NITTE MEENAKSHI INSTITUTE OF TECHNOLOGY

(An autonomous institution affiliated to VTU)

Govindapura, Gollahalli, Bengaluru – 64

A Mini Project Report On

"Medical Anomaly Detection using KNN"

Submitted by

Khushi Ranganatha 1NT18EC078

Nikita Sharma 1NT18EC101

Under the Guidance of

Dr. Rajesh N

Professor, Dept of

ECE NMIT,

Bengaluru.

DEPARTMENT OF ELECTRONICS AND

COMMUNICATION ENGINEERING

NITTE MEENAKSHI INSTITUTE OF TECHNOLOGY

YELAHANKA – 560064

2021 – 2022

NITTE MEENAKSHI INSTITUTE OF TECHNOLOGY

(An autonomous institution affiliated to VTU)

Govindapura, Gollahalli, Bengaluru – 64

A Mini Project Report On

"Medical Anomaly Detection using KNN"

Submitted by

Khushi Ranganatha 1NT18EC078

Nikita Sharma 1NT18EC101

Under the Guidance of

Dr. Rajesh N

Professor, Dept of

ECE NMIT,

Bengaluru.

DEPARTMENT OF ELECTRONICS AND

COMMUNICATION ENGINEERING

NITTE MEENAKSHI INSTITUTE OF TECHNOLOGY

YELAHANKA – 560064

2021 – 2022

NITTE MEENAKSHI INSTITUTE OF TECHNOLOGY

(AN AUTONOMOUS INSTITUTE UNDER VTU, BELGAUM)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

Certificate

This is to certify that the project work entitled "**Medical Anomaly Detection using KNN**" is successfully completed by **Khushi Ranganatha (1NT18EC078) and Nikita Sharma (1NT18EC101)** during the period October 2021 to February 2022. It is certified that all the corrections/suggestions indicated for internal assessment have been incorporated in the report deposited in the department library. The project has been approved as it satisfies the requirements prescribed for Bachelor of Engineering Degree.

Name and Signature of internal guide	***Name and Signature of HOD***	***Name and Signature of Principal***
		
[Dr. Rajesh N]	**[Dr. Ramachandra A C]**	**[Dr. H C Nagaraj]**

Declaration by Students

*We the students **KHUSHI RANGANATHA (1NT18EC078)** and **NIKITA SHARMA (1NT18EC101)** are aware of project conduction procedure and evaluation rubrics. We are also aware that the project phases are evaluated through continuous assessment.*

Name of the Student	***Signature***
KHUSHI RANGANATHA	
NIKITA SHARMA	

Confirmation by Guide

*I, **Dr. Rajesh Nandalike**, shall guide the above-mentioned students in the project entitled "**Medical Anomaly Detection**" and direct the students to submit the project for the academic year of 2021-2022.*

Signature of the Guide

Acknowledgement

A mini project is a golden opportunity for learning and self-development. We consider ourselves very lucky and honored to have a so many wonderful people lead us through in completion of this mini project.

We would like to thank **Dr. N.R. Shetty**, Director, NMIT, for providing such a wonderful environment to carry our Mini project.

We would like to express our gratitude to **Dr. H.C Nagaraj**, principal of NMIT, for his support in bringing this Mini project to completion. We wish to express our gratitude to the head of the department, **Dr. Ramachandra A.C,** Dept. of E&CE for providing a congenial working environment.

I wish to convey my deep sense of gratitude to **Dr. Rajesh N**, Professor, ECE Department, NMIT for his valuable guidance through the conduction of this mini project.

I take this opportunity to extend my sincere thanks to all remaining staff of our department for their necessary help and co-operation.

Abstract

Over the last decade, heart disease remains the primary basis of death worldwide. An estimate by the World Health Organization, that over 17.9 million deaths occur every year worldwide because of cardiovascular disease, and of these deaths, 80% are because of coronary artery disease and cerebral stroke. The silver lining is that heart attacks are highly preventable and simple lifestyle modifications (such as reducing alcohol and tobacco use; eating healthily and exercising) coupled with early treatment greatly improves its prognosis. It is, however, difficult to identify high risk patients because of the multi-factorial nature of several contributory risk factors such as diabetes, high blood pressure, high cholesterol, et cetera. Such information, if predicted well in advance, can provide important insights to doctors who can then adapt their diagnosis and treatment per patient basis. This is where machine learning and data mining come to the rescue.

In addition to the heart disease dataset, this project also utilizes the Breast Cancer Wisconsin (Diagnostic) Data Set to predict whether the cancer is benign or malignant using the same algorithm KNN. The purpose of this project is to compare the results of the algorithm applied to two different medical datasets.

Machine learning classification techniques can significantly benefit the medical field by providing an accurate and quick diagnosis of diseases. Hence, save time for both doctors and patients. This project mainly focuses on which patient is more likely to have a heart disease based on various medical attributes. We prepared a heart disease prediction system to predict whether the patient is likely to be diagnosed with a heart disease or not using the medical history of the patient. We have used a supervised machine learning algorithm K-nearest neighbours to predict and classify the patient with heart disease. The given heart disease prediction system enhances medical care and reduces the cost.

Contents

Table of Figures

Chapter 1 Introduction

The following are the processes executed in the mini-project:

1) Data Collection:

Processing of system start with the data collection for this we use the UCI repository dataset which is well verified by number of researchers and authority of the UCI. The dataset used here consists of 303 instances of 14 attributes each.

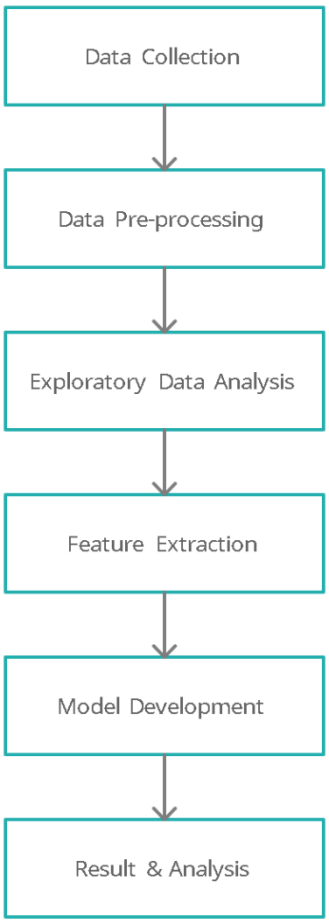

Figure 1.1 Flowchart for model implementation

2) Data Pre-processing:

Data preprocessing deals with the missing values, cleaning of data and normalization depending on algorithms used. Various attributes of the dataset may contain missing values which can lead to imprecise result and may reduce the accuracy of model. To overcome this problem, missing value way can be replaced by mean of column. Here it is observed that the dataset is already clean and has no missing null values.

3) Exploratory Data Analysis:

Data exploration technique to understand various aspects of data and to gain important statistical insights from the data. Here we plot various graphs of the categorical and distributive data in our dataset.

4) Feature Selection:

Having irrelevant features in a data set can decrease the accuracy of the models applied. Main aim of feature extraction is to extract a set of features which maximises the recognition rate with least number of elements.

5) Model development:

Here we use KNN algorithm for classification. This classifier looks for the classes of K nearest neighbors of a given data point and based on the majority class, it assigns a class to this data point.

6) Result and Analysis:

The following accuracy measures are done to analyse the accuracy of the model:

- Accuracy = (TP + TN) / (TP + TN + FP + FN)
- Sensitivity = TP / (TP+FN) * 100
- Specificity = TN/ (TN + FP) *100
- Positive Predictive Value= TP / (TP+FP) *100
- Negative Predictive Value= TN / (TN+FN) *100

where TP= True positive, TN= True negative, FP= False positive, FN= False negative.

Chapter 2 Literature survey and objectives

2.1 Literature survey

Citations	***Concepts/Techniques Used***	***Conclusion***
Mai Shouman, Tim Turner and Rob Stocker, **"*Applying K-Nearest Neighbour in Diagnosing Heart Disease Patients*"** 2012 International Conference of Knowledge Discovery	• KNN	• Achieved an accuracy of 97.4% in prediction using K-NN
D. Bajpai and L. He, **"*Evaluating KNN Performance on WESAD Dataset,*"** 2020 12th International Conference on Computational Intelligence and Communication Networks (CICN), 2020	• KNN • K-fold cross validation parameter	• Irrespective of size and shape of the dataset large value of nearest neighbors in KNN model will result in overfitting and complicate decision boundary • Very small value of nearest neighbors will result in under fitting
S. Karimifard, A. Ahmadian, M. Khoshnevisan and M. S. Nambakhsh, **"*Morphological Heart Arrhythmia Detection Using Hermitian Basis Functions and kNN Classifier,*"** 2006 International Conference of the IEEE Engineering in Medicine and Biology Society, 2006	• KNN • Voting System	• Achieved a sensitivity of 99% with specificity of 99.84%
Q. Yunneng, **"*A new stock price prediction model based on improved KNN,*"** 2020 7th International Conference on	• KNN	• Improved KNN algorithm has better predictive performance than the traditional KNN algorithm.

Information Science and Control Engineering (ICISCE), 2020		
C. C, **"*Prediction of Heart Disease using Different KNN Classifier,*"** 2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS), 2021	• K-NN Classifiers; • Fine • Medium • Coarse • Cosine • Cubic • Weighted	• Various classifiers are implemented and their training time and demerits are noted
Das, R., I. Turkoglu, and A. Sengur, **"*Effective diagnosis of heart disease through neural networks ensembles.*"** Expert Systems with Applications, Elsevier, 2009	• Multilayer Neural Network	• A classification accuracy of 89.01% classification accuracy is obtained and also 80.95% and 95.91% sensitivity and specificity values, respectively, in heart disease diagnosis.
Lee, I.-N., S.-C. Liao, and M. Embrechts, **"*Data mining techniques applied to medical information. Med. inform*"**, 2000	• Data visualisation • Correlation analysis • Discriminant analysis • Neural networks supervised classification	• Various common data mining techniques are used to formulate models which can extract knowledge from medical data.
G. Li and J. Zhang, **"*Music personalized recommendation system based on improved KNN algorithm,*"** 2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), 2018	• Improved algorithm KNN-Improved	• Effectively improve the problem of high error rate of KNN algorithm due to rating problem. • Overcame high rating (too good or too ad) influence
H. Yigit, **"*A weighting approach for KNN classifier,*"** 2013 International Conference on Electronics, Computer and Computation (ICECCO), 2013	• KNN • Artificial Bee Colony	• Found the optimal weights via Artificial Bee Colony (ABC) algorithm • ABC algorithm is applicable to kNN algorithm.

S. Rajathi and G. Radhamani, "***Prediction and analysis of Rheumatic heart disease using kNN classification with ACO,***" 2016 International Conference on Data Mining and Advanced Computing (SAPIENCE), 2016	• KNN • Ant Colony Optimisation	• KNN with ACO algorithm is effectively used to analyze Rheumatic Heart Disease • Accuracy is 70.26%

2.2 Objectives of the project

The following are the proposed objectives of the mini project

- To find a suitable datasets
- To understand and implement KNN
- To predict potential patients
- Therefore mitigate deaths

Chapter 3 Software tools

3.1 Python

Python has been the go-to choice for Machine Learning and Artificial Intelligence developers for a long time. Python offers some of the best flexibilities and features to developers that not only increase their productivity but the quality of the code as well, not to mention the extensive libraries helping ease the workload. Python libraries are also called as 'modules. Various features that put Python among the top programming languages for Machine Learning are listed below:

1) Free and open-source nature makes it community friendly and guarantees improvements in the long run
2) Exhaustive libraries ensure there's a solution for every existing problem
3) Smooth implementation and integration make it accessible for people with the varying skill level to adapt it
4) Increased productivity by reducing the time to code and debug
5) Can be used for Soft Computing, Natural Language Processing as well
6) Works seamlessly with C and C++ code modules

3.2 Google Colab Notebook

Technically, we can use any other code editor, but they are not ideal for machine learning projects. This is because we frequently need to inspect the data and it is really hard in VS code and terminal. In these machine learning projects we usually work with data of 10 or 20 rows and columns or more because of which visualisation of this data in terminal is very difficult and messy.

3.3 Python libraries used

- **NumPy:** The NumPy library for Python concentrates on handling extensive multi-dimensional data and the intricate mathematical functions operating on the data. NumPy offers speedy computation and execution of complicated functions working on arrays. Few of the points in favor of NumPy are:

    1) Support for mathematical and logical operations
    2) Shape manipulation
    3) Sorting and Selecting capabilities
    4) Discrete Fourier transformations
    5) Basic linear algebra and statistical operations

6) Random simulations
7) Support for n-dimensional arrays

Core task: Data cleaning and manipulation

- **Pandas:** Pandas is a Python data analysis library and is used primarily for data manipulation and analysis. It comes into play before the dataset is prepared for training. Pandas make working with time series and structured multi-dimensional data effortless for machine- learning programmers. Pandas make use of data frames, which is just a technical term for a two-dimensional representation of data by offering programmers with data frame objects. Some of the great features of Pandas when it comes to handling data are:

  1) Dataset reshaping and pivoting
  2) Merging and joining of datasets
  3) Handling of missing data and data alignment
  4) Various indexing options such as Hierarchical axis and Fancy indexing
  5) Data filtration options

  Core task: Data manipulation and analysis

- **Matplotlib:** It is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays for creating plots and graphs.

- **Scikit – learn:** Scikit-learn is another actively used machine learning library for Python. It includes easy integration with different ML programming libraries like NumPy and Pandas. Scikit-learn comes with the support of various algorithms such as:

  1) Classification
  2) Regression
  3) Clustering
  4) Dimensionality Reduction
  5) Model Selection
  6) Preprocessing

  Built around the idea of being easy to use but still be flexible, Scikit-learn is focused on data

modelling and not on other tasks such as loading, handling, manipulation and visualization of data. It is

considered sufficient enough to be used as an end-to-end ML, from the research phase to the deployment.

Core task: Modelling

Seaborn: It is a data visualization library for python.it is a plotting library that offers a simpler interface, sensible defaults for plots needed for ML, and most importantly, the plots are aesthetically better looking than those of matplotlib.

Chapter 4 Dataset-1

This project uses the existing dataset from the Cleveland database of UCI repository of **heart disease patients**. The dataset comprises 303 instances and 76 attributes. Of these 76 attributes, only 14 attributes are considered for testing, important to substantiate the performance of different algorithms.

1. age - age in years
2. sex - (1 = male; 0 = female)
3. cp - chest pain type

– 0: Typical angina: chest pain related decrease blood supply to the heart
– 1: Atypical angina: chest pain not related to heart
– 2: Non-anginal pain: typically, oesophageal spasms (typically oesophageal related)
– 3: Asymptomatic: chest pain not showing signs of disease

4. trestbps - resting blood pressure (in mm Hg on admission to the hospital) anything above 130-140 is typically cause for concern
5. chol - serum cholesterol in mg/dl

– serum = LDL + HDL + .2 * triglycerides
– above 200 is cause for concern

6. fbs - (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)

'>126' mg/dL signals diabetes

7. restecg - resting electrocardiographic results

– 0: Nothing to note
– 1: ST-T Wave abnormality can range from mild symptoms to severe problems , ignals non-normal heart beat
– 2: Possible or definite left ventricular hypertrophy enlarged heart's main pumping chamber

8. thalach - maximum heart rate achieved
9. exang - exercise induced angina (1 = yes; 0 = no)

10. oldpeak - ST depression induced by exercise relative to rest looks at stress of heart during exercise unhealthy heart will stress more

11. slope - the slope of the peak exercise ST segment

- Upsloping: better heart rate with exercise (uncommon)
- Flatsloping: minimal change (typical healthy heart)
- Downslopins: signs of unhealthy heart

12. ca - number of major vessels (0-3) coloured by fluoroscopy; coloured vessel means the doctor can see the blood passing through the more blood movement the better (no clots)

13. thal - thallium stress result

- 1,3: normal
- 6: fixed defect: used to be defect but ok now
- 7: reversable defect: no proper blood movement when exercising

14. target - have disease or not (1=yes, 0=no) (= the predicted attribute)

Chapter 5 Dataset-2

This project uses the existing dataset of **Breast Cancer** Wisconsin Diagnostic. The dataset comprises 569 instances and 32 attributes. Of these 32 attributes, only 10 attributes are considered for testing, important to substantiate the performance of different algorithms

1. radius (mean of distances from centre to points on the perimeter)
2. texture (standard deviation of grey-scale values)
3. perimeter
4. area
5. smoothness (local variation in radius lengths)
6. compactness (perimeter^2 / area - 1.0)
7. concavity (severity of concave portions of the contour)
8. concave points (number of concave portions of the contour)
9. symmetry
10. fractal dimension ("coastline approximation" - 1)

Chapter 6 About the algorithm

6.1 Introduction

K-Nearest Neighbor is one of the simplest Machine Learning algorithms based on Supervised Learning technique. KNN algorithm assumes the similarity between the new case/data and available cases and puts the new case into the category that is most similar to the available categories. It stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well-suited category by using KNN algorithm. It can be used for Regression as well as for Classification but mostly it is used for the Classification problems.

KNN 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 classification, 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 classifies 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 whether it is a cat or dog. So, for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find 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.

Figure 6.1 KNN Classifier

6.2 How does KNN 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 distance and weights of **K number of neighbors.**

Step-3: Take the K nearest neighbors as per the calculated distance and weights. (In the project we are using Euclidean distance and Uniform weights)

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.

6.3 K in KNN

K-denotes number of nearest neighbors which are the voting class for new data. When K=7, labels of 7 nearest neighbors are checked and the most common label is given to the new data. Choosing the value of K is called **Parameter Tuning** and it is important for better accuracy.

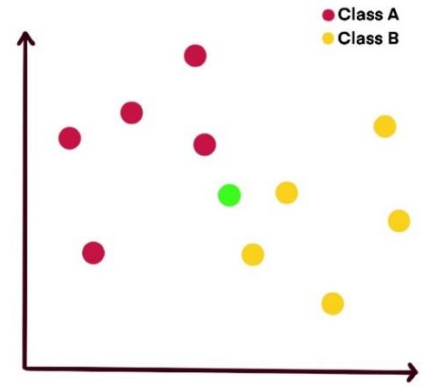

Figure 6.2 example: distribution of datapoints

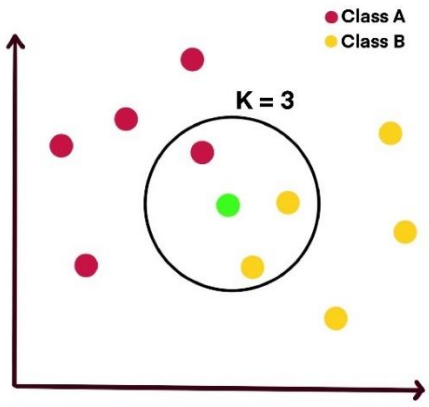

Figure 6.3 model with the number of neighbors as 3

Example: the dataset given is classified into two groups class A and class B. Now the model should predict which class the new data belongs to. When the value of **k** is 3, the model selects 3 closest points to the new data. In this case new data(star) belongs to class B. When the value of **k** is 6, the model selects 6 nearest points to the new data. In this case new data(star) belongs to class A.

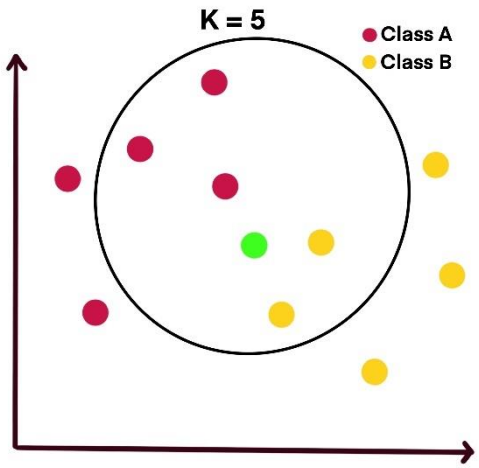

Figure 6.4 model with number of neighbors as 5

6.4 How to choose value of K?

- Cross validation technique (it's similar to "trial and error" method), where different values of K are chosen and the best value is selected.
- Take Sqrt(n), where n is the total number of datapoints.
- It's preferable to select odd numbers for K to avoid confusion between two classes of data.
- The most preferred value for K is 5.
- A very low value for K such as K=1 or K=2, can be noisy and lead to the effects of outliers in the model.
- Large values for K are good, but it may find some difficulties.

6.5 How is the prediction done?

- Based on distance and weights.
- KNN uses least distance measure to find its nearest neighbors and predict the class for new data.
- Types of distances KNN uses are Euclidean, Manhattan, Minkowski, Cosine, etc

There are different formulas to calculate these different types of distances. In the project we are using Euclidean distance.

Euclidean Distance: Direct/least possible distance between two points. It is the most widely used distance. It is the default metric that the Sklearn library of Python uses.

Manhattan Distance: Distance between two points is measured along the axis at right angles. It is also known as Taxi cab or City block distance.

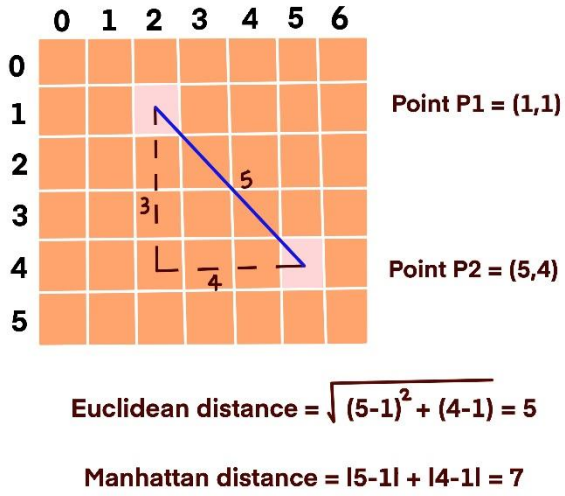

Figure 6.5 difference between Euclidean and Manhattan formulae

KNN algorithm have **2 types of weights**:

1. Uniform: All data points are given equal weights.
2. Distance: Weights are assigned according to distance between the new point and the existing point.
   Distance type is usually not preferred because it does not provide accurate output.

6.6 When to use KNN Algorithm?

- KNN can be used in both regression and predictive classification problems.
- When data is labeled.
- When data is noise free.
- When the dataset is small.
- If you have a dataset with many different points and precise information, this is a great place to start exploring machine learning with KNN.

6.7 Advantages of KNN Algorithm:

- Simple to implement.
- No learning phase.
- Fast computing time.
- High performance accuracy.

6.8 Disadvantages of KNN Algorithm:

- Always needs to determine the value of K which may be complex some time.
- The computation cost is high because of calculating the distance between the data points for all the training samples.
- Time consuming as it has to read and classify all the data in the dataset.
- Needs more memory to store.

Chapter 7 Code implementation

This section describes the various steps required to implement the model of KNN. For a detailed view of the code please refer to the Appendix.

7.1 Importing the libraries

Libraries like *pandas, numpy, matplotlib, seaborn and sklearn* are imported into the code. The advantage of using Google Colab is that these libraries need not be installed in order to be imported. They get imported from online repositories by Google that stores all the required modules.

7.2 Reading CSV file

The dataset is in the form of an excel sheet. This CSV file needs to be uploaded to Google Colab for it to be used for further data analysis. The datasets can be uploaded using the google.colab() module, and saved into a variable using the pandas module.

7.3 Exploratory Data Analysis

The first step into data analysis is to visualize the data in order to understand its content and structure. This understanding enables the coder to come up with specifics in the code i.e which rows and columns to utilize, removal of null values, dropping of rows and columns that are not required etc.

The two most popular modules that are utilized to visualize the data are Seaborn and Matplotlib.

7.4 Splitting the data

Our input data file consists of medical characteristics of the patients, along with their target values (determination of whether they tested positive for presence of heart disease or not). Now we need to split that data into 2; x contains solely the medical attributes of the patients, y contains only the diagnosis of the patients.

This x and y are further split into training and testing data respectively. The conventional split ratio is 80% for training and the remaining 20% for testing.

7.5 Standardizing Data

When using multiple data sources, not all data points may contribute equally due to internal inconsistencies. To make the data uniform, we transform it through standardization. This way we make the data have the same type of content and format, making it easy to compare and compute required information.

Some standardization methods that can be utilized are Normalizer, MinMaxScaler, RobustScaler. Here we are using *StandardScaler*. The basic principles of standardization of data include making the mean and standard deviation of each data point 0 and 1 respectively.

Here StandardScaler subtracts the mean from each feature and scales it to unit variance, making it similar to Normal Distribution. After standardizing the data, we now check to see if it has been standardized accurately using functions .mean() and .std()

7.6 Hyper-parameter Tuning

Parameter tuning is the concept of choosing the optimal parameters to be considered for training the model. Hyper-parameters are the parameters that we give for consideration (here, contents of the dictionary grid_params are hyper-parameters).

GridSeachCV is used here to identify the best parameters to be chosen for training. The results are stored and used to fit the training data of x and y.

7.7 Statistics

The following is a Confusion Matrix. X-axis represents the values that have been given to the model, and the y-axis represents the predicted values by the model.

If the values on both the axes match then we obtain the corresponding True Values i.e

(0,0) = True Negative

(1,1) = True Positive

However if the values on both the axes do not match then we obtain the corresponding False Values i.e

(0,1) = False Negative

(1,0) = False Positive

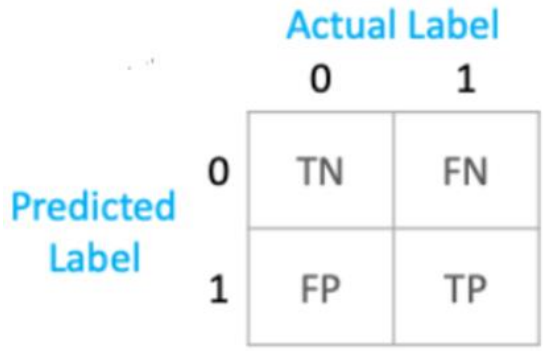

Figure 7.1 Structure of a confusion matrix

Consider a person is being tested for a disease:

True Positive – If the results come out positive and the person actually has the disease.

True Negative - If the results come out negative and the person actually does not have the disease.

False Positive – If the results come out positive while the patient does not actually have the disease

False Negative – If the results come out negative while the patient does actually have the disease

A confusion matrix is computated and its values are used to calculate the following:

- *Accuracy*

  The proportion of true results (both true positive and true negative) in the selected population

  Formula: (TP + TN) / (TP + TN + FP + FN)

- *Sensitivity*

  The proportion of True Positive results in the population with True Positive and False Negative values

  Formula: TP / (TP + FN)

- *Specificity*

  The proportion of True Negative results in the population with True Negative and False Positive values

  Formula: TN / (TN + FP)

- *Positive predictive value*

  Proportion of all the truly positive cases that were classified positive

  Formula: TP / (TP + FP)

- *Negative predictive values*

  Proportion of all the truly positive cases that were classified positive

  Formula: TN / (TN + FN)

7.8 Accuracy with different KNN values

From Hyper-parameter tuning, we obtained that the best number of neighbours for this model would be 11. Now by using KNeighborsClassifier(), we can implement the KNN algorithm onto our datapoints to obtain the accuracy of our model's predictions.

Chapter 8 Result and analysis

8.1 Exploratory Data Analysis:

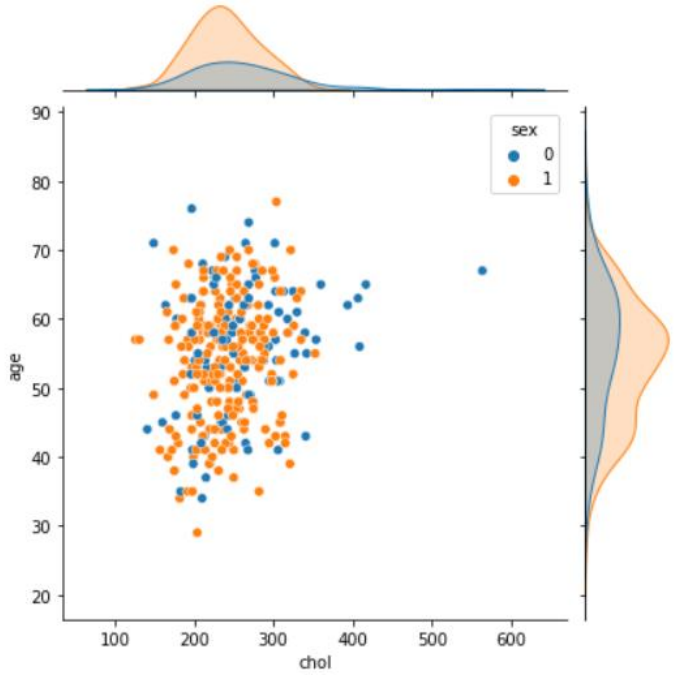

Figure 8.1 chol v. age scatter plot graph

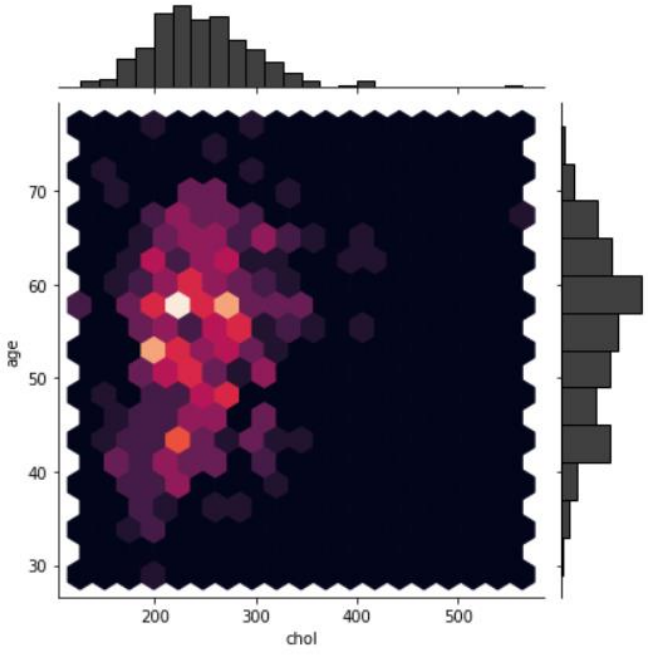

Figure 8.2 histogram of chol v. age using hexagonal bins

There are numerous ways to visualize the dataset. The graphs above depict Age versus Cholesterol Characteristics. The two most popular types of visualization are the Scatter Plot and Hexagonal Plot which are shown above. The sexes of the persons are also depicted by the use of color.

In the scatter plot, the datapoints are marked with respect to the x and y (chol and age) values

respectively. Then the datapoints are coloured in accordance to the sex of the patient.
The histogram with hexagonal bins utilize the plotted datapoints, and show us the concentration of the population. The brighter the color, the higher is the concentration of the datapoints in that hex bin. In contrast, the darker the color, the population is concentrated lesser in comparison.

8.2 Confusion Matrix

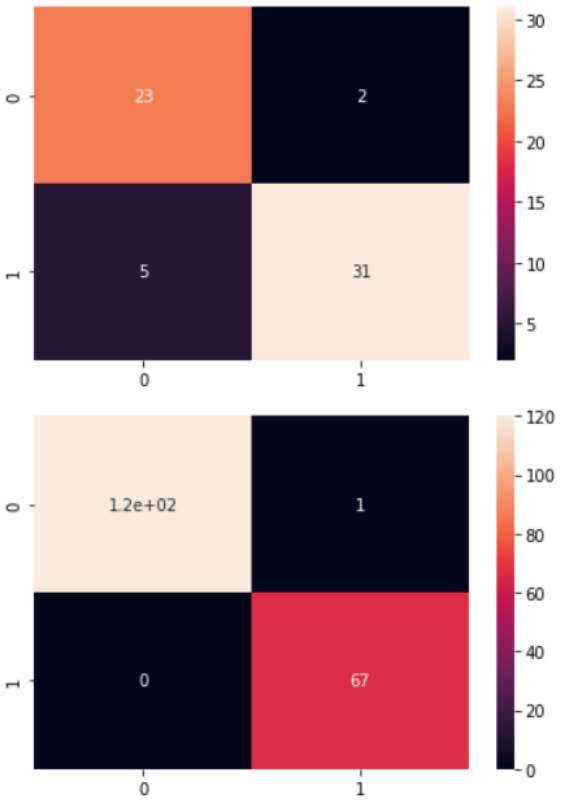

Figure 8.4 Confusion Matrix obtained from the implemented code

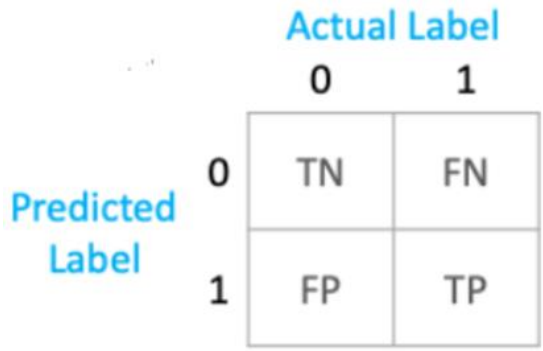

Figure 8.3 Structure of a confusion matrix

The following is a Confusion Matrix. X-axis represents the values that have been given to the model, and the y-axis represents the predicted values by the model.
If the values on both the axes match then we obtain the corresponding True Values i.e
(0,0) = True Negative
(1,1) = True Positive

However if the values on both the axes do not match then we obtain the corresponding False Values i.e
(0,1) = False Negative
(1,0) = False Positive

Consider a person is being tested for a disease:
True Positive – If the results come out positive and the person actually has the disease.
True Negative - If the results come out negative and the person actually does not have the disease.
False Positive – If the results come out positive while the patient does not actually have the

disease

False Negative – If the results come out negative while the patient does actually have the disease

A confusion matrix is computated and its values are used to calculate the following:

- *Accuracy*

  The proportion of true results (both true positive and true negative) in the selected population

  Formula: (TP + TN) / (TP + TN + FP + FN)

- *Sensitivity*

  The proportion of True Positive results in the population with True Positive and

  False Negative values

  Formula: TP / (TP + FN)

- *Specificity*

  The proportion of True Negative results in the population with True Negative and False Positive values

  Formula: TN / (TN + FP)

- *Positive predictive value*

  Proportion of all the truly positive cases that were classified positive

  Formula: TP / (TP + FP)

- *Negative predictive values*

  Proportion of all the truly positive cases that were classified positive

  Formula: TN / (TN + FN)

8.3 Final Result

```
Heart Disease Dataset
The accuracy score achieved using KNN is at n= 3  is: 70.49 %
The accuracy score achieved using KNN is at n= 5  is: 57.38 %
The accuracy score achieved using KNN is at n= 7  is: 63.93 %
The accuracy score achieved using KNN is at n= 9  is: 63.93 %
The accuracy score achieved using KNN is at n= 11  is: 68.85 %


Breast Cancer Dataset
The accuracy score achieved using KNN is at n= 3  is: 93.62 %
The accuracy score achieved using KNN is at n= 5  is: 95.21 %
The accuracy score achieved using KNN is at n= 7  is: 97.34 %
The accuracy score achieved using KNN is at n= 9  is: 97.34 %
The accuracy score achieved using KNN is at n= 11  is: 97.87 %
```

Figure 8.5 accuracy for various n values

Logically the accuracy must've increased with N, up to a certain point. But here as is observed in the above figure, the accuracy growth was inconsistent. Therefore it is obvious that the accuracy at the highest value of n is not necessarily the best accuracy that can be obtained.

Dataset	Accuracy %	Sensitivity %	Specificity %	positive predictive value in %	negative predictive value in %
Heart	70.4918	77.7778	60	73.6842	65.2174
Breast	96.2766	94.0299	97.5207	95.4545	96.7213

Figure 8.6 Accuracy of model without standardization

Dataset	Accuracy %	Sensitivity %	Specificity %	positive predictive value in %	negative predictive value in %
Heart	88.5246	86.1111	92	93.9394	82.1429
Breast	97.3404	95.5224	98.3471	96.9697	97.541

Figure 8.7 Accuracy of model with standardization

Here we see that the highest accuracy we obtained with a dataset that was not standardized was 65.21%, for heart disease detection and 96.72% for breast cancer detection. Moreover, comparing with the accuracy of our standardized data (***Error! Reference source not found.****4*), t he accuracy came up to 82.14% and 97.41% respectively. Therefore, standardization increases the accuracy of the model prediction hugely.

Chapter 9 Conclusion

Through this project we have attempted to analyze and anticipate if someone in particular, given different individual attributes and indications, will have heart disease or not using machine learning algorithm called K-Nearest neighbor. The primary thought process of our project was to looking at the exactness and analyzing result using KNN algorithm of machine learning. We have used Cleveland dataset for heart diseases which contains 303 instances and used percent split to divide the data into two sections which are training and testing datasets. We have considered 14 attributes among 76 attributes and implemented KNN algorithm to examine the accuracy. By the end of the implementation part, we have discovered that the accuracy level in our dataset is 83.61 percent. Also, on the off chance that we increment the number of training data, maybe we can find more accurate result. But if the increment in the number of training data is large then it will take more time to process and the system will be slower than now as it will be more perplexing and will be handling more data.

Chapter 10 Future scope

In this project we have implemented a machine learning model to predict if a person has heart disease or not using KNN algorithm and here the accuracy was 83.61 percent. Hence further we can try implementing the same model using other machine learning algorithms like random forest, decision tree, linear regression, logistic regression and so on. From these best suited algorithms for implementing our model with more accuracy can be selected.

The dataset used in this model has 303 instances and 76 attributes. Among these 76 attributes we have considered 14 of them for implementing our model. This dataset used is small hence our accuracy is quiet less i.e., 83.16 percent. By increasing the size of the dataset (number of instances) the accuracy of the model can be increased significantly. But while doing so we should be careful because if the increment in the number is large then the model maybe slower while handling more data.

Hence there are two ways we can increase the performance of our model:

1) By selecting suitable algorithm that gives maximum accuracy
2) By increasing the size of the dataset

Chapter 11 References

*[1] Mai Shouman, Tim Turner and Rob Stocker, "**Applying K-Nearest Neighbour in Diagnosing Heart Disease Patients**" 2012 International Conference of Knowledge Discovery*

*[2] S. Karimifard, A. Ahmadian, M. Khoshnevisan and M. S. Nambakhsh, "**Morphological Heart Arrhythmia Detection Using Hermitian Basis Functions and kNN Classifier**," 2006 International Conference of the IEEE Engineering in Medicine and Biology Society, 2006*

*[3] D. Bajpai and L. He, "**Evaluating KNN Performance on WESAD Dataset**," 2020 12th International Conference on Computational Intelligence and Communication Networks (CICN), 2020*

*[4] Sandhya, J., et al., "**Classification of Neurodegenerative Disorders Based on Major Risk Factors Employing Machine Learning Techniques**" International Journal of Engineering and Technology, 2010. Vol.2, No.4*

*[5] C. C, "**Prediction of Heart Disease using Different KNN Classifier**," 2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS), 2021*

*[6] Lee, I.-N., S.-C. Liao, and M. Embrechts, "**Data mining techniques applied to medical information. Med. inform**", 2000*

*[7] Q. Yunneng, "**A new stock price prediction model based on improved KNN**," 2020 7th International Conference on Information Science and Control Engineering (ICISCE), 2020*

*[8] G. Li and J. Zhang, "**Music personalized recommendation system based on improved KNN algorithm**," 2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), 2018*

*[9] H. Yigit, "**A weighting approach for KNN classifier,**" 2013 International Conference on Electronics, Computer and Computation (ICECCO), 2013*

*[10] S. Rajathi and G. Radhamani, "**Prediction and analysis of Rheumatic heart disease using kNN classification with ACO**," 2016 International Conference on Data Mining and Advanced Computing (SAPIENCE), 2016*

Chapter 12 Appendix

The code used to produce the above output is shown below:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
from sklearn import metrics
%matplotlib inline
```

```
from google.colab import files
uploaded = files.upload()
data1 = pd.read_csv('data1.csv')
```

Choose Files data1.csv

- **data1.csv**(application/vnd.ms-excel) - 11328 bytes, last modified: 2/6/2022 - 100% done

```
Saving data1.csv to data1.csv
```

```
from google.colab import files
uploaded = files.upload()
data2 = pd.read_csv('data2.csv')
```

Choose Files data2.csv

- **data2.csv**(application/vnd.ms-excel) - 125204 bytes, last modified: 2/6/2022 - 100% done

```
Saving data2.csv to data2.csv
```

```
#different plot types
sns.jointplot(data=data1,
x='chol',
y='age',
kind='kde',
cmap='rocket',
fill=True,
color='black',
#hue='sex'
)

sns.jointplot(data=data1, x='chol', y='age', kind='scatter', cmap='rocket', color='black', hue='sex')
sns.jointplot(data=data1, x='chol', y='age', kind='hist', fill=True, cmap='rocket', color='black', hue='sex')
sns.jointplot(data=data1, x='chol', y='age', kind='hex', cmap='rocket', color='black')
sns.jointplot(data=data1, x='chol', y='age', kind='reg', color='black')
sns.jointplot(data=data1, x='chol', y='age', kind='resid', color='black')
```

```
#define x train,test y train,test
#data.drop excludes target attribute from x, axis 0 is row and 1 is for column so column target is excluded
x1 = data1.drop('target',axis=1)
#only target attribute is considered for y because target is know quantity ie we know who has heart disease and who does not
y1 = data1['target']
#x is basically all the characteristics of a person and y is the diagnosis of the person
#now y has known data and x has all other data attributes, and we split both into train and test groups
x1_train, x1_test, y1_train, y1_test = train_test_split(x1,y1,test_size=0.2,random_state=4)

#-----------------------------------------------------------------------------------------------------------------------------

x2 = data2.drop(['Unnamed: 32', 'id'], axis = 1)
y2 = data2['diagnosis']

# Converting the diagnosis value of M and B to a numerical value where M (Malignant) = 1 and B (Benign) = 0
def diagnosis_value(diagnosis):
    if diagnosis == 'M':
        return 1
    else:
        return 0

data2['diagnosis'] = data2['diagnosis'].apply(diagnosis_value)

x2=data2[['radius_mean','perimeter_mean','area_mean','compactness_mean','concave points_mean','radius_se','perimeter_se','area_se','compactness_se'
y2=data2[['diagnosis']]

x2_train, x2_test, y2_train, y2_test = train_test_split(x2,y2,test_size=0.33,random_state=42)
```

```
#standardize data method 1
#making sure that one data set can be compared to other data sets
sc1 = StandardScaler()
#making sure all data points in x set contributes equally to the model
#calling the function fit_transform in the class StandardScalar to transform data with mean=0 and std=1
x1_train = sc1.fit_transform(x1_train)
x1_test = sc1.fit_transform(x1_test)
```

```
#standardize data method 2
sc2 = StandardScaler().fit(x2_train)
x2_train = sc2.transform(x2_train)

sc21 = StandardScaler().fit(x2_test)
x2_test = sc2.transform(x2_test)
```

```
#verifying that mean=0 and std=1 of training data
print("\n\t",x1_train.mean(),x1_train.std(),"\n\t",x1_test.mean(),"\t",x1_test.std())
print("\n\t",x2_train.mean(),x2_train.std(),"\n\t",x2_test.mean(),"\t",x2_test.std())
```

```
        -5.2229182340916454e-17 1.0
        6.944144012787864e-17   1.0

        -2.4815754677930857e-18 1.0
        0.007697421330925424    0.9843896308279503
```

```
#initializing the algo KNeighboursClassifier to knn
knn1 = KNeighborsClassifier

#for parameter tuning
#parameters and grid_param are dictionaries
grid_params1 = {'n_neighbors': [3,5,7,9,11], 'weights': ['uniform','distance'] , 'metric' : ['euclidean', 'manhattan']  }

#here the grid parameters are the hyper parameters used for selecting the best model
#GridSearchCV is the algorithm that selects the best parametes from the given dictionary above
#the algorithm considers the the parameters we have mentioned in grid_parameters and determines the best possible combination for knn
gs1 = GridSearchCV(knn1() , grid_params1 , cv = 4 , scoring='accuracy')
#applying the obtained optimal parameters to the training data
gs_results1 = gs1.fit(x1_train, y1_train)

#printing the chosen paramater values y the algorithm
print("Heart dataset\n",gs_results1.best_params_)

#now we apply those parameter values to generate a model
model1 = gs_results1.best_estimator_
model1.score(x1_test, y1_test)

#-------------------------------------------------------------------------------------------------------------------------------

knn2 = KNeighborsClassifier
grid_params2 = {'n_neighbors': [3,5,7,9,11], 'weights': ['uniform','distance'] , 'metric' : ['euclidean', 'manhattan']  }
gs2 = GridSearchCV(knn1() , grid_params2 , cv = 4 , scoring='accuracy')
gs_results2 = gs2.fit(x2_train, y2_train)
print("\nBreast dataset\n",gs_results2.best_params_)
model2 = gs_results2.best_estimator_
model2.score(x2_test, y2_test)
```

```
Heart dataset
 {'metric': 'manhattan', 'n_neighbors': 11, 'weights': 'uniform'}

Breast dataset
 {'metric': 'euclidean', 'n_neighbors': 3, 'weights': 'uniform'}
0.9946808510638298
```

```
#confusion matrix
#applying the trained model to x_test data
predictions1 = model1.predict(x1_test)
#generatinf cm (x,y)
cm1 = metrics.confusion_matrix(y1_test,predictions1)
cm1 = pd.DataFrame(cm1)
#constructing the heat map
sns.heatmap(cm1, annot=True)
plt.show()

#-----------------------------------------------------

predictions2 = model2.predict(x2_test)
cm2 = metrics.confusion_matrix(y2_test,predictions2)

cm2 = pd.DataFrame(cm2)

sns.heatmap(cm2, annot=True)
plt.show()
```

```
print(cm1[0][0],cm1[1][0],cm1[0][1],cm1[1][1]);
print(cm2[0][0],cm2[1][0],cm2[0][1],cm2[1][1]);
```

```
23 2 5 31
120 1 0 67
```

```
TN1 = cm1[0][0]
FN1 = cm1[0][1]
FP1 = cm1[1][0]
TP1 = cm1[1][1]

TN2 = cm2[0][0]
FN2 = cm2[0][1]
FP2 = cm2[1][0]
TP2 = cm2[1][1]

#all true values by total values
accu1 = (TP1 + TN1) / (TP1 + TN1 + FP1 + FN1)*100
accu2 = (TP2 + TN2) / (TP2 + TN2 + FP2 + FN2)*100
#actual positives that got predicted as positive
sensitivity1 = TP1 / (TP1+FN1 )* 100
sensitivity2 = TP2 / (TP2+FN2 )* 100
#actual negatives that got predicted as negative
specificity1 = TN1/(TN1 + FP1)*100
specificity2 = TN2/(TN2 + FP2)*100
#positive predictive value - proportion of all positively classifed cases that were truly positive
ppv1 = TP1 / (TP1+FP1) *100
ppv2 = TP2 / (TP2+FP2) *100
#positive predictive value - proportion of all negatively classifed cases that were truly negative
npv1 = TN1 / (TN1+FN1) *100
npv2 = TN2 / (TN2+FN2) *100

from tabulate import tabulate
info = {'Dataset': ['Heart', 'Breast'],'Accuracy %':[accu1,accu2],'Sensitivity %': [sensitivity1,sensitivity2],'Specificity %': [specificity1,specificity2],'positive predictive value
print(tabulate(info, headers='keys', tablefmt='fancy_grid'))
```

Dataset	Accuracy %	Sensitivity %	Specificity %	positive predictive value in %	negative predictive value in %
Heart	88.5246	86.1111	92	93.9394	82.1429
Breast	97.3404	95.5224	98.3471	96.9697	97.541

```
#show the accuracy with different knn values
knn1 = KNeighborsClassifier(n_neighbors=11)
knn1.fit(x1_train,y1_train)
Y_pred_knn1=knn1.predict(x1_test)

from sklearn.metrics import accuracy_score

score_knn1 = round(accuracy_score(Y_pred_knn1,y1_test)*100,2)

print("The accuracy score achieved using KNN is: "+str(score_knn1)+" %")

# ------------------------------------------------------------------------

knn2 = KNeighborsClassifier(n_neighbors=11)
knn2.fit(x2_train,y2_train)
Y_pred_knn2=knn2.predict(x2_test)

from sklearn.metrics import accuracy_score

score_knn2 = round(accuracy_score(Y_pred_knn2,y2_test)*100,2)

print("The accuracy score achieved using KNN is: "+str(score_knn2)+" %")
```

```
The accuracy score achieved using KNN is: 83.61 %
The accuracy score achieved using KNN is: 96.28 %
```