

CHAPTER- 1

INTRODUCTION

Now-a-days there is a continuous expansion of data availability in many areas of engineering and science, identifying patterns from vast amounts of data and identifying members of a predefined class, which is called classification, have become critical tasks. Therefore, classification is a fundamental problem, especially in pattern recognition and data mining.

1.1 Classification: -

Classification is a data mining function that assigns items in a collection to target categories or classes. The goal of classification is to accurately predict the target class for each case in the data. For example, a classification model could be used to identify loan applicants as low, medium, or high credit risks.

A classification task begins with a data set in which the class assignments are known. For example, a classification model that predicts credit risk could be developed based on observed data for many loan applicants over a period of time. In addition to the historical credit rating, the data might track employment history, home ownership or rental, years of residence, number and type of investments, and so on. Credit rating would be the target, the other attributes would be the predictors, and the data for each customer would constitute a case.

Classifications are discrete and do not imply order. Continuous, floating-point values would indicate a numerical, rather than a categorical, target. A predictive model with a numerical target uses a regression algorithm, not a classification algorithm.

The simplest type of classification problem is binary classification. In binary classification, the target attribute has only two possible values: for example, high credit rating or low credit rating. Multiclass targets have more than two values: for example, low, medium, high, or unknown credit rating.

In the model build (training) process, a classification algorithm finds relationships between the values of the predictors and the values of the target. Different classification algorithms use different techniques for finding relationships. These relationships are

summarized in a model, which can then be applied to a different data set in which the class assignments are unknown.

Classification models are tested by comparing the predicted values to known target values in a set of test data. The historical data for a classification project is typically divided into two data sets: one for building the model; the other for testing the model.

Scoring a classification model results in class assignments and probabilities for each case. For example, a model that classifies customers as low, medium, or high value would also predict the probability of each classification for each customer.

Classification has many applications in customer segmentation, business modeling, marketing, credit analysis, and biomedical and drug response modeling.

Several effective algorithms have been successfully applied in many real-world applications. Some of the classification algorithms are as follows:

- Decision Tree
- SVM
- Naive Bayes

Decision Tree

Decision tree is the most powerful and popular tool for classification and prediction. A Decision tree is a flowchart like tree structure, where each internal node

denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.

A tree can be “learned” by splitting the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called *recursive partitioning*. The recursion is completed when the subset at a node all has the same value of the target variable, or when splitting no longer adds value to the predictions. The construction of decision tree classifier does not require any domain knowledge or parameter setting, and therefore is appropriate for exploratory knowledge discovery. Decision trees can handle high dimensional data. In general decision tree classifier has good accuracy. Decision tree induction is a typical inductive approach to learn knowledge on classification.

Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance. An instance is classified by starting at the root node of the tree, testing the attribute specified by this node and then moving down the tree branch corresponding to the value of the attribute as shown in the above figure. This process is then repeated for the subtree rooted at the new node.

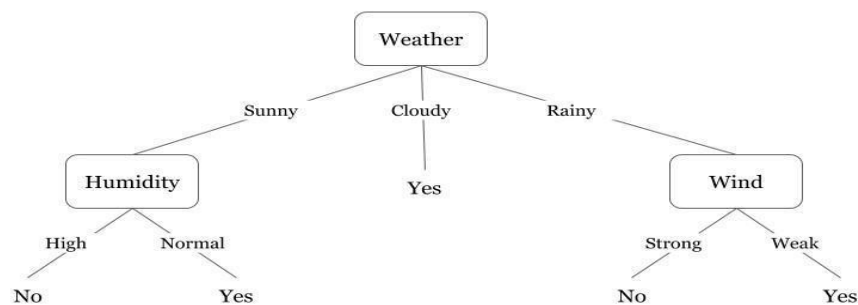
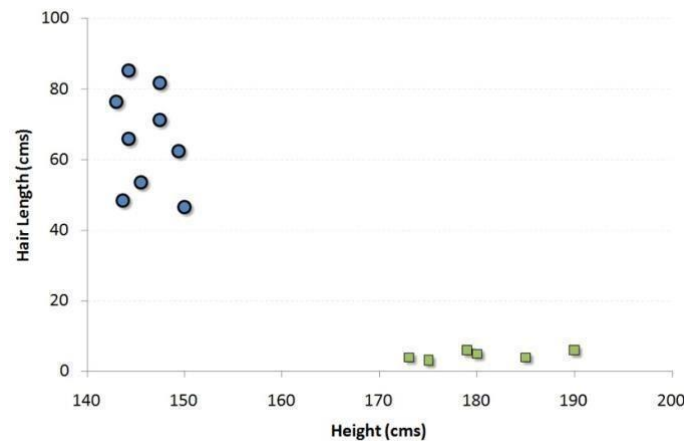


Fig: - Decision tree for suitable for playing a cricket match

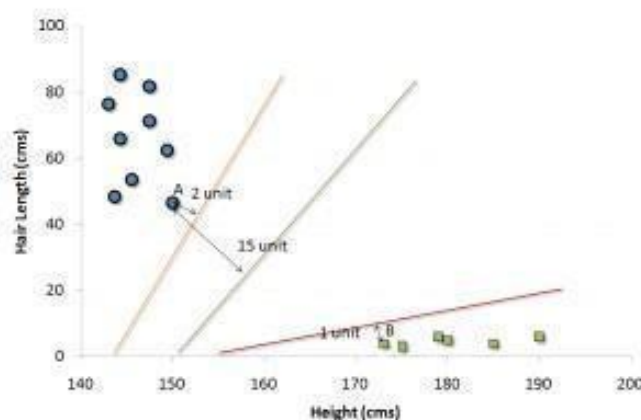
SVM (Support Vector Machine)

It is a classification method. In this algorithm, we plot each data item as a point in n dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate.

For example, if we only had two features like Height and Hair length of an individual, we'd first plot these two variables in two dimensional space where each point has two coordinates (these co-ordinates are known as **Support Vectors**)



Now, we will find some *line* that splits the data between the two differently classified groups of data. This will be the line such that the distances from the closest point in each of the two groups will be farthest away.

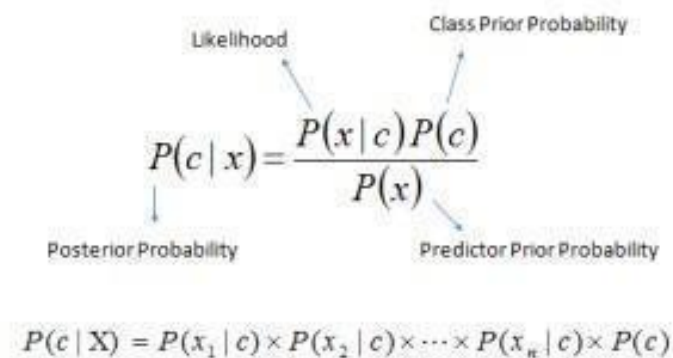


Naive Bayes

It is a classification technique based on Bayes theorem with an assumption of independence between predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, a naive Bayes classifier would consider all of these properties to independently contribute to the probability that this fruit is an apple.

Naive Bayesian model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

Bayes theorem provides a way of calculating posterior probability $P(c|x)$ from $P(c)$, $P(x)$ and $P(x|c)$. Look at the equation below:



The diagram shows the equation $P(c|x) = \frac{P(x|c)P(c)}{P(x)}$ with arrows pointing from labels to the terms: 'Likelihood' points to $P(x|c)$, 'Class Prior Probability' points to $P(c)$, 'Posterior Probability' points to $P(c|x)$, and 'Predictor Prior Probability' points to $P(x)$. Below this, the expanded equation is shown: $P(c|X) = P(x_1|c) \times P(x_2|c) \times \dots \times P(x_n|c) \times P(c)$.

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \dots \times P(x_n|c) \times P(c)$$

Here,

- $P(c|x)$ is the posterior probability of *class (target)* given *predictor (attribute)*.
- $P(c)$ is the prior probability of *class*.
- $P(x|c)$ is the likelihood which is the probability of *predictor* given *class*.
- $P(x)$ is the prior probability of *predictor*.

1.2 Parametric Classifier:

Any algorithm which it learns from predefined mapped function is called Parametric Classifier. No matter how much amount of data you give to this algorithm it would not change its mind to how many parameters do it require to learn in order to predict a classifier.

Example: - $y = B_0 + B_1(x)$

In the above equation we can simply specify the values of B_0 and B_1 and it will be easy to find the value of y . This is called **Simple Linear Regression (LR)**.

But there is a problem, what if the data which we will use to train the above equation does not follow a **Linear Distribution (LD)**. Hence it is the major drawback of

Parametric Classifier. Some of the algorithms that follows Parametric Classifier are **Logistic Regression (LR)**, **Linear Discriminant Analysis (LDA)** and **Deep Learning**.

Linear Regression

It is used to estimate real values (cost of houses, number of calls, total sales etc.) based on continuous variable(s). Here, we establish relationship between independent and dependent variables by fitting a best line. This best fit line is known as regression line and represented by a linear equation $Y = a * X + b$.

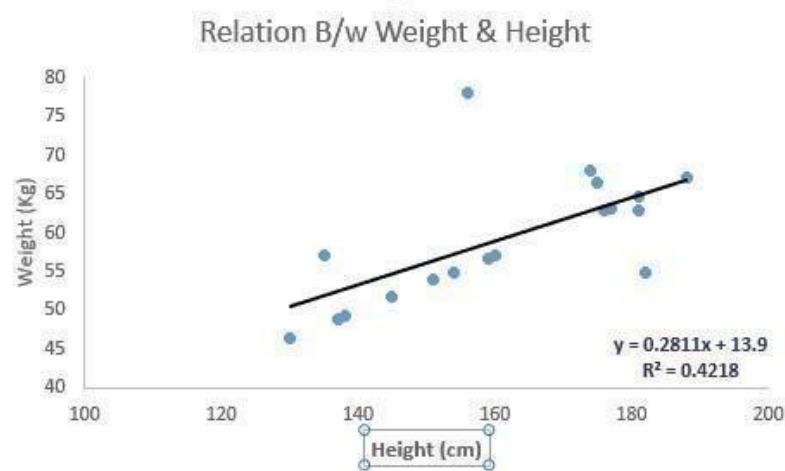
The best way to understand linear regression is to relive this experience of childhood. Let us say, you ask a child in fifth grade to arrange people in his class by increasing order of weight, without asking them their weights! What do you think the child will do? He / she would likely look (visually analyse) at the height and build of people and arrange them using a combination of these visible parameters. This is linear regression in real life! The child has actually figured out that height and build would be correlated to the weight by a relationship, which looks like the equation above.

In this equation:

- Y – Dependent Variable
- a – Slope
- X – Independent variable
- b – Intercept

These coefficients a and b are derived based on minimizing the sum of squared difference of distance between data points and regression line.

Look at the below example. Here we have identified the best fit line having linear equation $y = 0.2811x + 13.9$. Now using this equation, we can find the weight, knowing the height of a person.



Linear Regression is of mainly two types: Simple Linear Regression and Multiple Linear Regression. Simple Linear Regression is characterized by one independent variable. And, Multiple Linear Regression (as the name suggests) is characterized by multiple (more than 1) independent variables. While finding best fit line, you can fit a polynomial or curvilinear regression. And these are known as polynomial or curvilinear regression.

Logistic Regression

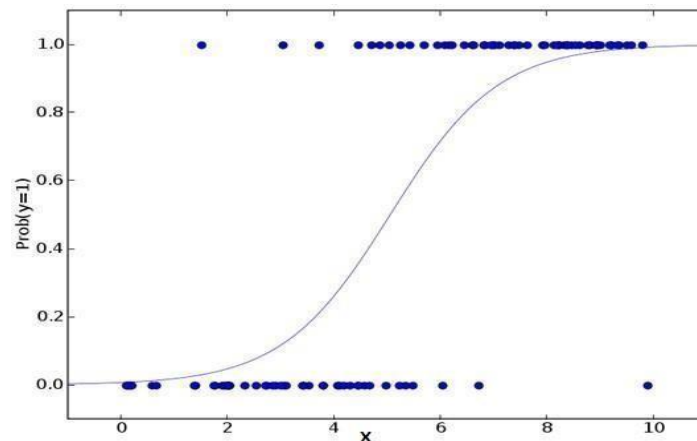
Don't get confused by its name! It is a classification not a regression algorithm. It is used to estimate discrete values (Binary values like 0/1, yes/no, true/false) based on given set of independent variable(s). In simple words, it predicts the probability of occurrence of an event by fitting data to a logit function. Hence, it is also known as **logit regression**. Since, it predicts the probability, its output values lies between 0 and 1 (as expected).

Let's say your friend gives you a puzzle to solve. There are only 2 outcome scenarios – either you solve it or you don't. Now imagine, that you are being given wide range of puzzles / quizzes in an attempt to understand which subjects you are good at. The outcome to this study would be something like this – if you are given a trigonometry based tenth grade problem, you are 70% likely to solve it. On the other hand, if it is grade fifth history question, the probability of getting an answer is only 30%. This is what Logistic Regression provides you.

Coming to the math, the log odds of the outcome is modelled as a linear combination of the predictor variables. Odds = $p / (1-p)$ = probability of event occurrence / probability

of not event occurrence $\ln(\text{odds}) = \ln(p/(1-p))$ $\text{logit}(p) = \ln(p/(1-p)) = b_0 + b_1X_1 + b_2X_2 + b_3X_3 + \dots + b_kX_k$

Above, p is the probability of presence of the characteristic of interest. It chooses parameters that maximize the likelihood of observing the sample values rather than that minimize the sum of squared errors (like in ordinary regression).



1.3 Non Parametric Classifier:

Any algorithm which does not make any assumptions regarding the form of mapping function are called Non Parametric Classifier.

They are free to learn. It is really good when we have lot of data but have no prior knowledge and do not want to worry too much about the features that we need to select correctly for your data set.

Some of the algorithms that follows Non Parametric Classifier are **K- Nearest Neighbors (K-NN)**, **Decision Trees**, **Naïve Bayes**, **Support Vector Machines (SVM)** and **Neural Networks**.

***k*- Nearest Neighbor (*k*-NN):-**

K Nearest Neighbor is a simple algorithm that stores all the available cases and classifies the new data or case based on a similarity measure. It is mostly used to classify a data point based on how its neighbors are classified.

It is non-parametric, meaning, it does not make any underlying assumptions about the distribution of a data.

Suppose we have a dataset which can be plotted as follows –

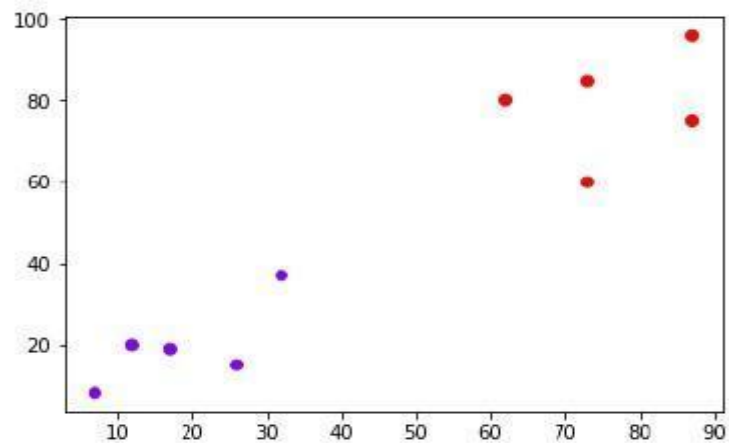


Fig: - 1.4 K- Nearest Neighbor

Now, we need to classify new data point with black dot (at point 60, 60) into blue or red class. We are assuming $K = 3$ i.e. it would find three nearest data points. It is shown in the next diagram –

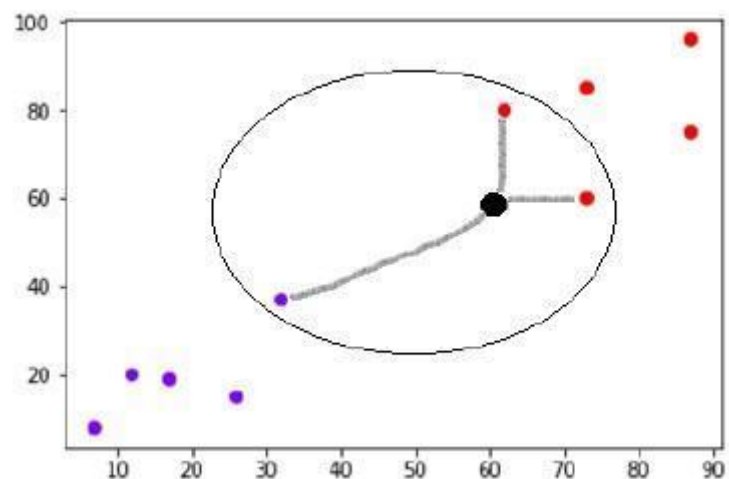


Fig: - 1.4 K- Nearest Neighbor

We can see in the above diagram the three nearest neighbors of the data point with black dot. Among those three, two of them lies in Red class hence the black dot will also be assigned in red class.

Few ideas on picking a value for ‘ k ’:-

1. There is no structured method to find the best value for “ k ”. We need to find out with various values by trial and error and assuming that training data is unknown.

2. Choosing smaller values for k can be noisy and will have a higher influence on the result.
3. Larger values of k will have smoother decision boundaries which mean lower variance but increased bias. Also, computationally expensive.
4. Another way to choose k is through cross-validation. One way to select the cross-validation dataset from the training dataset. Take the small portion from the training dataset and call it a validation dataset, and then use the same to evaluate different possible values of k . This way we are going to predict the label for every instance in the validation set using with k equals to 1, k equals to 2, k equals to 3. And then we look at what value of k gives us the best performance on the validation set and then we can take that value and use that as the final setting of our algorithm so we are minimizing the validation error.
5. In general, practice, choosing the value of k is $k = \text{sqrt}(\mathbf{N})$ where \mathbf{N} stands for the **number of samples in your training dataset**.
6. Try and keep the value of k odd in order to avoid confusion between two classes of data

CHAPTER-2

LITERATURE SURVEY

- **Ji Feng, Yan Wei proposed a system in December 2018 over Data mining at 14th International Conference. In this paper they have presented a model in which they overcome the problems detected on k -NN by NNBCA (Natural Neighborhood Based Classification Algorithm).**

The efficiency of NN classification heavily depends on the type of distance measure, especially in a large-scale and high-dimensional database. In some applications, the data structure is so complicated that the corresponding distance measure is computationally expensive. The NN classifier simply compares the data structure with all other examples in the database for each query. Such comparison may be impractical.

Traditional k -NN adopts a fixed k for all query samples regardless of their geometric location and related specialties. Furthermore, those k -nearest neighbors may not distribute symmetrically around the query sample if the neighborhood in the training set is not spatially homogeneous. The geometrical placement might be more important than the actual distance to depict a query sample's neighborhood.

- **Qingshang Zhu a Chongqing University professor proposed a parameter free k -NN algorithm which dealt the problems occurred in the conventional k -NN Algorithm. He proposed his system at 13th International Conference over the concept of Data Mining.**

Many other algorithms have been proposed to improve neighborhood based classifiers. Recently, on the basis of the mutual k -nearest neighborhood method, Tang and He proposed an Extended Nearest Neighbor (ENN) method for classification, which uses the two-way communication style. Unlike the classic k -NN rule which considers only the nearest neighbors of a test sample to make a classification decision, ENN method considers not only who the nearest neighbors of the test sample are but also who considers the test sample their nearest neighbors.

This two-way communication style is both an advantage and disadvantage of the ENN method. The advantage is that the classification decision of ENN method depends on all

variable training data. The disadvantage is that the problem of parameter selection such as the selection of k in k -NN still exists. ENN has a parameter of k , on the basis of which the size and shape of the graph can be modified. This feature makes additional optimization possible; however, the problem of parameter selection continues to persist.

To overcome the above limitations of classification algorithms, we propose a new method called the Natural Neighborhood-Based Classification Algorithm (NNBCA). With the help of our previously proposed Natural Neighbor (NaN) method, the solution of choosing the optimum value of k both in the training and testing stages is presented in this paper.

The selection of k is crucial to a successful k -NN learning algorithm. Usually, a large value of k corresponds to a smooth decision boundary of the classifier, however, its efficiency is questionable. By contrast, k -NN is sensitive to noisy data if k is small. Predetermining the value of k is difficult especially when the instances are uniformly distributed. To address this issue, an empirical strategy that is frequently used in practice is to determine a proper value of k by using Cross Validation (CV) or other heuristic techniques. However, the CV method requires considerable computational cost. Although many endeavours have attempted to determine k and several methods have been effective, determining a method of selecting the optimal value of k for different applications remains a challenge.

CHAPTER-3

SYSTEM ANALYSIS

3.1 Existing system

Nearest neighbouring (k -NN) is a non-parametric method used for classification. Here k is a parameter which is a user-defined constant. Based on the value of k the current tuple is classified by assigning the label which is most frequent among the k training samples nearest to the query point. The commonly used distance metric for continuous variable is Euclidean Distance.

Problem Definition

The existing algorithm requires a query point along with a parameter constant k , which has to be manually declared. But the optimal number of neighbours is to be considered which classify the new data entry which is not performed with the current algorithm (k -NN).

3.2 Proposed system

This proposed system uses an algorithm called Enhanced k -NN where the optimized k value which the highest accuracy for that particular dataset will be found out and that k value can be used for classifying new tuples. This algorithm provides a good classification result with artificially selecting the neighborhood parameter. The original k -NN based algorithm needs a prior k , but current algorithm predicts different k for different samples. Thus it provides a good classification result when compared to other methods.

Advantages

- The choice of parameter k is done automatically which reduces the pressure on the user.
- The automatic choice of k is more accurate than manual parameter k .
- This model provides a good classification result when compared to other methods.
- Outlier's detection and separation is more accurate in this model.

3.2.1 Architecture

A **system architecture** or **systems architecture** is the conceptual model that defines the structure, behaviour, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviours of the system.

A system architecture can consist of system components and the sub-systems developed, that will work together to implement the overall system.

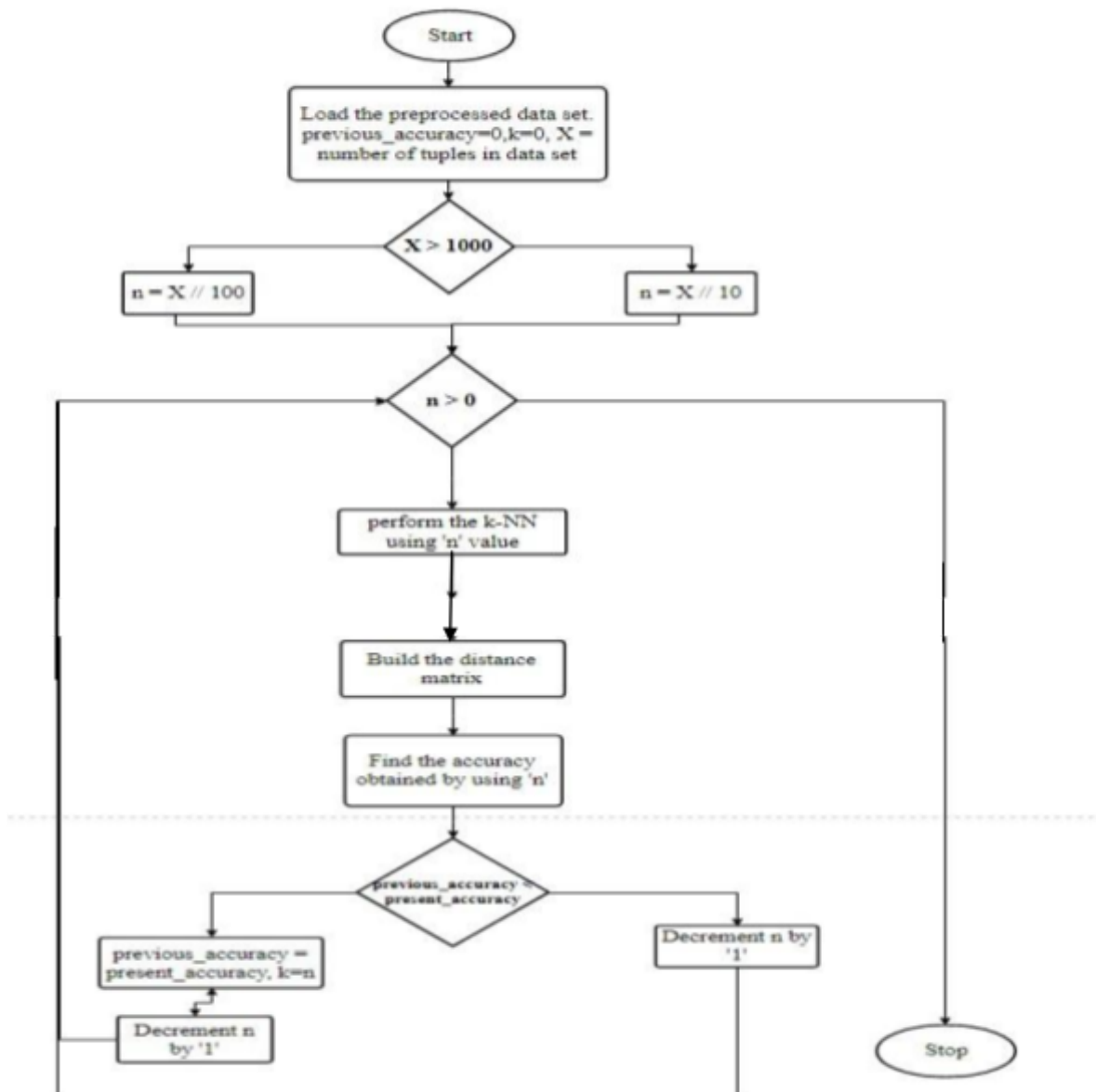


Fig: 3.2.1 Architecture

3.3 MODULES

- Data Preprocessing.
- Building the Distance Matrix.

- Determining Optimal ' k '.

3.3.1 Data Preprocessing

Data preprocessing is a data mining technique which is used to transform the raw data in a useful and efficient format.

Steps involved in data preprocessing are as follows:

- Data cleaning
- Data transformation
- Data reduction

Data Cleaning: -

The data can have many irrelevant and missing parts. To handle this part, data cleaning is done. It involves handling of missing data, noisy data etc.

• Missing Data

This situation arises when some data is missing in the data. It can be handled in various ways.

1. Ignore the tuples:

This approach is suitable only when the dataset we have is quite large and multiple values are missing within a tuple.

2. Fill the Missing values:

There are various ways to do this task. You can choose to fill the missing values manually, by attribute mean or the most probable value.

• Noisy Data

Noisy data is a meaningless data that can't be interpreted by machines. It can be generated due to faulty data collection, data entry errors etc. It can be handled in following ways:

1. Binning Method:

This method works on sorted data in order to smooth it. The whole data divided into segments of equal size and the various methods are performed to

complete the task. Each segmented is handled separately. One can replace all data in a segmented by its mean or boundary values can be used to complete the task.

2. Regression:

Here data can be made smooth by fitting it to a regression function. The regression used may be linear (having one independent variable) or multiple (having multiple independent variables).

3. Clustering:

This approach groups the similar data in a cluster. The outliers may be undetected or it will fall outside the clusters.

Data Transformation:

This step is taken in order to transform the data in appropriate forms suitable for mining process. This involves following ways:

1. Normalization:

It is done in order to scale the data values in a specified range (-1.0 to 1.0 or 0.0 to 1.0)

2. Attribute Selection:

In this strategy, new attributes are constructed from the given set of attributes to help the mining process.

3. Discretization:

This is done to replace the raw values of numeric attribute by interval levels or conceptual levels.

4. Concept Hierarchy Generation:

Here attributes are converted from level to higher level in hierarchy. For Example-The attribute “city” can be converted to “country”.

Data Reduction:

Since data mining is a technique that is used to handle huge amount of data. While working with huge volume of data, analysis became harder in such cases. In order to get rid of this, we use data reduction technique. It aims to increase the storage efficiency and reduce data storage and analysis costs.

The various steps to data reduction are:

1. Data Cube Aggregation:

Aggregation operation is applied to data for the construction of the data cube.

2. Attribute Subset Selection:

The highly relevant attributes should be used, rest all can be discarded. For performing attribute selection, one can use level of significance and p- value of the attribute. The attribute having p-value greater than significance level can be discarded.

3. Numerosity Reduction:

This enable to store the model of data instead of whole data, for example: Regression Models.

4. Dimensionality Reduction:

This reduce the size of data by encoding mechanisms. It can be lossy or lossless. If after reconstruction from compressed data, original data can be retrieved, such reduction are called lossless reduction else it is called lossy reduction. The two effective methods of dimensionality reduction are: Wavelet transforms and PCA (Principal Component Analysis).

Algorithm: Data Preprocessing

Input: Actual Data set

Output: Preprocessed Data set

Begin

1. read CSV file into data array.
2. For each element i in class
3. If i == "p"
4. i <- 1
5. Else
6. i <- 0
7. For each column in data.columns

```

8.      value <- 0
9.      step <- 1 / (len(data[column].unique())-1)
10.     For i in data[column].unique():
11.         data[column] <- [value if letter == i else letter for letter in data[column]]
12.         value += step
End

```

The preprocessing is different for different data sets. So, here we have shown the preprocessing of a data set called “**mushroom**” data set which is of type CSV. In this data set there is column called “**class**” where it consists of two class labels which are ‘p’ and ‘q’.

Since the class column is categorical we have to convert it into numerical for further calculations. And the data set also contains a column called “**veil-type**” which is not a considering factor for calculations we have drop that column. The data set also contains many columns which are of categorical type. So by considering all the columns we are going to find the unique values of each column and converting into numerical by fitting all the values in range of ‘0’ and ‘1’.

5.3.2 Building the Distance Matrix:

In this module we have used an n value which is generated by dividing the actual size of data set by a number. We are going to perform this step in two ways which is based on the size of the data set. After generating the n value we will perform k -NN and build the distance matrix.

Building the distance matrix:

The k -NN all instances are viewed as an array of n -dimensional space and their distances are defined using a standard measure such as a Euclidean distance. Each feature that describes the instances will be considered as part of the elements used for calculating the distances and thus used for classifying the new query. This means that k -NN is heavily dependent on how the distance is measured and represented. Furthermore, k -NN classification works by taking the majority voting or labels of its knearest neighbors in the examples set. It is a distance measurement

based on correlation of variables in random vectors in non-spherical symmetric distribution. It takes into account of the mean, standard deviation, and covariance of each group in the sample set as defined in the following squared distance metric:

$$d(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

The Raw data which contains the important attributes of all the data entries are converted into a distance matrix in which we are going to find the distance between one entry point to every other entry point in the data set and it is maintained in the form of n-dimensional array which is shown as below:

RAW DATA

X	Y
9	49
24	54
51	28
81	54
81	23
86	32

DISTANCE MATRIX

A	B	C	D	E	F
0	16	47	72	77	79
16	0	37	57	65	66
47	37	0	40	30	35
72	57	40	0	31	23
77	65	30	31	0	10
79	66	35	23	10	0

Algorithm: Choosing the n.

Input: D (length of the data set).

Output: generated n value.

Begin

1. $D \leftarrow \text{length of the data set}$
2. if ($D > 1000$)
3. $n \leftarrow D / 100$
4. else
5. $n \leftarrow D / 10$

End

5.1.3 Determining Optimal 'k':

After finding the distance matrix and k value we have to assign the class label for the test instance by finding the majority of class label where the instances that are neighbor to the test instance.

For example, $k=5$ and class labels are C1 and C2. For The test instance 3 neighbors has class label C2 and 2 neighbors has class label as C1. Here the majority members have class label C2. Hence the test instance will be assigned the class label as C2.

After performing k -NN with different n values we will find the n value which gives highest accuracy for the data set will be found and it is assigned to k which is further used for finding the class label for the new tuples.

k -NN is a lazy learning model which takes heavy amount of time during its training phase, while building its model. After performing k -NN with different n values every time the accuracy is stored and it is compared with previous accuracy and at last we are going to find the n value which gives highest accuracy for the given data set and it is assigned to the k . That k value is the output of our project. By using that k value we can predict the class labels of new tuples with more accuracy.

Algorithm: Enhanced k -NN

Input: Classification data set

Output: The best k for the data set

Begin

1. $D = \text{Data_set_size}$
2. If ($D > 1000$)
3. $n \leftarrow D // 100$
4. Else
5. $n \leftarrow D // 10$
6. $\text{prev_acc} \leftarrow 0; k \leftarrow 0$
7. While $n > 0$
8. Split the data into training and testing data # training data = 0.8 and testing data = 0.2
9. Classify (training data, n)
10. For i in training data
11. Find Euclidean distance of i to every other tuple.

12. Build the distance matrix and train the model.
13. Call Classify(training data, n)
14. For each test record (X_i to X_n) in testing data
15. Compute k nearest neighbors for X_i with n .
16. Assign the majority class label among the nearest neighbors to X_i .
17. Compare the predicted class labels of test data with the actual class labels.
18. `present_accuracy <-` Compute the accuracy of the model
19. If `previous_accuracy < present_accuracy` and `present_accuracy != 1.0`:
20. `previous_accuracy <- present_accuracy`
21. $k <- n$
22. Decrement n by 1
23. Display ' k ' and the 'previous accuracy'

End

Confusion Matrix:

A confusion matrix is a table that is often used to describe the performance of a classification model or (classifier) on a set of test data for which the true values are known. Each row of the matrix represents the instances in a predicted class while each column represents the instances in an actual class.

The confusion matrix is as follows:

		Actual class	
		P	N
Predicted class	P	TP	FP
	N	FN	TN

Where: P = positive; N = Negative; TP = True Positive;

FP = False Positive; TN = True Negative; FN = False Negative.

True positives: data points labelled as positive that are actually positive.

False positives: data points labelled as positive that are actually negative.

True negatives: data points labelled as negative that are actually negative.

False negatives: data points labelled as negative that are actually positive.

Precision: Ability of a classification model to return only relevant instances.

Recall : Ability of a classification model to identify all relevant instances.

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

3.3 SYSTEM REQUIREMENTS:

Hardware requirements:

- Hard Disk : 60 GB (Min).
- System : Intel i3 processor (Min).
- RAM : 2 GB (Min).

Software requirements:

- Operating system : Windows 7/8/10
- Language : Python
- Tools : Jupyter Notebook, Python IDLE.

CHAPTER-4

SYSTEM DESIGN

4.1 UML DIAGRAMS

The unified modeling language is a standard visual modeling language intended to be used for modeling business and similar processes, analysis, design, and implementation of software-based systems. UML is a common language for business analysts, software architects and developers used to describe, specify, design, and document existing or new business processes, structure and behavior of artifacts of software systems. UML can be applied to diverse **application domains** (e.g., banking, finance, internet, aerospace, healthcare, etc.) It can be used with all major object and component **software development methods** and for various **implementation platforms** (e.g., J2EE, .NET).

UML is a standard modelling **language**, not a **software development process**. UML Specification explained that process:

- provides guidance as to the order of a team's activities,
- specifies what artifacts should be developed,
- Directs the tasks of individual developers and the team as a whole.

UML is intentionally **process independent** and could be applied in the context of different processes. Still, it is most suitable for use case driven, iterative and incremental development processes. An example of such process is **Rational Unified Process** (RUP).

4.1.1 FLOW CHART

A flowchart is a formalized graphic representation of a logic sequence, work or manufacturing process, organization chart, or similar formalized structure. The purpose of a flow chart is to provide people with a common language or reference point when dealing with a project or process.

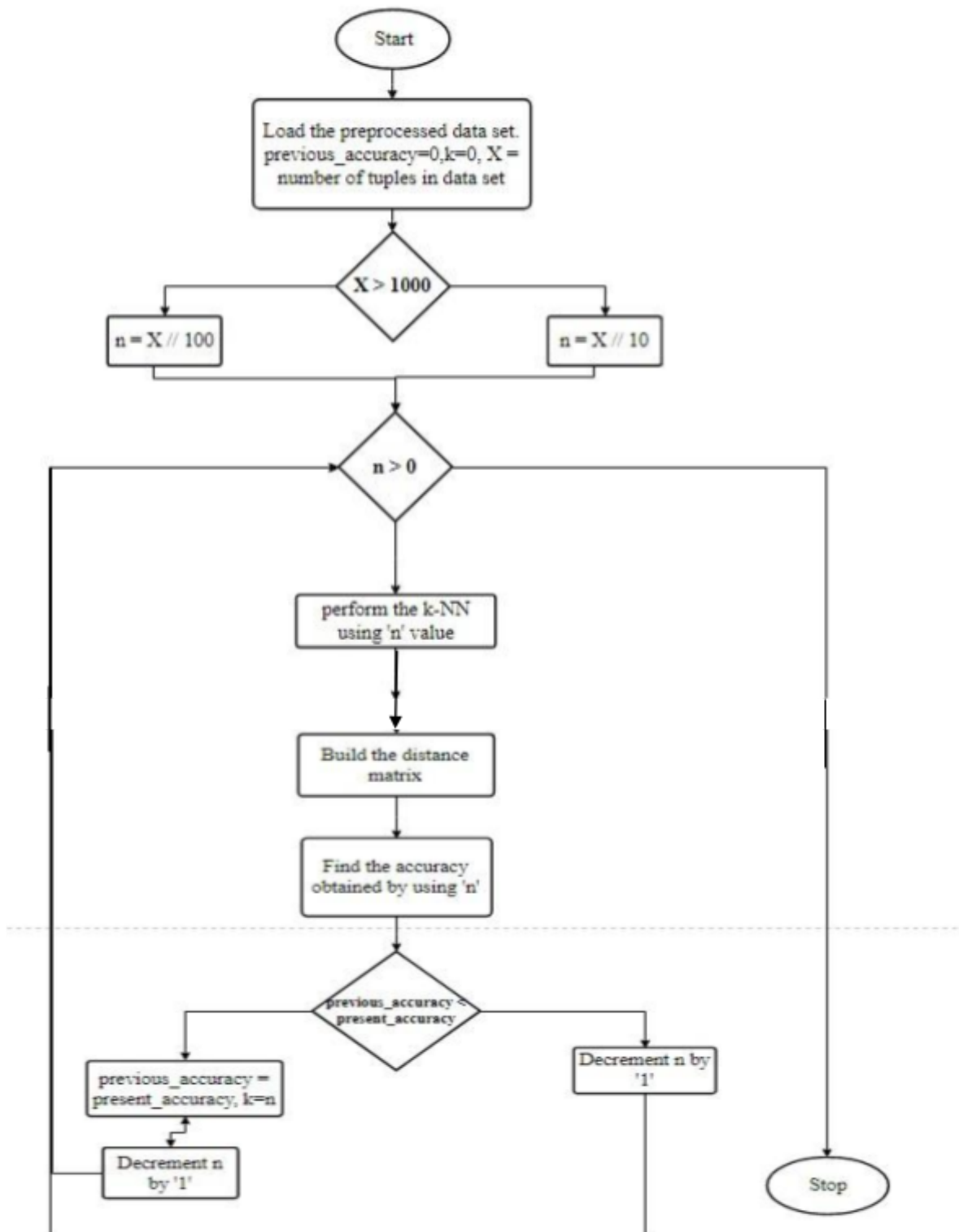


Fig 4.1.1 Flow Chart Flowcharts use simple geometric symbols and arrows to define relationships.

Some of them are as follows:

- The beginning or end of a program is represented by an oval.

- A process is represented by a rectangle.
- A decision is represented by a diamond.
- An I/O process is represented by a parallelogram.

4.1.2 SEQUENCE DIAGRAM:

A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagram represents the sequential flow of the system based on the time event.

Sequence Diagrams captures:

- the interaction that takes place in a collaboration that either realizes a use case or an operation (instance diagrams or generic diagrams)
- high-level interactions between user of the system and the system, between the system and other systems, or between subsystems (sometimes known as system sequence diagrams)

Purpose of Sequence Diagram

- Model high-level interaction between active objects in a system
- Model the interaction between object instances within a collaboration that realizes a use case
- Model the interaction between objects within a collaboration that realizes an operation
- Either model generic interactions (showing all possible paths through the interaction) or specific instances of a interaction (showing just one path through the interaction)

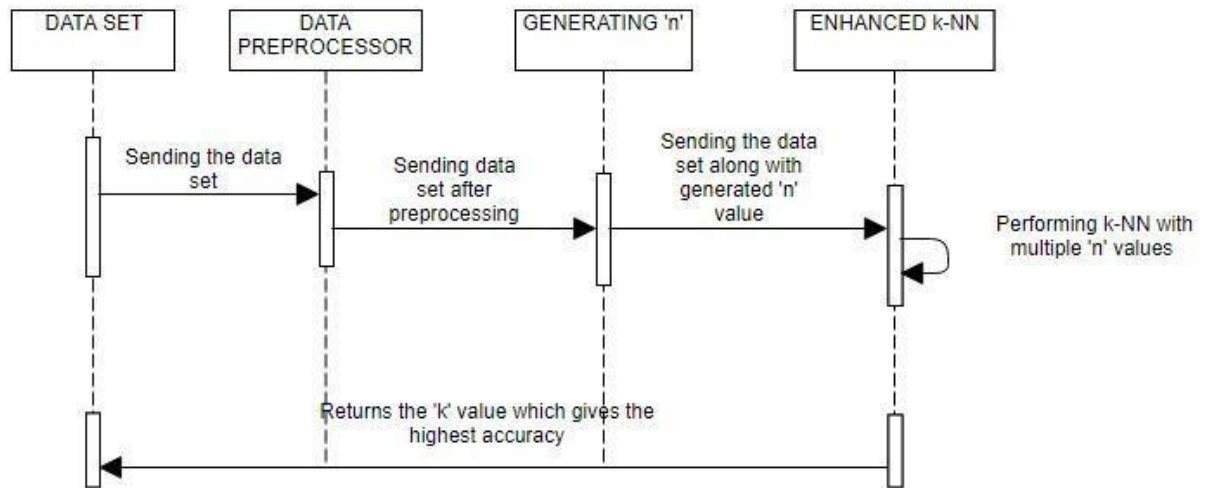


Fig 4.1.3 Sequence diagram

4.1.3 COMPONENT DIAGRAM

Component diagrams are used in modeling the physical aspects of object-oriented systems that are used for visualizing, specifying, and documenting component-based systems and also for constructing executable systems through forward and reverse engineering. Component diagrams are essentially class diagrams that focus on a system's components that often used to model the static implementation view of a system.

A component diagram breaks down the actual system under development into various high levels of functionality. Each component is responsible for one clear aim within the entire system and only interacts with other essential elements on a need-to-know basis.

A component diagram consists of following:

1. A rectangle with the component's name
2. A rectangle with the component icon
3. A rectangle with the stereotype text and/or icon

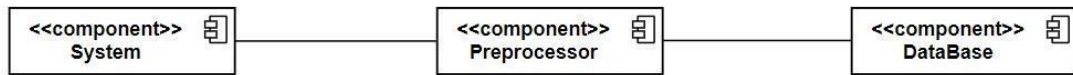


Fig: - 4.1.3 Component diagram

4.1.4 ACTIVITY DIAGRAM

An activity diagram shows the structure of a process or the flow of control and data from step to step within the computation. Activity diagrams are constructed from a limited number of shapes, connected with arrows.

The most important shape types:

- Rounded rectangles represent actions;
- Diamonds represent decisions;
- Bars represent the start (split) or end (join) of concurrent activities;
- A black circle represents the start (initial state) of the workflow;
- An encircled black circle represents the end (final state).

Arrows run from the start towards the end and represent the order in which activities happen.

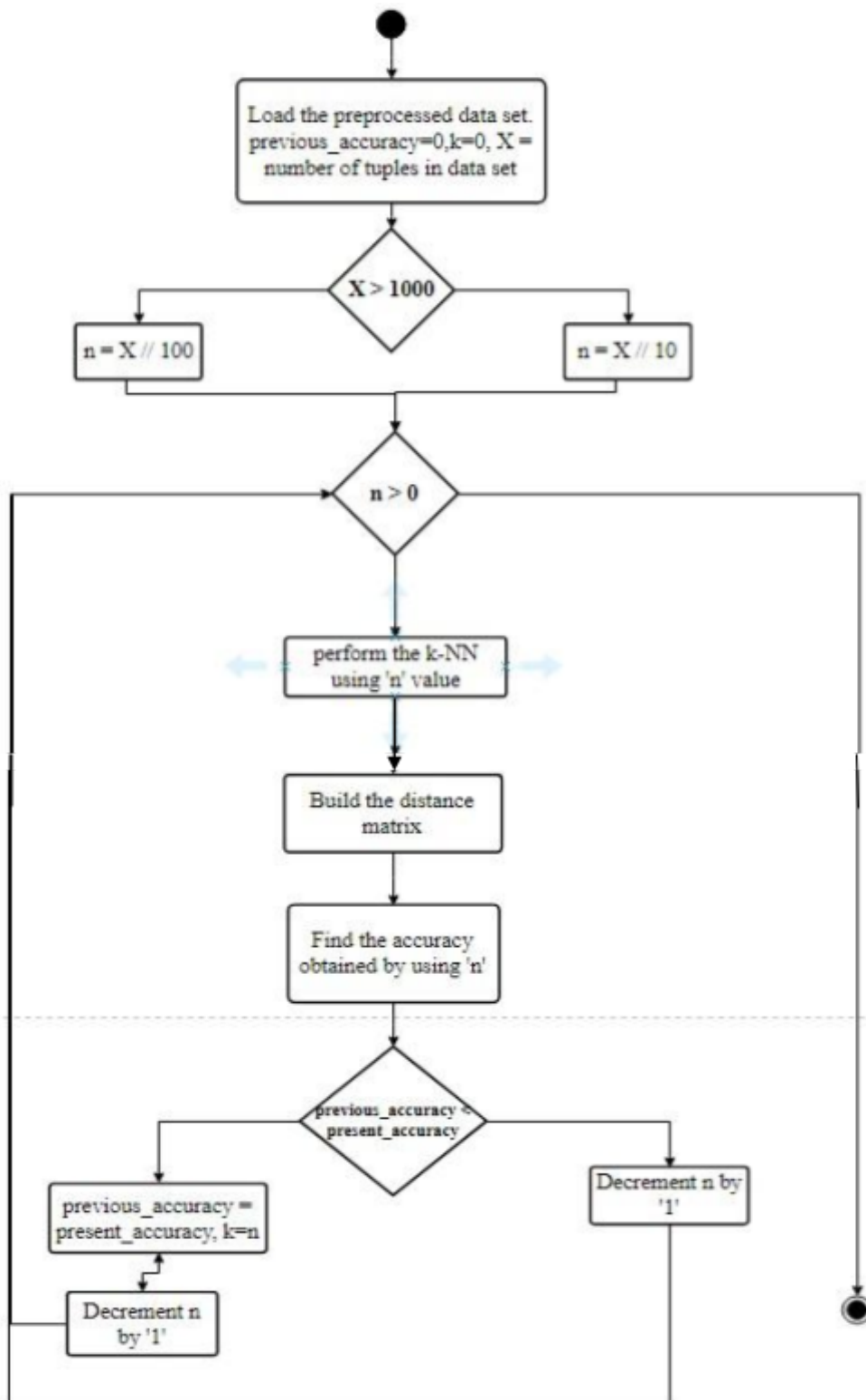


Fig: - 4.1.4 Activity diagram
LIST OF ACTIVITIES

User

System
Upload dataset
Dataset loaded
Apply Data Preprocessing
Split Training and Testing data
Apply k -NN
Find best k for the data set
Find the Accuracy
Display k and Accuracy

Table 4.1.4 Activity Diagram

CHAPTER-5

IMPLEMENTATION

Python:

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and objectoriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically typed and garbage-collected, ever since its inception has been one of the most widely used languages along with C++, Java, etc.

In python there are many pre-defined packages which can be easily installed and run on our python platform. So we have used those packages to make our work easy and faster when compared to other programming platforms.

There are some packages for performing Enhanced k -NN. They are:

- tkinter
- sklearn
- pandas

TKINTER

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps –

- Import the tkinter module.
- Create the GUI application main window.
- Add one or more of the above-mentioned widgets to the GUI application.
- Enter the main event loop to take action against each event triggered by the user

Tkinter Widgets:

Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.

There are many widgets available in tkinter. Some of them which are used in our project are as follows:

1. Button
2. Label
3. Entry

For importing files from local directory python provides a sub-packages in tkinter called “**filedialog**”

File dialog:

File dialogs help you open, save files or directories. This is the type of dialog you get when you click file, open. This dialog comes out of the module, there's no need to write all the code manually.

The tkinter filedialog comes in several types. Which type you need really depends on your applications needs. All of them are methods calls.

Algorithm: Uploading a file

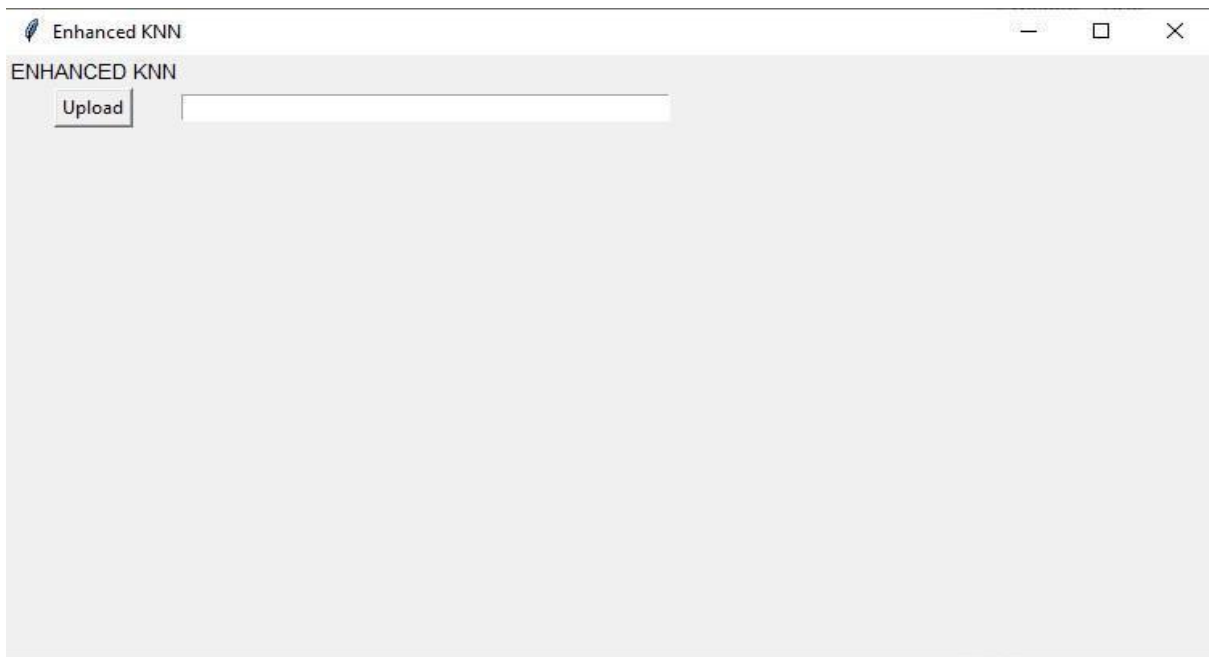
Input: Click on Upload Button

Output: File uploaded

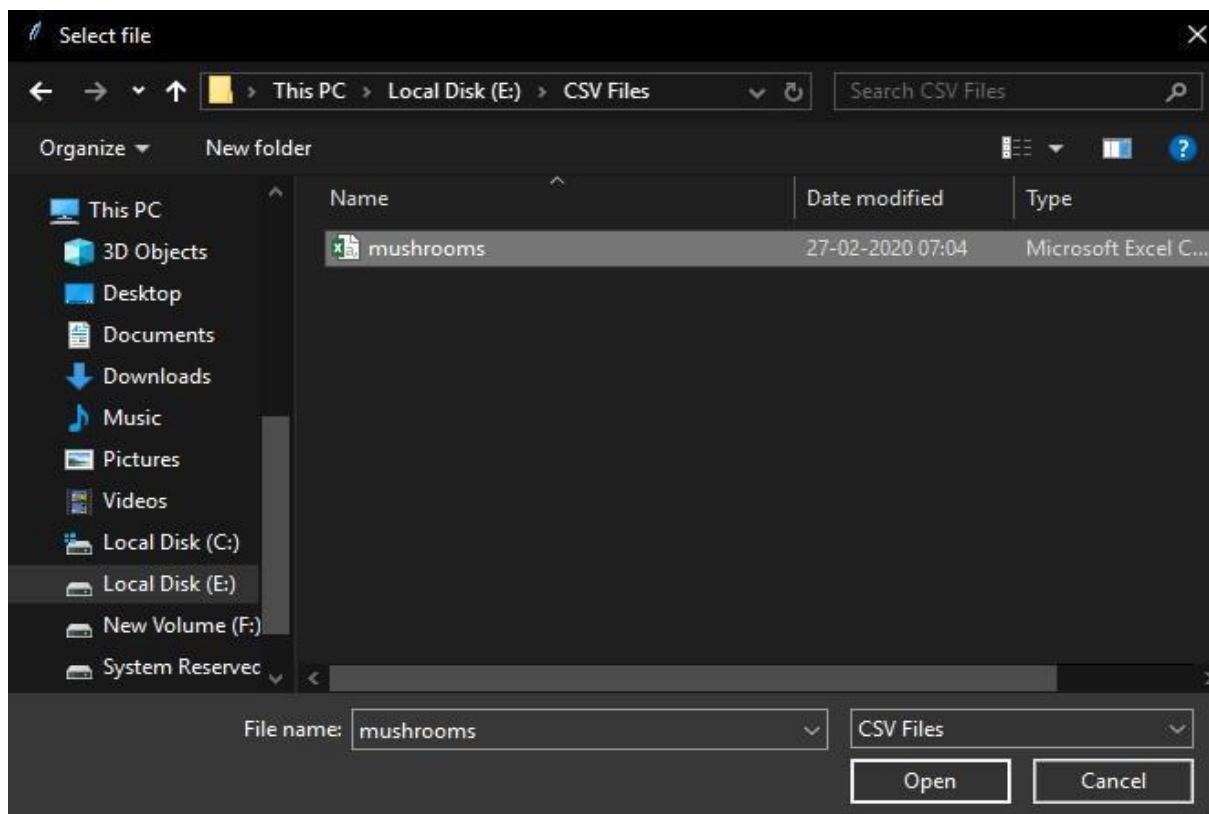
Begin

1. import tkinter as tk
2. from tkinter import filedialog
3. window = tk.Tk()
4. window.title("Enhanced KNN")
5. lbl = tk.Label(window, text = "ENHANCED KNN", font = ("Arial", 10))
6. lbl.grid(column = 0, row = 0)
7. txt = tk.Entry(window, font = ("Times new roman", 8), width = 50)
8. txt.grid(column = 1, row = 2)
9. def Uploading(event = None):
10. filename = filedialog.askopenfilename(title = "Select file", filetypes = (("CSV Files", "*.csv"), ("All", "*.*")))
11. txt.insert(0, filename)
12. b1 = tk.Button(window, text = "Upload", command = Uploading)
13. b1.grid(column = 0, row = 2, padx = 2)
14. window.geometry('750x400')
15. window.mainloop()

End



Clicking the upload button will display a window which allows user to select the files which are of format “.csv”.



SKLEARN

Scikit-learn is a free machine learning library for Python. It features various algorithms like support vector machine, random forests, and k-neighbors, and it also supports Python numerical and scientific libraries like Numpy and Scipy.

Scikitlearn. Neighbors provides functionality for unsupervised and supervised neighborsbased learning methods. Unsupervised nearest neighbors is the foundation of many other learning methods, notably manifold learning and spectral clustering. Supervised neighborsbased learning comes in two flavours: classification for data with discrete labels, and regression for data with continuous labels.

The principle behind nearest neighbor methods is to find a predefined number of training samples closest in distance to the new point, and predict the label from these. The number of samples can be a user-defined constant (k-nearest neighbor learning), or vary based on the local density of points (radius-based neighbor learning). The distance can, in general, be any metric measure: standard Euclidean distance is the most common choice. Neighbors-based methods are known as non-generalizing machine learning methods, since they simply

“remember” all of its training data (possibly transformed into a fast indexing structure such as KD tree).

Despite its simplicity, nearest neighbors has been successful in a large number of classification and regression problems, including handwritten digits and satellite image scenes. Being a non-parametric method, it is often successful in classification situations where the decision boundary is very irregular.

Nearest Neighbor Classification

Neighbors-based classification is a type of instance-based learning or non-generalizing learning: it does not attempt to construct a general internal model, but simply stores instances of the training data. Classification is computed from a simple majority vote of the nearest neighbors of each point: a query point is assigned the data class which has the most representatives within the nearest neighbors of the point.

Scikit-learn implements two different nearest neighbors classifiers: KNeighborsClassifier implements learning based on the k nearest neighbors of each query point, where k is an integer value specified by the user. RadiusNeighborsClassifier implements learning based on the number of neighbors within a fixed radius r of each training point, where r is a floating point value specified by the user.

The k-neighbors classification in `KNeighboursClassifier` is the most commonly used technique. The optimal choice of the value k is highly data-dependent: in general a larger k suppresses the effects of noise, but makes the classification boundaries less distinct.

In cases where the data is not uniformly sampled, radius-based neighbors classification in `RadiusNeighborsClassifier` can be a better choice. The user specifies a fixed radius r , such that points in sparser neighborhoods use fewer nearest neighbors for the classification. For high-dimensional parameter spaces, this method becomes less effective due to the so-called “curse of dimensionality”.

The basic nearest neighbors classification uses uniform weights: that is, the value assigned to a query point is computed from a simple majority vote of the nearest neighbors. Under some circumstances, it is better to weight the neighbors such that nearer neighbors contribute more to the fit. This can be accomplished through the `weights` keyword. The default value, `weights='uniform'`, assigns uniform weights to each neighbor. `weights='distance'` assigns weights proportional to the inverse of the distance from the query point. Alternatively, a user-defined function of the distance can be supplied to compute the weights.

PANDAS

Pandas is an open-source, BSD-licensed Python library providing high-performance, and easy-to-use data structures and data analysis tools for the Python programming language. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Prior to Pandas, Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data — load, prepare, manipulate, model, and analyse.

Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Features of Pandas:

- Fast and efficient Data Frame object with default and customized indexing.
- Tools for loading data into in-memory data objects from different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of date sets.
- Label-based slicing, indexing and sub setting of large data sets.
- Columns from a data structure can be deleted or inserted.
- Group by data for aggregation and transformations.
- High performance merging and joining of data.
- Time Series functionality.

To read files like comma - separated values (csv) file from local directory to perform k-NN on that data set.

```
import pandas as pd  
data = pd.read_csv('file_path')
```

The function `read_csv()` is a pandas in-built function which reads csv files from the local directory.

The loaded file can be stored in a variable for further process. In the above code the csv file that is read by `pd.read_csv()` function is stored in 'data' variable.

Source Code:

```
1.  import tkinter as tk
2.  from tkinter import filedialog
3.  window = tk.Tk()
4.  window.title("Enhanced KNN")
5.  lbl = tk.Label(window, text="ENHANCED KNN", font=("Arial", 10))
6.  lbl.grid(column=0, row=0)
7.  txt = tk.Entry(window, font=("Times new roman", 8), width=50)
8.  txt.grid(column=1, row=2)
9.  def Uploading(event = None):
10.  filename = filedialog.askopenfilename(title = "Select file",filetypes = (("CSV
    Files","*.csv"),("All","*..*")))
11.  txt.insert(0, filename)
12.  b1 = tk.Button(window, text="Upload", command=Uploading)
13.  b1.grid(column=0, row=2, padx=2)
14.  def ok_clicked():
15.  from sklearn.neighbors import KNeighborsClassifier
16.  from sklearn import datasets
17.  from sklearn import metrics
18.  from sklearn.model_selection import train_test_split
19.  import pandas as pd
20.  data = pd.read_csv(txt.get())
21.  data["class"] = [1 if i == "p" else 0 for i in data["class"]]
22.  data.drop("veil-type",axis=1,inplace=True)
23.  for column in data.drop(["class"], axis=1).columns:
24.  value = 0
25.  step = 1/(len(data[column].unique())-1)
26.  for i in data[column].unique():
27.  data[column] = [value if letter == i else letter for letter in data[column]]
```

```

28. value += step
29. y = data['class'].values
30. X = data.drop(["class"], axis=1).values
31. if( len(X)>1000):
32.     n=len(X)//100
33. else:
34.     n=len(X)//10
35.     prev_acc=0;k=0
36.     while(n>0):
37.         x_train,x_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=42)
38.         neigh = KNeighborsClassifier(n_neighbors=n)
39.         neigh.fit(x_train, y_train)
40.         y_pred=neigh.predict(x_test)
41.         pres_acc=metrics.accuracy_score(y_test, y_pred)
42.         if(prev_acc<pres_acc and pres_acc!=1.0):
43.             prev_acc=pres_acc
44.             k=n
45.         n-=1
46.         lb3 = tk.Label(window, text=k)
47.         lb3.grid(column=1, row=4)
48.         lb5 = tk.Label(window, text=format(pres_acc*100)+"%")
49.         lb5.grid(column=1, row=5)
50.         print("For k value as {}".format(k if k!=0 else 1),"we got the highest accuracy
percentage : {}".format(pres_acc*100))
51.         def cancel_clicked():
52.             txt.delete(0, 'end')
53.             b2 = tk.Button(window, text="Ok", command=ok_clicked)
54.             b2.grid(column=1, row=3)
55.             b3 = tk.Button(window, text="Cancel", command=cancel_clicked)
56.             b3.grid(column=2, row=3)
57.             lb2 = tk.Label(window, text=" The Best 'K' value for the above data set is =",
font=("Times new roman", 8))
58.             lb2.grid(column=0, row=4)
59.             lb4 = tk.Label(window, text=" The Highest accuracy obtained by using the above
'K' value for the data set is =", font=("Times new roman", 8))

```


- 60.** `lb4.grid(column=0, row=5)`
- 61.** `window.geometry('750x400')`
- 62.** `window.mainloop()`

CHAPTER-6

SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or a finished product it is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

The aim of the testing process is to identify all defects in a software product. It is not possible to guarantee that the software is error free. This is because of the fact that the input data domain of most software projects is very large. We can safely conclude that testing provides way of reducing defects in a system and increasing the user's confidence in a developed system.

6.1 Test Cases

Testing the input by checking whether the input data i.e. taking input as only xmi files. Checking whether extraction of data has done successfully on given files, finding correlation, calculating mean and standard deviation, storing past projects data, maintaining recent history and user accessing has done successfully by using manual testing.

Test Case Name	Uploading a file.
Test Case Description	Clicking on Upload button and accepting file from local directory.
Input Parameters	Click on Upload button.
Requirements Required	Single Click.
Expected Results	Shows a window that displays files from local directory And selecting a file from it of type CSV.
Actual Results	A window of local directory files and allows user to select a file of type CSV.
Test Result	Pass
Remarks	No Remarks

The tables 6.1 to 6.4 shows the test cases written on Effort Estimation Tool.

Table: 6.1 Test Case-1

Table: 6.2 Test Case-2

Test Case Name	Displaying the path of file.
Test Case Description	A Entry field that displays the path of selected file.
Input Parameters	A CSV file selected by user.
Requirements Required	Single Click
Expected Results	After clicking the upload d button, It has to direct to location of the data set.
Actual Results	After clicking the upload button, It has to direct to location of the data set.
Test Result	Pass
Remarks	No Remarks

Table: 6.3 Test Case-3

Test Case Name	Apply Enhanced k-NN
Test Case Description	Clicking on Ok
Input Parameters	click on ok
Requirements Required	Single Click
Expected Results	After clicking Ok button, it displays the best K and the accuracy obtained by using that K for the data set. .
Actual Results	After clicking Ok button, it displays the best K and the accuracy obtained by using that K for the data set.
Test Result	Pass
Remarks	No Remarks

Table: 6.4 Test Case-4

Test Case Name	Clear the Entry
Test Case Description	Clicking on Cancel to clear the path of file and to select a new file.
Input Parameters	Click on Cancel.
Requirements Required	Single Click.
Expected Results	After clicking Cancel, it clears the path in the entry field.
Actual Results	After clicking Cancel, it clears the path in the entry field.
Test Result	Pass
Remarks	No Remarks

6.2 TYPES OF TESTS

6.2.1 Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

6.2.2 Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

6.2.3 Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centred on the following items:

- Valid Input : Identified classes of valid input must be accepted.
- Invalid Input : Identified classes of invalid input must be rejected.
- Functions : Identified functions must be exercised.
- Output : Identified classes of application outputs must be exercised.
- Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must

be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

6.2.4 System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

6.2.5 White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

6.2.6 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

6.3 RESULTS

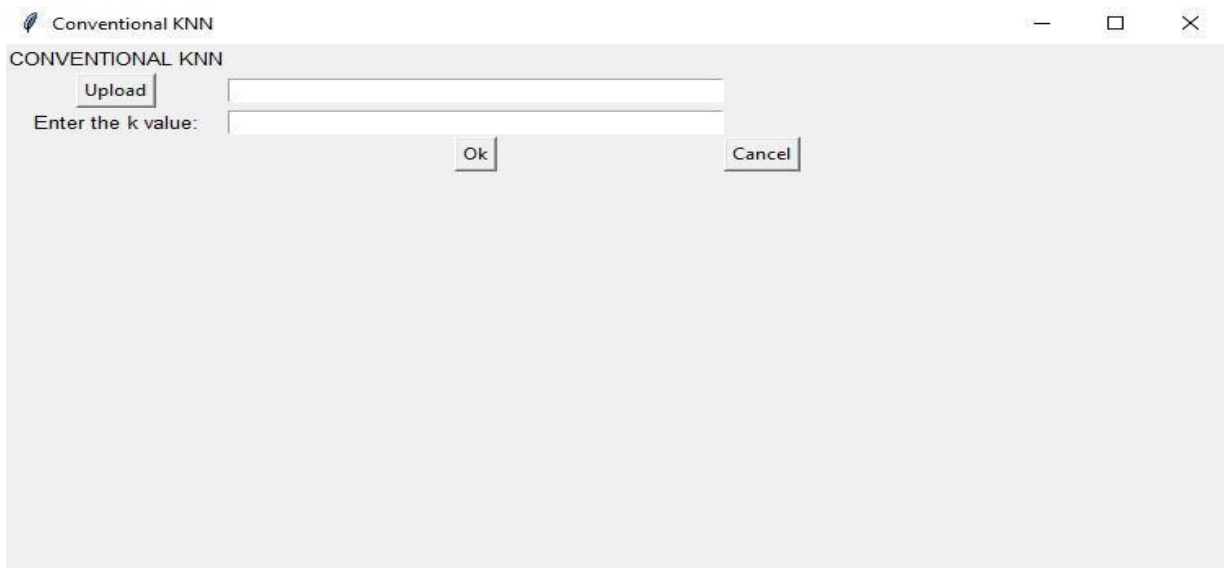


Fig 6.3.1 Home Screen of Conventional k -NN

Description: This is the home screen of Conventional k -NN. Click on Upload to move on to next step.

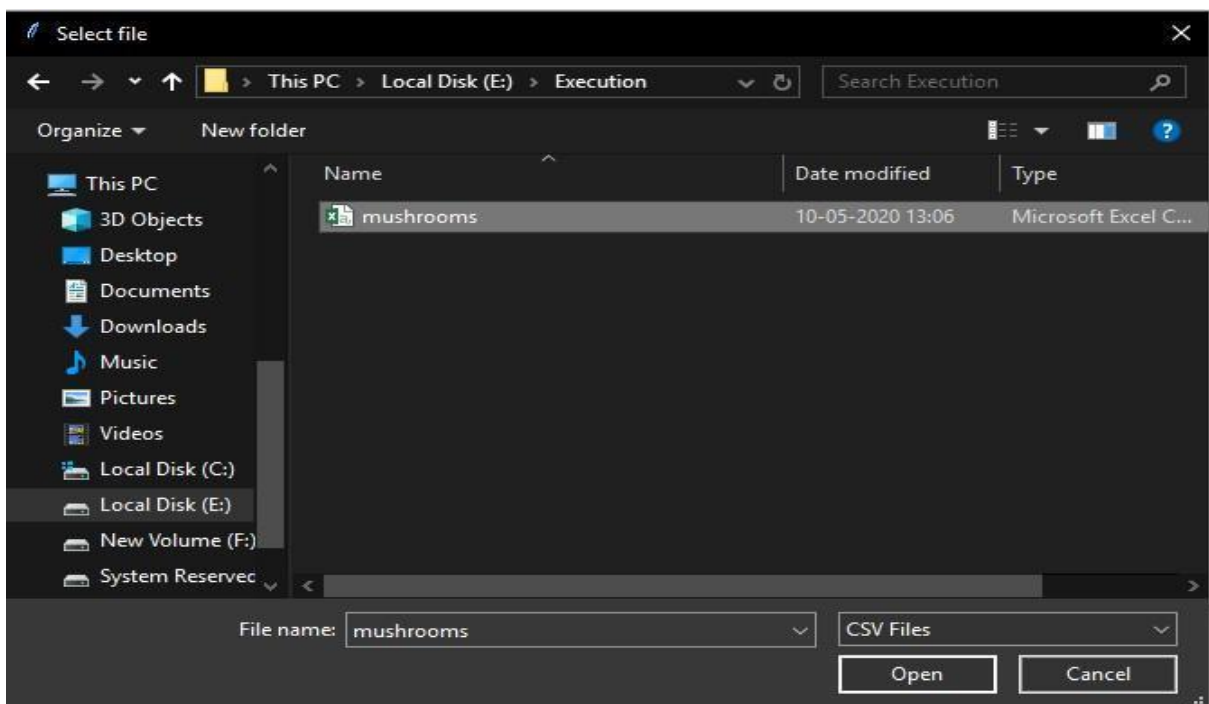


Fig 6.3.2 selecting a CSV file to Conventional k -NN

Description: This is the screen which allows the user to select a CSV (comma separated values) file from the local directory. Click on open to move to next screen.

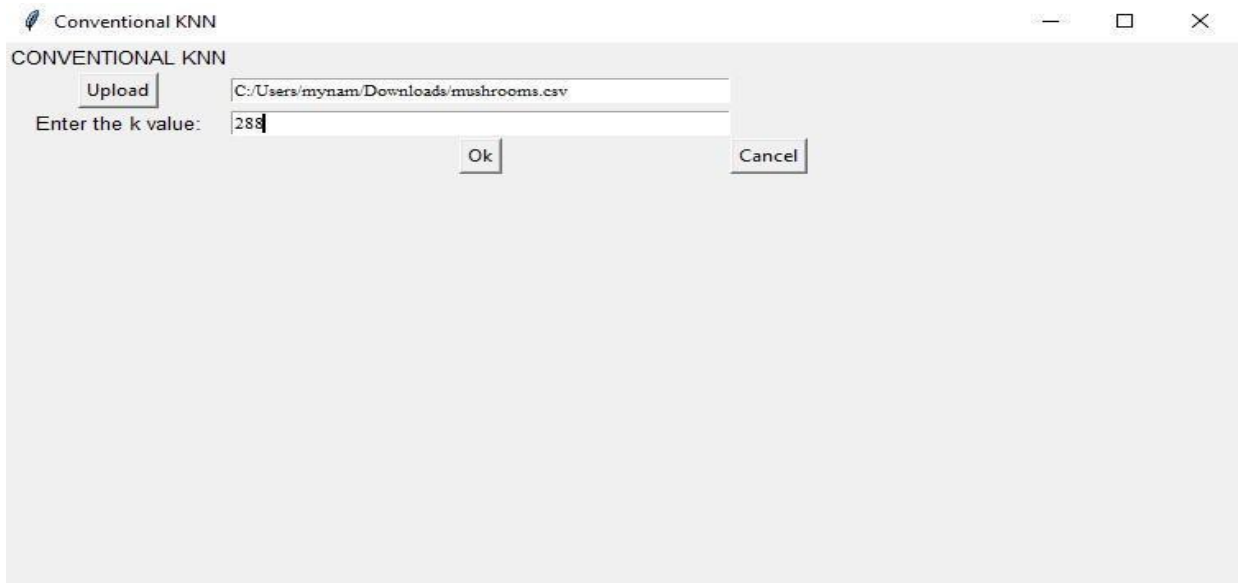


Fig 6.3.3 After Uploading a CSV file and entering ‘ k ’ value

Description: This screen shows the path of the CSV file that is selected from Local directory.

```

Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 22:22:05) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\mynam\Desktop\Batch_11\conventiona_k-NN.py =====
For k value as 281 we got the highest accuracy percentage : 95.13846153846154%
[[793  50]
 [ 29 753]]
The precision obtained: 0.9406880189798339
The Recall obtained: 0.9647201946472019
|

```

Fig 6.3.4 Accuracy, precision and recall obtained by conventional k -NN

Description: This screen shows the accuracy, confusion matrix, precision and recall obtained by performing the conventional k -NN with a user defined ‘ k ’ value.



Fig. 6.3.5 Home Screen of Enhanced k -NN

Description: This is the home screen of Enhanced k -NN. Click on Upload to move on to next page.

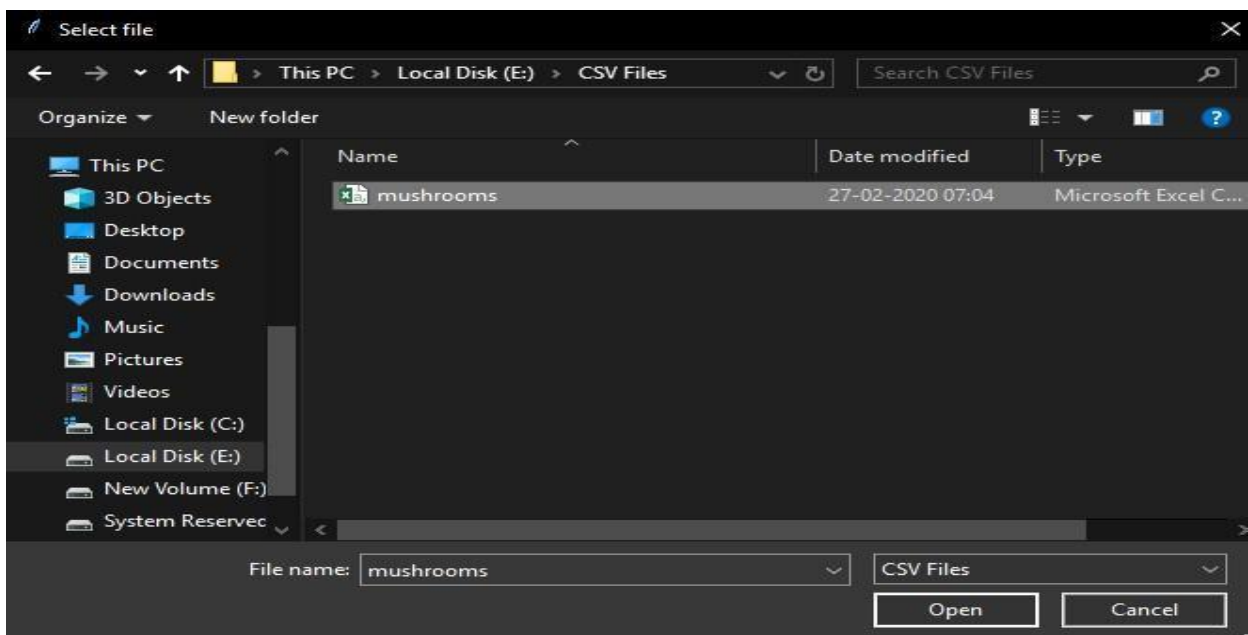


Fig. 6.3.6 selecting a CSV file to Enhanced k -NN

Description: This is the screen which allows the user to select a CSV (comma separated values) file from the local directory. Click on open to move to next screen.

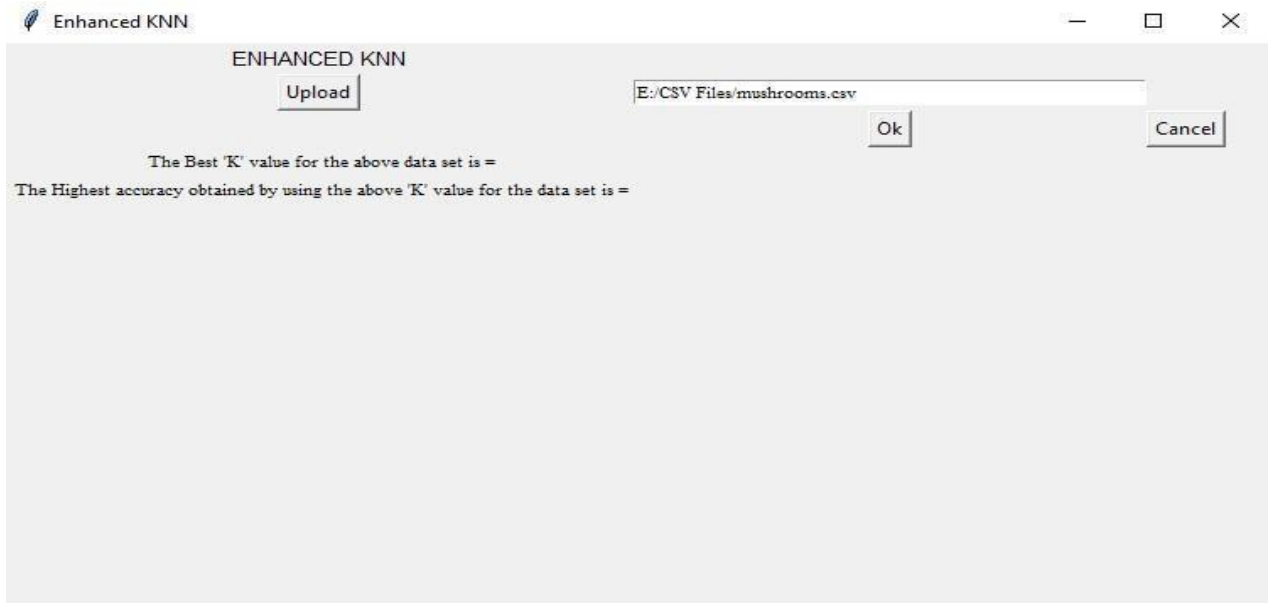


Fig. 6.3.7 after selecting a file

Description: This screen shows the path of the file in Entry field that was selected from by user from local directory. Click on Ok to move on to next screen.

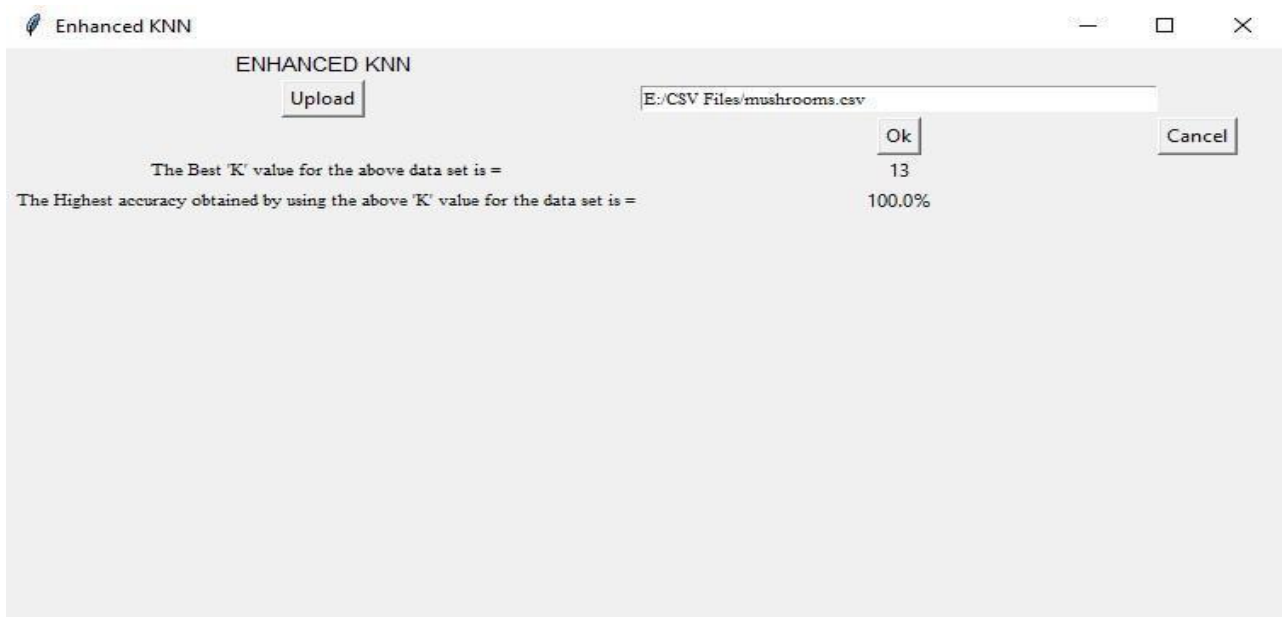


Fig. 6.3.8 after performing Enhanced k-NN

Description: This screen shows the best 'k' value and the accuracy obtained by using that value for the data set after performing Enhanced k-NN is displayed.

```

Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 22:22:05) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\mynam\Desktop\Batch_11\my_gui_for_project.py =====
For k value as 13 we got the highest accuracy percentage : 100.0%
[[843  0]
 [ 0 782]]
The precision obtained: 1.0
The Recall obtained: 1.0
|

```

Fig 6.3.9 Accuracy, precision and recall obtained by Enhanced k -NN

Description: This screen shows the accuracy, confusion matrix, precision and recall obtained after performing Enhanced k -NN.

	Conventional k -NN			Enhanced k -NN		
	Precision	Recall	Accuracy (%)	Precision	Recall	Accuracy (%)
Mushrooms data set	0.94068	0.96472	95.1384	1.0	1.0	100
Iris data set	0.91225	0.93543	92.81793	0.95367	0.98435	98.77142
Average	0.926465	0.950075	93.978165	0.976835	0.992175	99.38571

Table 6.3.1 comparing the results of Conventional k -NN & Enhanced k -NN

CHAPTER-7

CONCLUSION

The algorithm that is designed to perform recursive k -NN called Enhanced k -NN provides results accurately when compared to conventional k -NN. To increase the performance of k -NN we have dynamically determined the k value and for the new instance the derived k accuracy will be high. Though it takes heavy time while training the model it achieves the highest accuracy. It can also achieve maximum of 100 percent accuracy because it is a lazy learning model. The choice of parameter k is done automatically in Enhanced k -NN which reduces the pressure on the user to provide the accurate value of ' k ' for the data set. The experimental results proves that the proposed algorithm performs better than the conventional classification method.

REFERENCE

- [1] Z. H. Zhou, N. V. Chawla, Y. C. Jin, and G. J. Williams, Big data opportunities and challenges: Discussions from data analytics perspectives, *IEEE Comput. Intell. Mag.*, vol. 9, no. 4, pp. 62–74, 2014.
- [2] Y. T. Zhai, Y. S. Ong, and I. W. Tsang, The emerging “big dimensionality”, *IEEE Comput. Intell. Mag.*, vol. 9, no. 3, pp. 14–26, 2014.
- [3] E. Fix and J. L. Jr. Hodges, Discriminatory analysis nonparametric discrimination: Consistency properties, *Int. Stat. Rev.*, vol. 57, no. 3, pp. 238–247, 1951.
- [4] T. Cover and P. Hart, nearest neighbor pattern classification, *IEEE Trans. Inf. Theory*, vol. 13, no. 1, pp. 21–27, 1967.
- [5] H. Zhang, A. C. Berg, M. Maire, and J. Malik, SVM- KNN: Discriminative nearest neighbor classification for visual category recognition, in *Proc. 2006 IEEE Computer Society Conf. Computer Vision and Pattern Recognition*, New York, NY, USA, 2006, pp. 2126–2136.
- [6] S. H. Wang, Q. M. Huang, S. Q. Jiang, Q. Tian, and L. Qin, Nearest-neighbor method using multiple neighborhood similarities for social media data mining, *Neurocomputing*, vol. 95, pp. 105–116, 2012.
- [7] O. Kucuktunc and H. Ferhatosmanoglu, -diverse nearest neighbors browsing for multidimensional data, *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 3, pp. 481–493, 2013.
- [8] D. Lunga and O. Ersoy, Spherical nearest neighbor classification: Application to hyperspectral data, in *Machine Learning and Data Mining in Pattern Recognition*, P. Perner, ed. Springer, 2011.
- [9] P. Horton and K. Nakai, Better prediction of protein cellular localization sites with the k nearest neighbors classifier, *Proc. Int. Conf. Intell. Syst. Mol. Biol.*, vol. 5, pp. 147–152, 1997.
- [10] R. M. Parry, W. Jones, T. H. Stokes, J. H. Phan, R. A. Moffitt, H. Fang, L. Shi, A. Oberthuer, M. Fischer, W. Tong, et al., K-nearest neighbor models for microarray gene

expression analysis and clinical outcome prediction, *Pharmacogenomics J.*, vol. 10, no. 4, pp. 292–309, 2010.

[11] J. Macqueen, Some methods for classification and analysis of multivariate observations, in *Proc. the 5th Berkeley Symp. Mathematical Statistics and Probability*, Berkeley, CA, USA, 1967, pp. 281–297.

[12] N. S. Altman, An introduction to kernel and nearest- neighbor nonparametric regression, *Am. Stat.*, vol. 46, no. 3, pp. 175–185, 1992.

[13] G. R. Terrell and D. W. Scott, Variable kernel density estimation, *Ann. Stat.*, vol. 20, no. 3, pp. 1236–1265, 1992.

[14] M. M. Breunig, H. P. Kriegel, R. T. Ng, and J. Sander, LOF: Identifying density-based local outliers, *ACM SIGMOD Record*, vol. 29, no. 2, pp. 93–104, 2000.

[15] H. Zhang and J. Malik, Learning a discriminative classifier using shape context distances, in *Proc. 2003 IEEE Computer Society Conf. Computer Vision and Pattern Recognition*, Madison, WI, USA, 2003, pp. 242–247.

[16] Y. G. Chen, K. Chen, and M. A. Nascimento, Effective and efficient shape-based pattern detection over streaming time series, *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 2, pp. 265–278, 2012.

[17] B. Tang and H. B. He, ENN: Extended nearest neighbor method for pattern recognition, *IEEE Comput. Intell. Mag.*, vol. 10, no. 3, pp. 52–60, 2015.

[18] Y. Song, J. Huang, D. Zhou, H. Y. Zha, and C. Lee Giles, IKNN: Informative k-nearest neighbor pattern classification, in *Knowledge Discovery in Databases: PKDD 2007*, J. N.

Kok, J. Koronacki, R. L. de Mantaras, S. Matwin, D. Mladenic, and A. Skowron, eds. Springer, 2007, pp. 248–264.

[19] M. Z. Jahromi, E. Parvinnia, and R. John, A method of learning weighted similarity function to improve the performance of nearest neighbor, *Inf. Sci.*, vol. 179, no. 17, pp. 2964–2973, 2009.

[20] J. Toyama, M. Kudo, and H. Imai, Probably correct K-nearest neighbor search in high dimensions, *Pattern Recognit.*, vol. 43, no. 4, pp. 1361–1372, 2010.