Answer	Coding Efficiency	Viva	Timely Completion	Total	Dated Sign of Subject Teacher
5	5	5	5	20	

Expected Date of Completion:..... Actual Date of Completion:.....

Group B

Assignment No:4

Title of the Assignment: Implement K-Nearest Neighbors algorithm on diabetes.csv dataset. Compute confusion matrix, accuracy, error rate, precision and recall on the given dataset.

Dataset Description: This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective is to predict based on diagnostic measurements whether a patient has diabetes

Link for Dataset: https://www.kaggle.com/datasets/abdallamahgoub/diabetes

Objective of the Assignment:
Students should be able to learn the concept of K-Nearest Neighbours and Confusion Matrix

Prerequisite:

1. Basic knowledge of Python
2. Concept of Confusion Matrix
3. Concept of K-Nearest Neighbour

Contents of the Theory:

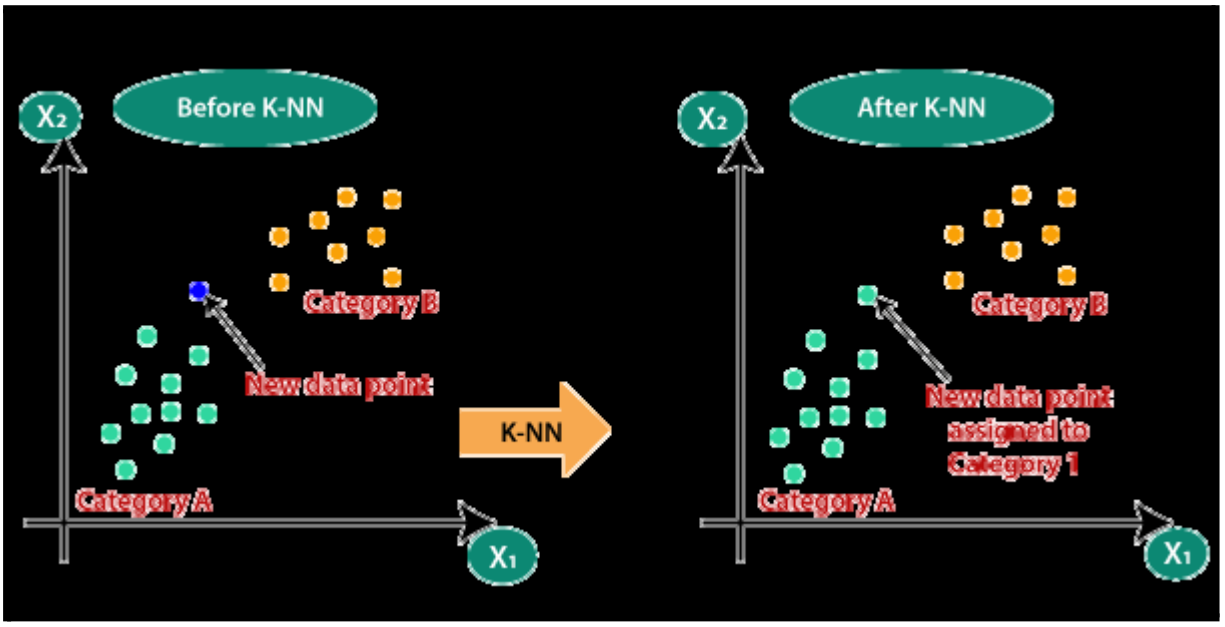
1. K-Nearest Neighbour
2. Confusion Matrix
3. Scikit learn

K-Nearest Neighbour:

- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifi es a new data point based on the similarity. This means when new data appears then it can be easily classifi ed into a well suite category by using K- NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classifi cation but mostly it is used for the Classifi cation problems.
- K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.
- It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classifi cation, it performs an action on the dataset.
- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifi es that data into a category that is much similar to the new data.
- **Example:** Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identifi cation, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will fi nd the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.

Why do we need a K-NN Algorithm?

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:

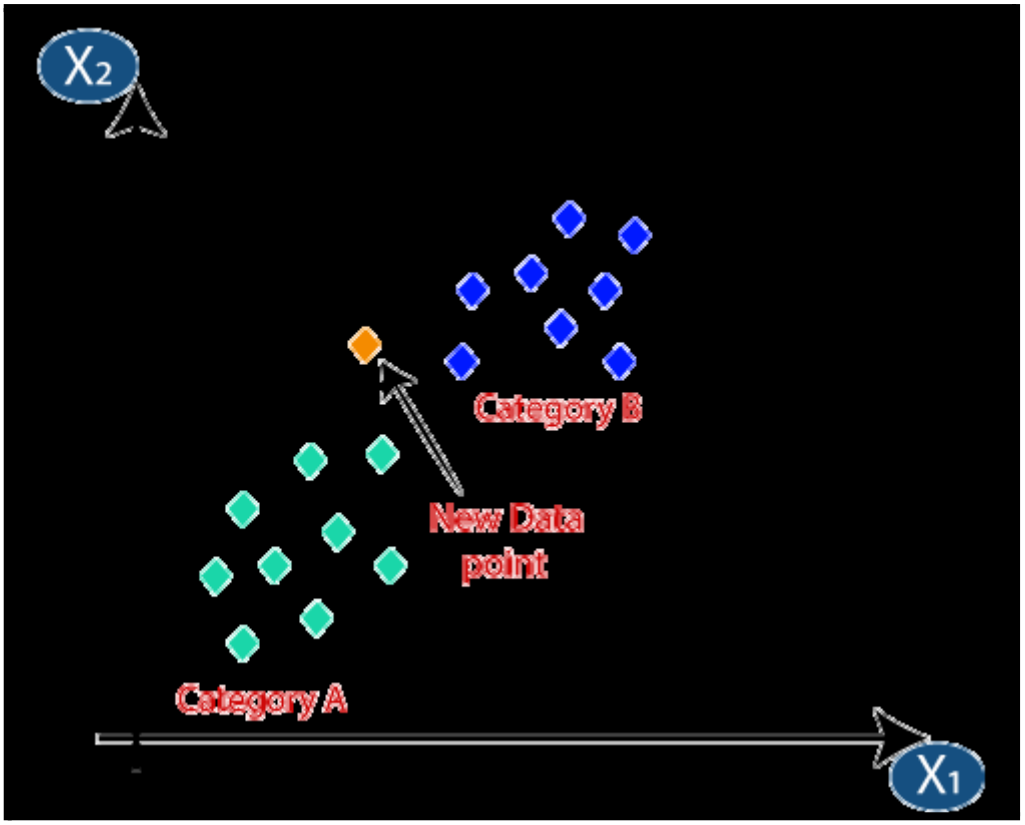


How does K-NN work?

The K-NN working can be explained on the basis of the below algorithm:

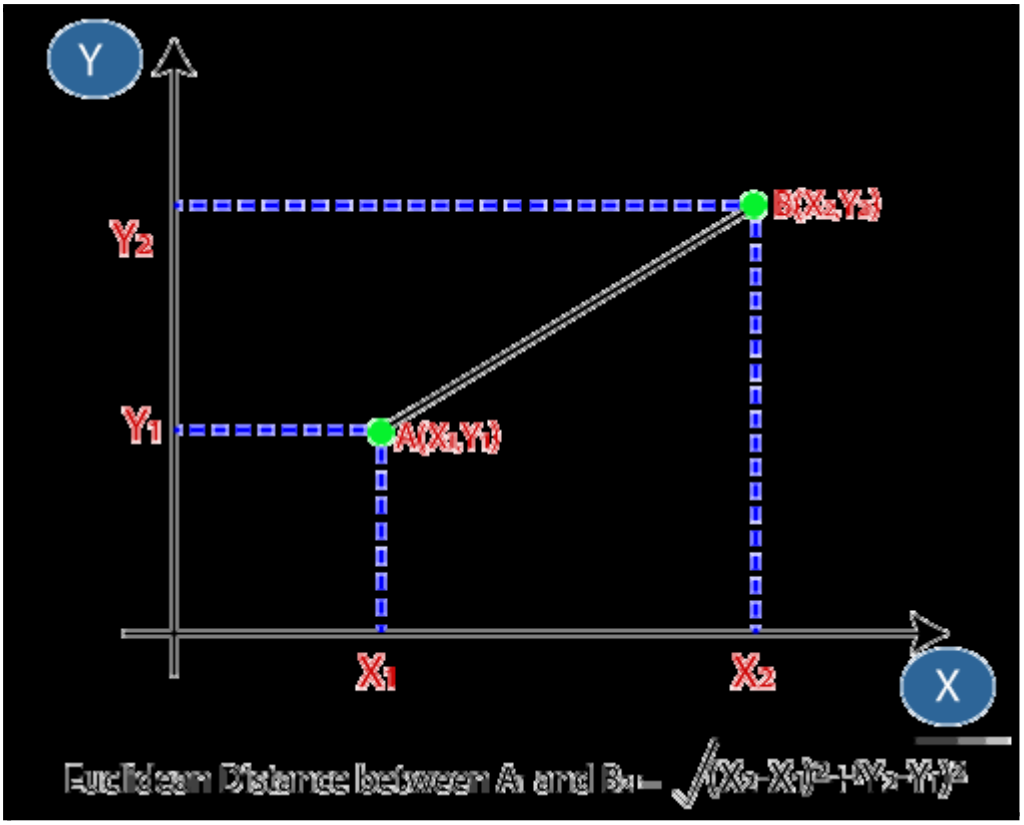
- **Step-1:** Select the number K of the neighbors
- **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
- **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
- **Step-6:** Our model is ready.

Suppose we have a new data point and we need to put it in the required category. Consider the below image:

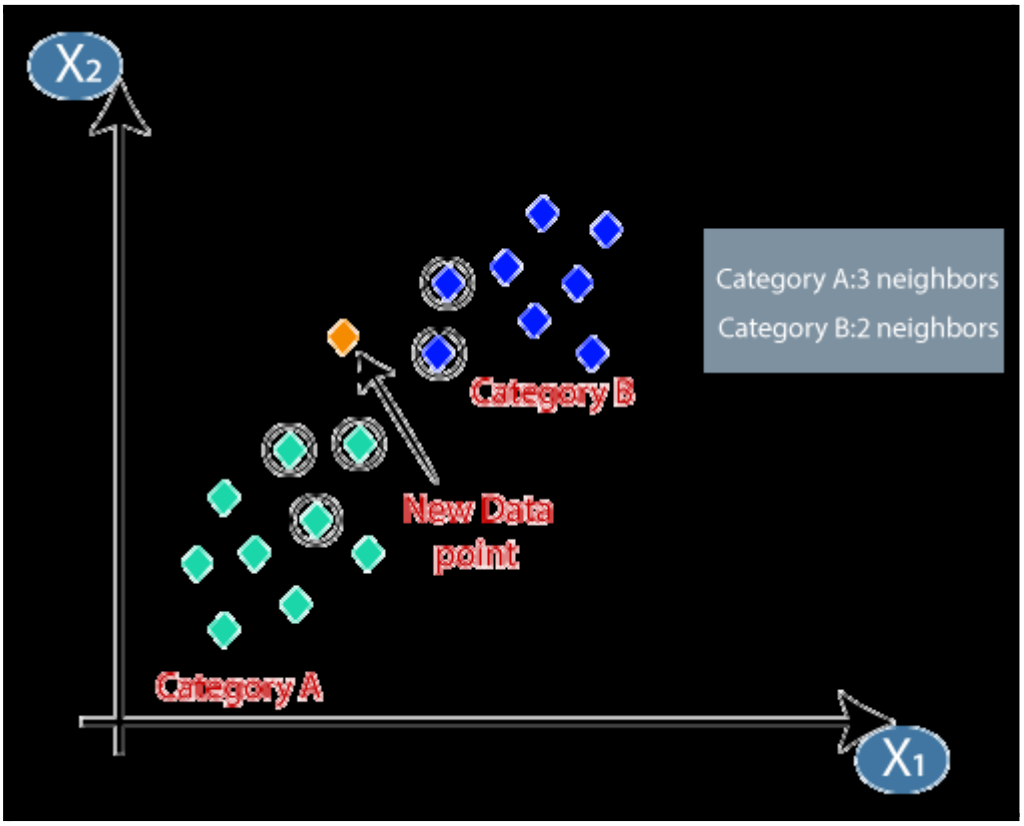


- Firstly, we will choose the number of neighbors, so we will choose the k=5.

- Next, we will calculate the **Euclidean distance** between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:



- By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:



Confusion Matrix:

The confusion matrix is a matrix used to determine the performance of the classifi cation models for a given set of test data. It can only be determined if the true values for test data are known. The matrix itself can be easily understood, but the related terminologies may be confusing. Since it shows the errors in the model performance in the form of a matrix, hence also known as an **error matrix**. Some features of Confusion matrix are given below:

- For the 2 prediction classes of classifi ers, the matrix is of 2*2 table, for 3 classes, it is 3*3 table, and so on.
- The matrix is divided into two dimensions, that are **predicted values** and **actual values** along with the total number of predictions.
- Predicted values are those values, which are predicted by the model, and actual values are the true values for the given observations.
- It looks like the below table:

The above table has the following cases:

- **True Negative:** Model has given prediction No, and the real or actual value was also No.
- **True Positive:** The model has predicted yes, and the actual value was also true.
- **False Negative:** The model has predicted no, but the actual value was Yes, it is also called as **Type-II error**.
- **False Positive:** The model has predicted Yes, but the actual value was No. It is also called a **Type-I error.**

Need for Confusion Matrix in Machine learning

- It evaluates the performance of the classifi cation models, when they make predictions on test data, and tells how good our classifi cation model is.
- It not only tells the error made by the classifi ers but also the type of errors such as it is either type-I or type-II error.
- With the help of the confusion matrix, we can calculate the different parameters for the model, such as accuracy, precision, etc.

Example: We can understand the confusion matrix using an example.

Suppose we are trying to create a model that can predict the result for the disease that is either a person has that disease or not. So, the confusion matrix for this is given as:

n = 100	Actual: No	Actual: Yes	
Predicted: No	TN: 65	FP: 3	68
Predicted: Yes	FN: 8	TP: 24	32
	73	27	

From the above example, we can conclude that:

- The table is given for the two-class classifi er, which has two predictions "Yes" and "NO." Here, Yes defi nes that patient has the disease, and No defi nes that patient does not has that disease.

- The classifier has made a total of **100 predictions**. Out of 100 predictions, **89 are true predictions**, and **11 are incorrect predictions**.
- The model has given prediction "yes" for 32 times, and "No" for 68 times. Whereas the actual "Yes" was 27, and actual "No" was 73 times.

Calculations using Confusion Matrix:

We can perform various calculations for the model, such as the model's accuracy, using this matrix. These calculations are given below:

- **Classification Accuracy:** It is one of the important parameters to determine the accuracy of the classification problems. It defines how often the model predicts the correct output. It can be calculated as the ratio of the number of correct predictions made by the classifier to all number of predictions made by the classifiers. The formula is given below:

$$\text{Accuracy}=\frac{TP+TN}{TP+FP+FN+TN}$$

- **Misclassification rate:** It is also termed as Error rate, and it defines how often the model gives the wrong predictions. The value of error rate can be calculated as the number of incorrect predictions to all number of the predictions made by the classifier. The formula is given below:

$$\text{Error rate}=\frac{FP+FN}{TP+FP+FN+TN}$$

- **Precision:** It can be defined as the number of correct outputs provided by the model or out of all positive classes that have predicted correctly by the model, how many of them were actually true. It can be calculated using the below formula:

$$\text{Precision}=\frac{TP}{TP+FP}$$

- **Recall:** It is defined as the out of total positive classes, how our model predicted correctly. The recall must be as high as possible.

$$\text{Recall}=\frac{TP}{TP+FN}$$

- **F-measure:** If two models have low precision and high recall or vice versa, it is difficult to compare these models. So, for this purpose, we can use F-score. This score helps us to evaluate the recall and precision at the same time. The F-score is maximum if the recall is equal to the precision. It can be calculated using the below formula:

$$\text{F-measure} = \frac{2*Recall*Precision}{Recall+Precision}$$

Other important terms used in Confusion Matrix:

- **Null Error rate:** It defines how often our model would be incorrect if it always predicted the majority class. As per the accuracy paradox, it is said that "*the best classifier has a higher error rate than the null error rate.*"
- **ROC Curve:** The ROC is a graph displaying a classifier's performance for all possible thresholds. The graph is plotted between the true positive rate (on the Y-axis) and the false Positive rate (on the x-axis).

Scikit Learn:

French research scientist David Cournapeau's scikits.learn is a Google Summer of Code venture where the scikit-learn project first began. Its name refers to the idea that it's a modification to SciPy called "SciKit" (SciPy Toolkit), which was independently created and published. Later, other programmers rewrote the core codebase.

The French Institute for Research in Computer Science and Automation at Rocquencourt, France, led the work in 2010 under the direction of Alexandre Gramfort, Gael Varoquaux, Vincent Michel, and Fabian Pedregosa. On February 1st of that year, the institution issued the project's first official release. In November 2012, scikit-learn and scikit-image were cited as examples of scikits that were "well-maintained and popular". One of the most widely used machine learning packages on GitHub is Python's scikit-learn.

Steps to Build a Model in Sklearn

Let us now learn the modelling process.

Step 1: Loading a Dataset

Simply put, a dataset is a collection of sample data points. A dataset typically consists of two primary parts:

Features: Features are essentially the variables in our dataset, often called predictors, data inputs, or attributes. A feature matrix, which is frequently symbolised by the letter "X," can be used to represent them since many of them may exist. The term "feature names" refers to a list of names of all the features.

Response: (sometimes referred to as the target feature, label, or output) Based on the variables feature, this variable is the output. In most cases, we only have one response column, which is depicted by a response column or vector (the letter 'y' is frequently used to denote a response vector). Target names refer to all the various values a response vector could take.

Step 2: Splitting the Dataset

The correctness of each machine learning model is a crucial consideration. Now, one may train a model with the provided dataset and then use that model to predict the target values for another set of the dataset to ascertain the correctness of the model.

To sum it up:

- Make a training dataset and a testing dataset out of the given dataset.
- On the practise set, train the model.
- Test the model using the testing dataset and assess its performance.

Step 3: Training the Model

It's time to use the training dataset to train the model, which will make predictions. A variety of machine learning techniques with an easy-to-use interface for fi tting, prediction accuracy, etc., are offered by Scikit-learn.

Our classifi er must now be tested using the testing dataset. For this, we can use the

.predict() model class method, giving back the predicted values.

By comparing the actual values of the testing dataset and the predicted values, we can assess the model's performance with the help of sklearn methods. The accuracy_score function of the metrics package is used for this.

Conclusion:

In this we have successfully implemented the above assignment.

Assignment Questions:

1) Explain Confusion Matrix ?
2) Explain Scikit Learn?
3) Explain the need of Confusion Matrix?
4) What Ratios can be calculated from the Confusion Matrix?