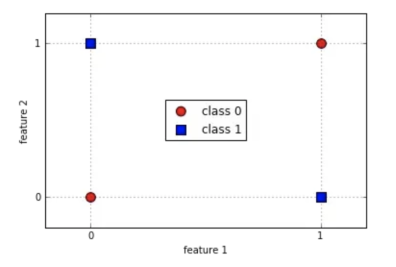
It is a widely used technique because it is very efficient, does not require too many computational resources, it’s highly interpretable, it doesn’t require input features to be scaled, it doesn’t require any tuning, it’s easy to regularize, and it outputs well-calibrated predicted probabilities.

Like linear regression, logistic regression does work better when you remove attributes that are unrelated to the output variable as well as attributes that are very similar (correlated) to each other. Therefore Feature Engineering plays an important role in regards to the performance of Logistic and also Linear Regression. Another advantage of Logistic Regression is that it is incredibly easy to implement and very efficient to train. I typically start with a Logistic Regression model as a benchmark and try using more complex algorithms from there on.

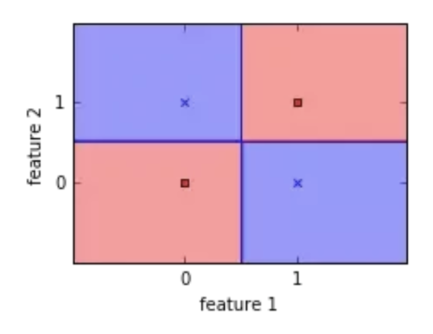
Because of its simplicity and the fact that it can be implemented relatively easy and quick, Logistic Regression is also a good baseline that you can use to measure the performance of other more complex Algorithms.

**Cons:**

A disadvantage of it is that we can’t solve non-linear problems with logistic regression since it’s decision surface is linear. Just take a look at the example below that has 2 binary features from 2 examples.



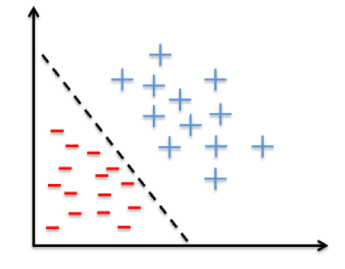
It is clearly visible that we can’t draw a line that separates these 2 classes without a huge error. To use a simple decision tree would be a much better choice.



Logistic Regression is also not one of the most powerful algorithms out there and can be easily outperformed by more complex ones. Another disadvantage is its high reliance on a proper presentation of your data. This means that logistic regression is not a useful tool unless you have already identified all the important independent variables. Since its outcome is discrete, Logistic Regression can only predict a categorical outcome. It is also an Algorithm that is known for its vulnerability to overfitting.

**When to use it?**

Logistic Regression separates your input into two “regions” by a linear boundary, one for each class. Therefore it is required that your data is linearly separable, like the data points in the image below:



In other words: You should think about using logistic regression when your Y variable takes on only two values (e.g when you are facing a classification problem). Note that you could also use Logistic Regression for multiclass classification, which will be discussed in the next section.

Multiclass Classification

Out there are algorithms that can deal by themselves with predicting multiple classes, like Random Forest classifiers or the Naive Bayes Classifier. There are also algorithms that can’t do that, like Logistic Regression, but with some tricks, you can predict multiple classes with it too.

Let’s discuss the most common of these “tricks” at the example of the MNIST Dataset, which contains handwritten images of digits, ranging from 0 to 9. This is a classification task where our Algorithm should tell us which number is on an image.

1) One-versus-All (OvA)

With this strategy, you train 10 binary classifiers, one for each number. This simply means training one classifier to detect 0s, one to detect 1s, one to detect 2s and so on. When you then want to classify an image, you just look at which classifier has the best decision score

2) One-versus-One (OvO)

Here you train a binary classifier for every pair of digits. This means training a classifier that can distinguish between 0s and 1s, one that can distinguish between 0s and 2s, one that can distinguish between 1s and 2s etc. If there are N classes, you would need to train NxN(N-1)/2 classifiers, which are 45 in the case of the MNIST dataset.

When you then want to classify images, you need to run each of these 45 classifiers and choose the best performing one. This strategy has one big advantage over the others and this is, that you only need to train it on a part of the training set for the 2 classes it distinguishes between. Algorithms like Support Vector Machine Classifiers don’t scale well at large datasets, which is why in this case using a binary classification algorithm like Logistic Regression with the OvO strategy would do better, because it is faster to train a lot of classifiers on a small dataset than training just one at a large dataset.

At most algorithms, sklearn recognizes when you use a binary classifier for a multiclass classification task and automatically uses the OvA strategy. There is an exception: When you try to use a Support Vector Machine classifier, it automatically runs the OvO strategy.

**Naïve Bayes Classifier**

Naive Bayes classifier is a straightforward and powerful algorithm for the [**classification**](https://dataaspirant.com/2016/09/24/classification-clustering-alogrithms/) task. Even if we are working on a data set with millions of records with some attributes, it is suggested to try Naive Bayes approach.

Naive Bayes classifier gives great results when we use it for textual data analysis. Such as Natural Language Processing.

To understand the naive Bayes classifier we need to understand the Bayes theorem. So let’s first discuss the Bayes Theorem.

**What is Bayes Theorem?**

Bayes theorem named after Rev. Thomas Bayes. It works on conditional**probability**. Conditional probability is the probability that something will happen, ***given that something else* has already occurred**. Using the conditional probability, we can calculate the probability of an event using its prior knowledge.

Below is the formula for calculating the conditional probability.

\textrm{P(H \textbar E) = }  \frac{\textrm{ P(E \textbar H) * P(H)}} {\textrm{P(E)}}

**where**

* P(H) is the probability of hypothesis H being true. This is known as the prior probability.
* P(E) is the probability of the evidence(regardless of the hypothesis).
* P(E|H) is the probability of the evidence given that hypothesis is true.
* P(H|E) is the probability of the hypothesis given that the evidence is there.

Let’s consider an example to understand how the above formula of Bayes theorem works.

**Problem:**

A Path Lab is performing a Test of disease say “D” with two results “Positive” & “Negative.” They guarantee that their test result is 99% accurate: if you have the disease, they will give test positive 99% of the time. If you don’t have the disease, they will test negative 99% of the time. If 3% of all the people have this disease and test gives “positive” result, **what is the probability that you actually have the disease?**

For solving the above problem, we will have to use conditional probability.  
Probability of people suffering from Disease D, P(D) = 0.03 = 3%  
Probability that test gives “positive” result and patient have the disease, P(Pos | D) = 0.99 =99%

Probability of people not suffering from Disease D, P(~D) = 0.97 = 97%  
Probability that test gives “positive” result and patient does have the disease, P(Pos | ~D) = 0.01 =1%

For calculating the probability that the patient actually have the disease i.e, **P( D | Pos)** we will use Bayes theorem:

\textrm{P(D \textbar Pos) = }  \frac{\textrm{ P(Pos \textbar D) * P(D)}} {\textrm{P(Pos)}}

We have all the values of numerator but we need to calculate P(Pos):  
**P(Pos)** = P(D, pos) + P( ~D, pos)  
= P(pos|D)\*P(D) + P(pos|~D)\*P(~D)  
= 0.99 \* 0.03 + 0.01 \* 0.97  
= 0.0297 + 0.0097  
= 0.0394

Let’s calculate, **P( D | Pos) = (P(Pos | D) \* P(D)) / P(Pos)**  
= (0.99 \* 0.03) / 0.0394  
= 0.753807107

So, Approximately **75%** chances are there that the patient is actually suffering from disease.

Naive Bayes is a kind of classifier which uses the Bayes Theorem. It predicts membership probabilities for each class such as the probability that given record or data point belongs to a particular class.  The class with the highest probability is considered as the most likely class. This is also known as **Maximum A Posteriori (MAP)**.

***The MAP for a hypothesis is:***

**MAP(H)**   
= max( P(H|E) )  
=  max( (P(E|H)\*P(H))/P(E))  
= max(P(E|H)\*P(H))

P(E) is evidence probability, and it is used to normalize the result. It remains same so, removing it won’t affect.

Naive Bayes classifier assumes that all the features are**unrelated** to each other. Presence or absence of a feature does not influence the presence or absence of any other feature. We can use Wikipedia example for explaining the logic i.e.,

A fruit may be considered to be an apple if it is red, round, and about 4″ in diameter.  Even if these features depend on each other or upon the existence of the other features, a naive Bayes classifier considers all of these properties to independently contribute to the probability that this fruit is an apple.

In real datasets, we test a hypothesis given multiple evidence(feature). So, calculations become complicated. To simplify the work, the feature independence approach is used to ‘uncouple’ multiple evidence and treat each as an independent one.

**P(H|Multiple Evidences)** =  P(E1| H)\* P(E2|H) ……\*P(En|H) \* P(H) / P(Multiple Evidences)

**Example of Naive Bayes Classifier**

For understanding a theoretical concept, the best procedure is to try it on an example.

Naive Bayes classification features

Let’s consider a training dataset with 1500 records and 3 classes. We presume that there are no missing values in our data. We have

We have 3 classes associated with Animal Types:

* Parrot,
* Dog,
* Fish.

The Predictor features set consists of 4 features as follows:

* Swim
* Wings
* Green Color
* Dangerous Teeth.

Green Color, Dangerous Teeth.  All the features are categorical variables with either of the 2 values: **T (True) or F ( False)**.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Swim** | **Wings** | **Green Color** | **Dangerous Teeth** | **Animal Type** |
| 50 | 500/500 | 400/500 | 0 | Parrot |
| 450/500 | 0 | 0 | 500/500 | Dog |
| 500/500 | 0 | 100/500 | 50/500 | Fish |

The above table shows a frequency table of our data. In our training data:

* Parrots have 50(10%) value for Swim, i.e., 10% parrot can swim according to our data, 500 out of 500(100%) parrots have wings, 400 out of 500(80%) parrots are Green and 0(0%) parrots have Dangerous Teeth.
* Classes with Animal type Dogs shows that 450 out of 500(90%) can swim, 0(0%) dogs have wings, 0(0%) dogs are of Green color and 500 out of 500(100%) dogs have Dangerous Teeth.
* Classes with Animal type Fishes shows that 500 out of 500(100%) can swim, 0(0%) fishes have wings, 100(20%) fishes are of Green color and 50 out of 500(10%) dogs have Dangerous Teeth.

Now, it’s time to work on predict classes using the Naive Bayes model. We have taken 2 records that have values in their feature set, but the target variable needs to predicted.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Swim** | **Wings** | **Green** | **Teeth** |
| 1. | True | False | True | False |
| 2. | True | False | True | True |

We have to predict animal type using the feature values. We have to predict whether the animal is a Dog, a Parrot or a Fish

We will use the Naive Bayes approach  
**P(H|Multiple Evidences)** =  P(E1| H)\* P(E2|H) ……\*P(En|H) \* P(H) / P(Multiple Evidences)

Let’s consider the first record.  
The Evidence here is **Swim & Green**. The Hypothesis can be an animal type to be Dog, Parrot, Fish.

***For Hypothesis testing for the animal to be a Dog:***

P(Dog | Swim, Green) = P(Swim|Dog) \* P(Green|Dog) \* P(Dog) / P(Swim, Green)  
=  0.9 \* 0 \* 0.333 / P(Swim, Green)  
= 0

***For Hypothesis testing for the animal to be a Parrot:***

P(Parrot| Swim, Green) = P(Swim|Parrot) \* P(Green|Parrot) \* P(Parrot) / P(Swim, Green)  
=  0.1 \* 0.80 \* 0.333 / P(Swim, Green)  
= 0.0264/ P(Swim, Green)

***For Hypothesis testing for the animal to be a Fish:***

P(Fish| Swim, Green) = P(Swim|Fish) \* P(Green|Fish) \* P(Fish) / P(Swim, Green)  
=  1 \* 0.2 \* 0.333 / P(Swim, Green)  
= 0.0666/ P(Swim, Green)

The denominator of all the above calculations is same i.e, P(Swim, Green). The value of P(Fish| Swim, Green) is greater that P(Parrot| Swim, Green).

Using Naive Bayes, we can predict that the class of this record is Fish.

Let’s consider the second record.  
The Evidence here is Swim, Green & Teeth. The Hypothesis can be an animal type to be Dog, Parrot, Fish.

***For Hypothesis testing for the animal to be a Dog:***

P(Dog | Swim, Green, Teeth) = P(Swim|Dog) \* P(Green|Dog) \* P(Teeth|Dog) \* P(Dog) / P(Swim, Green, Teeth)  
=  0.9 \* 0 \* 1 \* 0.333 / P(Swim, Green, Teeth)  
= 0

***For Hypothesis testing for the animal to be a Parrot:***

P(Parrot| Swim, Green, Teeth) = P(Swim|Parrot) \* P(Green|Parrot)\* P(Teeth|Parrot) \* P(Parrot) / P(Swim, Green, Teeth)  
=  0.1 \* 0.80 \*  0 \*0.333 / P(Swim, Green, Teeth)  
= 0

***For Hypothesis testing for the animal to be a Fish:***

P(Fish|Swim, Green, Teeth) = P(Swim|Fish) \* P(Green|Fish) \* P(Teeth|Fish) \*P(Fish) / P(Swim, Green, Teeth)  
=  1 \* 0.2 \* 0.1 \* 0.333 / P(Swim, Green, Teeth)  
= 0.00666 / P(Swim, Green, Teeth)

The denominator of all the above calculations is same i.e, P(Swim, Green, Teeth). The value of P(Fish| Swim, Green, Teeth) is the only positive value greater than 0. Using Naive Bayes, we can predict that the class of this record is Fish.

As the calculated value of probabilities is very less. To normalize these values, we need to use denominators.

Let’s proceed to learn the various type of **Naive Bayes Methods**.

**Types of Naive Bayes Classifier Algorithms**

**Gaussian Naive Bayes**

When attribute values are continuous, an assumption is made that the values associated with each class are distributed according to Gaussian i.e., Normal Distribution.

If in our data, an attribute say “x” contains continuous data. We first segment the data by the class and then compute mean \mu_{y}  & Variance {\sigma_{y}}^{2}  of each class.  
P(x_i \mid y) &= \frac{1}{\sqrt{2\pi\sigma^2_y}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma^2_y}\right)

**MultiNomial Naive Bayes**

MultiNomial Naive Bayes is preferred to use on data that is multinomially distributed. It is one of the standard classic algorithms. Which is used in text categorization (classification). Each event in text classification represents the occurrence of a word in a document.

**Bernoulli Naive Bayes**

Bernoulli Naive Bayes is used on the data that is distributed according to multivariate Bernoulli distributions.i.e., multiple features can be there, but each one is assumed to be a binary-valued (Bernoulli, boolean) variable. So, it requires features to be binary valued.

**Pros and Cons of Naïve Bayes:**

**Pros:**

* Naive Bayes Algorithm is a fast, highly scalable algorithm.
* Naive Bayes can be use for Binary and Multiclass classification. It provides different types of Naive Bayes Algorithms like GaussianNB, MultinomialNB, BernoulliNB.
* It is a simple algorithm that depends on doing a bunch of counts.
* Great choice for Text Classification problems. It’s a popular choice for spam email classification.
* It can be easily train on small dataset

**Cons:**

* It considers all the features to be unrelated, so it cannot learn the relationship between features. E.g., Let’s say Remo is going to a part. While cloth selection for the party, Remo is looking at his cupboard. Remo likes to wear a white color shirt. In Jeans, he likes to wear a brown Jeans, But Remo doesn’t like wearing a white shirt with Brown Jeans. Naive Bayes can learn individual features importance but can’t determine the relationship among features.

**Decision Tree**

A decision tree is a [flowchart](https://en.wikipedia.org/wiki/Flowchart)-like structure in which each internal node represents a "test" on an attribute (e.g. whether a coin flip comes up heads or tails), each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). The paths from root to leaf represent classification rules.

In [decision analysis](https://en.wikipedia.org/wiki/Decision_analysis), a decision tree and the closely related [influence diagram](https://en.wikipedia.org/wiki/Influence_diagram) are used as a visual and analytical decision support tool, where the [expected values](https://en.wikipedia.org/wiki/Expected_value) (or [expected utility](https://en.wikipedia.org/wiki/Expected_utility)) of competing alternatives are calculated.

A decision tree consists of three types of nodes:[[1]](https://en.wikipedia.org/wiki/Decision_tree" \l "cite_note-1)

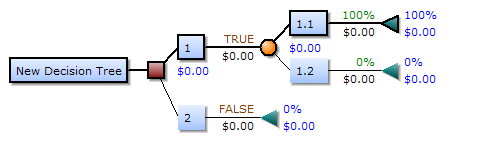
1. Decision nodes – typically represented by squares
2. Chance nodes – typically represented by circles
3. End nodes – typically represented by triangles

Decision trees are commonly used in [operations research](https://en.wikipedia.org/wiki/Operations_research) and [operations management](https://en.wikipedia.org/wiki/Operations_management). If, in practice, decisions have to be taken online with no recall under incomplete knowledge, a decision tree should be paralleled by a [probability](https://en.wikipedia.org/wiki/Probability) model as a best choice model or online selection model [algorithm](https://en.wikipedia.org/wiki/Algorithm). Another use of decision trees is as a descriptive means for calculating [conditional probabilities](https://en.wikipedia.org/wiki/Conditional_probability).

Decision trees, [influence diagrams](https://en.wikipedia.org/wiki/Influence_diagrams), [utility functions](https://en.wikipedia.org/wiki/Utility_function), and other [decision analysis](https://en.wikipedia.org/wiki/Decision_analysis) tools and methods are taught to undergraduate students in schools of business, health economics, and public health, and are examples of operations research or [management science](https://en.wikipedia.org/wiki/Management_science) methods.

**Decision tree building block**

**Decision tree elements**

[](https://en.wikipedia.org/wiki/File:Decision-Tree-Elements.png)

Drawn from left to right, a decision tree has only burst nodes (splitting paths) but no sink nodes (converging paths). Therefore, used manually, they can grow very big and are then often hard to draw fully by hand. Traditionally, decision trees have been created manually — as the aside example shows — although increasingly, specialized software is employed.

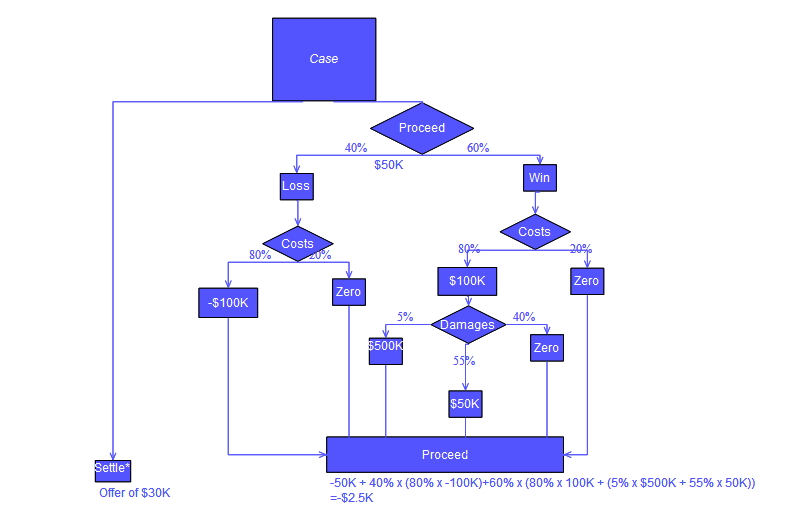
**Decision rule**

The decision tree can be [linearized](https://en.wikipedia.org/wiki/Linearization" \o "Linearization) into **decision rules**,[[2]](https://en.wikipedia.org/wiki/Decision_tree#cite_note-2) where the outcome is the contents of the leaf node, and the conditions along the path form a conjunction in the if clause. In general, the rules have the form:

*if* condition1 *and* condition2 *and* condition3 *then* outcome.

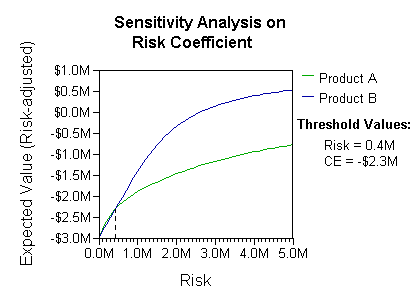
Decision rules can be generated by constructing [association rules](https://en.wikipedia.org/wiki/Association_rule_learning) with the target variable on the right. They can also denote [temporal](https://en.wikipedia.org/wiki/Time) or [causal](https://en.wikipedia.org/wiki/Causal) relations.[[3]](https://en.wikipedia.org/wiki/Decision_tree#cite_note-3)

**Decision tree using flowchart symbols**

Commonly a decision tree is drawn using [flowchart](https://en.wikipedia.org/wiki/Flowchart) symbols as it is easier for many to read and understand. [](https://en.wikipedia.org/wiki/File:DecisionCalcs.jpg)

**Analysis example**

Analysis can take into account the decision maker's (e.g., the company's) [preference](https://en.wikipedia.org/wiki/Preference) or [utility function](https://en.wikipedia.org/wiki/Utility_function), for example:

[](https://en.wikipedia.org/wiki/File:RiskPrefSensitivity2Threshold.png)

The basic interpretation in this situation is that the company prefers B's risk and payoffs under realistic risk preference coefficients (greater than $400K—in that range of risk aversion, the company would need to model a third strategy, "Neither A nor B").

Another example, commonly used in [operations research](https://en.wikipedia.org/wiki/Operations_research) courses, is the distribution of lifeguards on beaches (a.k.a. the "Life's a Beach" example).[[4]](https://en.wikipedia.org/wiki/Decision_tree#cite_note-4) The example describes two beaches with lifeguards to be distributed on each beach. There is maximum budget *B* that can be distributed among the two beaches (in total), and using a marginal returns table, analysts can decide how many lifeguards to allocate to each beach.

|  |  |  |
| --- | --- | --- |
| **Lifeguards on each beach** | **Drownings prevented in total, beach #1** | **Drownings prevented in total, beach #2** |
| 1 | 1 | 3 |
| 2 | 4 | 0 |

In this example, a decision tree can be drawn to illustrate the principles of [diminishing returns](https://en.wikipedia.org/wiki/Diminishing_returns) on beach.

[](https://en.wikipedia.org/wiki/File:Beachdecisiontree.png)

Beach Decision Tree

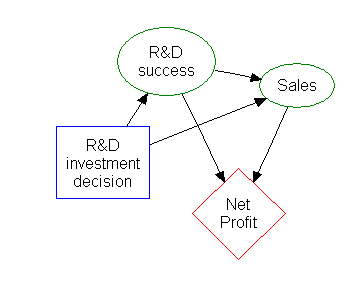
The decision tree illustrates that when sequentially distributing lifeguards, placing a first lifeguard on beach #1 would be optimal if there is only the budget for 1 lifeguard. But if there is a budget for two guards, then placing both on beach #2 would prevent more overall drownings.

[](https://en.wikipedia.org/wiki/File:Lifeguards.png)

Lifeguards

**Influence diagram**

Much of the information in a decision tree can be represented more compactly as an [influence diagram](https://en.wikipedia.org/wiki/Influence_diagram), focusing attention on the issues and relationships between events.

[](https://en.wikipedia.org/wiki/File:Factory2_InfluenceDiagram.png)

The rectangle on the left represents a decision, the ovals represent actions, and the diamond represents results.

**Pros and Cons:**

**Pros:**

Among decision support tools, decision trees (and [influence diagrams](https://en.wikipedia.org/wiki/Influence_diagrams)) have several advantages. Decision trees:

* Are simple to understand and interpret. People are able to understand decision tree models after a brief explanation.
* Have value even with little hard data. Important insights can be generated based on experts describing a situation (its alternatives, probabilities, and costs) and their preferences for outcomes.
* Help determine worst, best and expected values for different scenarios.
* Use a [white box](https://en.wikipedia.org/wiki/White_box_(software_engineering)) model. If a given result is provided by a model.
* Can be combined with other decision techniques.

**Cons:**

* They are unstable, meaning that a small change in the data can lead to a large change in the structure of the optimal decision tree.
* They are often relatively inaccurate. Many other predictors perform better with similar data. This can be remedied by replacing a single decision tree with a [random forest](https://en.wikipedia.org/wiki/Random_forest) of decision trees, but a random forest is not as easy to interpret as a single decision tree.
* For data including categorical variables with different number of levels, [information gain in decision trees](https://en.wikipedia.org/wiki/Information_gain_in_decision_trees) is biased in favor of those attributes with more levels.[[7]](https://en.wikipedia.org/wiki/Decision_tree#cite_note-7)
* Calculations can get very complex, particularly if many values are uncertain and/or if many outcomes are linked.

**4. Experimental set up and evaluation**

**Python:**

Python is an [interpreted](https://en.wikipedia.org/wiki/Interpreted_language" \o "Interpreted language), [high-level](https://en.wikipedia.org/wiki/High-level_programming_language), [general-purpose programming language](https://en.wikipedia.org/wiki/General-purpose_programming_language). Created by [Guido van Rossum](https://en.wikipedia.org/wiki/Guido_van_Rossum) and first released in 1991, Python has a design philosophy that emphasizes [code readability](https://en.wikipedia.org/wiki/Code_readability), notably using [significant whitespace](https://en.wikipedia.org/wiki/Significant_whitespace). It provides constructs that enable clear programming on both small and large scales. In July 2018, Van Rossum stepped down as the leader in the language community.

Python features a [dynamic type](https://en.wikipedia.org/wiki/Dynamic_type) system and automatic [memory management](https://en.wikipedia.org/wiki/Memory_management). It supports multiple [programming paradigms](https://en.wikipedia.org/wiki/Programming_paradigm), including [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming),[imperative](https://en.wikipedia.org/wiki/Imperative_programming), [functional](https://en.wikipedia.org/wiki/Functional_programming) and [procedural](https://en.wikipedia.org/wiki/Procedural_programming), and has a large and comprehensive[standard library](https://en.wikipedia.org/wiki/Standard_library).[[29]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-About-29)

Python interpreters are available for many [operating systems](https://en.wikipedia.org/wiki/Operating_system). [CPython](https://en.wikipedia.org/wiki/CPython" \o "CPython), the[reference implementation](https://en.wikipedia.org/wiki/Reference_implementation) of Python, is [open source](https://en.wikipedia.org/wiki/Open-source_software) software and has a community-based development model, as do nearly all of Python's other implementations. Python and CPython are managed by the non-profit [Python Software Foundation](https://en.wikipedia.org/wiki/Python_Software_Foundation).

Here, we have used Python 3.6 as it is the stable version as on now.

**Python Packages:**

They are simply directories, but with a twist. Each **package** in **Python** is a directory which MUST contain a special file called \_\_init\_\_.py . This file can be empty, and it indicates that the directory it contains is a **Python package**, so it can be imported the same way a **module** can be imported.   
Packages are a way of structuring Python’s module namespace by using "dotted module names". A.B stands for a submodule named B in a package named A. Two different packages like P1 and P2 can both have modules with the same name, let's say A, for example. The submodule A of the package P1 and the submodule A of the package P2 can be totally different.   
A package is imported like a "normal" module.

**4.1 Components to be installed:**

* 1. Python 3.6
  2. Python Packages installed

2.1. pandas- used to provide fast, flexible and expressive data structures to make working with “relational” or “labeled” data both easy and intuitive.

2.2. tkinter- used for developing Graphical User Interface (GUI).

2.3. sklearn- efficient for data mining and data analysis. It contains a number of algorithms which can be implemented directly when required.

2.4. matplotlib- contains modules which are used for plotting graphs.

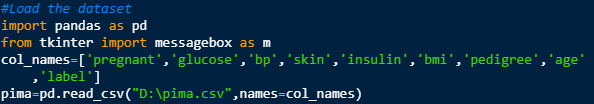
**4.2 Methodology**

**4.2.1 *Collect the dataset:***

At first the dataset was collected from the UCI Repository and was saved as csv (coma separated value) file.

**4.2.2 *Load the dataset:***

Then the dataset was loaded into the system through the code using read\_csv() present in ***pandas*** package.



*to load the dataset*

**4.2.3 *Describe the dataset:***

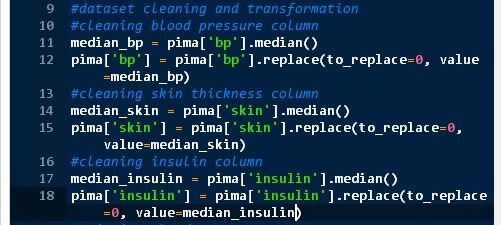
Then the dataset was described (i.e. the names were given to the columns for easy access) through the code.

12.png

*to describe the dataset*

**4.2.4 *Analysis of the dataset for cleaning purpose:***

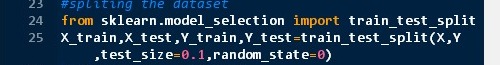
Then the analysis of the dataset was done. Some anomalies were found during this process which was mandatory to correct to achieve an accurate result. Thus, the mandatory columns (like blood pressure, skin thickness, insulin) which should not have 0 attribute were cleaned and replaced with the median value of the respective whole column.



*to clean the dataset*

**4.2.5 Split the dataset for training and testing:**

Then the dataset was split in ratio 7:3 for training and testing purpose respectively. The dataset was split through the code using train\_test\_split() which was present in model\_selection module of ***sklearn*** package.



*to split the dataset*

**4.2.6 Train the dataset using algorithms**

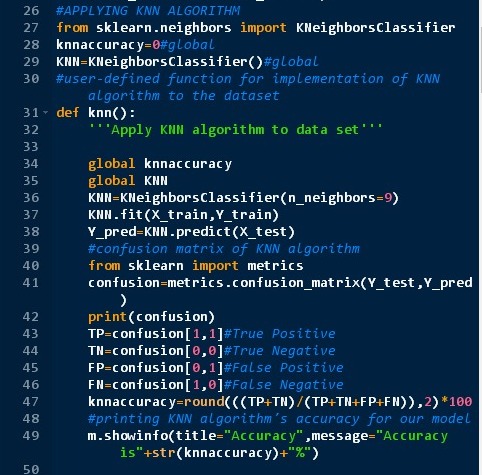
***Confusion matrix***

To check the adequacy of classiﬁers performance measures need to be taken into account. True positives (TP) refers to the positive cases that were correctly labeled by the classiﬁer, while true negatives (TN) are the negative cases that were correctly labeled by the classiﬁer. False positives (FP) are the negative cases that were incorrectly labeled, while false negatives (FN) are the positive cases that were incorrectly labeled . Evaluation measures of classiﬁers such as accuracy was calculated. The accuracy of the classiﬁer on a given test data set is the percentage of test cases that are correctly classiﬁed by classiﬁer.

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

* 1. **Using KNN Algorithm**

The system was thus trained using KNN Algorithm.



*Training the model using KNN Algorithm*

1. **Using Logistic Regression Algorithm**

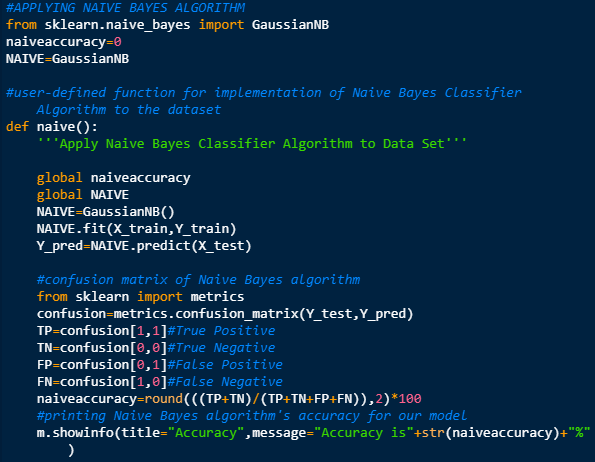
The system was then trained using Logistic Regression Algorithm.

****

*Training the model using Logistic Regression Algorithm*

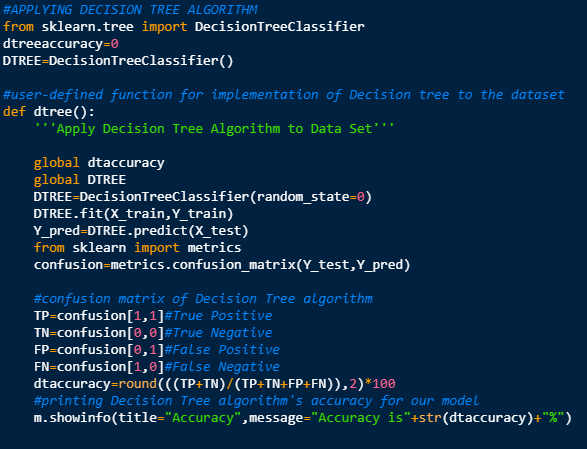
1. **Using Naïve Bayes Algorithm**

The system was then trained using Gaussian Naïve Bayes Algorithm.



*Training the model using Gaussian Naïve Bayes Algorithm*

1. **Using Decision Tree Algorithm**

The system was then trained using Decision Tree Algorithm. *Training the model using Decision Tree Algorithm*

**4.2.7 Compare the accuracy of all the Algorithm**

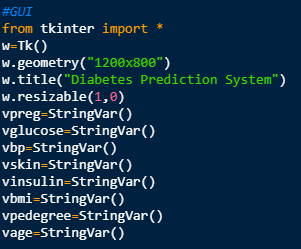
After finding out the accuracy of both the algorithms, they were compared so as to find the optimized algorithm. And accuracy of both the algorithms were displayed in graphical representation.



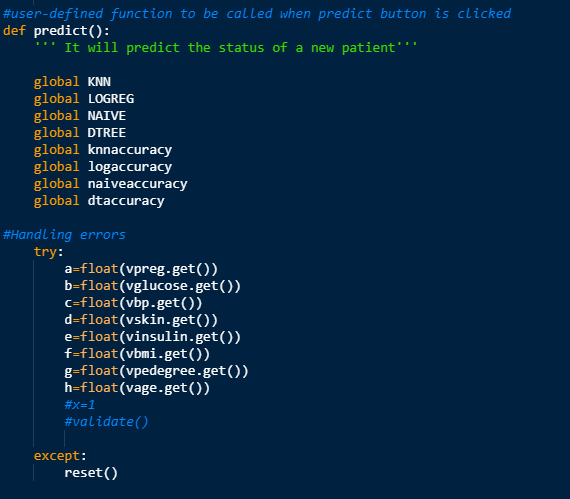
*To compare all the algorithm and display it as a graph*

**4.2.8 Create GUI for the model**

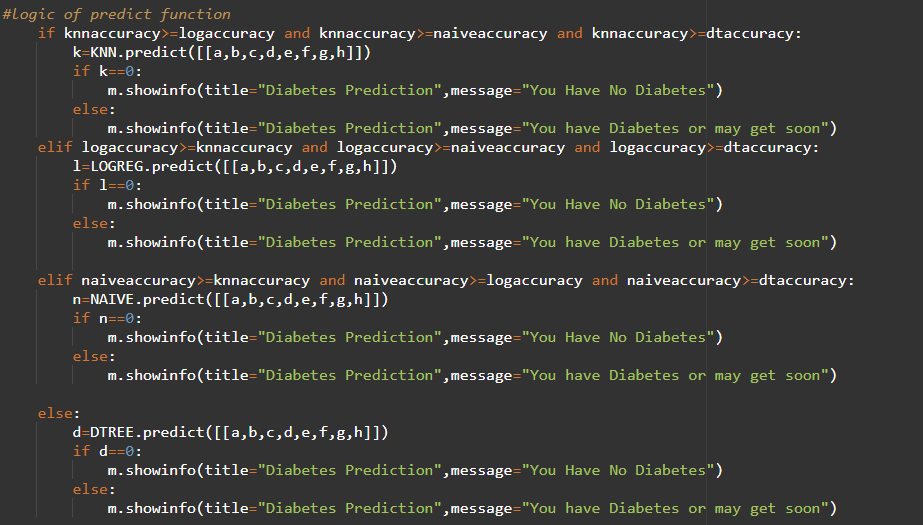
Then the Graphical User Interfaces for the whole model were model were created.



*To create space for the GUI*



*Function of Predict button*



*Logic of Predict button*



*Function of Reset button*



*Designing the page*



*Designing the page cont..*



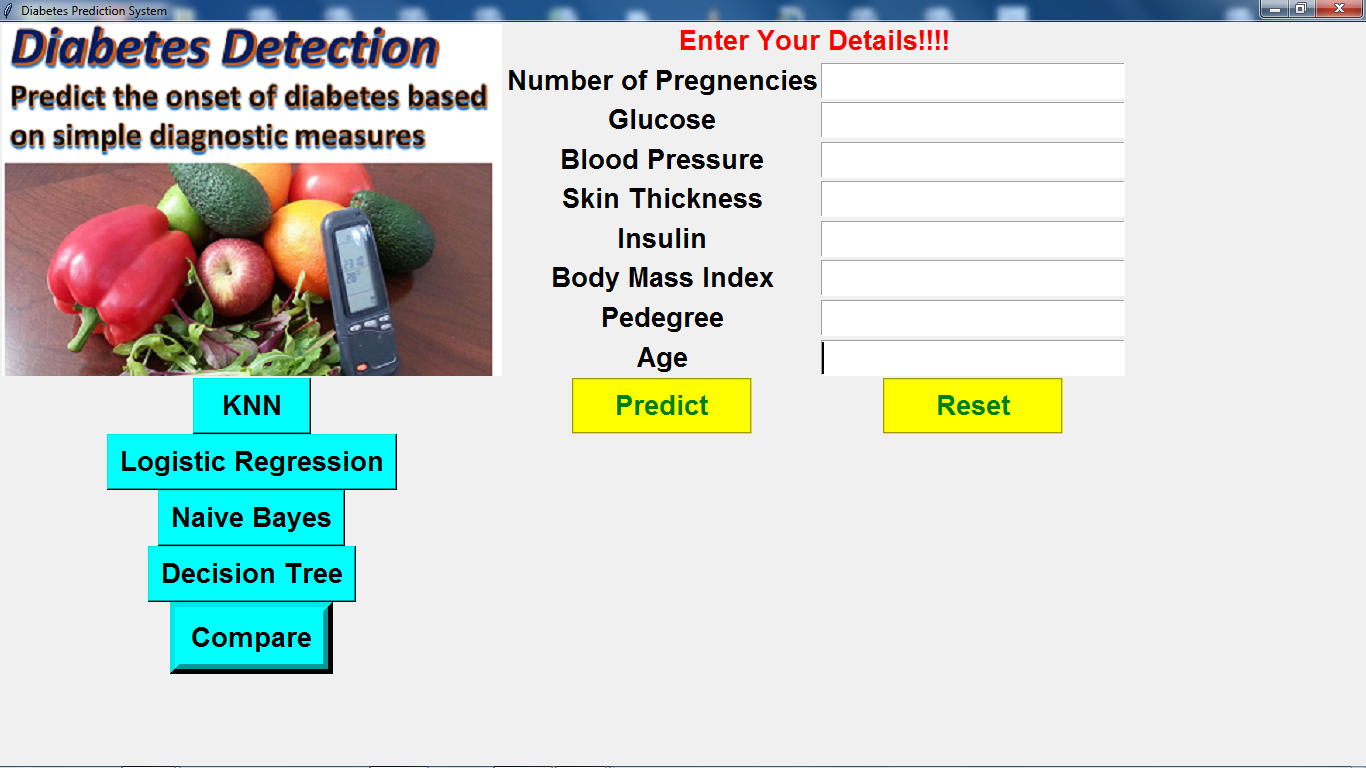
*Designing the page cont..*



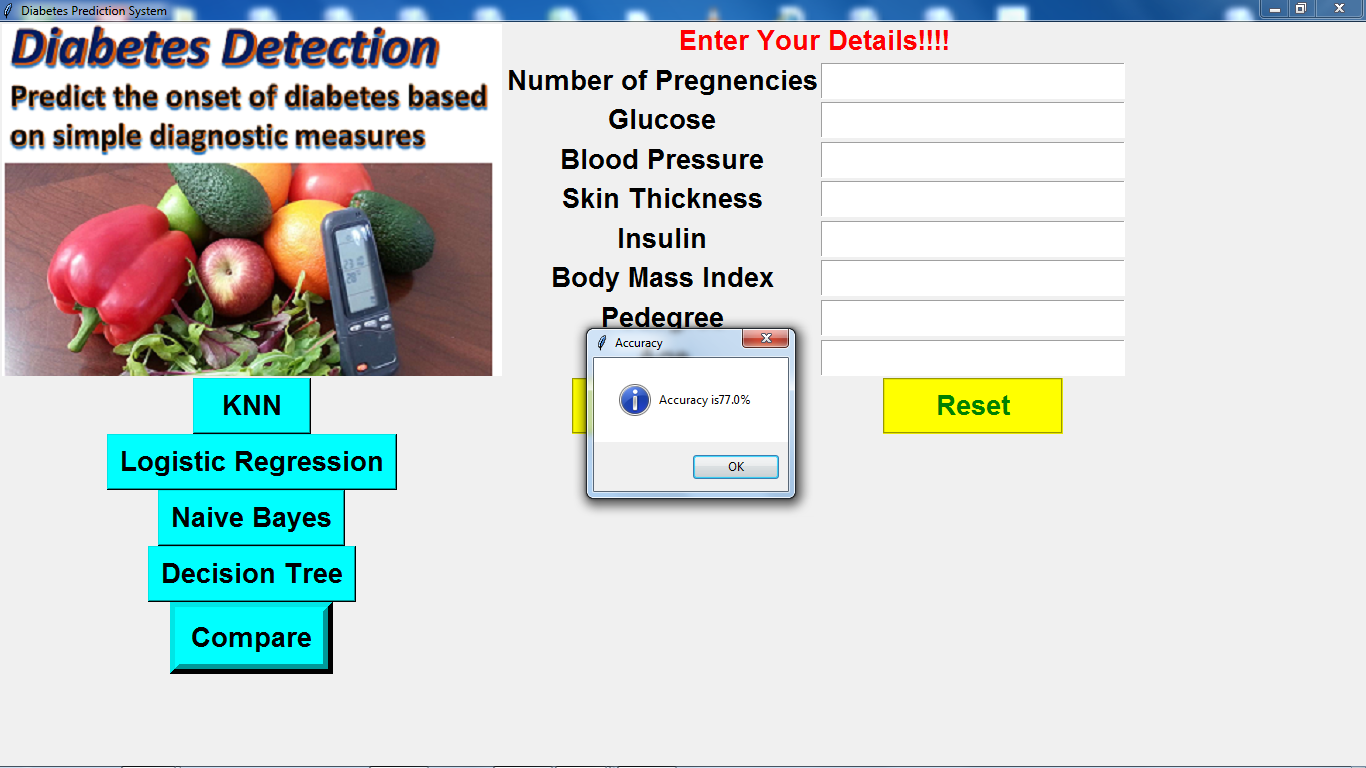
*Designing the page cont..*

**4.3 System Overview**

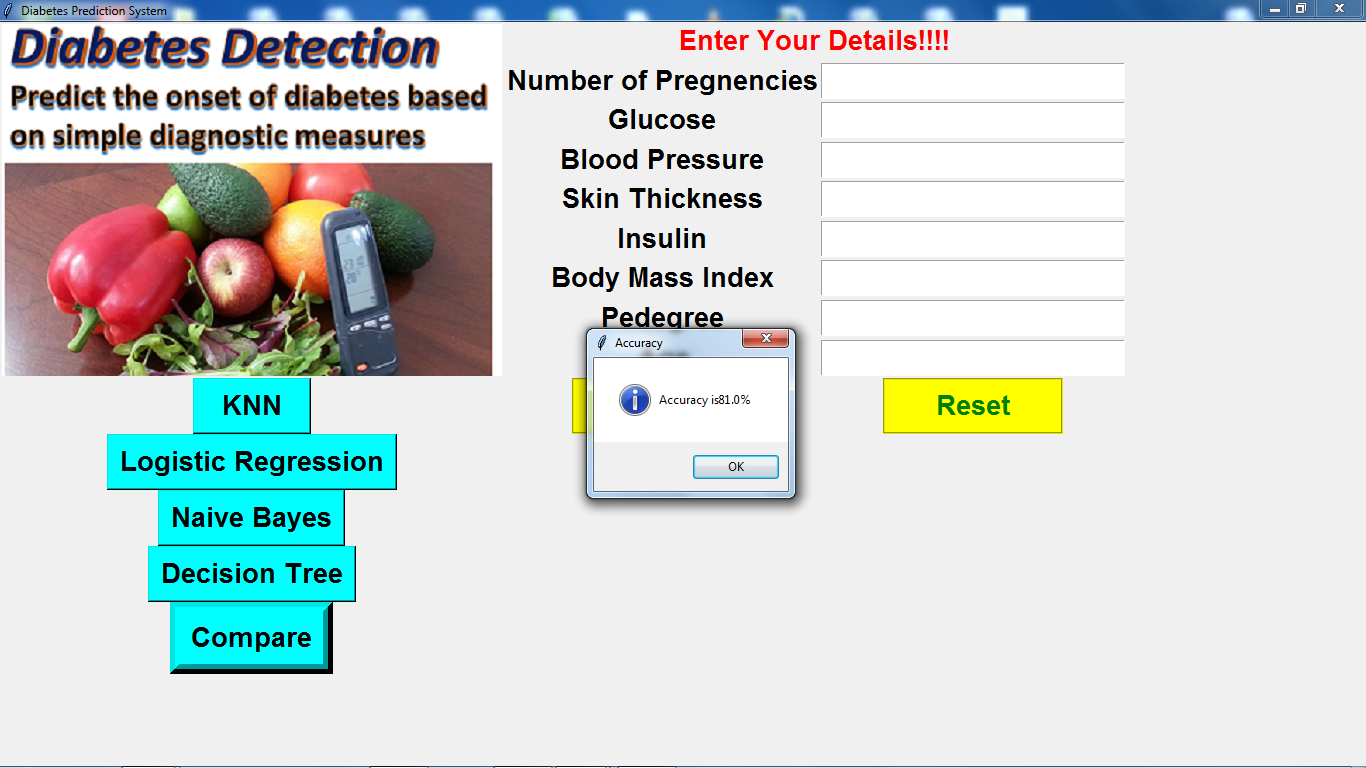
**4.3.1 Data Entry page**

****

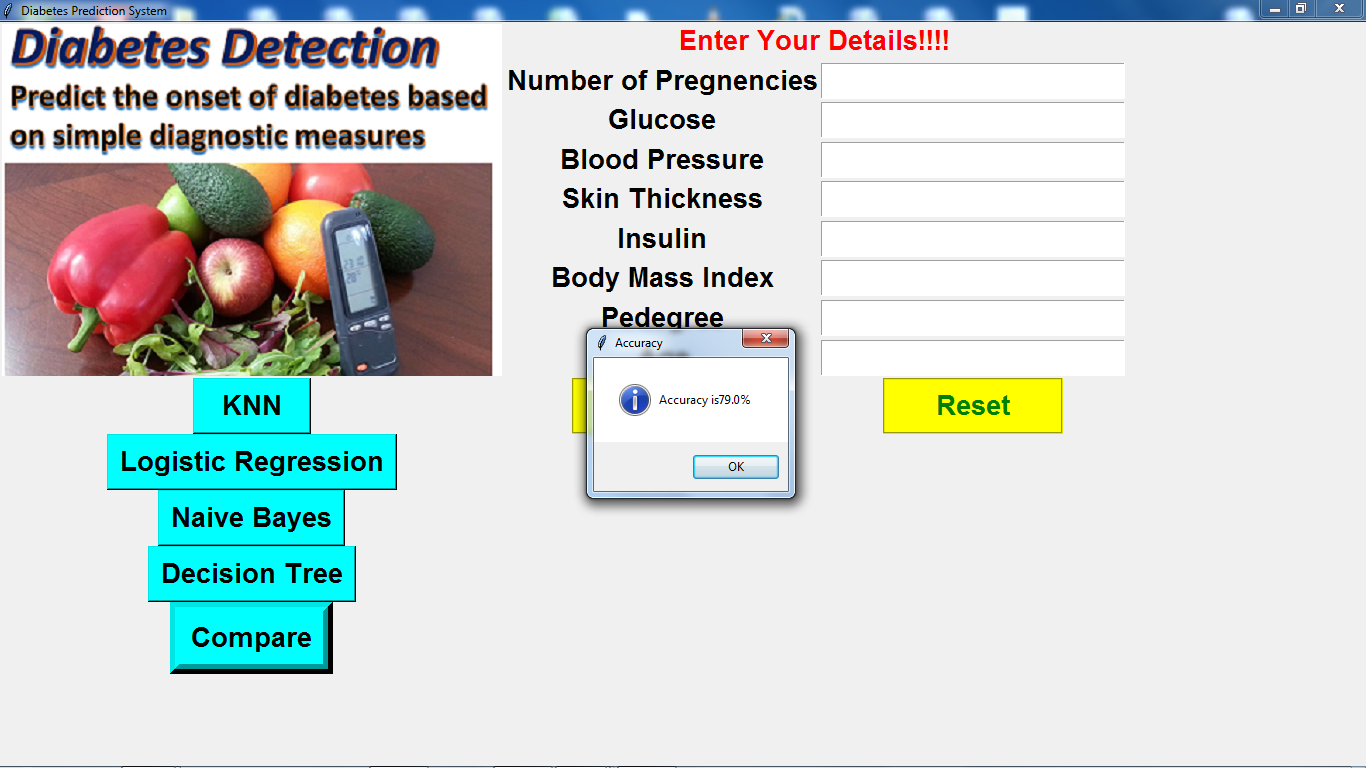
**4.3.2 After clicking on the KNN button**

****

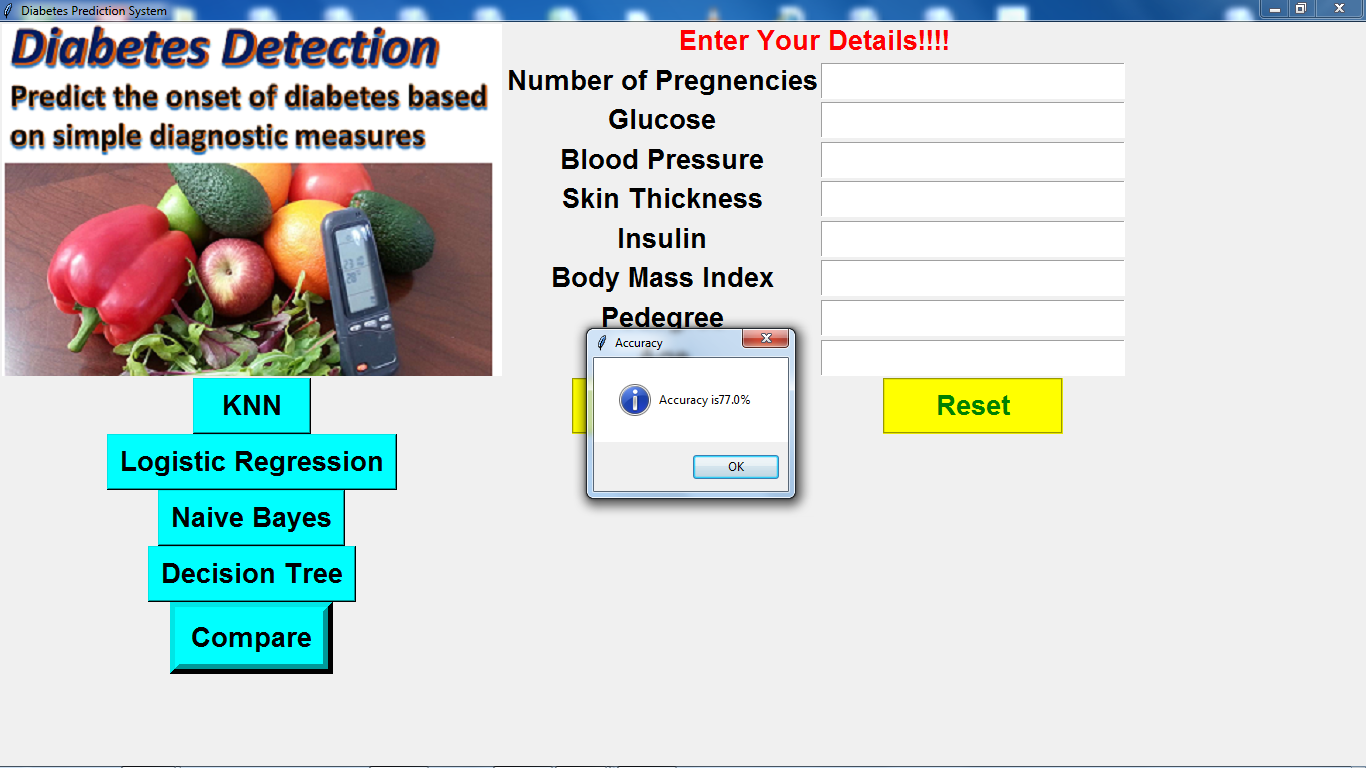
**4.3.3 After clicking on the Logistic Regression button**

****

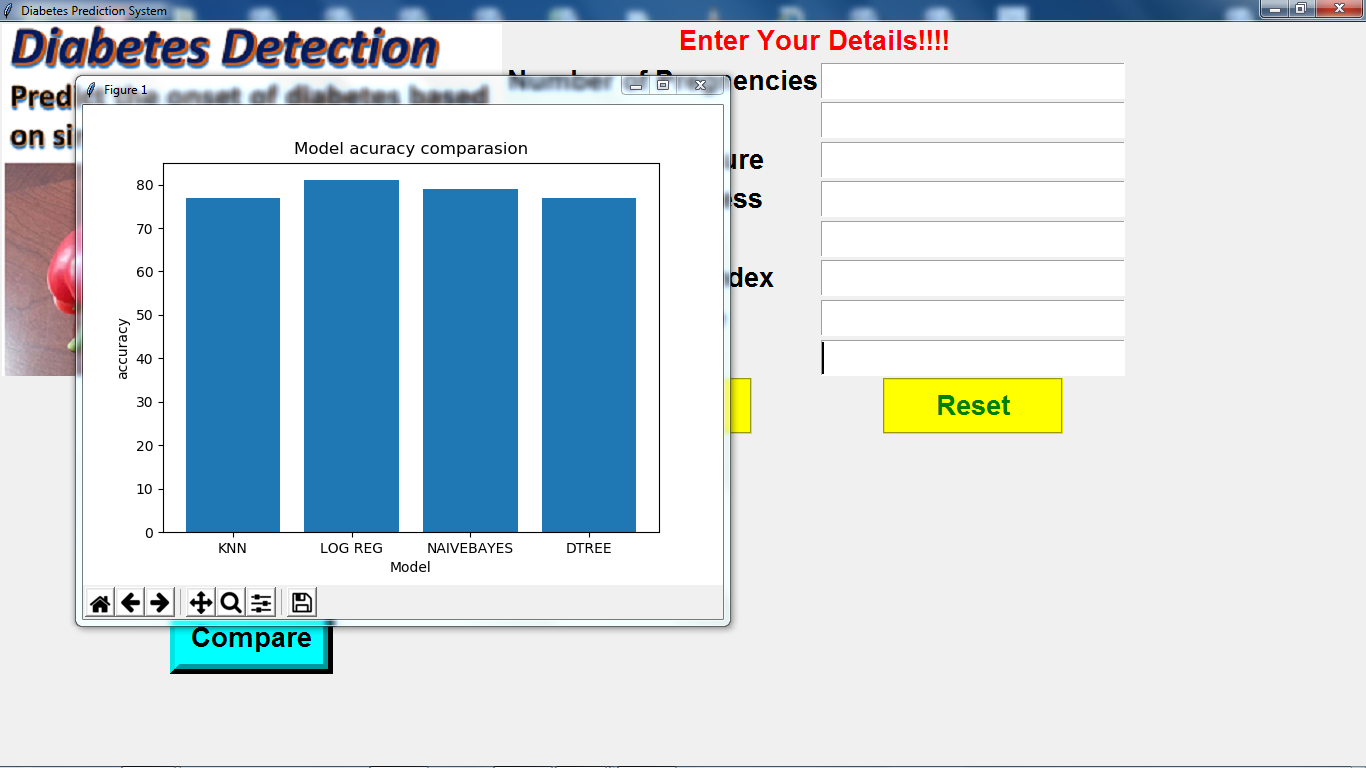
**4.3.4 After clicking on the Naïve Bayes button**

****

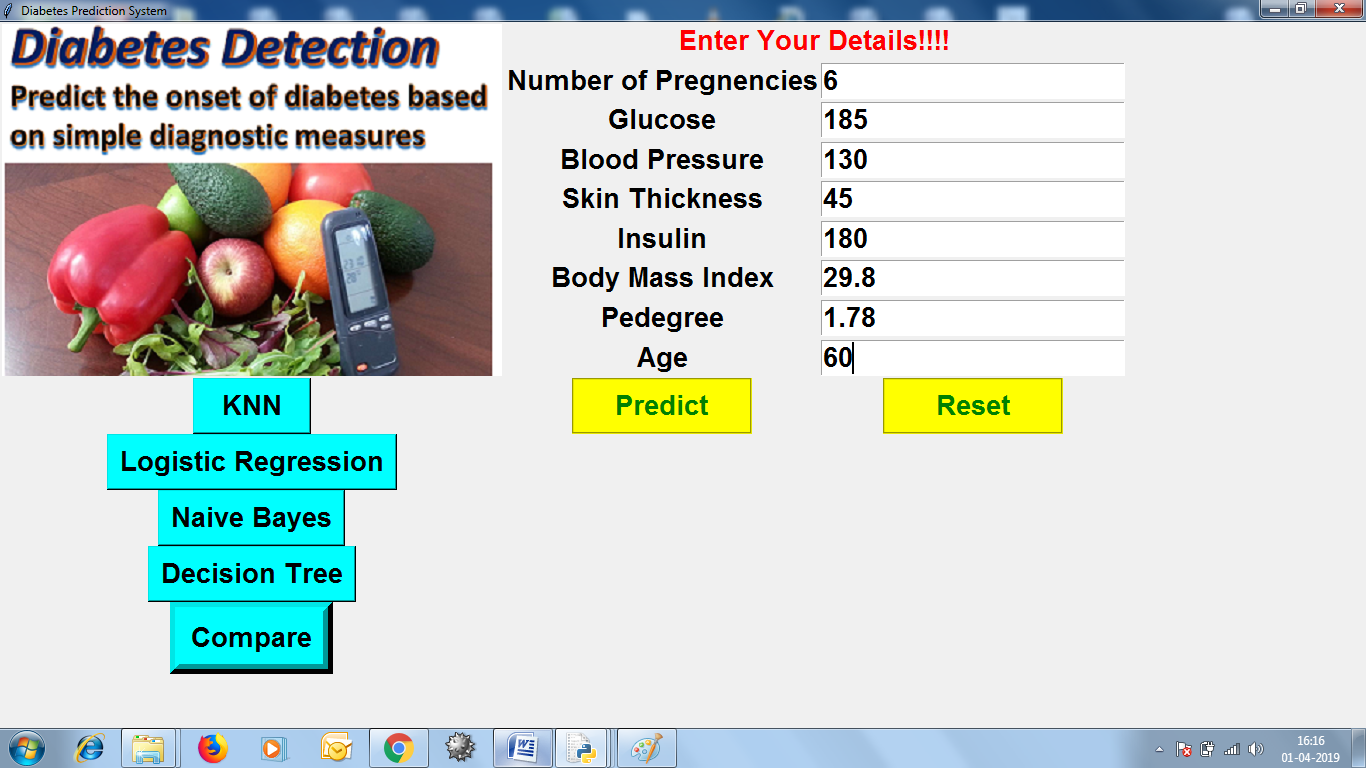
**4.3.5 After clicking on the Decision Tree button**

****

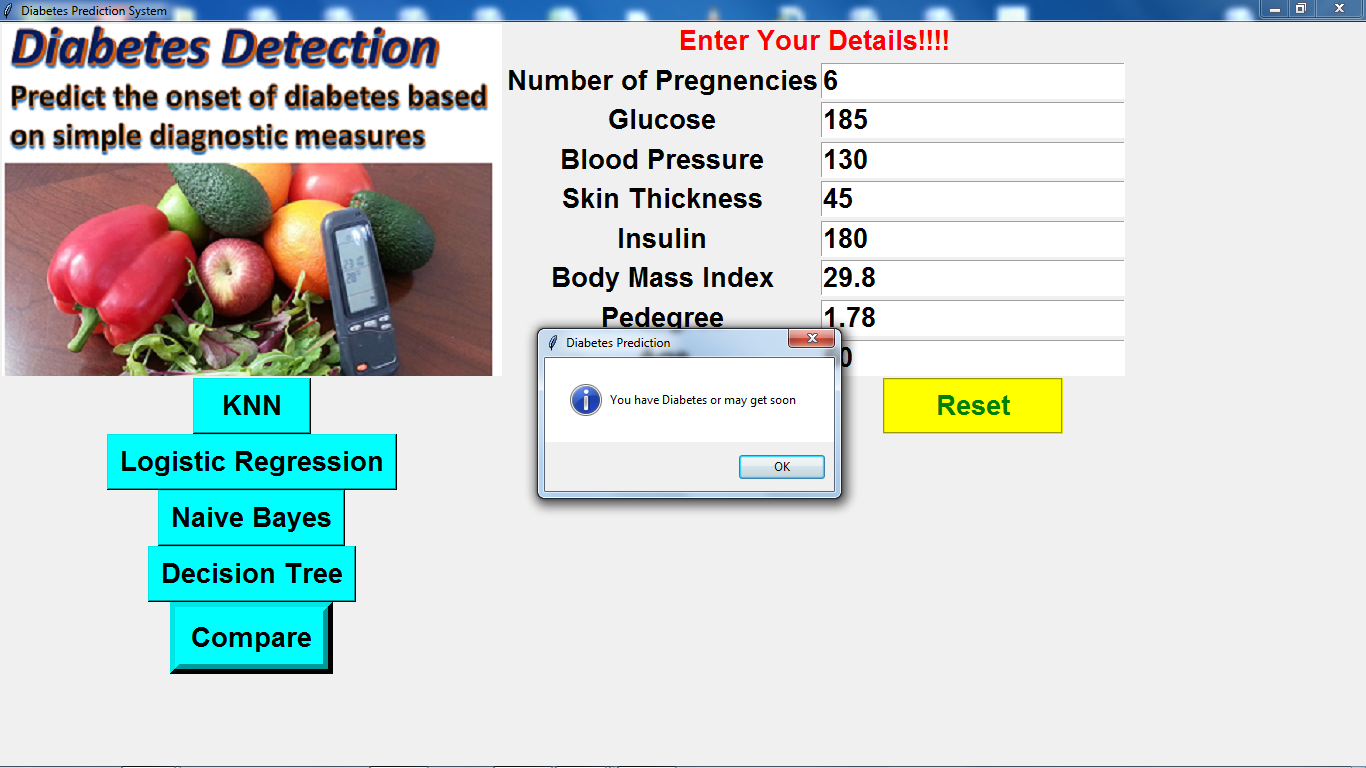
**4.3.6 After clicking on the Compare button**

****

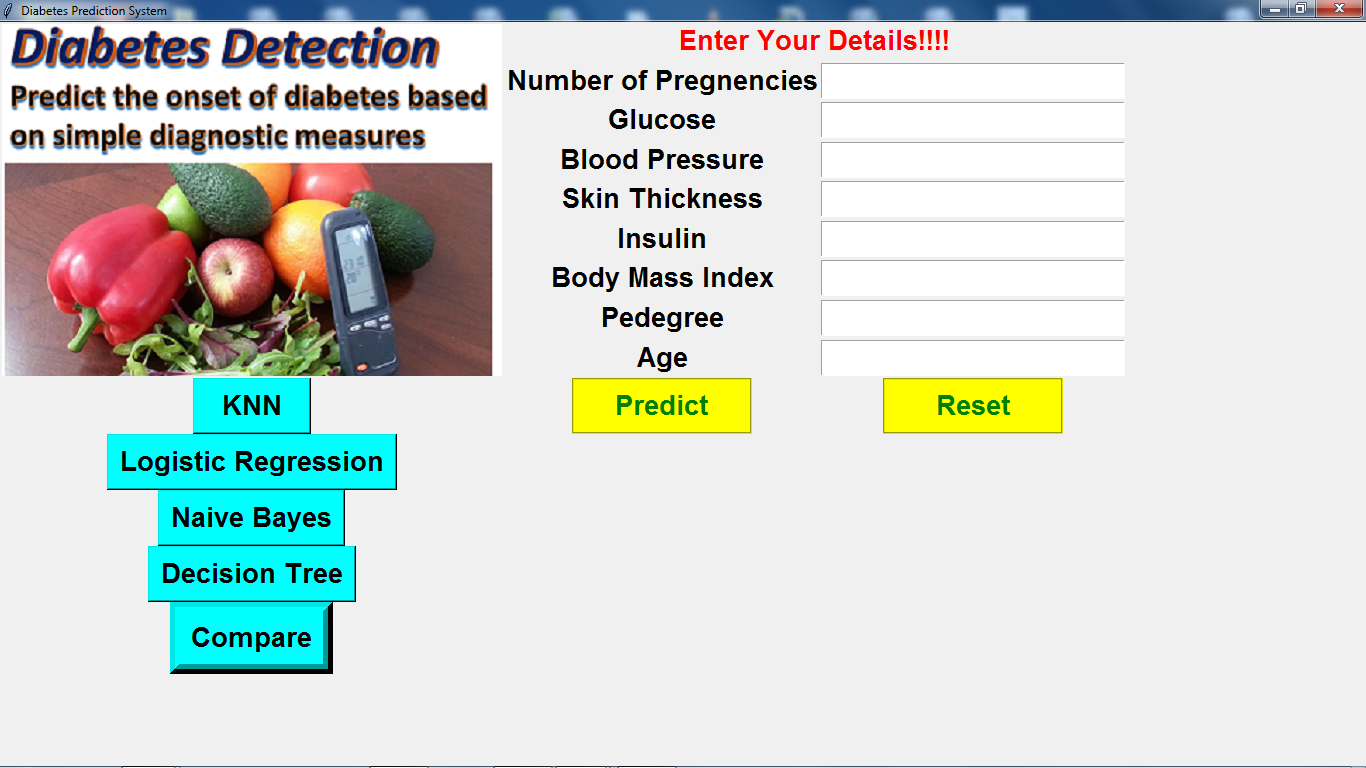
**4.3.5 After entering the data**

****

**4.3.6 Then after clicking on the Predict button**

****

**4.3.7 Then after clicking on the Reset button**

****

**5. Related work**

1. Using Bayes Network for Prediction of Type-2 Diabetes -Yang Guo School of computing Blekinge Institute of Technology Karlskrona, Sweden xyg@bth.se,Guohua Bai School of computing Blekinge Institute of Technology Karlskrona, Sweden gba@bth.se,Yan Hu School of computing Blekinge Institute of Technology Karlskrona, Sweden yhx@bth.se
2. A study of machine learning performance in the prediction of juvenile diabetes from clinical test results Shibendra Pobi University of South Florida,2006
3. Analyze Data Mining Algorithms For Prediction Of Diabetes -1 Priya B. Patel, 2 Parth P. Shah, 3 Himanshu D. Patel 1,2,3Student 1 Computer Engineering Department, 1 BVM Engineering College, Vallabh Vidyanagar, India.
4. Comparative Study of Decision Tree and Neural Network for the Analysis of Diabetes Data -Gauri D. Kalyankar Dept. of CSE, Rajarambapu Institute of Technology, Islampur,Shivananda R. Poojara Dept. of CSE, Rajarambapu Institute of Technology, Islampur, Nagaraj V. Dharwadkar Dept. of CSE, Rajarambapu Institute of Technology, Islampur dharwadkarn@gmail.com

**6. Conclusion**

Machine learning algorithms in the medical field extracts different hidden patterns from the medical data. They can be used for the analysis of important clinical parameters, prediction of various diseases, forecasting tasks in medicine, extraction of medical knowledge, therapy planning support and patient management. A number of algorithms were proposed for the prediction and diagnosis of diabetes. These algorithms provide more accuracy than the available traditional systems. We tried and optimize every algorithm and we found out Logistic regression algorithm best suitable for over application.

**7. Future work**

We tried and optimize every algorithm and we found out Logistic regression algorithm best suitable for over application.

But this is just not the end. We can also implement other classification algorithms to obtain more accurate and optimized result.

**8. Reference**

1. UCI Machine Learning Repository- Center for Machine Learning and Intelligent System, <http://archive.ics.uci.edu/ml/>
2. Gaganjot Kaur, Amit Chhabra, “Improved J48 Classification Algorithm for the Prediction of Diabetes”, International Journal of Computer Applications (0975 – 8887) Volume 98 – No.22, July 2014.
3. Machine Learning tutorials and examples <https://www.toptal.com/machine-learning/machine>- learningtheory- an-introductory-primer .
4. Sudip Mandal, Goutam Saha, Rajat K. Pal, “A Comparative Study on Disease Classification using Different Soft Computing Techniques”, The SIJ Transactions on Computer Science Engineering & its Applications (CSEA), Vol. 2, No. 3, May 2014.
5. Classification Techniques" IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 1 Issue 6, August 2014.
6. <http://www.diabetes.ca/about-diabetes/types-of-diabetes>
7. <http://www.diabetes.org/diabetes-basics/type>
8. G. F. Cooper. A simple constraint-based algorithm for eﬃciently mining observational databases for causal relationships. Machine Learning, 1(2):203–224, 1997