**2.3 Supervised Machine Learning Classifiers**

In this step, we will describe some supervised machine learning classifiers named Logistic Regression, k-nearest neighbors, Support Vector Machine, Decision Tree, Gaussian Naive Bayes, Random Forest, Gradient Boosting and Linear Discriminant Analysis.

**2.3.1 Logistic Regression (LR)**

Logistic Regression (LR) is a supervised machine learning data classification algorithm that mines real-valued features from the input, multiplies each of them by a weight, adds them, and transfers the sum through a sigmoid function to produce a probability. A threshold is used to finalize a decision [2]. A solution for classification of our data set is LR which Instead of fitting a straight line or hyperplane uses the logistic function to squeeze the output of a linear equation between 0 and 1. The logistic function is defined as:

*Logistic*(*η*) = 1*/*(1 + *exp*(*-η*))

As *η* goes from *-∞* to *∞*, logistic (*η*) goes from 0 to 1, a “squashing function”. In our study, we used a maximum 4000 iterations to converge the output.

**2.3.2 k-nearest neighbors (KNN)**

(KNN) is a non-parametric process we used for diabetic data classification. In KNN a data is classified by a majority vote of its neighbors, with the data being allotted to the class most mutual amongst its K nearest neighbors estimated by a distance function. If K = 1, then the data is simply allotted to the class of its nearest neighbor. KNN algorithm is as below [3]:

**Algorithm 1** KNN

1. Let *m* be the number of training data samples. Let *p* be an unknown point that needs to be classified
2. Storing the training samples in an array of data points *arr*[]. Each element of this array denotes a tuple (*x, y*).
3. **for** *i* = 0 to *m* **do**
4. Calculating distance *d*(*arr*[*i*]*, p*)
5. **end for**
6. Making set *S* of *K* smallest distances achieved. Each of these distances resembles an already classified data point
7. Returning the majority label among *S*

**2.3.3 Decision Tree (DT)**

A DT is a classifier that recursively performs partition of the instance space. The decision tree contains nodes that form a tree, a node called “root” that has no incoming edges is the starting point of the tree. All other nodes have one incoming edge. The leaf nodes are known as decision nodes. The child node is nominated by computing Information Gain (IG).

Information Gain = Entropy(parent) - [weights average] \* Entropy(children)

Entropy(*Ci*) = -P(*xi*) log P(*xi*), where P(*xi*) is the probability of child node *i*.

Node with the highest IG will be the parent for next level. This process is continued until it gets a leaf node and completed decision tree.

The algorithm for generating a decision tree is as below [5]:

**Algorithm 2** DT

1. Create (T)
2. Calculate frequencies (Ci, T)
3. If all instances belong to the same class, returning leaf
4. for every attribute a test is set for splitting criteria. An attribute that satisfies the test is test node K
5. Repeating Create (Ti) on each partition Ti.Adding those nodes as children of node K

**2.3.4 Gaussian Naive Bayes (GNB)**

The GNB classifier is a probability distribution function having the effect of associating neural activation to the means and variances of activation in various impulse conditions. The production of the classifier is a condition-label [6]. The classifier creates hypothesis that the classes have Gaussian normal distributions.

The z-score distance between the inputted point and each class-mean is estimated for each data point, namely the distance from the class mean divided by the standard deviation of that class.

*Z*A = (*x - µ*A)/*σ*A

According to the equation for a Gaussian normal distribution, each z-score is then converted into a probability value which is used for observing data point x. The co-variance between dimensions is not modelled by GNB classifier.

**2.3.5 Random Forest (RF)**

RF is a collective algorithm which was modelled from trees algorithm and Bagging algorithm. It works fine with a data set with a large number of input variables. It is a meta estimator that creates a number of decision tree classifiers on different subsamples of the data set and uses mean value to increase the accuracy of the model and control over-fitting. Suppose training data set is given as: [X1, X2, X3, X4] with labels as [L1, L2, L3, L4] respectively, random forest algorithm may create three decision trees taking input of subset for example, [X1, X3, X4], [X2, X3, X4] and [X1, X2, X4]. Finally, it predicts class based on the majority of votes from each of the decision trees generated. Generally, the more trees in the forest the more robust and reliable the forest is. The random forest classifier works in the same way, the higher the number of trees in the forest gives higher accuracy output [7].

**2.3.6 Gradient boosting (GB)**

GB includes three components: a loss function that is to be optimized, a weak learner that makes predictions and an additive model which will add weak learners to minimize the loss function.

**2.3.7 Linear Discriminant Analysis (LDA)**

The goal of an LDA is to convert a feature space (n-dimensional data set) into a smaller dimension k (where k *≤* n-1) while saving the class-discriminatory information. 5 general steps for performing a linear discriminant analysis are given below: [8]

1. Computing the d-dimensional mean vectors for different classes.
2. Generating the scatter matrices.