MIT | Academy of Engineering

Lab Manual

Soft Computing

(ET363)

TY BTECH

Autonomous Pattern 2019-23

Department of Electronics & Telecommunication Engineering

MIT Academy of Engineering, Alandi, Pune

Vision & Mission of MIT Academy of Engineering

&

Vision, Mission, PEOs, PO & PSO of E&TC Engg.

Vision & Mission of MIT Academy of Engineering

Vision

To develop MITAOE into a new-age learning center with an excellent ambiance for academics and research conjugated with a vibrant environment for honing the extra and curricular skills of all its stakeholders, to enable them to solve real-world problems and bring a positive change in the society.

Mission

To leave no stone unturned in our endeavor to ensure that every alumnus looks back at us and says MITAOE has not merely taught me, it has educated me.

Vision & Mission of E&TC Engineering

Vision

To develop the students towards an exemplary career in Telecommunication and its cognate disciplines, possessing a sound social awareness, sense of responsibility, and moral ethos.

Mission

- To develop the Department into a well-established education hub in the domain of Electronics & Telecommunication engineering.
- To provide students with a multi-faceted learning environment complemented by adequate engineering practice and research, preparing them to solve real-life engineering problems.
- To facilitate inclusive growth of all its student community and enabling them to be leaders of tomorrow.

Program Educational Objectives (PEOs)

The graduates of BTECH in Electronics & Telecommunication Engineering, four years after completion of their degrees, are expected:

PEO 1. To achieve a high level of technical competence in the electronics and telecommunication domain or any other associated areas, be it an Engineering Practice or Research.

PEO 2. To address real-world complex engineering problems by formulating solutions and designs that are technically sound, economically viable, practically feasible, and environmentally sustainable.

PEO 3. To aim towards career enhancement by pursuing lifelong learning and evolve as a leader in professional and personal life.

Program Outcomes (POs)

After successfully completing the BTECH program students will be able to -

PO1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO 2 Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO 3 Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO 4 Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO 5 Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

PO 6 The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO 7 Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO 8 Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO 9 Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO 10 Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO 11 Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO 12 Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcomes (PSOs)

After successfully completing the BTECH E&TC Engg. program students will be able to –

PSO 1 Analyze and simulate diverse problems in the field of communication.

PSO 2 Design and analyze a system with applications in signal and image processing.

PSO 3 Build, test and evaluate an embedded system with real time constraints.

PSO 4 Design and implement a system towards automatic control in varied engineering problems.

MIT Academy of Engineering AN AUTONOMOUS INSTITUTE	**INDEX & CERTIFICATE**	
	ACADEMIC YEAR	**2021-22**
Alandi (D), Pune – 412105	**SEM/TRI**	VI
DEPARTMENT of E&TC ENGG.	**CLASS & BLOCK**	TY BTECH

Experiment no	**Title**	**Mapped CO**	**Page no**	**Assessment points**	**Remark**
01	Experimental Data Analysis: Perform following operations on any open dataset available in Python/Kaggle	**CO.1**			
02	Liner Regression and Logistic Regression Model Implementation on Given Dataset.	**CO.1,2, 4**			
03	Implementation of Decision Tree, Random Forest, KNN, Naïve Bayes with hyperparameter tunning.	**CO.2,4, 5**			
04	Machine Learning for Image Classification	**CO.4**			
05	Implementation of Unsupervised Machine Learning	**CO.2,4**			
06	Implementation of IOT Solution using Machine Learning	**CO.1,4**			
07	ANN for Computer Vision	**CO.4**			
08	Open CV for Computer Vision	**CO.1**			

This is to certify that Mr Kshitij Vitthal Darwhekar

Roll No TETB19 has successfully completed the experiments for the course

Soft Computing for academic year 2021-22

Sign
Student

Sign
Course instructor

Expt. No. 1 Date: ______________

Experimental Data Analysis: Perform following operations on any open dataset available in Python/Kaggle

Objectives:

- To perform basic operations for data computations on given data set using Python/Kaggle

Software Requirement:

- Python jupyter notebook

Theory:

Python is an all-rounder programming language that deals with various kinds of fundamentals; these are going to use to develop Python programs, which include variables, identifiers, data types, strings arrays, etc.

Perform basic following operations on given data set

- Load data into a data frame from a csv or any other file format

  To load data into Pandas DataFrame from a CSV file, use pandas.read_csv() function

```
import pandas as pd

#load dataframe from csv
df = pd.read_csv("data.csv")

#print dataframe
print(df)
```

Output

```
   name  physics  chemistry  algebra
0  Somu       68         84       78
1  Kiku       74         56       88
2  Amol       77         73       82
3  Lini       78         69       87
```

- Identification of variables and data types.

  In statistical research, a variable is defined as an attribute of an object of study.

- Find Missing Values, Replace/eliminate missing values, Drop unessential columns. Initially load the file and look at the structure of the file. When you have a big dataset with high number of columns it is hard to look at each columns and study the types of columns. Then separate the categorical and numerical columns in the data frame to find missing values.
  Use **isnull()** function to identify the missing values in the data frame
  Use **sum()** functions to get sum of all missing values per column.
  use **sort_values(ascending=False)** function to get columns with the missing values in descending order.
- Find average/min/max of numeric columns.
- Display summary of data frame.

<u>Steps:</u>

- Selection of appropriate data set as per the ML algorithm
- Data preprocessing
- ML Model Implementation with the applicable software
- Performance analysis of implemented ML model
- To improve the model performance use Hyperparameter tuning

<u>Conclusion</u>

Assessment Rubrics

Performance Area	**Rating = 3**	**Rating = 2**	**Rating = 1**	**Rating= 0**	**Score**
Presentation of Markdown Cell	In Markdown cell mentions all details of ML Algorithm and comment properly about Python Steps	In Markdown cell mentions either details of ML Algorithm or comment properly about Python Steps	Student fails to define the application adequately.	Markdown Cell not prepared	
Data Set	Data Set Visualization, Analysis and Preprocessing is done properly.	Only Data Set Visualization or Analysis or Preprocessing is done.	Only Data Set Visualization done.	Data Set operation is not done properly	
ML Model Implementation	ML model train, test with good accuracy and predication done properly . Optimization Parameter Shown	ML model train , test with better accuracy and predication is done , optimization parameter not shown	ML model train , test but accuracy is poor and predication is done , optimization parameter not shown	ML model train , test is not proper and predication is not done , optimization parameter not shown	
				Timely Submission	1 Mark
				Total	10

Expt. No. 2 Date: ____________

Linear Regression and Logistic Regression Model Implementation on Given Dataset.

Objectives:

- To implement linear Regression model using given dataset
- To implement logistic Regression model using given dataset

Software Requirement:

- Python jupyter notebook

Theory:

- **Linear Regression:**

Linear regression is one of the most well-known algorithms in statistics and machine learning. It is a linear model that assumes a linear relationship between the input variables (x) and the single output variable (y). More specifically, that y can be calculated from a linear combination of the input variables (x).

1) Simple linear regression

$$Y = \beta_0 + \beta_1 X \quad \text{.......................................1}$$

2) Multiple Linear Regression

When there is a single input variable (x), the method is referred to as simple linear regression. When there are multiple input variables, literature from statistics often refers to the method as multiple linear regression.

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \ldots\ldots + \beta_n X_n \quad \text{........................2}$$

Advantages of Linear Regression:

1Linear Regression is simple to implement.

2Less complexity compared to other algorithms.

3) Linear Regression may lead to over-fitting but it can be avoided using some dimensionality reduction techniques, regularization techniques, and cross-validation.

- **Logistic Regression**

  The Logistic Regression is a regression model in which the response variable (dependent variable) has categorical values such as True/False or 0/1. It actually measures the probability of a binary response as the value of response variable based on the mathematical equation relating it with the predictor variables.

  Logistic regression becomes a classification technique only when a decision threshold is brought into the picture. The setting of the threshold value is a very important aspect of Logistic regression and is dependent on the classification problem itself.

  The decision for the value of the threshold value is majorly affected by the values of precision and recall. Ideally, we want both precision and recall to be 1, but this seldom is the case. In the case of a Precision-Recall tradeoff, we use the following arguments to decide upon the threshold

  1. Low Precision/High Recall: In applications where we want to reduce the number of false negatives without necessarily reducing the number of false positives, we choose a decision value that has a low value of Precision or a high value of Recall. For example, in a cancer diagnosis application, we do not want any affected patient to be classified as not affected without giving much heed to if the patient is being wrongfully diagnosed with cancer. This is because the absence of cancer can be detected by further medical diseases but the presence of the disease cannot be detected in an already rejected candidate.

2. High Precision/Low Recall: In applications where we want to reduce the number of false positives without necessarily reducing the number of false negatives, we choose a decision value that has a high value of Precision or a low value of Recall. For example, if we are classifying customers whether they will react positively or negatively to a personalized advertisement, we want to be absolutely sure that the customer will react positively to the advertisement because otherwise, a negative reaction can cause a loss of potential sales from the customer.

Hypothesis representation for logistic regression

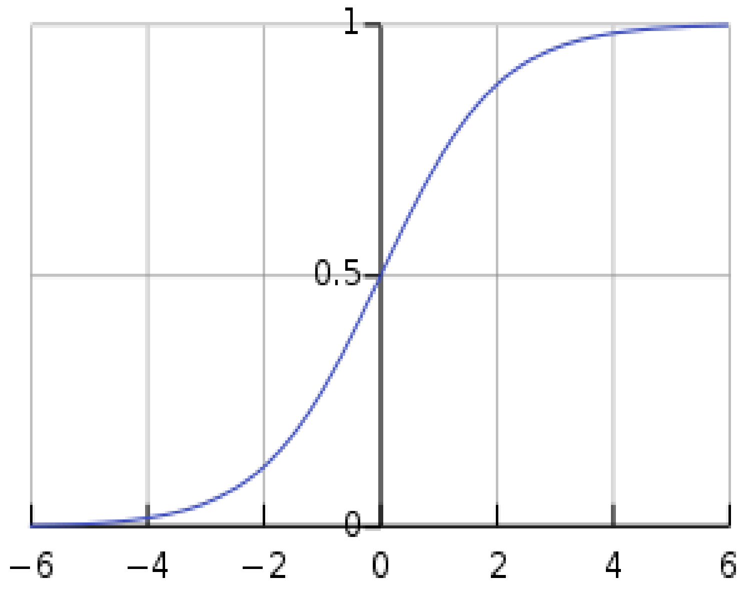
Want $0 \le h_\theta(x) \le 1$

$h_\theta(x) = g(\theta^\top x)$

where $g(z) = \frac{1}{1+e^{-z}}$

$$h_\theta(x) = \frac{1}{1+e^{-\theta^\top x}}$$

Sigmoid function



<u>Steps for algorithm implementation:</u>

- Selection of appropriate data set as per the ML algorithm
- Data preprocessing
- ML Model Implementation with the applicable software
- Train the model with given dataset and test the model by training and test split
- Performance analysis of implemented ML model
- To improve the model performance use Hyperparameter tuning
- Observe the model
- Submit the python notebook

<u>Conclusion</u>

Assessment Rubrics

Performance Area	Rating = 3	Rating = 2	Rating = 1	Rating= 0	Score
Presentation of Markdown Cell	In Markdown cell mentions all details of ML Algorithm and comment properly about Python Steps	In Markdown cell mentions either details of ML Algorithm or comment properly about Python Steps	Student fails to define the application adequately.	Markdown Cell not prepared	
Data Set	Data Set Visualization, Analysis and Preprocessing is done properly.	Only Data Set Visualization or Analysis or Preprocessing is done.	Only Data Set Visualization done.	Data Set operation is not done properly	
ML Model Implementation	ML model train, test with good accuracy and predication done properly . Optimization Parameter Shown	ML model train , test with better accuracy and predication is done , optimization parameter not shown	ML model train , test but accuracy is poor and predication is done , optimization parameter not shown	ML model train , test is not proper and predication is not done , optimization parameter not shown	
				Timely Submission	1 Mark
				Total	10

Expt. No. 3 Date: ____________

Implementation of Decision Tree, Random Forest, KNN, Naïve Bayes with hyperparameter tunning.

Objectives:

- To implement Decision Tree model using given dataset
- To implement Random Forest model using given dataset
- To implement KNN model using given dataset
- To implement Naïve Bayes model using given dataset

Software Requirement:

- Python jupyter notebook

A. Decision Tree

The Decision Tree Algorithm is one such algorithm that is used to solve both Regression and Classification problems. Decision Tree is considered to be one of the most useful Machine Learning algorithms since it can be used to solve a variety of problems. It is considered to be the most understandable Machine Learning algorithm and it can be easily interpreted.
It can be used for classification and regression problems. Unlike most Machine Learning algorithms, it works effectively with non-linear data. Constructing a Decision Tree is a very quick process since it uses only one feature per node to split the data.

Decision Tree Terminology

- **Root Node:** The root node is the starting point of a tree. At this point, the first split is performed.
- **Internal Nodes:** Each internal node represents a decision point (predictor variable) that eventually leads to the prediction of the outcome.
- **Leaf/ Terminal Nodes:** Leaf nodes represent the final class of the outcome and therefore they're also called terminating nodes.
- **Branches:** Branches are connections between nodes, they're represented as arrows. Each branch represents a response such as yes or no.

Two measures are used to decide the best attribute:

1. Information Gain: Information Gain (IG) is the most significant measure used to build a Decision Tree. It indicates how much "information" a particular feature/ variable gives us about the final outcome.
2. Entropy: Entropy measures the impurity or uncertainty present in the data. It is used to decide how a Decision Tree can split the data

Steps for Decision Tree algorithm implementation:

- Selection of appropriate data set as per the ML algorithm
- Data preprocessing
- ML Model Implementation with the applicable software
- Train the model with given dataset and test the model by training and test split
- Performance analysis of implemented ML model
- To improve the model performance use Hyperparameter tuning
- Observe the model
- Submit the python notebook

B. Random Forest

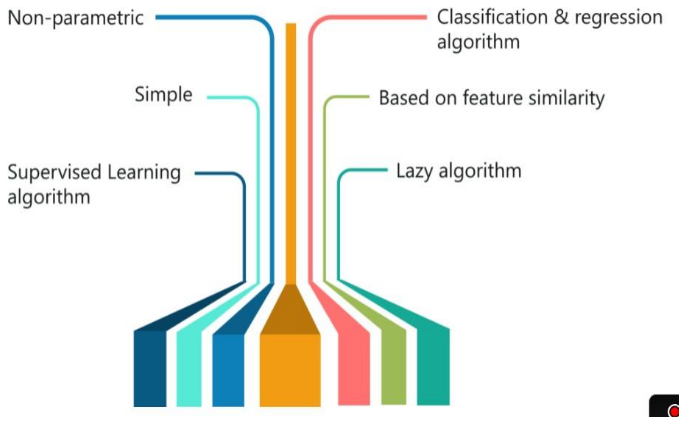
Random forests is an ensemble learning algorithm. The basic premise of the algorithm is that building a small decision-tree with few features is a computationally cheap process. If we can build many small, weak decision trees in parallel, we can then combine the trees to form a single, strong learner by averaging or taking the majority vote. In practice, random forests are often found to be the most accurate learning algorithms to date. The random forest algorithm uses the bagging technique for building an ensemble of decision trees. Bagging is known to reduce the variance of the algorithm.

<u>Steps for Random Forest algorithm implementation:</u>

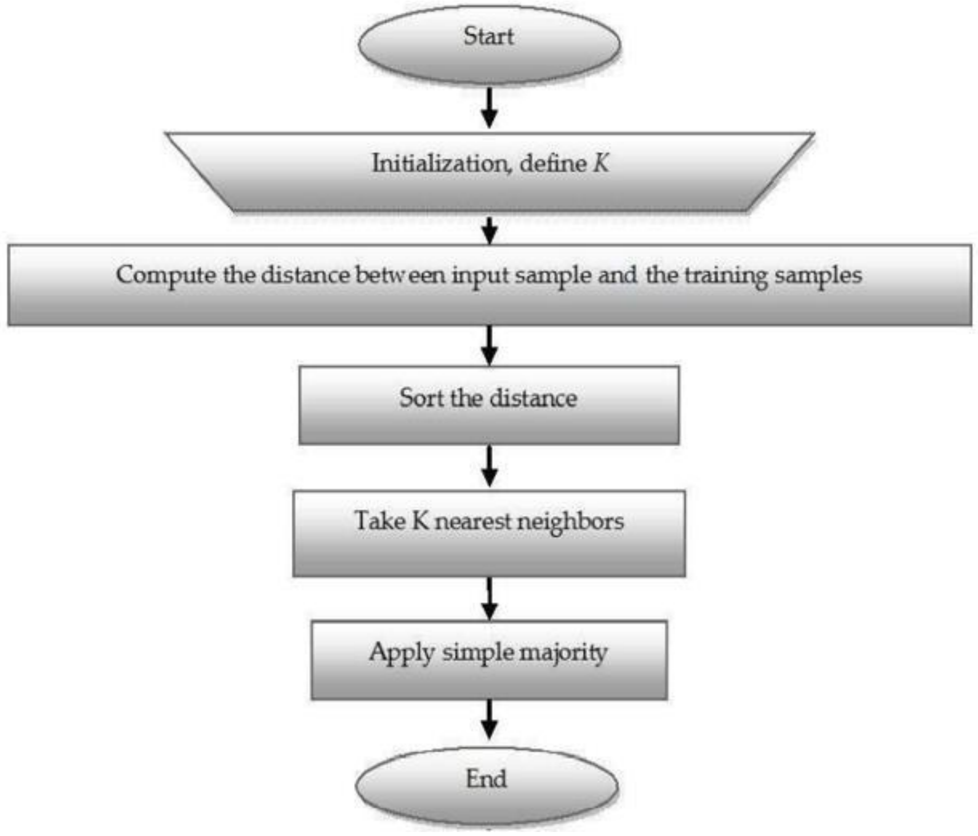
- Selection of appropriate data set as per the ML algorithm
- Data preprocessing
- ML Model Implementation with the applicable software
- Train the model with given dataset and test the model by training and test split
- Performance analysis of implemented ML model
- To improve the model performance use Hyperparameter tuning
- Observe the model
- Submit the python notebook

C. <u>KNN</u>

KNN is a Supervised Learning algorithm that uses labeled input data set to predict the output of the data points. It is one of the simplest Machine learning algorithms and it can be easily implemented for a varied set of problems. It is mainly based on feature similarity. KNN checks how similar a data point is to its neighbor and classifies the data point into the class it is most similar to.


Non-parametric
Simple
Supervised Learning algorithm
Classification & regression algorithm
Based on feature similarity
Lazy algorithm

To determine a value for k, start with k = 1, use a test where the error rate of the classifier set gets estimated. The k value that gives the minimum error rate may be selected.


Start
Initialization, define *K*
Compute the distance between input sample and the training samples
Sort the distance
Take K nearest neighbors
Apply simple majority
End

Steps for KNN algorithm implementation:

- Selection of appropriate data set as per the ML algorithm
- Data preprocessing
- ML Model Implementation with the applicable software
- Classify digits (0 to 9) using KNN classifier. You can use different values for k neighbors and need to figure out a value of K that gives you a maximum score. You can manually try different values of K or use gridsearchcv
- Train the model with given dataset and test the model by training and test split
- Plot confusion matrix
- Observe the model
- Plot classification report
- Submit the python notebook

D. <u>Naïve Bayes</u>

Naive Bayes is among one of the most simple and powerful algorithms for classification based on Bayes' Theorem. Based on an assumption of independence among predictors. Naive Bayes model is easy to build and particularly useful for very large data sets.

There are two parts to this algorithm. The Naive Bayes classifier assumes that the presence of a feature in a class is unrelated to any other feature.

a. Naive
b. Bayes

All of these properties independently contribute to the probability that a particular fruit is an apple or an orange or a banana and that is why it is known as **"Naive".** In Statistics and probability theory, **Bayes'** theorem describes the probability of an event, based on prior knowledge of conditions that might be related to the event.

<u>Applications</u>

1. Spam Classification

Given an email, predict whether it is spam or not

2. Medical Diagnosis

Given a list of symptoms, predict whether a patient has disease X or not

3. Weather

Based on temperature, humidity, etc. predict if it will rain tomorrow

Given a Hypothesis H and evidence E, Bayes' Theorem states that the relationship between the probability of Hypothesis before getting the evidence P(H) and the probability of the hypothesis after getting the evidence P(H|E) is:

$$P(H|E) = \frac{P(E|H).P(H)}{P(E)}$$

This relates the probability of the hypothesis before getting the evidence P(H), to the probability of the hypothesis after getting the evidence, P(H|E).

P(H) is called the prior probability, while P(H|E) is called the posterior probability. The factor that relates the two, P(H|E) / P(E), is called the likelihood ratio.

Steps for algorithm implementation:

- Selection of appropriate data set as per the ML algorithm
- Data preprocessing
- ML Model Implementation with the applicable software
- Train the model with given dataset and test the model by training and test split
- Performance analysis of implemented ML model
- To improve the model performance use Hyperparameter tuning
- Observe the model
- Submit the python notebook

Conclusion

Assessment Rubrics

Performance Area	**Rating = 3**	**Rating = 2**	**Rating = 1**	**Rating= 0**	**Score**
Presentation of Markdown Cell	In Markdown cell mentions all details of ML Algorithm and comment properly about Python Steps	In Markdown cell mentions either details of ML Algorithm or comment properly about Python Steps	Student fails to define the application adequately.	Markdown Cell not prepared	
Data Set	Data Set Visualization, Analysis and Preprocessing is done properly.	Only Data Set Visualization or Analysis or Preprocessing is done.	Only Data Set Visualization done.	Data Set operation is not done properly	
ML Model Implementation	ML model train, test with good accuracy and predication done properly . Optimization Parameter Shown	ML model train , test with better accuracy and predication is done , optimization parameter not shown	ML model train , test but accuracy is poor and predication is done , optimization parameter not shown	ML model train , test is not proper and predication is not done , optimization parameter not shown	
				Timely Submission	1 Mark
				Total	10

Expt. No. 4 Date: ____________

Machine Learning for Image Classification

Objectives:

- To implement SVM for Image Classification
- To implement PCA for Classification

Software Requirement:

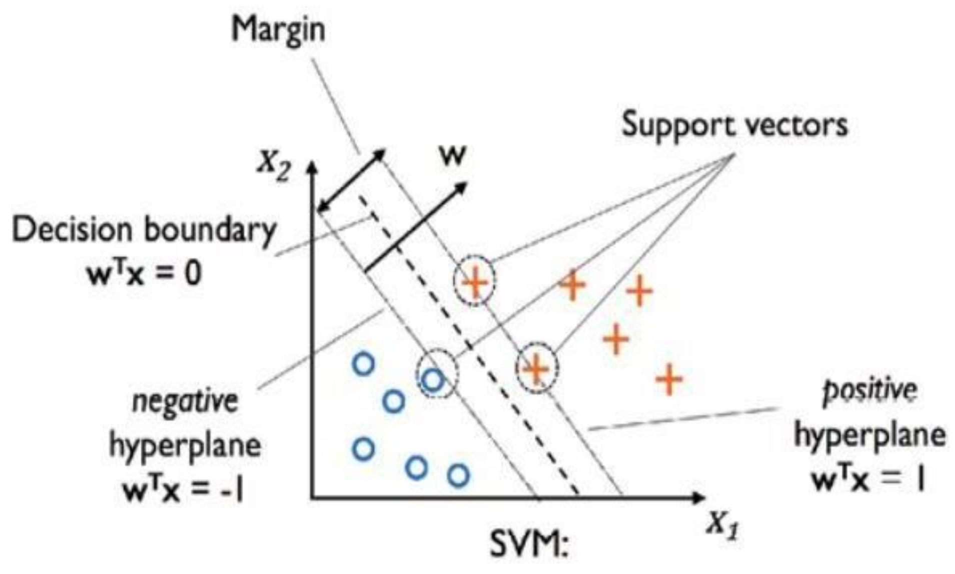
- Python jupyter notebook

Theory

Support Vector Machine is a discriminative classifier that is formally designed by a separative hyperplane. It is a representation of examples as points in space that are mapped so that the points of different categories are separated by a gap as wide as possible. In addition to this, an SVM can also perform non-linear classification. The main objective of a support vector machine is to segregation.

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane. SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.



In some cases, hyperplanes cannot be very efficient. In those cases, the support vector machine uses a kernel trick to transform the input into a higher-dimensional space. With this, it becomes easier to segregate the points. Let us take a look at the SVM kernels.

To perform SVM on multi-class problems, we can create a binary classifier for each class of the data.

The two results of each classifier will be either:

The data point belongs to that class

The data point does not belong to that class.

SVM for complex (Non-Linearly Separable)

SVM works very well without any modifications for linearly separable data. **Linearly Separable Data** is any data that can be plotted in a graph and can be separated into classes using a straight line.

There are various kernel functions available, but two of are very popular:

Radial Basis Function Kernel (RBF): The similarity between two points in the transformed feature space is an exponentially decaying function of the distance between the vectors and the original input space as shown below. RBF is the default kernel used in SVM.

Polynomial Kernel: The Polynomial kernel takes an additional parameter, 'degree' that controls the model's complexity and computational cost of the transformation

PCA

The Principal Component Analysis is a popular unsupervised learning technique for reducing the dimensionality of data. It increases interpretability yet, at the same time, it minimizes information loss.

It helps to find the most significant features in a dataset and makes the data easy for plotting in 2D and 3D. PCA helps in finding a sequence of linear combinations of variables. In the figure, we have several points plotted on a 2-D plane. There are two principal components. PC1 is the primary principal component that explains the maximum variance in the data. PC2 is another principal component that is orthogonal to PC1.

The Principal Components are a straight line that captures most of the variance of the data. They have a direction and magnitude. Principal components are orthogonal projections (perpendicular) of data onto lower-dimensional space. The main idea behind PCA is to figure out patterns and correlations among various features in the data set. On finding a strong correlation between different variables, a final decision is made about reducing the dimensions of the data in such a way that the significant data is still retained. Such a process is very essential in solving complex data-driven problems that involve the use of high-dimensional data sets. PCA can be achieved via a series of steps. Let's discuss the whole end-to-end process.

Steps for algorithm implementation:

- Selection of appropriate data set as per the ML algorithm
- Data preprocessing
- ML Model Implementation with the applicable software
- Train the model with given dataset and test the model by training and test split
- Performance analysis of implemented ML model
- To improve the model performance use Hyperparameter tuning
- Observe the model
- Submit the python notebook
- Measure accuracy of your model using different kernels such as rbf and linear.
- Tune your model further using regularization and gamma parameters and try to come up with highest accuracy score
- Use 80% of samples as training data size

Conclusion

Assessment Rubrics

Performance Area	**Rating = 3**	**Rating = 2**	**Rating = 1**	**Rating= 0**	**Score**
Presentation of Markdown Cell	In Markdown cell mentions all details of ML Algorithm and comment properly about Python Steps	In Markdown cell mentions either details of ML Algorithm or comment properly about Python Steps	Student fails to define the application adequately.	Markdown Cell not prepared	
Data Set	Data Set Visualization, Analysis and Preprocessing is done properly.	Only Data Set Visualization or Analysis or Preprocessing is done.	Only Data Set Visualization done.	Data Set operation is not done properly	
ML Model Implementation	ML model train, test with good accuracy and predication done properly . Optimization Parameter Shown	ML model train , test with better accuracy and predication is done , optimization parameter not shown	ML model train , test but accuracy is poor and predication is done , optimization parameter not shown	ML model train , test is not proper and predication is not done , optimization parameter not shown	
				Timely Submission	1 Mark
				Total	10

Expt. No. 5 Date: ____________

Implementation of Unsupervised Machine Learning

Objectives:

- To implement both the k-means algorithm and the Hierarchical Agglomerative Clustering (HAC) algorithm

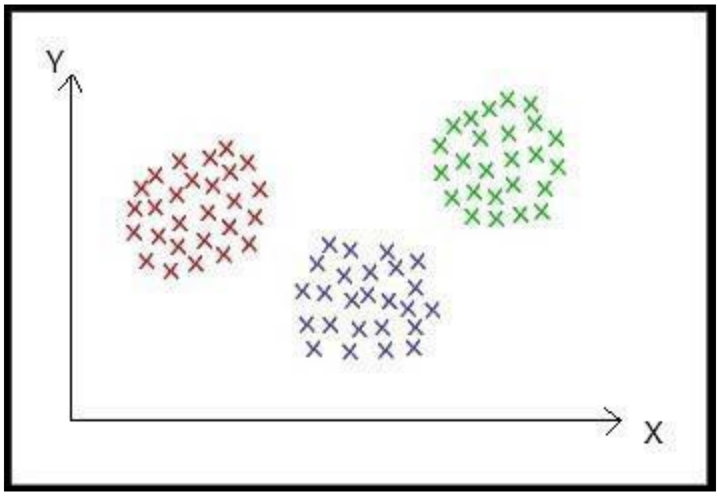
Software Requirement:

- Python jupyter notebook

Theory

Agglomerative Clustering is a type of hierarchical clustering algorithm. It is an unsupervised machine learning technique that divides the population into several clusters such that data points in the same cluster are more similar and data points in different clusters are dissimilar.

- Points in the same cluster are closer to each other.
- Points in the different clusters are far apart.



Sample 2-dimension Dataset

The intuition behind Agglomerative Clustering:

Agglomerative Clustering is a bottom-up approach, initially, each data point is a cluster of its own, further pairs of clusters are merged as one moves up the hierarchy.

- Steps of Agglomerative Clustering:

a) Initially, all the data-points are a cluster of its own.

b) Take two nearest clusters and join them to form one single cluster.

c) Proceed recursively step 2 until you obtain the desired number of clusters.

   To obtain the desired number of clusters, the number of clusters needs to be reduced from initially being n cluster (n equals the total number of data-points).

   Two clusters are combined by computing the similarity between them.

   There are some methods which are used to calculate the similarity between two clusters:

- Distance between two closest points in two clusters.
- Distance between two farthest points in two clusters.
- The average distance between all points in the two clusters.
- Distance between centroids of two clusters.
- There are several pros and cons of choosing any of the above similarity metrics.

<u>Steps for algorithm implementation:</u>

- Selection of appropriate data set as per the ML algorithm
- Data preprocessing
- ML Model Implementation with the applicable software
- Train the model with given dataset and test the model by training and test split
- Performance analysis of implemented ML model
- To improve the model performance use Hyperparameter tuning
- Observe the model
- Submit the python notebook
- Measure accuracy of your model using different kernels such as rbf and linear.
- Tune your model further using regularization and gamma parameters and try to come up with highest accuracy score
- Use 80% of samples as training data size

<u>Conclusion</u>

Assessment Rubrics

Performance Area	Rating = 3	Rating = 2	Rating = 1	Rating= 0	Score
Presentation of Markdown Cell	In Markdown cell mentions all details of ML Algorithm and comment properly about Python Steps	In Markdown cell mentions either details of ML Algorithm or comment properly about Python Steps	Student fails to define the application adequately.	Markdown Cell not prepared	
Data Set	Data Set Visualization, Analysis and Preprocessing is done properly.	Only Data Set Visualization or Analysis or Preprocessing is done.	Only Data Set Visualization done.	Data Set operation is not done properly	
ML Model Implementation	ML model train, test with good accuracy and predication done properly . Optimization Parameter Shown	ML model train , test with better accuracy and predication is done , optimization parameter not shown	ML model train , test but accuracy is poor and predication is done , optimization parameter not shown	ML model train , test is not proper and predication is not done , optimization parameter not shown	
				Timely Submission	1 Mark
				Total	10

Expt. No. 6 Date: _____________

Implementation of IoT Solution using Machine Learning

Objectives:

- To implement IoT application using Machine Learning

Software Requirement:

- Python IDE

Theory

Collect training data: The process begins by collecting training data. In some cases, data has already been collected and is available in a database, or in form of data files. In other cases, especially for IoT scenarios, the data needs to be collected from IoT devices and sensors and stored in the cloud.

We assume that you don't have a collection of turbofan engines, so the project files include a simple device simulator that sends the NASA device data to the cloud.

Prepare data. In most cases, the raw data as collected from devices and sensors will require preparation for machine learning. This step may involve data clean up, data reformatting, or preprocessing to inject additional information machine learning can key off.

For our airplane engine machine data, data preparation involves calculating explicit time-to-failure times for every data point in the sample based on the actual observations on the data. This information allows the machine learning algorithm to find correlations between actual sensor data patterns and the expected remaining life time of the engine. This step is highly domain-specific.

Build a machine learning model. Based on the prepared data, we can now experiment with different machine learning algorithms and parameterizations to train models and compare the results to one another.

In this case, for testing we compare the predicted outcome computed by the model with the real outcome observed on a set of engines. In Azure Machine Learning, we can manage the different iterations of models we create in a model registry.

Deploy the model. Once we have a model that satisfies our success criteria, we can move to deployment. That involves wrapping the model into a web service app that can be fed with data using REST calls and return analysis results. The web service app is then packaged into a docker container, which in turn can be deployed either in the cloud or as an IoT Edge module. In this example, we focus on deployment to IoT Edge. **Maintain and refine the model**. Our work is not done once the model is deployed. In many cases, we want to continue collecting data and periodically upload that data to the cloud. We can then use this data to retrain and refine our model, which we then can redeploy to IoT Edge.

Steps for algorithm implementation:

1. Select IoT application based on Machine Learning with reference to research paper
2. Prepare poster presentation as per following guidelines:

   a. Define Problem Statement of selected IoT application.
   b. Describe objectives.
   c. Flow diagram/block diagram
   d. Explanation of data collection using the applied IoT sensor
   e. Insights of IoT sensor collected data preprocessing/feature engineering and analysis
   f. Explain/Implement ML model for prediction/classification for selected IoT applications.
   g. Describe ML model deployment using cloud
   h. Result and conclusion
   i. References.

Conclusion

Assessment Rubrics

Performance Area	Rating = 3	Rating = 2	Rating = 1	Score
Defining the Application	Student states the application clearly and identifies underlying issues.	Student adequately defines the application.	Student fails to define the application adequately.	
Machine Learning model implementation after analyzing the application with proper dataset	Data Set Visualization, Analysis and implementation is done properly.	Only Data Set Visualization or Analysis or Preprocessing is done.	Only Data Set Visualization done.	
Presentation Quality	Presentation flows well and logically. Poster presentation reflects extensive use of tools in a creative way.	Presentation flows well. Some tools used to show acceptable Understanding on Poster.	Presentation is unorganized. Tools are not used in relevant manner nor represented appropriate information in poster.	
			Timely Submission	1 Mark
			Total	10

Expt. No. 7 Date: _____________

ANN for Computer Vision

Objectives:

- To implement IoT application using Machine Learning

Software Requirement:

Python, Keras Environment

Theory

A. **Computer vision** is a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos and other visual inputs — and take actions or make recommendations based on that information. If AI enables computers to think, computer vision enables them to see, observe and understand.

Computer vision works much the same as human vision, except humans have a head start. Human sight has the advantage of lifetimes of context to train how to tell objects apart, how far away they are, whether they are moving and whether there is something wrong in an image.

Computer vision trains machines to perform these functions, but it has to do it in much less time with cameras, data and algorithms rather than retinas, optic nerves and a visual cortex. Because a system trained to inspect products or watch a production asset can analyse thousands of products or processes a minute, noticing imperceptible defects or issues, it can quickly surpass human capabilities.

Computer vision is used in industries ranging from energy and utilities to manufacturing and automotive – and the market is continuing to grow.

B. Artificial Neural Network

Neural Networks is a computational learning system that uses a network of functions to understand and translate a data input of one form into a desired output, usually in another form. The concept of the artificial neural network was inspired by human biology and the way neurons of the human brain function together to understand inputs from human senses.

In simple words, Neural Networks are a set of algorithms that tries to recognize the patterns, relationships, and information from the data through the process which is inspired by and works like the human brain/biology.

An Artificial Neural Network is specified by:

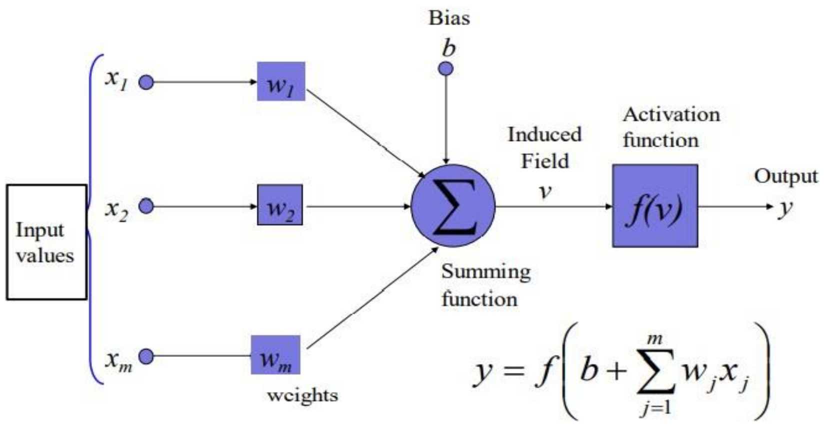
1. Neuron Model: The processing unit of the ANN which performs a linear combination of inputs.
2. Architecture: Just like the set of neurons in brain. The neurons are connected by links which have weight.
3. Learning Algorithm: Modifies the weight of links to model a specific task. The training relies heavily on the data fed to the neurons

Neuron

The neuron provides a linear combination of the input provided to it and the applies a non-linear activation function to it.

The weights of the links are represented as w_j and the inputs as x_j for the j th input and neuron.

Activation function: 'b' represents bias. y = f (u + b)



Perceptron

The perceptron is used for binary classification.

First train a perceptron for a classification task.

Find suitable weights in such a way that the training examples are correctly classified.

The perceptron can only model linearly separable classes.

Steps for algorithm implementation

- Download the dataset
- Image preprocessing
- Design ANN model using Keras
- Train ANN model for selected dataset
- Measure accuracy of your model

Conclusion

Assessment Rubrics

Performance Area	**Rating = 3**	**Rating = 2**	**Rating = 1**	**Rating= 0**	**Score**
Presentation of Markdown Cell	In Markdown cell mentions all details of ML Algorithm and comment properly about Python Steps	In Markdown cell mentions either details of ML Algorithm or comment properly about Python Steps	Student fails to define the application adequately.	Markdown Cell not prepared	
Data Set	Data Set Visualization, Analysis and Preprocessing is done properly.	Only Data Set Visualization or Analysis or Preprocessing is done.	Only Data Set Visualization done.	Data Set operation is not done properly	
ML Model Implementation	ML model train, test with good accuracy and predication done properly . Optimization Parameter Shown	ML model train , test with better accuracy and predication is done , optimization parameter not shown	ML model train , test but accuracy is poor and predication is done , optimization parameter not shown	ML model train , test is not proper and predication is not done , optimization parameter not shown	
				Timely Submission	1 Mark
				Total	10

Expt. No. 8 Date: ____________

Open CV for Computer Vision

Objectives:

- To Open CV Library for Image Processing

Software Requirement:

Theory

OpenCV supports a wide variety of programming languages such as C++, Python, Java, etc., and is available on different platforms including Windows, Linux, OS X, Android, and iOS. Interfaces for high-speed GPU operations based on CUDA and OpenCL are also under active development.

OpenCV-Python is the Python API for OpenCV, combining the best qualities of the OpenCV C++ API and the Python language.

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library.

OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.

Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available.

Steps for algorithm implementation

- Open PyCharm
- Click on new project
- Select the location and name for project.
- Select the version of python to be use as base interpreter.
- Click on create.
- It will create a virtual environment for your project.
- To Run your First Project, click on Run ‘main’
- See the output on terminal, which is included in PyCharm IDE.

Conclusion

Assessment Rubrics

Performance Area	Rating = 3	Rating = 2	Rating = 1	Rating= 0	Score
Presentation of Markdown Cell	In Markdown cell mentions all details of ML Algorithm and comment properly about Python Steps	In Markdown cell mentions either details of ML Algorithm or comment properly about Python Steps	Student fails to define the application adequately.	Markdown Cell not prepared	
Data Set	Data Set Visualization, Analysis and Preprocessing is done properly.	Only Data Set Visualization or Analysis or Preprocessing is done.	Only Data Set Visualization done.	Data Set operation is not done properly	
ML Model Implementation	ML model train, test with good accuracy and predication done properly . Optimization Parameter Shown	ML model train , test with better accuracy and predication is done , optimization parameter not shown	ML model train , test but accuracy is poor and predication is done , optimization parameter not shown	ML model train , test is not proper and predication is not done , optimization parameter not shown	
				Timely Submission	1 Mark
				Total	10

Name : Kshitij V Darwhekar

Roll No: TETB19

Sub: Soft Computitng

Batch: B2

Experiment 1 : Experimental Data Analysis: Perform following operations on any open dataset available in Python/Kaggle

```
import numpy as np
import pandas as pd
```

```
from google.colab import drive
drive.mount('/content/drive/')

#data = open('ML/penguins_size','r')
```

```
    Mounted at /content/drive/
```

```
database = pd.read_csv('/content/drive/MyDrive/ML/penguins_size.csv')
```

```
database.head()
```

	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mas
0	Adelie	Torgersen	39.1	18.7	181.0	375
1	Adelie	Torgersen	39.5	17.4	186.0	380
2	Adelie	Torgersen	40.3	18.0	195.0	325
3	Adelie	Torgersen	NaN	NaN	NaN	N
4	Adelie	Torgersen	36.7	19.3	193.0	345

```
database.head(10)
```

	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mas
0	Adelie	Torgersen	39.1	18.7	181.0	375
1	Adelie	Torgersen	39.5	17.4	186.0	380
2	Adelie	Torgersen	40.3	18.0	195.0	325
3	Adelie	Torgersen	NaN	NaN	NaN	N
4	Adelie	Torgersen	36.7	19.3	193.0	345
5	Adelie	Torgersen	39.3	20.6	190.0	365
6	Adelie	Torgersen	38.9	17.8	181.0	362

```
database.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 344 entries, 0 to 343
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   species            344 non-null    object 
 1   island             344 non-null    object 
 2   culmen_length_mm   342 non-null    float64
 3   culmen_depth_mm    342 non-null    float64
 4   flipper_length_mm  342 non-null    float64
 5   body_mass_g        342 non-null    float64
 6   sex                334 non-null    object 
dtypes: float64(4), object(3)
memory usage: 18.9+ KB
```

```
print(database.isnull().sum())
```

```
species               0
island                0
culmen_length_mm      2
culmen_depth_mm       2
flipper_length_mm     2
body_mass_g           2
sex                  10
dtype: int64
```

```
database = database.dropna()
database.head()
```

	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mas
0	Adelie	Torgersen	39.1	18.7	181.0	375
1	Adelie	Torgersen	39.5	17.4	186.0	380
2	Adelie	Torgersen	40.3	18.0	195.0	325
4	Adelie	Torgersen	36.7	19.3	193.0	345
5	Adelie	Torgersen	39.3	20.6	190.0	365

```
len(database)
```

```
334
```

```
len(database.columns)
```

```
7
```

```
database.loc[(database['sex'] != 'FEMALE')& (database['sex'] != 'MALE')]
```

	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mas
336	Gentoo	Biscoe	44.5	15.7	217.0	487

```
database['culmen_depth_mm'].fillna((database['culmen_depth_mm'].mean()), inplace=True)
database['flipper_length_mm'].fillna((database['flipper_length_mm'].mean()), inplace=True)
database['body_mass_g'].fillna((database['body_mass_g'].mean()), inplace=True)
database['culmen_length_mm'].fillna((database['culmen_length_mm'].mean()), inplace=True)
database['sex'].fillna((database['sex'].value_counts().index[0]), inplace=True)

database.reset_index()
database.head()
```

	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mas
0	Adelie	Torgersen	39.1	18.7	181.0	375
1	Adelie	Torgersen	39.5	17.4	186.0	380
2	Adelie	Torgersen	40.3	18.0	195.0	325
4	Adelie	Torgersen	36.7	19.3	193.0	345
5	Adelie	Torgersen	39.3	20.6	190.0	365

```
col_new = ['new_species','new_island','new_culmen_length_mm','new_culmen_depth_mm','new_fl
database.columns = col_new
col_new
```

```
['new_species',
 'new_island',
 'new_culmen_length_mm',
 'new_culmen_depth_mm',
 'new_flipper_length',
 'new_body_mass_g',
 'new_sex']
```

```
database.head()
```

	new_species	new_island	new_culmen_length_mm	new_culmen_depth_mm	new_flipper_l
0	Adelie	Torgersen	39.1	18.7	
1	Adelie	Torgersen	39.5	17.4	
2	Adelie	Torgersen	40.3	18.0	

```
database_new = database.drop(['new_island','new_culmen_length_mm','new_flipper_length'],ax
database.head()
```

	new_species	new_island	new_culmen_length_mm	new_culmen_depth_mm	new_flipper_l
0	Adelie	Torgersen	39.1	18.7	
1	Adelie	Torgersen	39.5	17.4	
2	Adelie	Torgersen	40.3	18.0	
4	Adelie	Torgersen	36.7	19.3	
5	Adelie	Torgersen	39.3	20.6	

```
database_new.head()
```

	new_species	new_culmen_depth_mm	new_body_mass_g	new_sex
0	Adelie	18.7	3750.0	MALE
1	Adelie	17.4	3800.0	FEMALE
2	Adelie	18.0	3250.0	FEMALE
4	Adelie	19.3	3450.0	FEMALE
5	Adelie	20.6	3650.0	MALE

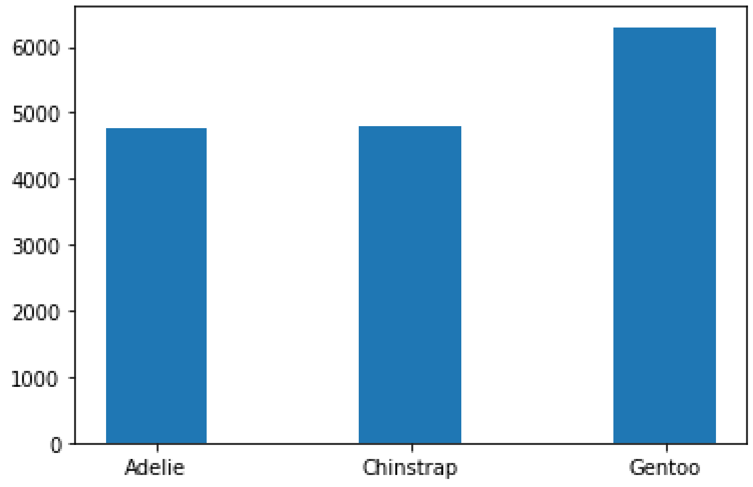
```
database_new["islands"] = "Torgersen"
database_new.head()
```

	new_species	new_culmen_depth_mm	new_body_mass_g	new_sex	islands
0	Adelie	18.7	3750.0	MALE	Torgersen
1	Adelie	17.4	3800.0	FEMALE	Torgersen
2	Adelie	18.0	3250.0	FEMALE	Torgersen
4	Adelie	19.3	3450.0	FEMALE	Torgersen
5	Adelie	20.6	3650.0	MALE	Torgersen

```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
X = database['new_species']
Y = database['new_body_mass_g']
plt.bar(X,Y,width = 0.4)
```

```
<BarContainer object of 334 artists>
```



+ Code + Text

✓ 0s completed at 1:41 PM

Name : Kshitij V Darwhekar

Roll No: TETB19

Sub: Soft Computitng

Batch: B2

Experiment 2: Liner Regression and Logistic Regression Model Implementation on Given Dataset.

```
import matplotlib.pyplot as plt
import seaborn as sns
import·pandas·as·pd
import·numpy·as·np
```

```
from google.colab import drive
drive.mount('/content/drive/')
```

```
     Mounted at /content/drive/
```

```
pf = pd.read_csv("/content/drive/MyDrive/ML/heart.csv")
```

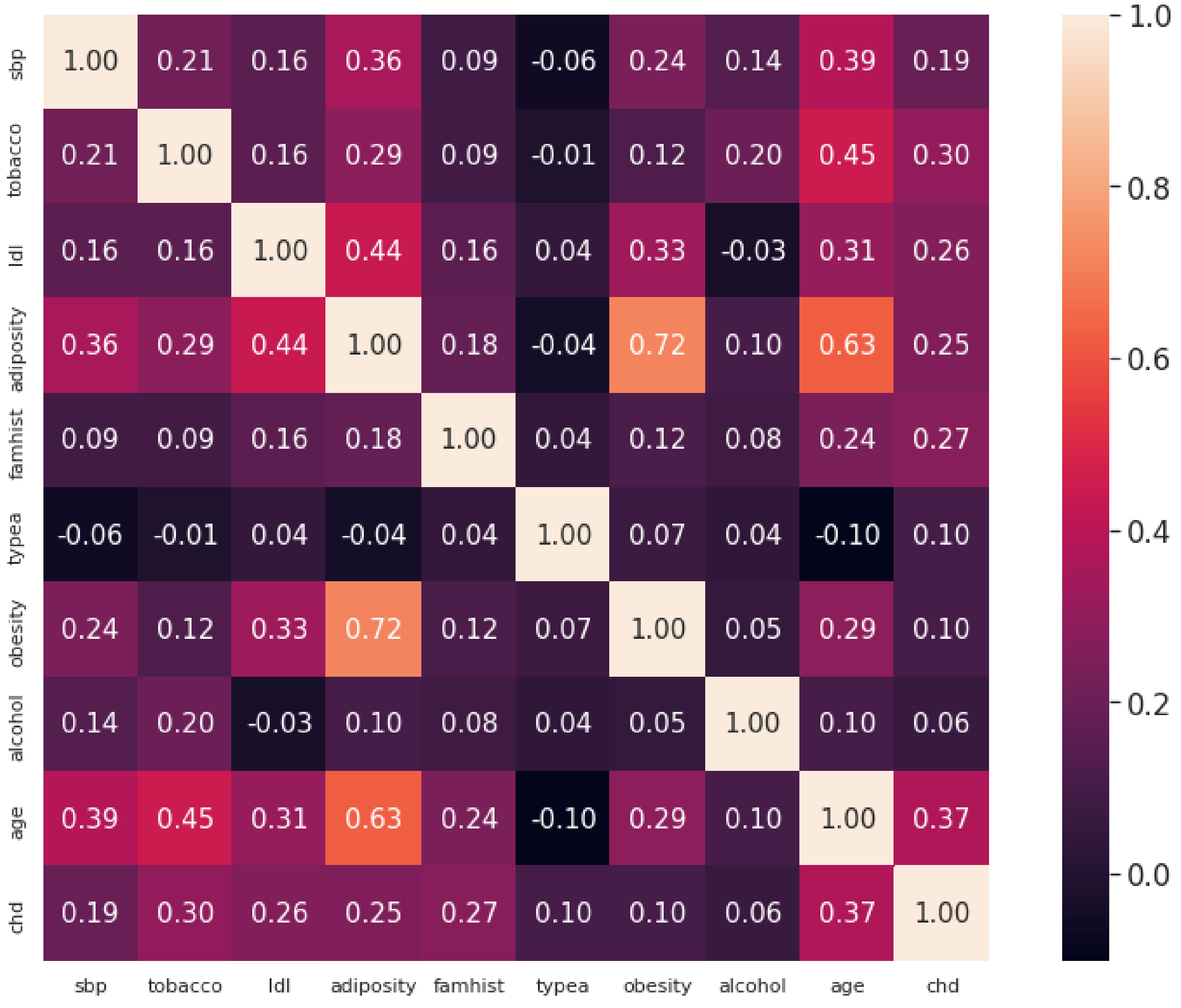
```
pf.head()
```

	sbp	tobacco	ldl	adiposity	famhist	typea	obesity	alcohol	age	chd
0	160	12.00	5.73	23.11	Present	49	25.30	97.20	52	1
1	144	0.01	4.41	28.61	Absent	55	28.87	2.06	63	1
2	118	0.08	3.48	32.28	Present	52	29.14	3.81	46	0
3	170	7.50	6.41	38.03	Present	51	31.99	24.26	58	1
4	134	13.60	3.50	27.78	Present	60	25.99	57.34	49	1

```
history_mapping = {'Absent': 0,'Present': 1}
pf["famhist"] = pf["famhist"].map(history_mapping)
pf.head()
```

	sbp	tobacco	ldl	adiposity	famhist	typea	obesity	alcohol	age	chd
0	160	12.00	5.73	23.11	1	49	25.30	97.20	52	1
1	144	0.01	4.41	28.61	0	55	28.87	2.06	63	1
2	118	0.08	3.48	32.28	1	52	29.14	3.81	46	0

```
sns.set(style='whitegrid', context='notebook')
cols = ['sbp','tobacco','ldl','adiposity','famhist','typea','obesity', 'alcohol','age', 'c
f, ax = plt.subplots(figsize=(15, 10))
cm = np.corrcoef(pf[cols].values.T)
sns.set(font_scale=1.5)
hm = sns.heatmap(cm,
                 cbar=True,
                 annot=True,
                 square=True,
                 fmt='.2f',
                 annot_kws={'size': 15},
                 yticklabels=cols,
                 xticklabels=cols)

plt.show()
```



```
X = pf[['tobacco','ldl','adiposity','famhist','typea','obesity','alcohol','age']].values
y = pf[['chd']].values
```

```
from sklearn.model_selection import train_test_split
```

```
X_train , X_test , y_train,y_test = train_test_split(X,y,train_size = 0.9)
```

```
# Apply logistic regression

from sklearn.linear_model import LogisticRegression
```

```
model = LogisticRegression(C=1,penalty='l2')
model.fit(X_train,y_train)
y_pred=model.predict(X_test)
```

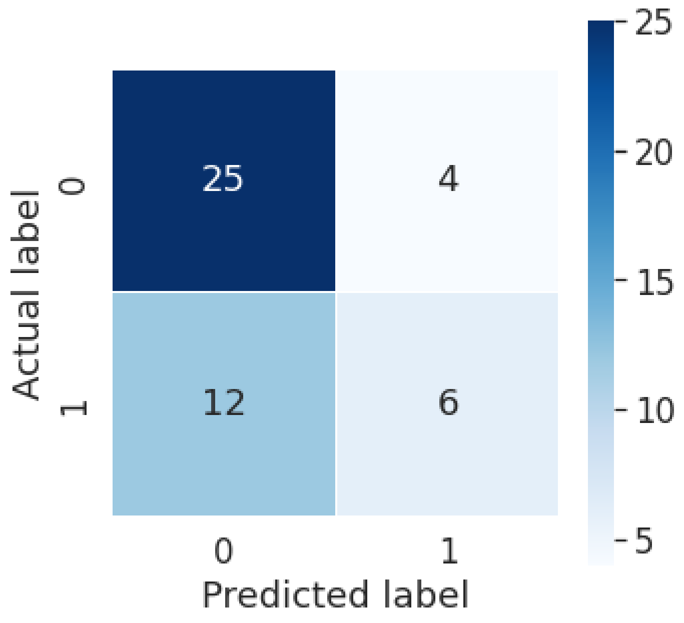
```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:993: DataConversic
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818: Converg
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
```

```
print ('Training Accuracy: %.2f' % model.score(X_train,y_train))

print ('Test Accuracy: %.2f' % model.score(X_test,y_test))
```

```
Training Accuracy: 0.74
Test Accuracy: 0.66
```

```
import seaborn as sns
from sklearn.tree import plot_tree
from sklearn import tree
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test,y_pred)
plt.figure(figsize=(5,5))
sns.heatmap(data=cm,linewidths=.5, annot=True,square =True, cmap ='Blues')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

```
Text(0.5, 37.79999999999998, 'Predicted label')
```



Name- Kshitij V Darwhekar

Roll No - TETB19

Sub : Soft Computitng

Batch -B2

Experiment 3 : Implementation of Decision Tree, Random Forest, KNN, Naïve Bayes with hyperparameter tunning.

1. DECISION TREE

```
import pandas as pd
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
     Mounted at /content/drive
```

```
df = pd.read_csv("/content/drive/MyDrive/ML/Titanic-Dataset.csv")
df.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th	female	38.0	1	0	PC 17599	71.

```
df.drop(['PassengerId','Name','SibSp','Parch','Ticket','Cabin','Embarked'],axis='columns',
```

```
df.head()
```

	Survived	Pclass	Sex	Age	Fare
0	0	3	male	22.0	7.2500
1	1	1	female	38.0	71.2833
2	1	3	female	26.0	7.9250

```
inputs = df.drop('Survived',axis='columns')
target = df.Survived
```

```
inputs.Sex = inputs.Sex.map({'male': 1, 'female': 2})
```

```
inputs.Age[:10]
```

```
0    22.0
1    38.0
2    26.0
3    35.0
4    35.0
5     NaN
6    54.0
7     2.0
8    27.0
9    14.0
Name: Age, dtype: float64
```

```
inputs.Age = inputs.Age.fillna(inputs.Age.mean())
```

```
inputs.head()
```

	Pclass	Sex	Age	Fare
0	3	1	22.0	7.2500
1	1	2	38.0	71.2833
2	3	2	26.0	7.9250
3	1	2	35.0	53.1000
4	3	1	35.0	8.0500

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(inputs,target,test_size=0.2)
```

```
len(X_train)
```

```
712
```

```
len(X_test)
```

```
     179
```

```
from sklearn import tree
model = tree.DecisionTreeClassifier()
```

```
model.fit(X_train,y_train)
```

```
     DecisionTreeClassifier()
```

```
model.score(X_test,y_test)
```

```
     0.7877094972067039
```

✓ 0s completed at 1:47 PM ● ×

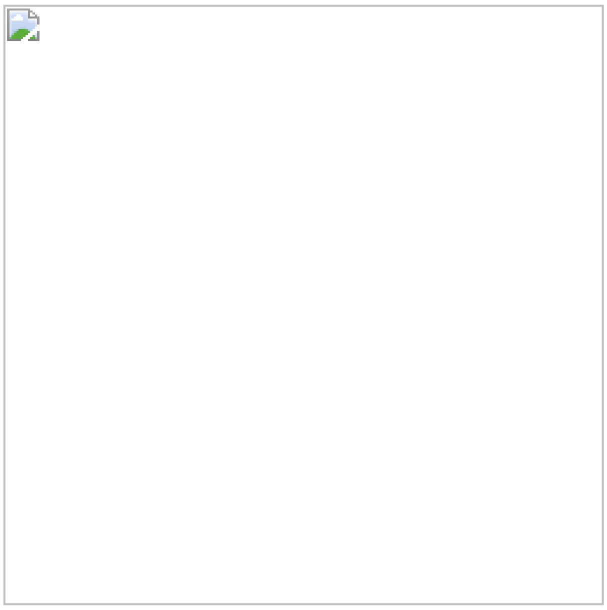
Name- Kshitij V Darwhekar

Roll No - TETB19

Batch -B2

2. KNN (K Nearest Neighbors) Classification

```
import pandas as pd
from sklearn.datasets import load_iris
iris = load_iris()
```



```
iris.feature_names
```

```
     ['sepal length (cm)',
      'sepal width (cm)',
      'petal length (cm)',
      'petal width (cm)']
```

```
iris.target_names
```

```
     array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

```
df = pd.DataFrame(iris.data,columns=iris.feature_names)
df.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2

```
df['target'] = iris.target
df.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

```
df[df.target==1].head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
50	7.0	3.2	4.7	1.4	1
51	6.4	3.2	4.5	1.5	1
52	6.9	3.1	4.9	1.5	1
53	5.5	2.3	4.0	1.3	1
54	6.5	2.8	4.6	1.5	1

```
df[df.target==2].head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
100	6.3	3.3	6.0	2.5	2
101	5.8	2.7	5.1	1.9	2
102	7.1	3.0	5.9	2.1	2
103	6.3	2.9	5.6	1.8	2
104	6.5	3.0	5.8	2.2	2

```
df['flower_name'] =df.target.apply(lambda x: iris.target_names[x])
df.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	flower_name
0	5.1	3.5	1.4	0.2	0	setosa
1	4.9	3.0	1.4	0.2	0	setosa
2	4.7	3.2	1.3	0.2	0	setosa
3	4.6	3.1	1.5	0.2	0	setosa

```
df[45:55]
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	flower_name
45	4.8	3.0	1.4	0.3	0	setosa
46	5.1	3.8	1.6	0.2	0	setosa
47	4.6	3.2	1.4	0.2	0	setosa
48	5.3	3.7	1.5	0.2	0	setosa
49	5.0	3.3	1.4	0.2	0	setosa
50	7.0	3.2	4.7	1.4	1	versicolor
51	6.4	3.2	4.5	1.5	1	versicolor
52	6.9	3.1	4.9	1.5	1	versicolor
53	5.5	2.3	4.0	1.3	1	versicolor
54	6.5	2.8	4.6	1.5	1	versicolor

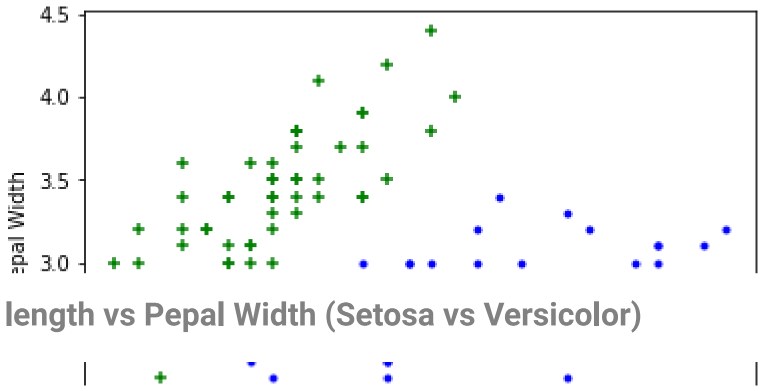
```
df0 = df[:50]
df1 = df[50:100]
df2 = df[100:]
```

```
import matplotlib.pyplot as plt
%matplotlib inline
```

Sepal length vs Sepal Width (Setosa vs Versicolor)

```
plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')
plt.scatter(df0['sepal length (cm)'], df0['sepal width (cm)'],color="green",marker='+')
plt.scatter(df1['sepal length (cm)'], df1['sepal width (cm)'],color="blue",marker='.')
```

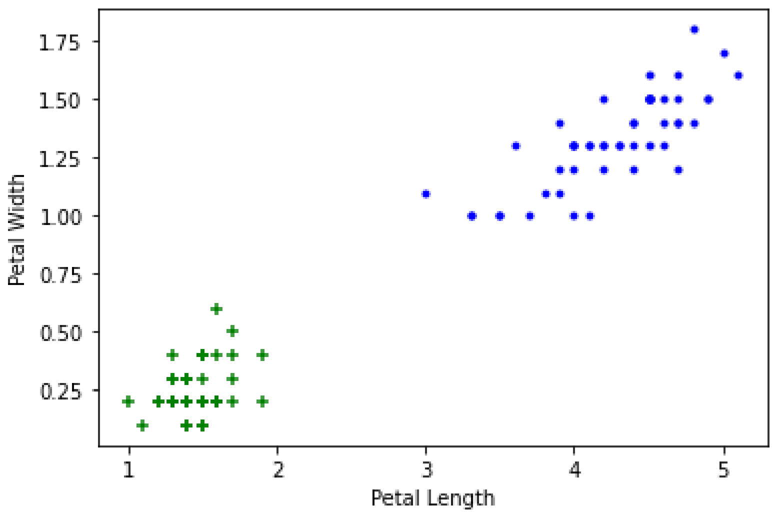
```
<matplotlib.collections.PathCollection at 0x7f8d07945f50>
```



Petal length vs Pepal Width (Setosa vs Versicolor)

```
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
plt.scatter(df0['petal length (cm)'], df0['petal width (cm)'],color="green",marker='+')
plt.scatter(df1['petal length (cm)'], df1['petal width (cm)'],color="blue",marker='.')
```

```
<matplotlib.collections.PathCollection at 0x7f8d07437910>
```



Train test split

```
from sklearn.model_selection import train_test_split
```

```
X = df.drop(['target','flower_name'], axis='columns')
y = df.target
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)
```

```
len(X_train)
```

```
120
```

```
len(X_test)
```

```
30
```

Create KNN (K Neighrest Neighbour Classifier)

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=10)
```

```
knn.fit(X_train, y_train)
```

```
     KNeighborsClassifier(n_neighbors=10)
```

```
knn.score(X_test, y_test)
```

```
     0.9666666666666667
```

```
knn.predict([[4.8,3.0,1.5,0.3]])
```

```
     /usr/local/lib/python3.7/dist-packages/sklearn/base.py:451: UserWarning: X does not h
       "X does not have valid feature names, but"
     array([0])
```

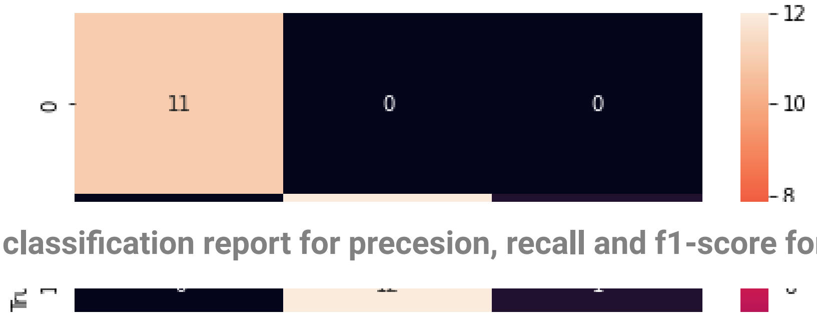
Plot Confusion Matrix

```
from sklearn.metrics import confusion_matrix
y_pred = knn.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
cm
```

```
     array([[11,  0,  0],
            [ 0, 12,  1],
            [ 0,  0,  6]])
```

```
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sn
plt.figure(figsize=(7,5))
sn.heatmap(cm, annot=True)
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

```
Text(42.0, 0.5, 'Truth')
```



Print classification report for precesion, recall and f1-score for each classes

```
from sklearn.metrics import classification_report

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
           1       1.00      0.92      0.96        13
           2       0.86      1.00      0.92         6

    accuracy                           0.97        30
   macro avg       0.95      0.97      0.96        30
weighted avg       0.97      0.97      0.97        30
```

✓ 0s completed at 1:46 PM

Name : Kshitij V Darwhekar

Roll No: TETB19

Sub: Soft Computitng

Batch: B2

3. RANDOM FOREST

```
import pandas as pd
from sklearn.datasets import load_digits
digits = load_digits()
```

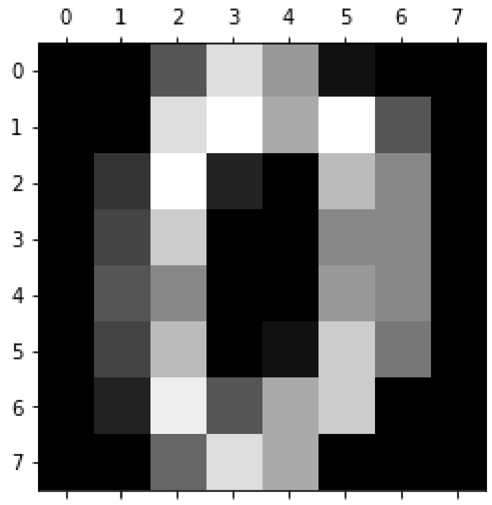
```
dir(digits)
```

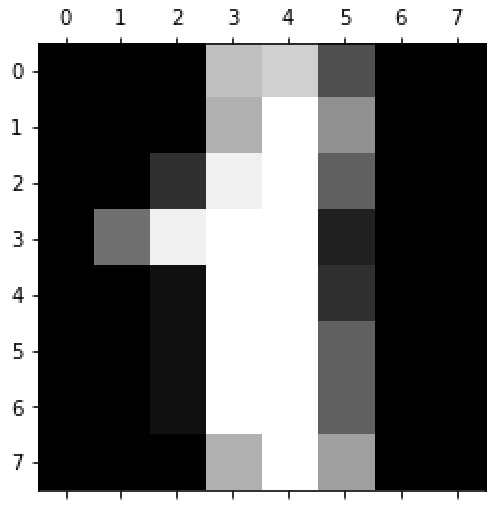
```
     ['DESCR', 'data', 'feature_names', 'frame', 'images', 'target', 'target_names']
```

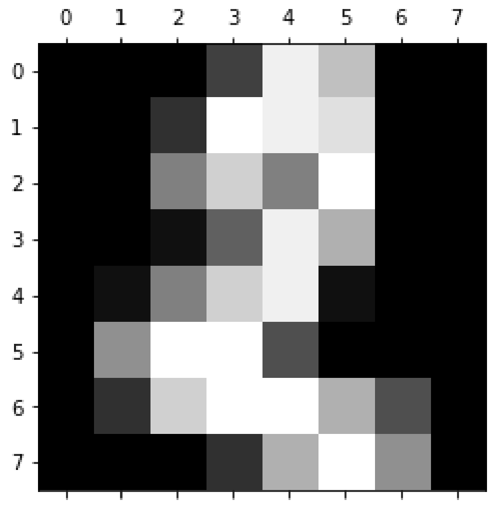
```
%matplotlib inline
import matplotlib.pyplot as plt
```

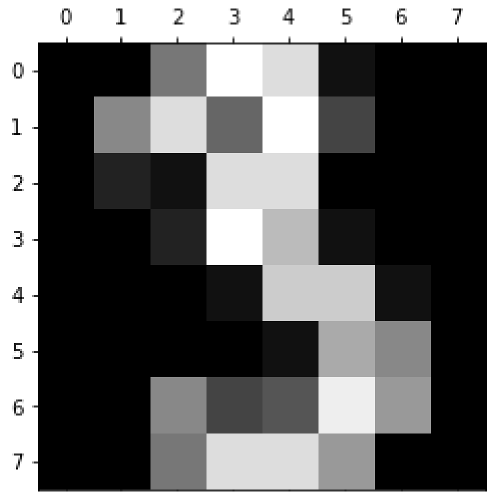
```
plt.gray()
for i in range(10):
  plt.matshow(digits.images[i])
```

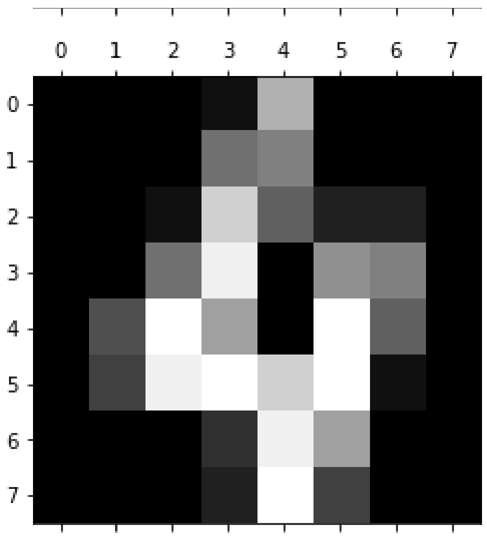
```
<Figure size 432x288 with 0 Axes>
```

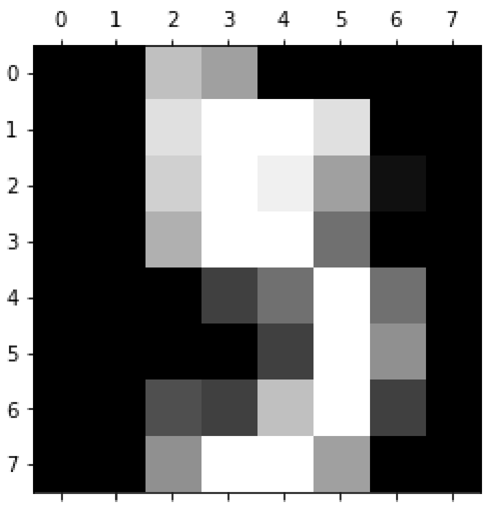

0
1
2
3
4
5
6
7

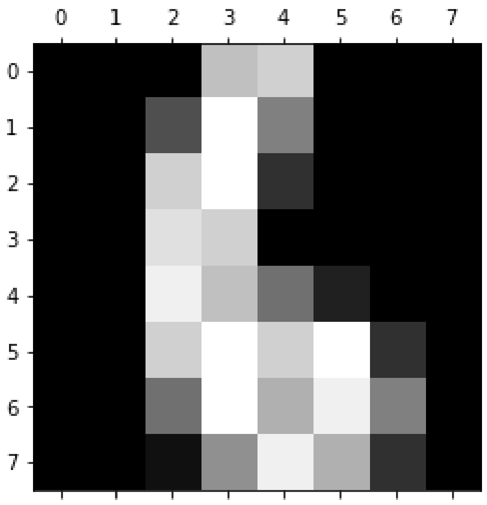

0
1
2
3
4
5
6
7

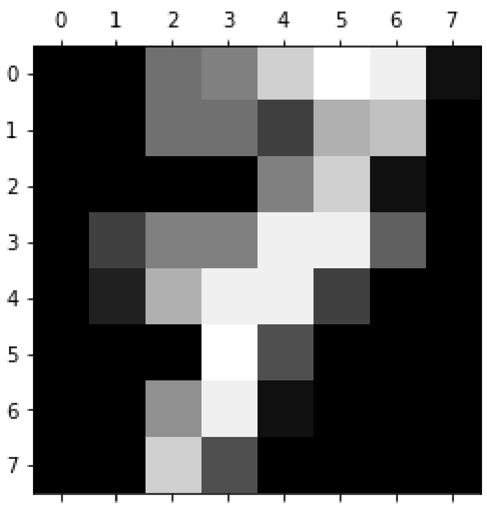

0
1
2
3
4
5
6
7

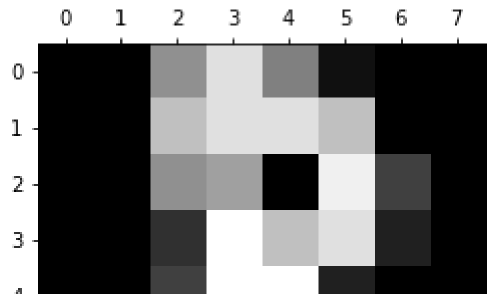

0
1
2
3
4
5
6
7


0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7


0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7


0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7


0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7



```
df = pd.DataFrame(digits.data)
df.head()
```

	0	1	2	3	4	5	6	7	8	9	...	54	55	56	57	58	59	
0	0.0	0.0	5.0	13.0	9.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	6.0	13.0	1
1	0.0	0.0	0.0	12.0	13.0	5.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	11.0	1
2	0.0	0.0	0.0	4.0	15.0	12.0	0.0	0.0	0.0	0.0	...	5.0	0.0	0.0	0.0	0.0	3.0	1
3	0.0	0.0	7.0	15.0	13.0	1.0	0.0	0.0	0.0	8.0	...	9.0	0.0	0.0	0.0	7.0	13.0	1
4	0.0	0.0	0.0	1.0	11.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	2.0	1

5 rows × 64 columns

```
df['target'] = digits.target
```

```
df[0:12]
```

	0	1	2	3	4	5	6	7	8	9	...	55	56	57	58	59	
0	0.0	0.0	5.0	13.0	9.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	6.0	13.0	1
1	0.0	0.0	0.0	12.0	13.0	5.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	11.0	1
2	0.0	0.0	0.0	4.0	15.0	12.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	3.0	1
3	0.0	0.0	7.0	15.0	13.0	1.0	0.0	0.0	0.0	8.0	...	0.0	0.0	0.0	7.0	13.0	1
4	0.0	0.0	0.0	1.0	11.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	2.0	1
5	0.0	0.0	12.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	9.0	16.0	1
6	0.0	0.0	0.0	12.0	13.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	1.0	9.0	1
7	0.0	0.0	7.0	8.0	13.0	16.0	15.0	1.0	0.0	0.0	...	0.0	0.0	0.0	13.0	5.0	
8	0.0	0.0	9.0	14.0	8.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	11.0	16.0	1
9	0.0	0.0	11.0	12.0	0.0	0.0	0.0	0.0	0.0	2.0	...	0.0	0.0	0.0	9.0	12.0	1
10	0.0	0.0	1.0	9.0	15.0	11.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	1.0	10.0	1
11	0.0	0.0	0.0	0.0	14.0	13.0	1.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	1.0	1

12 rows × 65 columns

```
## Train the model and prediction
```

```
X = df.drop('target',axis = 'columns')
y = df.target
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.1)
```

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=30)
model.fit(X_train, y_train)
```

```
     RandomForestClassifier(n_estimators=30)
```

```
model.score(X_test, y_test)
```

```
     0.95
```

```
y_predicted = model.predict(X_test)
```

```
from sklearn.datasets import make_classification
```

```
## Confusion Matrix
```

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_predicted)
cm
```

```
     array([[18,  0,  0,  0,  0,  0,  0,  0,  0,  0],
            [ 0, 17,  0,  0,  0,  0,  0,  0,  0,  0],
            [ 0,  0, 15,  0,  0,  0,  0,  0,  0,  0],
            [ 0,  0,  0, 25,  0,  0,  0,  0,  0,  0],
            [ 0,  0,  0,  0, 15,  0,  0,  0,  0,  0],
            [ 0,  0,  0,  0,  1, 18,  1,  0,  0,  1],
            [ 0,  0,  0,  0,  0,  0, 18,  0,  0,  0],
            [ 0,  0,  0,  0,  0,  0,  0, 15,  0,  0],
            [ 0,  0,  1,  2,  0,  0,  0,  0, 14,  0],
            [ 0,  1,  0,  0,  0,  1,  0,  0,  1, 16]])
```

```
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sn
plt.figure(figsize=(10,7))
sn.heatmap(cm, annot=True)
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

Name : Kshitij V Darwhekar

Roll No: TETB19

Sub: Soft Computitng

Batch: B2

4. NAIVE BAYES

```
import warnings
warnings.filterwarnings('ignore')
```

```
from sklearn import datasets
from sklearn import metrics
from sklearn.naive_bayes import GaussianNB
```

```
from sklearn.datasets import load_digits
dataset = load_digits()
```

```
model = GaussianNB()
model.fit(dataset.data, dataset.target)
```

```
     GaussianNB()
```

```
## Predictions
```

```
expected = dataset.target
predicted = model.predict(dataset.data)
```

```
print(metrics.classification_report(expected, predicted))
print(metrics.confusion_matrix(expected, predicted))
```

```
                   precision    recall  f1-score   support

                0       0.99      0.99      0.99       178
                1       0.83      0.85      0.84       182
                2       0.98      0.64      0.77       177
                3       0.94      0.79      0.86       183
                4       0.98      0.84      0.90       181
                5       0.91      0.93      0.92       182
                6       0.96      0.99      0.98       181
                7       0.72      0.99      0.83       179
                8       0.58      0.86      0.69       174
                9       0.94      0.71      0.81       180
```

```
    accuracy                           0.86      1797
   macro avg       0.88      0.86      0.86      1797
weighted avg       0.89      0.86      0.86      1797

[[176   0   0   0   1   0   0   1   0   0]
 [  0 154   0   0   0   0   3   5  14   6]
 [  0  13 113   0   0   1   1   0  49   0]
 [  0   2   2 145   0   6   0   7  20   1]
 [  1   1   0   0 152   1   2  21   3   0]
 [  0   0   0   3   0 169   1   6   2   1]
 [  0   1   0   0   0   1 179   0   0   0]
 [  0   0   0   0   1   1   0 177   0   0]
 [  0   8   0   1   0   3   0  12 150   0]
 [  1   6   0   5   1   3   0  17  20 127]]
```

```
# Multinomial Naive Bayes

from sklearn.naive_bayes import MultinomialNB
model = MultinomialNB()


model.fit(dataset.data, dataset.target)
expected = dataset.target
predicted = model.predict(dataset.data)
print(metrics.classification_report(expected, predicted))
print(metrics.confusion_matrix(expected, predicted))
```

```
              precision    recall  f1-score   support

           0       0.99      0.98      0.99       178
           1       0.87      0.75      0.81       182
           2       0.90      0.90      0.90       177
           3       0.99      0.87      0.93       183
           4       0.96      0.96      0.96       181
           5       0.97      0.86      0.91       182
           6       0.98      0.97      0.98       181
           7       0.89      0.99      0.94       179
           8       0.78      0.89      0.83       174
           9       0.76      0.88      0.82       180

    accuracy                           0.91      1797
   macro avg       0.91      0.91      0.91      1797
weighted avg       0.91      0.91      0.91      1797

[[175   0   0   0   3   0   0   0   0   0]
 [  0 137  14   0   0   1   2   0  13  15]
 [  0   7 160   0   0   0   0   0   8   2]
 [  0   0   2 159   0   2   0   5   8   7]
 [  1   0   0   0 173   0   0   4   3   0]
 [  0   0   0   0   1 157   1   1   2  20]
 [  0   2   0   0   1   1 176   0   1   0]
 [  0   0   0   0   0   0   0 178   1   0]
 [  0  11   1   0   1   0   1   1 154   5]
 [  0   1   0   1   1   1   0  11   7 158]]
```

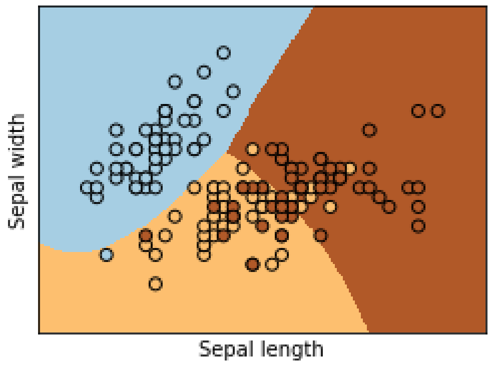
```
## Bernoulli Naive bayes
```

```
from sklearn.naive_bayes import BernoulliNB
model = BernoulliNB()

model.fit(dataset.data, dataset.target)
expected = dataset.target
predicted = model.predict(dataset.data)
print(metrics.classification_report(expected, predicted))
print(metrics.confusion_matrix(expected, predicted))
```

```
                   precision    recall  f1-score   support

                0       0.98      0.98      0.98       178
                1       0.76      0.62      0.68       182
                2       0.86      0.86      0.86       177
                3       0.91      0.86      0.88       183
                4       0.91      0.95      0.93       181
                5       0.93      0.82      0.87       182
                6       0.97      0.94      0.96       181
                7       0.88      0.98      0.93       179
                8       0.70      0.82      0.75       174
                9       0.76      0.81      0.78       180

         accuracy                           0.86      1797
        macro avg       0.87      0.86      0.86      1797
     weighted avg       0.87      0.86      0.86      1797

     [[175   1   0   0   2   0   0   0   0   0]
      [  0 112  21   0   3   1   1   1  32  11]
      [  0   6 153   6   0   0   0   1  11   0]
      [  1   1   3 157   0   2   0   3   7   9]
      [  0   1   0   0 172   0   0   7   1   0]
      [  2   3   0   2   1 149   2   0   3  20]
      [  0   5   0   0   2   2 171   0   1   0]
      [  0   0   0   0   3   0   0 175   1   0]
      [  0  13   1   4   0   3   2   2 142   7]
      [  0   6   0   3   7   3   0   9   6 146]]
```

```
##

import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline


def Naive_bayes(Model_Type):
        # import some data to play with
        iris = datasets.load_iris()
        X = iris.data[:, :2]  # we only take the first two features.
        Y = iris.target
        h = .02  # step size in the mesh
        # we create an instance of Neighbours Classifier and fit the data.
        if(Model_Type=='Gaussian'):
            model =  GaussianNB()
        elif (Model_Type=='Multinomial'):
                model =  MultinomialNB()
        else:
                model =  BernoulliNB()
        model.fit(X, Y)
        # Plot the decision boundary. For that, we will assign a color to each
        # point in the mesh [x_min, m_max]x[y_min, y_max].
        x_min, x_max = X[:, 0].min() - .5, X[:, 0].max() + .5
        y_min, y_max = X[:, 1].min() - .5, X[:, 1].max() + .5
        xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
        Z = model.predict(np.c_[xx.ravel(), yy.ravel()])

        # Put the result into a color plot
        Z = Z.reshape(xx.shape)
        plt.figure(1, figsize=(4, 3))
        plt.pcolormesh(xx, yy, Z, cmap=plt.cm.Paired)

        # Plot also the training points
        plt.scatter(X[:, 0], X[:, 1], c=Y, edgecolors='k', cmap=plt.cm.Paired)
        plt.xlabel('Sepal length')
        plt.ylabel('Sepal width')
        plt.xlim(xx.min(), xx.max())
        plt.ylim(yy.min(), yy.max())
        plt.xticks(())
        plt.yticks(())
        plt.show()

        model.fit(dataset.data, dataset.target)
        expected = dataset.target
        predicted = model.predict(dataset.data)
        print(metrics.classification_report(expected, predicted))
        print(metrics.confusion_matrix(expected, predicted))


from IPython.html import widgets
from IPython.html.widgets import interact
from IPython.display import display
import warnings
warnings.filterwarnings('ignore')


i = interact(Naive_bayes, Model_Type=['Gaussian','Multinomial','Bernoulli'])
```

Model_Type Gaussian



```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       178
           1       0.83      0.85      0.84       182
           2       0.98      0.64      0.77       177
           3       0.94      0.79      0.86       183
           4       0.98      0.84      0.90       181
           5       0.91      0.93      0.92       182
           6       0.96      0.99      0.98       181
           7       0.72      0.99      0.83       179
           8       0.58      0.86      0.69       174


    accuracy                           0.86      1797
   macro avg       0.88      0.86      0.86      1797
weighted avg       0.89      0.86      0.86      1797

[[176   0   0   0   1   0   0   1   0   0]
 [  0 154   0   0   0   0   3   5  14   6]
 [  0  13 113   0   0   1   1   0  49   0]
 [  0   2   2 145   0   6   0   7  20   1]
 [  1   1   0   0 152   1   2  21   3   0]
 [  0   0   0   3   0 169   1   6   2   1]
 [  0   1   0   0   0   1 179   0   0   0]
 [  0   0   0   0   1   1   0 177   0   0]
 [  0   8   0   1   0   3   0  12 150   0]
 [  1   6   0   5   1   3   0  17  20 127]]
```

Name : Kshitij V Darwhekar

Roll No: TETB19

Sub: Soft Computitng

Batch: B2

Experiment 4: Machine Learning for Image Classification (Support Vector Machine Tutorial Using Python Sklearn)

```
import pandas as pd
from sklearn.datasets import load_iris
iris = load_iris()
```

```
iris.feature_names
```

```
     ['sepal length (cm)',
      'sepal width (cm)',
      'petal length (cm)',
      'petal width (cm)']
```

```
iris.target_names
```

```
     array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

```
df = pd.DataFrame(iris.data,columns=iris.feature_names)
```

```
df.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

```
df['target'] = iris.target
df.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
df[df.target==1].head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
50	7.0	3.2	4.7	1.4	1
51	6.4	3.2	4.5	1.5	1
52	6.9	3.1	4.9	1.5	1
53	5.5	2.3	4.0	1.3	1
54	6.5	2.8	4.6	1.5	1

```
df[df.target==2].head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
100	6.3	3.3	6.0	2.5	2

```
df['flower_name'] =df.target.apply(lambda x: iris.target_names[x])
df.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	flower_name
0	5.1	3.5	1.4	0.2	0	setosa
1	4.9	3.0	1.4	0.2	0	setosa
2	4.7	3.2	1.3	0.2	0	setosa
3	4.6	3.1	1.5	0.2	0	setosa
4	5.0	3.6	1.4	0.2	0	setosa

```
df[45:55]
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	flower_name
45	4.8	3.0	1.4	0.3	0	setosa
46	5.1	3.8	1.6	0.2	0	setosa
47	4.6	3.2	1.4	0.2	0	setosa
48	5.3	3.7	1.5	0.2	0	setosa
49	5.0	3.3	1.4	0.2	0	setosa
50	7.0	3.2	4.7	1.4	1	versicolor
51	6.4	3.2	4.5	1.5	1	versicolor
52	6.9	3.1	4.9	1.5	1	versicolor
53	5.5	2.3	4.0	1.3	1	versicolor
54	6.5	2.8	4.6	1.5	1	versicolor

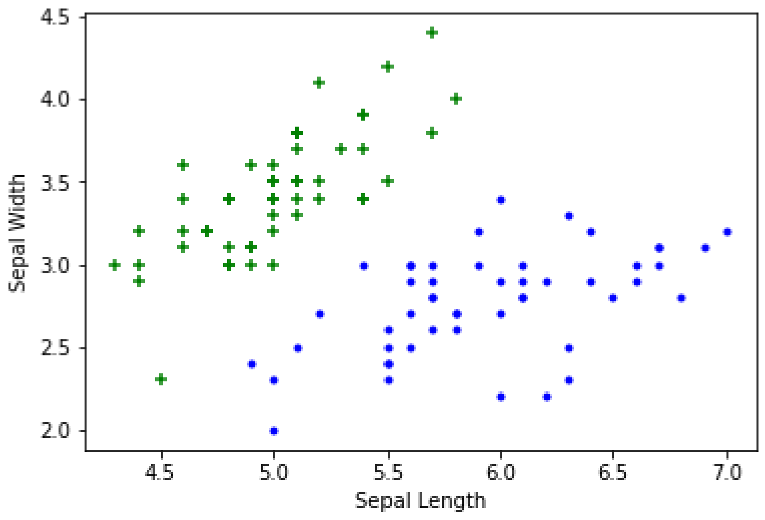
```
df0 = df[:50]
df1 = df[50:100]
df2 = df[100:]
```

```
import matplotlib.pyplot as plt
%matplotlib inline
```

Sepal length vs Sepal Width (Setosa vs Versicolor)

```
plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')
plt.scatter(df0['sepal length (cm)'], df0['sepal width (cm)'],color="green",marker='+')
plt.scatter(df1['sepal length (cm)'], df1['sepal width (cm)'],color="blue",marker='.')
```

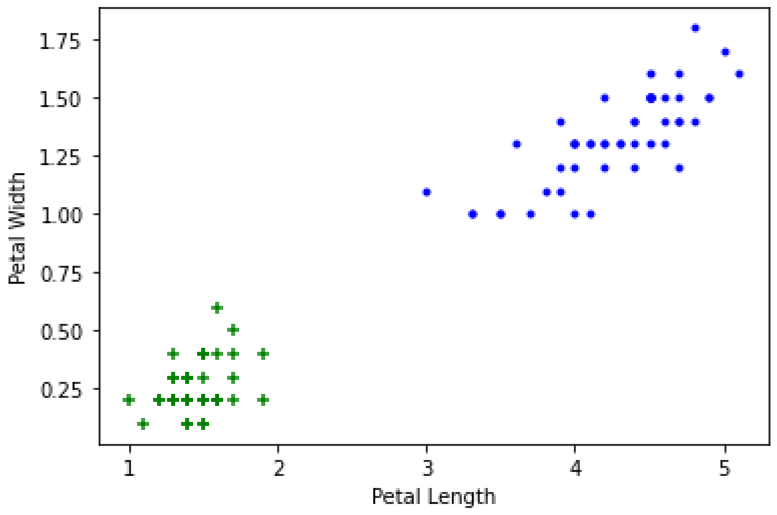
```
<matplotlib.collections.PathCollection at 0x7f14938e3a10>
```



Petal length vs Pepal Width (Setosa vs Versicolor)

```
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
plt.scatter(df0['petal length (cm)'], df0['petal width (cm)'],color="green",marker='+')
plt.scatter(df1['petal length (cm)'], df1['petal width (cm)'],color="blue",marker='.')
```

```
<matplotlib.collections.PathCollection at 0x7f14933d5b10>
```



Train Using Support Vector Machine (SVM)

```
from sklearn.model_selection import train_test_split
```

```
X = df.drop(['target','flower_name'], axis='columns')
y = df.target
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

```
len(X_train)
```

```
     120
```

```
len(X_test)
```

```
     30
```

```
from sklearn.svm import SVC
model = SVC()
```

```
model.fit(X_train, y_train)
```

```
     SVC()
```

```
model.score(X_test, y_test)
```

```
     0.9666666666666667
```

```
model.predict([[4.8,3.0,1.5,0.3]])
```

```
     /usr/local/lib/python3.7/dist-packages/sklearn/base.py:451: UserWarning: X does not h
       "X does not have valid feature names, but"
     array([0])
```

Tune parameters

1. Regularization (C)

The Regularization parameter (often termed as C parameter in python's sklearn library) tells the SVM optimization how much you want to avoid misclassifying each training example.

```
model_C = SVC(C=1)
model_C.fit(X_train, y_train)
model_C.score(X_test, y_test)
```

```
     0.9666666666666667
```

```
model_C = SVC(C=10)
model_C.fit(X_train, y_train)
model_C.score(X_test, y_test)
```

```
     0.9666666666666667
```

2. Gamma

Gamma parameter: gamma determines the distance a single data sample exerts influence. That is, the gamma parameter can be said to adjust the curvature of the decision boundary.

```
model_g = SVC(gamma=10)
model_g.fit(X_train, y_train)
model_g.score(X_test, y_test)
```

```
    0.9666666666666667
```

3. Kernel

A kernel is a specialized kind of similarity function. It takes two points as input, and returns their similarity as output, just as a similarity metric does. A mathematical result from linear algebra known as Mercer's theorem has the implication that a broad class of functions (e.g. similarity metrics) may be expressed in terms of a dot product in some (possibly very and even infinitely) high dimensional space. This means that calculations performed on points in high-dimensional spaces may be restated in terms of dot products

```
model_linear_kernal = SVC(kernel='linear')
model_linear_kernal.fit(X_train, y_train)
```

```
    SVC(kernel='linear')
```

```
model_linear_kernal.score(X_test, y_test)
```

```
    0.9666666666666667
```

Exercise

Train SVM classifier using sklearn digits dataset (i.e. from sklearn.datasets import load_digits) and then,

1. Measure accuracy of your model using different kernels such as rbf and linear.
2. Tune your model further using regularization and gamma parameters and try to come up with highest accurancy score
3. Use 80% of samples as training data size

```
import pandas as pd

import numpy as np

import sklearn

import matplotlib.pyplot as plt
```

```
from sklearn.datasets import load_digits

import seaborn as sns


digits=load_digits()

print(digits)
```

```
     {'data': array([[ 0.,  0.,  5., ...,  0.,  0.,  0.],
            [ 0.,  0.,  0., ..., 10.,  0.,  0.],
            [ 0.,  0.,  0., ..., 16.,  9.,  0.],
            ...,
            [ 0.,  0.,  1., ...,  6.,  0.,  0.],
            [ 0.,  0.,  2., ..., 12.,  0.,  0.],
            [ 0.,  0., 10., ..., 12.,  1.,  0.]]), 'target': array([0, 1, 2, ..., 8, 9, 8
             [ 0.,  0., 13., ..., 15.,  5.,  0.],
             [ 0.,  3., 15., ..., 11.,  8.,  0.],
             ...,
             [ 0.,  4., 11., ..., 12.,  7.,  0.],
             [ 0.,  2., 14., ..., 12.,  0.,  0.],
             [ 0.,  0.,  6., ...,  0.,  0.,  0.]],

            [[ 0.,  0.,  0., ...,  5.,  0.,  0.],
             [ 0.,  0.,  0., ...,  9.,  0.,  0.],
             [ 0.,  0.,  3., ...,  6.,  0.,  0.],
             ...,
             [ 0.,  0.,  1., ...,  6.,  0.,  0.],
             [ 0.,  0.,  1., ...,  6.,  0.,  0.],
             [ 0.,  0.,  0., ..., 10.,  0.,  0.]],

            [[ 0.,  0.,  0., ..., 12.,  0.,  0.],
             [ 0.,  0.,  3., ..., 14.,  0.,  0.],
             [ 0.,  0.,  8., ..., 16.,  0.,  0.],
             ...,
             [ 0.,  9., 16., ...,  0.,  0.,  0.],
             [ 0.,  3., 13., ..., 11.,  5.,  0.],
             [ 0.,  0.,  0., ..., 16.,  9.,  0.]],

            ...,

            [[ 0.,  0.,  1., ...,  1.,  0.,  0.],
             [ 0.,  0., 13., ...,  2.,  1.,  0.],
             [ 0.,  0., 16., ..., 16.,  5.,  0.],
             ...,
             [ 0.,  0., 16., ..., 15.,  0.,  0.],
             [ 0.,  0., 15., ..., 16.,  0.,  0.],
             [ 0.,  0.,  2., ...,  6.,  0.,  0.]],

            [[ 0.,  0.,  2., ...,  0.,  0.,  0.],
             [ 0.,  0., 14., ..., 15.,  1.,  0.],
             [ 0.,  4., 16., ..., 16.,  7.,  0.],
             ...,
             [ 0.,  0.,  0., ..., 16.,  2.,  0.],
             [ 0.,  0.,  4., ..., 16.,  2.,  0.],
             [ 0.,  0.,  5., ..., 12.,  0.,  0.]],

            [[ 0.,  0., 10., ...,  1.,  0.,  0.],
             [ 0.,  2., 16., ...,  1.,  0.,  0.],
             [ 0.,  0., 15., ..., 15.,  0.,  0.],
             ...,
             [ 0.,  4., 16., ..., 16.,  6.,  0.],
             [ 0.,  8., 16., ..., 16.,  8.,  0.],
             [ 0.,  1.,  8., ..., 12.,  1.,  0.]]]), 'DESCR': ".. _digits_dataset:\n\nOpti
```

```
digits.keys()
```

```
dict_keys(['data', 'target', 'frame', 'feature_names', 'target_names', 'images', 'DES
```

```
df=pd.DataFrame(digits.data)

print(df.head())

print(df.shape)
```

```
     0    1    2     3     4     5    6    7    8    9  ...   54   55   56  \
0  0.0  0.0  5.0  13.0   9.0   1.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0
1  0.0  0.0  0.0  12.0  13.0   5.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0
2  0.0  0.0  0.0   4.0  15.0  12.0  0.0  0.0  0.0  0.0  ...  5.0  0.0  0.0
3  0.0  0.0  7.0  15.0  13.0   1.0  0.0  0.0  0.0  8.0  ...  9.0  0.0  0.0
4  0.0  0.0  0.0   1.0  11.0   0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0

    57   58    59    60    61   62   63
0  0.0  6.0  13.0  10.0   0.0  0.0  0.0
1  0.0  0.0  11.0  16.0  10.0  0.0  0.0
2  0.0  0.0   3.0  11.0  16.0  9.0  0.0
3  0.0  7.0  13.0  13.0   9.0  0.0  0.0
4  0.0  0.0   2.0  16.0   4.0  0.0  0.0

[5 rows x 64 columns]
(1797, 64)
```

```
df.columns
```

```
RangeIndex(start=0, stop=64, step=1)
```

```
df.isnull().sum()
```

```
0     0
1     0
2     0
3     0
4     0
     ..
59    0
60    0
61    0
62    0
63    0
Length: 64, dtype: int64
```

```
df['target']=digits.target
```

```
df.head()
```

	0	1	2	3	4	5	6	7	8	9	...	55	56	57	58	59	60
0	0.0	0.0	5.0	13.0	9.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	6.0	13.0	10.0
1	0.0	0.0	0.0	12.0	13.0	5.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	11.0	16.0
2	0.0	0.0	0.0	4.0	15.0	12.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	3.0	11.0
3	0.0	0.0	7.0	15.0	13.0	1.0	0.0	0.0	0.0	8.0	...	0.0	0.0	0.0	7.0	13.0	13.0
4	0.0	0.0	0.0	1.0	11.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	2.0	16.0

5 rows × 65 columns

```
print(digits.data.shape)

print(digits.target.shape)
```

```
(1797, 64)
(1797,)
```

```
df.target
```

```
0       0
1       1
2       2
3       3
4       4
       ..
1792    9
1793    0
1794    8
1795    9
1796    8
Name: target, Length: 1797, dtype: int64
```

```
df.values
```

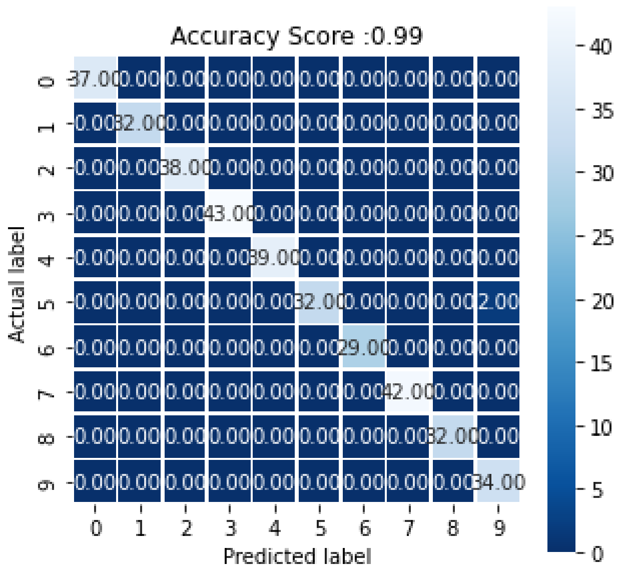
```
array([[ 0.,  0.,  5., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ...,  0.,  0.,  1.],
       [ 0.,  0.,  0., ...,  9.,  0.,  2.],
       ...,
       [ 0.,  0.,  1., ...,  0.,  0.,  8.],
       [ 0.,  0.,  2., ...,  0.,  0.,  9.],
       [ 0.,  0., 10., ...,  1.,  0.,  8.]])
```

```
from sklearn.model_selection import train_test_split

x=df.drop(['target'],axis='columns')
```

```
y=df.target

x_train,x_test,y_train,y_test= train_test_split(x,y,test_size=0.2,random_state=12)



print(len(x_train))

print(len(x_test))

     1437
     360

from sklearn.metrics import accuracy_score

from sklearn.svm import SVC

model1=SVC(kernel='rbf',random_state=0, probability=True)

model1.fit(x_train,y_train)

y_pred_1=model1.predict(x_test)

print("Model Score of Kernal(rbf) :", model1.score(x_test,y_test))


     Model Score of Kernal(rbf) : 0.9916666666666667

model2=SVC(kernel='linear',random_state=0, probability=True)

model2.fit(x_train,y_train)

y_pred_2=model2.predict(x_test)

print("Model Score of Kernal(linear) :", model2.score(x_test,y_test))



     Model Score of Kernal(linear) : 0.975

model3=SVC(kernel='poly',random_state=0, probability=True)

model3.fit(x_train,y_train)

y_pred_3=model3.predict(x_test)

print("Model Score of Kernal(poly) :", model3.score(x_test,y_test))

     Model Score of Kernal(poly) : 0.9944444444444445

accuracy=accuracy_score(y_test,y_pred_3)
```

```
print('ACCURACY is',accuracy)

     ACCURACY is 0.9944444444444445

from sklearn.metrics import confusion_matrix

cm=np.array(confusion_matrix(y_test,y_pred_3))

cm

     array([[37,  0,  0,  0,  0,  0,  0,  0,  0,  0],
            [ 0, 32,  0,  0,  0,  0,  0,  0,  0,  0],
            [ 0,  0, 38,  0,  0,  0,  0,  0,  0,  0],
            [ 0,  0,  0, 43,  0,  0,  0,  0,  0,  0],
            [ 0,  0,  0,  0, 39,  0,  0,  0,  0,  0],
            [ 0,  0,  0,  0,  0, 32,  0,  0,  0,  2],
            [ 0,  0,  0,  0,  0,  0, 29,  0,  0,  0],
            [ 0,  0,  0,  0,  0,  0,  0, 42,  0,  0],
            [ 0,  0,  0,  0,  0,  0,  0,  0, 32,  0],
            [ 0,  0,  0,  0,  0,  0,  0,  0,  0, 34]])

from sklearn.metrics import mean_squared_error

mse=mean_squared_error(y_test,y_pred_3)

mse

     0.08888888888888889



model1_C=SVC(C=3)

model1_C.fit(x_train,y_train)

model1_C.score(x_test,y_test)

     0.9944444444444445


model2_C=SVC(C=3)

model2_C.fit(x_train,y_train)

model2_C.score(x_test,y_test)


     0.9944444444444445


model3_C=SVC(C=3)

model3_C.fit(x_train,y_train)

model3_C.score(x_test,y_test)

     0.9944444444444445
```

```
plt.figure(figsize=(5,5))

sns.heatmap(cm, annot=True, fmt=".2f", linewidths=.5, square = True, cmap = 'Blues_r')

plt.ylabel('Actual label')

plt.xlabel('Predicted label')

A=f'Accuracy Score :{accuracy:.2f}'

plt.title(A)

plt.show()
```



✓ 1s completed at 1:54 PM

Name : Kshitij V Darwhekar

Roll No: TETB19

Sub: Soft Computitng

Batch: B2

Experiment 5 : To implement both the k-means algorithm and the Hierarchical Agglomerative Clustering (HAC) algorithm

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

```
from google.colab import drive
drive.mount('/content/drive')
df1 = pd.read_csv('/content/drive/MyDrive/ML/shopping-data.csv')
```

```
Mounted at /content/drive
```

Implementation of hierarchial clustering

```
df1.shape
```

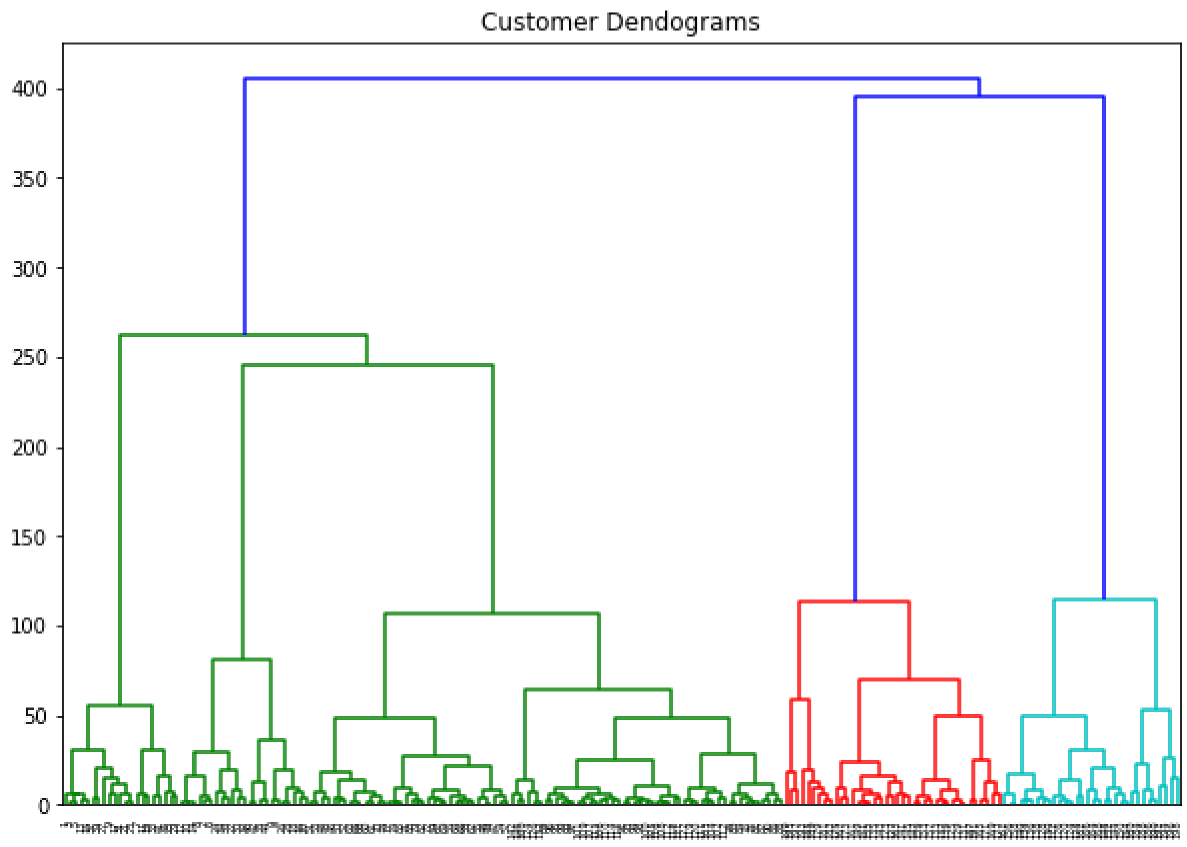
```
(200, 5)
```

```
df1.head()
```

	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

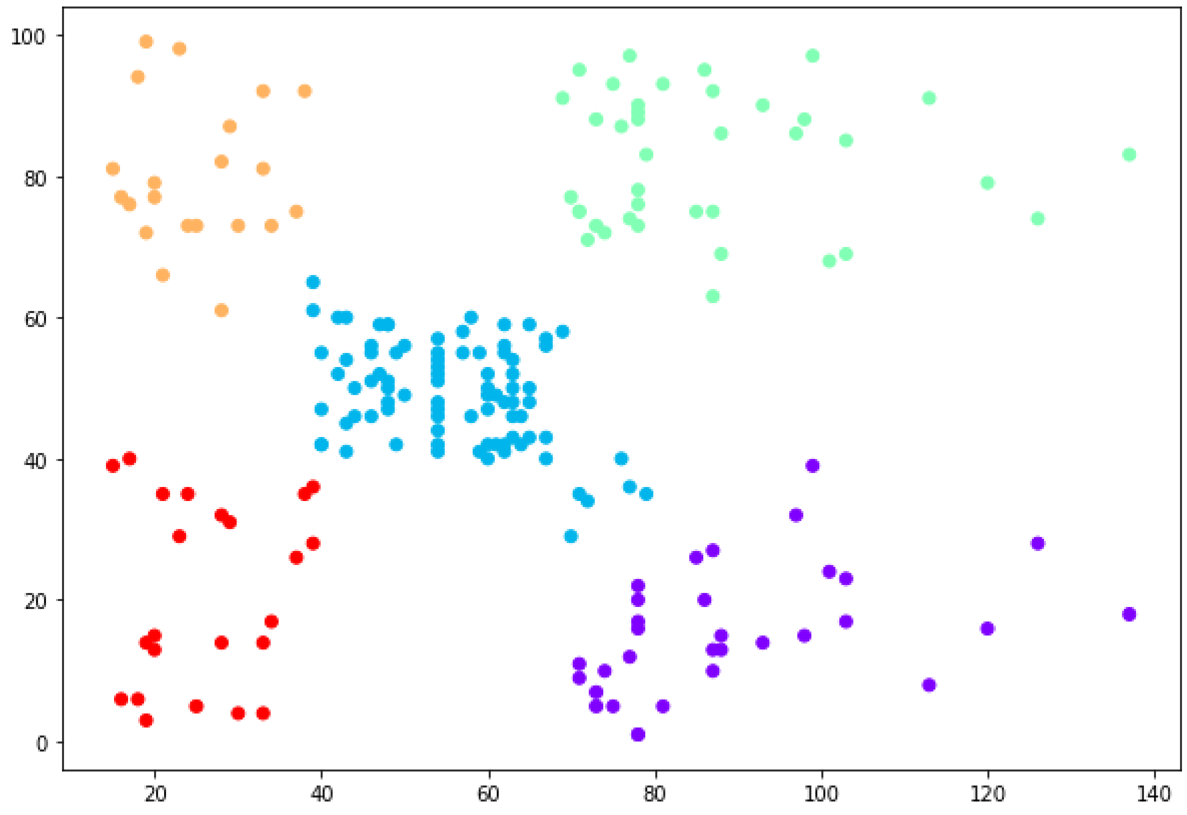
```
data = df1.iloc[:, 3:5].values
```

```
import scipy.cluster.hierarchy as shc

plt.figure(figsize=(10, 7))
plt.title("Customer Dendograms")
dend = shc.dendrogram(shc.linkage(data, method='ward'))
```



```
from sklearn.cluster import AgglomerativeClustering

cluster = AgglomerativeClustering(n_clusters=5, affinity='euclidean', linkage='ward')
cluster.fit_predict(data)

     array([4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3,
            4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 1,
            4, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
            1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
            1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
            1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2, 1, 2, 0, 2, 0, 2,
            1, 2, 0, 2, 0, 2, 0, 2, 0, 2, 1, 2, 0, 2, 1, 2, 0, 2, 0, 2, 0, 2,
            0, 2, 0, 2, 0, 2, 1, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2,
            0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2,
            0, 2])


plt.figure(figsize=(10, 7))
plt.scatter(data[:,0], data[:,1], c=cluster.labels_, cmap='rainbow')
```

```
<matplotlib.collections.PathCollection at 0x7f658ce67f90>
```



✓ 0s completed at 1:56 PM

Name : Kshitij V Darwhekar

Roll No: TETB19

Sub: Soft Computitng

Batch: B2

Experiment 6: Implementation of IOT Solution using Machine Learning

Importing the libraries

```
import sklearn
import numpy as np
import pandas as pd
```

Importing the dataset

```
from google.colab import drive
drive.mount('/content/drive/')
```

```
Drive already mounted at /content/drive/; to attempt to forcibly remount, call drive
```

```
dataset = pd.read_csv("/content/drive/MyDrive/ML/Crop_recommendation.csv")
```

```
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
```

```
dataset.head()
```

```
     N    P    K  temperature   humidity         ph    rainfall  label
```

Data Preprocessing

```
  2  60   55   44     23.004459   82.320763   7.840207   263.964248     rice
```

Taking care of missing data

```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
imputer.fit(X[:,:])
X[:, :] = imputer.transform(X[:, :])
```

Encoding categorical data

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)
```

Splitting the dataset into the Training set and Test set

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state =
```

```
print(X_train)
```

```
[[134.          56.          18.         ...  83.91902605   6.6912681
   70.97358303]
 [ 29.         122.         196.         ...  81.15595212   5.63832848
   73.06862952]
 [ 25.          68.          19.         ...  64.25510719   7.10845012
   67.47677295]
 ...
 [ 35.          64.          15.         ...  63.53604453   6.50014496
   69.5274407 ]
 [ 39.          65.          23.         ...  69.12613376   7.6859593
   41.02682925]
 [ 14.          22.           9.         ...  91.13772765   6.54319181
  112.5090516 ]]
```

```
print(y_train)
```

```
[ 6  7  2 ...  2 10 16]
```

```
print(X_test)
```

```
[[105.          14.          50.         ...  87.6883982    6.41905219
   59.65590798]
 [ 91.          12.          46.         ...  85.49938185   6.34394252
   48.31219031]
 [ 14.         121.         203.         ...  83.74765639   6.15868941
   74.46411148]
 ...
 [ 84.          27.          29.         ...  53.00366334   7.16709259
  168.2644287 ]
 [ 31.          13.          33.         ...  95.21224392   6.34246371
  148.3003692 ]
 [  5.          24.          40.         ...  93.87030088   6.29790758
  104.6735454 ]]
```

```
print(y_test)
```

```
[21 21  7  3  2 20 13  9 15  1 13  5 10 14 12  0  5 10  5 12  4  2  9  8
  6  5 10 16 13  9 19 20 11 15  4  6 12 12 21 13 11  2 18 21 18 14  9  9
  6 14 13  2  0 15 18  1 17 12 10  6 16 14 21 20 15  0  7  5  0 16  4 19
  9 11  7 13  3 11  8 12 20  2 21 21 15  6 11 10 13 17  2  8 14  7 14 11
  5  8 10  3 16  8 14  1  1 20 21  5 18 15 15 12  5  7 16 19 14 10 11  8
 19 10 16  3  3  2 19 16  3 17 13 13 15 14 11 14  4 19 16  2  2  7  0  5
  3  0  8 12 21 17 16  4 13  1 19  3 21  2  0  8 10 18  8  9  9 15 20 15
  1 16 18  0 13  4  6 14  9 19 17 16 20 17 17 18  9  1  4 18 20 17 11  8
 13 20 11  5 18  4  3 12  4 19 11 13 13 16 15 11 18  1  3  2 18 16 13 14
 12 17 15 19 20 20  2 17  2  5 11  5 16 20 13 14 16  9 19  4 12 14  6 20
  3 14  0 18  2 20 21  2 19 16 11  7  3 18  8 17 19  5 12 13  8 21 19 20
  7  4  8 10  3  5  5 17 19 11 20  3 18 16 19 18  4  9 19 15 13 12 10  1
  2 12  9 12  6 14 17  7  7 18 17  8 20  3 15  5 21 20  8 17  7 15  2 13
 13  3  2 12  1 12 19  8 16 15  3 10  6 17  7  9 10  0 20 15  0 17  2  8
  3 13 10  7  8  9 15 17  7 17 20  5 15 13  1 17 16  9 21 18  0 21 21 18
  9 13  9  8  4  6  9 16  6 18 19  6  6  0  6  0 16 11  7  1  0 13 20  9
  1 20 10  3 19  1  3 15 19  0 10 15 16  2 15 13 12  3 19 12  3  4 15  1
 18 17  8 10  6 20  1  4 20  2 11 16 21 20  0  7 18  7  3 12  8 19 11 12
  7  1 14 18  1  6  2  0  0  8  8 21  3  1 19  1  9  7 11  5 11  8  7  5
 14  2  8 16 18 18 15 13 21 14 21 17 14 14 14 19 16 13  0  5  4 11  4  7
  7  3  3 12  9 17 16 14 17 18  2 17 15  2  1 20  5  6  7  8  3 15  1  7
 21 15 18  8 18  6 21 19  5  4 11 20 14  9 21 14  0  0 21  1 13 14  0 14
  6 20 17  6 17  3  0 19 13 20  2 12 16  8  1 17  5  6 12  5  4 19]
```

Feature Scaling

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
print(X_train)
```

```
[[ 2.25367108  0.07555744 -0.59141091 ...  0.56115786  0.28639844
  -0.58838147]
 [-0.58434455  2.06834149  2.90385791 ...  0.43651791 -1.09903674
  -0.55053196]
 [-0.69245943  0.43788181 -0.57177457 ... -0.3258651   0.83531751
  -0.65155552]
 ...
 [-0.42217223  0.31710702 -0.65031993 ... -0.35830141  0.03492274
  -0.61450776]
 [-0.31405735  0.34730072 -0.4932292  ... -0.10613716  1.5951916
  -1.12940532]
 [-0.98977536 -0.95102828 -0.76813798 ...  0.88678747  0.09156286
   0.16200634]]
```

```
print(X_test)
```

```
[[ 1.46983819 -1.19257786  0.03695202 ...  0.73119109 -0.07177737
  -0.79284878]
 [ 1.09143611 -1.25296526 -0.04159334 ...  0.63244639 -0.17060505
  -0.99778658]
 [-0.98977536  2.03814779  3.0413123  ...  0.55342752 -0.41435709
  -0.52532091]
 ...
 [ 0.90223507 -0.80005979 -0.37541115 ... -0.83340833  0.91247799
   1.16929376]
 [-0.53028711 -1.22277156 -0.29686578 ...  1.07058549 -0.17255083
   0.8086192 ]
 [-1.23303384 -0.89064089 -0.15941139 ...  1.01005156 -0.23117683
   0.02044856]]
```

Random Forest

Training the Random Forest Classification model on the Training set

```
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_state
classifier.fit(X_train, y_train)
```

```
RandomForestClassifier(criterion='entropy', n_estimators=10, random_state=0)
```

Predicting the Test set results

```
y_pred_RF = classifier.predict(X_test)
print(np.concatenate((y_pred_RF.reshape(len(y_pred_RF),1), y_test.reshape(len(y_test),1)),
```

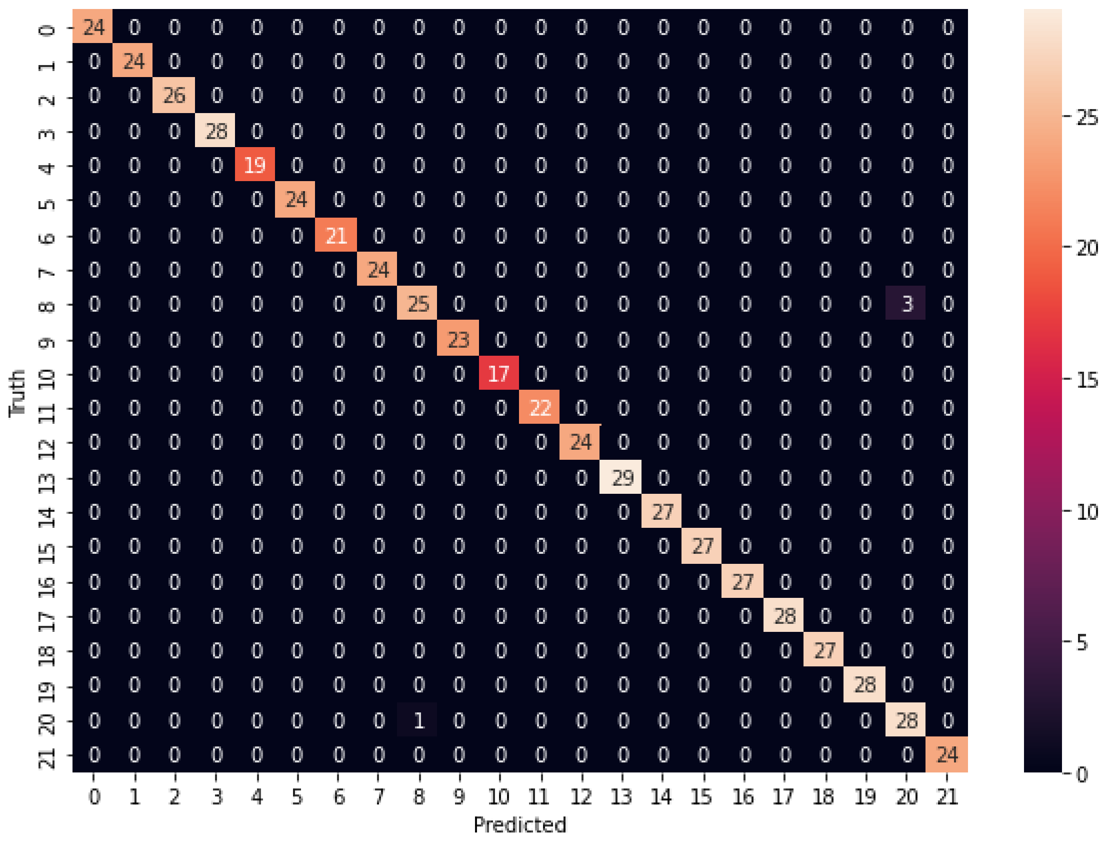
```
[[21 21]
 [21 21]
 [ 7  7]
 ...
 [ 5  5]
 [ 4  4]
 [19 19]]
```

Making the Confusion Matrix

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred_RF)
```

```
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sn
plt.figure(figsize=(10,7))
sn.heatmap(cm, annot=True)
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

```
Text(69.0, 0.5, 'Truth')
```



```
accuracy_score(y_test, y_pred_RF)
```

```
0.9927272727272727
```

Naive Bayes

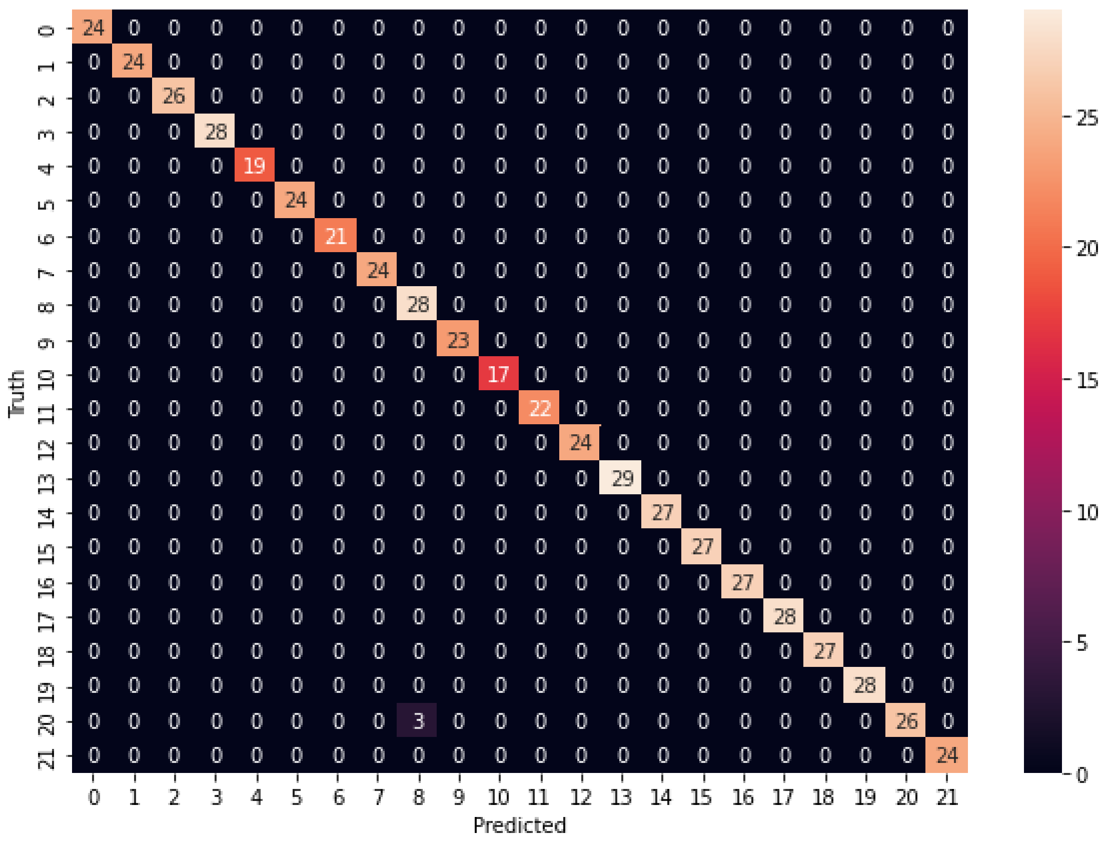
Training the Naive Bayes model on the Training set

```
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)

     GaussianNB()
```

```
y_pred_NV = classifier.predict(X_test)
print(np.concatenate((y_pred_NV.reshape(len(y_pred_NV),1), y_test.reshape(len(y_test),1)),

     [[21 21]
      [21 21]
      [ 7  7]
      ...
      [ 5  5]
      [ 4  4]
      [19 19]]
```

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred_NV)
```

```
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sn
plt.figure(figsize=(10,7))
sn.heatmap(cm, annot=True)
plt.xlabel('Predicted')
plt.ylabel('Truth')

     Text(69.0, 0.5, 'Truth')
```



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
0	24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	24	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	28	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	23	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	17	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	22	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	24	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	29	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	27	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	27	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	27	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	28	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	27	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	28	0	0
20	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	26	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	24

```
accuracy_score(y_test, y_pred_NV)

    0.9945454545454545
```

✓ 0s completed at 2:22 PM

Name : Kshitij V Darwhekar

Roll No: TETB19

Sub: Soft Computitng

Batch: B2

```
import tensorflow
```

```
from tensorflow import keras
```

```
import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
```

```
(X_train, y_train) , (X_test, y_test) = keras.datasets.mnist.load_data()
```

```
    Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mn
    11493376/11490434 [==============================] - 0s 0us/step
    11501568/11490434 [==============================] - 0s 0us/step
```

```
len(X_train)
```

```
    60000
```

```
len(X_test)
```

To undo cell deletion use Ctrl+M Z or the Undo option in the Edit menu

```
X_train[0].shape
```

```
    (28, 28)
```

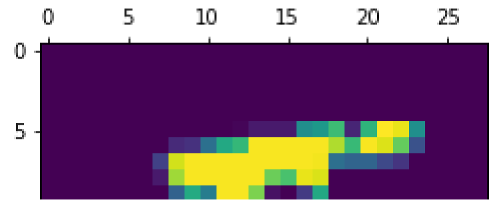
```
X_train[0]
```

```
              0,   0],
           [  0,   0,   0,   0,   0,   0,   0,   0,  80, 156, 107, 253, 253,
            205,  11,   0,  43, 154,   0,   0,   0,   0,   0,   0,   0,   0,
              0,   0],
           [  0,   0,   0,   0,   0,   0,   0,   0,   0,  14,   1, 154, 253,
             90,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
              0,   0],
           [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 139, 253,
            190,   2,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
              0,   0],
           [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  11, 190,
            253,  70,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
              0,   0],
           [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  35,
            241, 225, 160, 108,   1,   0,   0,   0,   0,   0,   0,   0,   0,
              0,   0],
           [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
             81, 240, 253, 253, 119,  25,   0,   0,   0,   0,   0,   0,   0,
              0,   0],
           [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
              0,  45, 186, 253, 253, 150,  27,   0,   0,   0,   0,   0,   0,
              0,   0],
           [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
              0,   0,  16,  93, 252, 253, 187,   0,   0,   0,   0,   0,   0,
              0,   0],
           [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
              0,   0,   0,   0, 249, 253, 249,  64,   0,   0,   0,   0,   0,
              0,   0],
           [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
              0,  46, 130, 183, 253, 253, 207,   2,   0,   0,   0,   0,   0,
              0,   0],
           [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  39,
            148, 229, 253, 253, 253, 250, 182,   0,   0,   0,   0,   0,   0,
              0,   0],
           [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  24, 114, 221,
            253, 253, 253, 253, 201,  78,   0,   0,   0,   0,   0,   0,   0,
              0,   0],
           [  0,   0,   0,   0,   0,   0,   0,   0,  23,  66, 213, 253, 253,
            253, 253, 198,  81,   2,   0,   0,   0,   0,   0,   0,   0,   0,
              0,   0],
           [  0,   0,   0,   0,   0,   0,  18, 171, 219, 253, 253, 253, 253,
            195,  80,   9,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
              0,   0],
           [  0,   0,   0,   0,  55, 172, 226, 253, 253, 253, 253, 244, 133,
             11,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
              0,   0],
           [  0,   0,   0,   0, 136, 253, 253, 253, 212, 135, 132,  16,   0,
              0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
              0,   0],
           [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                                                              0,   0,   0,
                                                              0,   0,   0,
              0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
              0,   0],
           [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
              0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
              0,   0]], dtype=uint8)
```

To undo cell deletion use Ctrl+M Z or the Undo option in the Edit menu ×

```
plt.matshow(X_train[0])
```

```
<matplotlib.image.AxesImage at 0x7efc976e77d0>
```



```
y_train[0]
```

```
5
```

```
# Scaling Technique

X_train = X_train / 255
X_test = X_test / 255
```

```
X_train[0]
```

```
         0.        , 0.        , 0.        , 0.        , 0.18039216,
         0.50980392, 0.71764706, 0.99215686, 0.99215686, 0.81176471,
         0.00784314, 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        ],
        [0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.15294118, 0.58039216, 0.89803922,
         0.99215686, 0.99215686, 0.99215686, 0.98039216, 0.71372549,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        ],
        [0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.09411765, 0.44705882, 0.86666667, 0.99215686, 0.99215686,
         0.99215686, 0.99215686, 0.78823529, 0.30588235, 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        ],
        [0.        , 0.        , 0.        , 0.        , 0.        ,
                                                       25882353,
                                                       99215686,
                                                     , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        ],
        [0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.07058824, 0.67058824, 0.85882353, 0.99215686,
         0.99215686, 0.99215686, 0.99215686, 0.76470588, 0.31372549,
         0.03529412, 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        ],
        [0.        , 0.        , 0.        , 0.        , 0.21568627,
         0.6745098 , 0.88627451, 0.99215686, 0.99215686, 0.99215686,
         0.99215686, 0.95686275, 0.52156863, 0.04313725, 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        ],
        [0.        , 0.        , 0.        , 0.        , 0.53333333,
         0.99215686, 0.99215686, 0.99215686, 0.83137255, 0.52941176,
         0.51764706, 0.0627451 , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
```

To undo cell deletion use Ctrl+M Z or the Undo option in the Edit menu

```
          0.        , 0.        , 0.        ],
         [0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        ],
         [0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        ],
         [0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        ]])
```

```
X_train_flattened = X_train.reshape(len(X_train), 28*28)
X_test_flattened = X_test.reshape(len(X_test), 28*28)
```

```
X_train_flattened.shape
```

```
    (60000, 784)
```

```
X_test_flattened.shape
```

```
    (10000, 784)
```

```
X_train_flattened[0]
```

```
          0.97647059, 0.25098039, 0.        , 0.        , 0.        ,
```

To undo cell deletion use Ctrl+M Z or the Undo option in the Edit menu

```
          0.        , 0.        , 0.        , 0.18039216, 0.50980392,
          0.71764706, 0.99215686, 0.99215686, 0.81176471, 0.00784314,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.15294118,
          0.58039216, 0.89803922, 0.99215686, 0.99215686, 0.99215686,
          0.98039216, 0.71372549, 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.09411765, 0.44705882, 0.86666667, 0.99215686, 0.99215686,
          0.99215686, 0.99215686, 0.78823529, 0.30588235, 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.09019608, 0.25882353, 0.83529412, 0.99215686,
          0.99215686, 0.99215686, 0.99215686, 0.77647059, 0.31764706,
          0.00784314, 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.07058824, 0.67058824, 0.85882353,
          0.99215686, 0.99215686, 0.99215686, 0.99215686, 0.76470588,
          0.31372549, 0.03529412, 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.21568627, 0.6745098 ,
          0.88627451, 0.99215686, 0.99215686, 0.99215686, 0.99215686,
          0.95686275, 0.52156863, 0.04313725, 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.53333333, 0.99215686, 0.99215686, 0.99215686,
          0.83137255, 0.52941176, 0.51764706, 0.0627451 , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        ])
```

```
# 10 Output neuron and 784 in input neuron
model = keras.Sequential([
                                                              oid')
```

To undo cell deletion use Ctrl+M Z or the Undo option in the Edit menu

```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(X_train_flattened, y_train, epochs=5)
```

```
    Epoch 1/5
    1875/1875 [==============================] - 3s 1ms/step - loss: 0.4724 - accuracy:
    Epoch 2/5
    1875/1875 [==============================] - 3s 2ms/step - loss: 0.3034 - accuracy:
    Epoch 3/5
    1875/1875 [==============================] - 4s 2ms/step - loss: 0.2833 - accuracy:
    Epoch 4/5
    1875/1875 [==============================] - 5s 3ms/step - loss: 0.2731 - accuracy:
    Epoch 5/5
    1875/1875 [==============================] - 5s 3ms/step - loss: 0.2664 - accuracy:
    <keras.callbacks.History at 0x7efc93077f50>
```

```
model.evaluate(X_test_flattened, y_test)
```

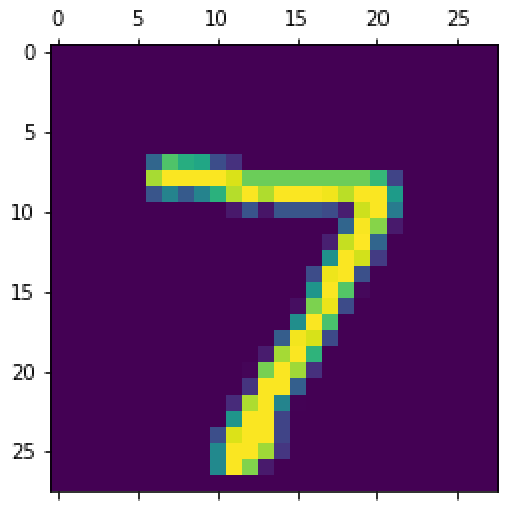
```
313/313 [==============================] - 1s 2ms/step - loss: 0.2695 - accuracy: 0.9
[0.2695145010948181, 0.9254000186920166]
```

```
y_predicted = model.predict(X_test_flattened)
y_predicted[0]
```

```
array([1.5027165e-02, 3.9325224e-07, 6.3410342e-02, 9.5975685e-01,
       3.2548308e-03, 1.0176152e-01, 1.0720740e-06, 9.9978119e-01,
       7.0441395e-02, 6.0749489e-01], dtype=float32)
```

```
plt.matshow(X_test[0])
```

```
<matplotlib.image.AxesImage at 0x7efc920031d0>
```



```
# np.argmax finds a maximum element from an array and returns the index of it
```

To undo cell deletion use Ctrl+M Z or the Undo option in the Edit menu

```
7
```

```
y_predicted_labels = [np.argmax(i) for i in y_predicted]
```

```
y_predicted_labels[:5]
```

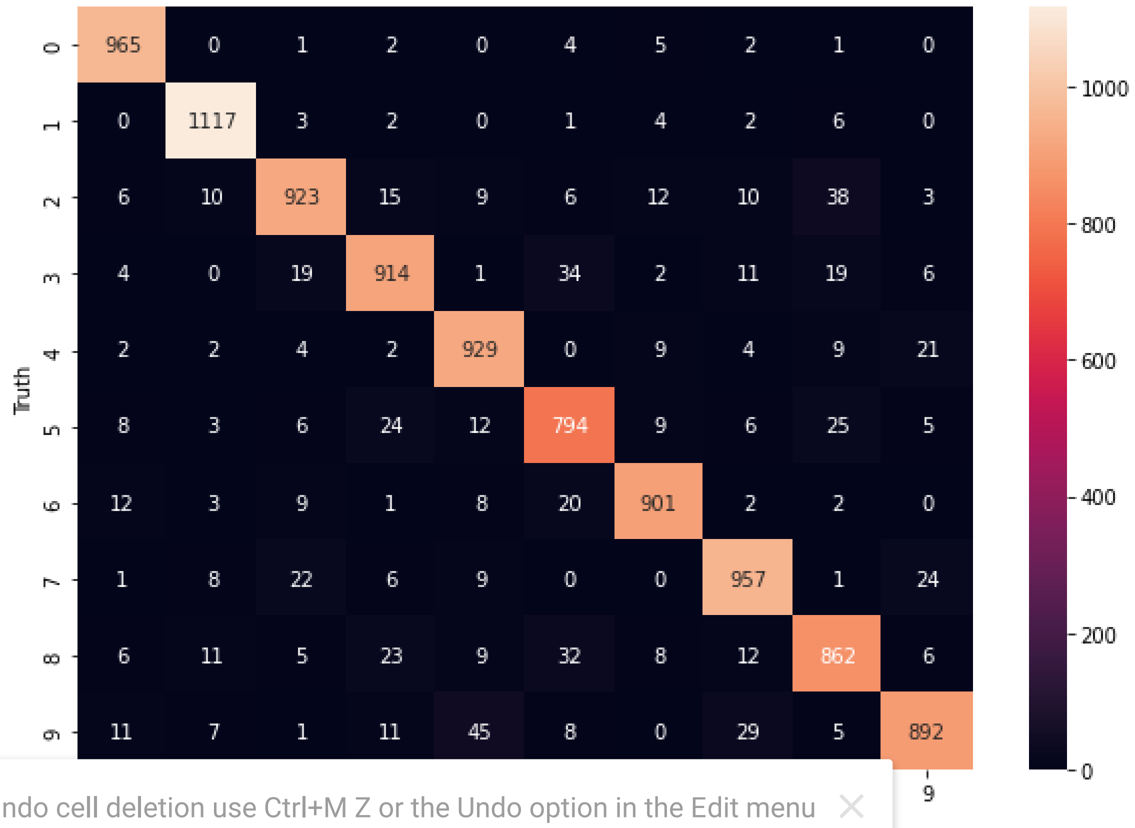
```
[7, 2, 1, 0, 4]
```

```
cm = tf.math.confusion_matrix(labels=y_test,predictions=y_predicted_labels)
cm
```

```
<tf.Tensor: shape=(10, 10), dtype=int32, numpy=
array([[ 965,    0,    1,    2,    0,    4,    5,    2,    1,    0],
       [   0, 1117,    3,    2,    0,    1,    4,    2,    6,    0],
       [   6,   10,  923,   15,    9,    6,   12,   10,   38,    3],
       [   4,    0,   19,  914,    1,   34,    2,   11,   19,    6],
       [   2,    2,    4,    2,  929,    0,    9,    4,    9,   21],
       [   8,    3,    6,   24,   12,  794,    9,    6,   25,    5],
       [  12,    3,    9,    1,    8,   20,  901,    2,    2,    0],
       [   1,    8,   22,    6,    9,    0,    0,  957,    1,   24],
       [   6,   11,    5,   23,    9,   32,    8,   12,  862,    6],
       [  11,    7,    1,   11,   45,    8,    0,   29,    5,  892]],
      dtype=int32)>
```

```
import seaborn as sn
plt.figure(figsize = (10,7))
sn.heatmap(cm, annot=True, fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

```
    Text(69.0, 0.5, 'Truth')
```



0	965	0	1	2	0	4	5	2	1	0
1	0	1117	3	2	0	1	4	2	6	0
2	6	10	923	15	9	6	12	10	38	3
3	4	0	19	914	1	34	2	11	19	6
4	2	2	4	2	929	0	9	4	9	21
5	8	3	6	24	12	794	9	6	25	5
6	12	3	9	1	8	20	901	2	2	0
7	1	8	22	6	9	0	0	957	1	24
8	6	11	5	23	9	32	8	12	862	6
9	11	7	1	11	45	8	0	29	5	892

To undo cell deletion use Ctrl+M Z or the Undo option in the Edit menu

Using hidden layer

```
model = keras.Sequential([
    keras.layers.Dense(100, input_shape=(784,), activation='relu'),
    keras.layers.Dense(10, activation='sigmoid')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(X_train_flattened, y_train, epochs=5)
```

```
    Epoch 1/5
    1875/1875 [==============================] - 3s 2ms/step - loss: 0.2726 - accuracy: 
```

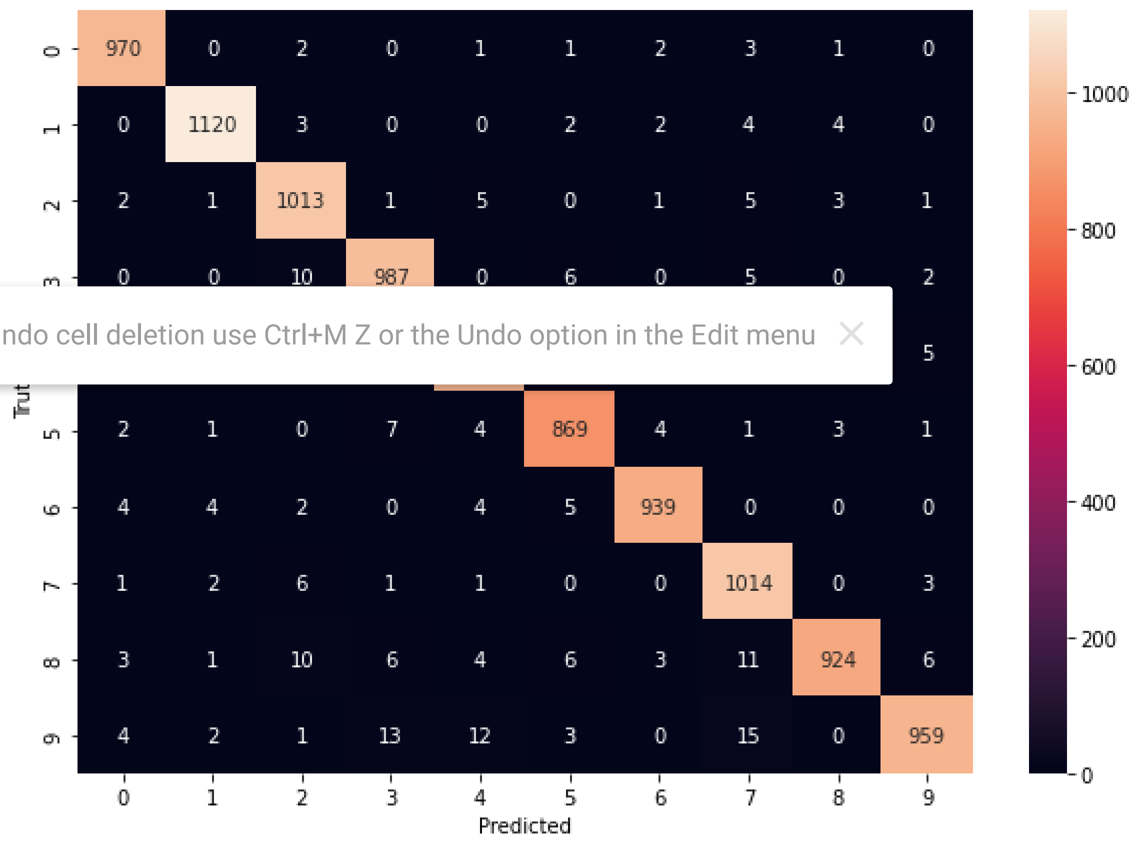
```
Epoch 2/5
1875/1875 [==============================] - 3s 2ms/step - loss: 0.1230 - accuracy:
Epoch 3/5
1875/1875 [==============================] - 3s 2ms/step - loss: 0.0852 - accuracy:
Epoch 4/5
1875/1875 [==============================] - 3s 2ms/step - loss: 0.0660 - accuracy:
Epoch 5/5
1875/1875 [==============================] - 3s 1ms/step - loss: 0.0521 - accuracy:
<keras.callbacks.History at 0x7efc8ed501d0>
```

```
model.evaluate(X_test_flattened,y_test)
```

```
313/313 [==============================] - 0s 1ms/step - loss: 0.0786 - accuracy: 0.
[0.07863084971904755, 0.9763000011444092]
```

```
y_predicted = model.predict(X_test_flattened)
y_predicted_labels = [np.argmax(i) for i in y_predicted]
cm = tf.math.confusion_matrix(labels=y_test,predictions=y_predicted_labels)

plt.figure(figsize = (10,7))
sn.heatmap(cm, annot=True, fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

```
Text(69.0, 0.5, 'Truth')
```



To undo cell deletion use Ctrl+M Z or the Undo option in the Edit menu

Using Flatten layer so that we don't have to call .reshape on input dataset

```
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(100, activation='relu'),
    keras.layers.Dense(10, activation='sigmoid')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(X_train, y_train, epochs=10)
```

```
Epoch 1/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.2693 - accuracy: 0
Epoch 2/10
1875/1875 [==============================] - 3s 1ms/step - loss: 0.1207 - accuracy: 0
Epoch 3/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.0839 - accuracy: 0
Epoch 4/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.0657 - accuracy: 0
Epoch 5/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.0490 - accuracy: 0
Epoch 6/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.0408 - accuracy: 0
Epoch 7/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.0332 - accuracy: 0
Epoch 8/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.0281 - accuracy: 0
Epoch 9/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.0225 - accuracy: 0
Epoch 10/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.0196 - accuracy: 0
<keras.callbacks.History at 0x7efc92f47c90>
```

To undo cell deletion use Ctrl+M Z or the Undo option in the Edit menu

```
313/313 [==============================] - 0s 1ms/step - loss: 0.0913 - accuracy: 0.9
[0.0912703275680542, 0.9740999937057495]
```

✓ 0s completed at 4:17 PM

To undo cell deletion use Ctrl+M Z or the Undo option in the Edit menu

Name : Kshitij V Darwhekar

Roll No: TETB19

Sub: Soft Computitng

Batch: B2

Experiment 8 : Open CV for Computer Vision

In [ ]:
```
import cv2
```

In [ ]:
```
scale_percent = 20 # percent of original size
width = int(img.shape[1] * scale_percent / 100)
height = int(img.shape[0] * scale_percent / 100)
dim = (width,height)
```

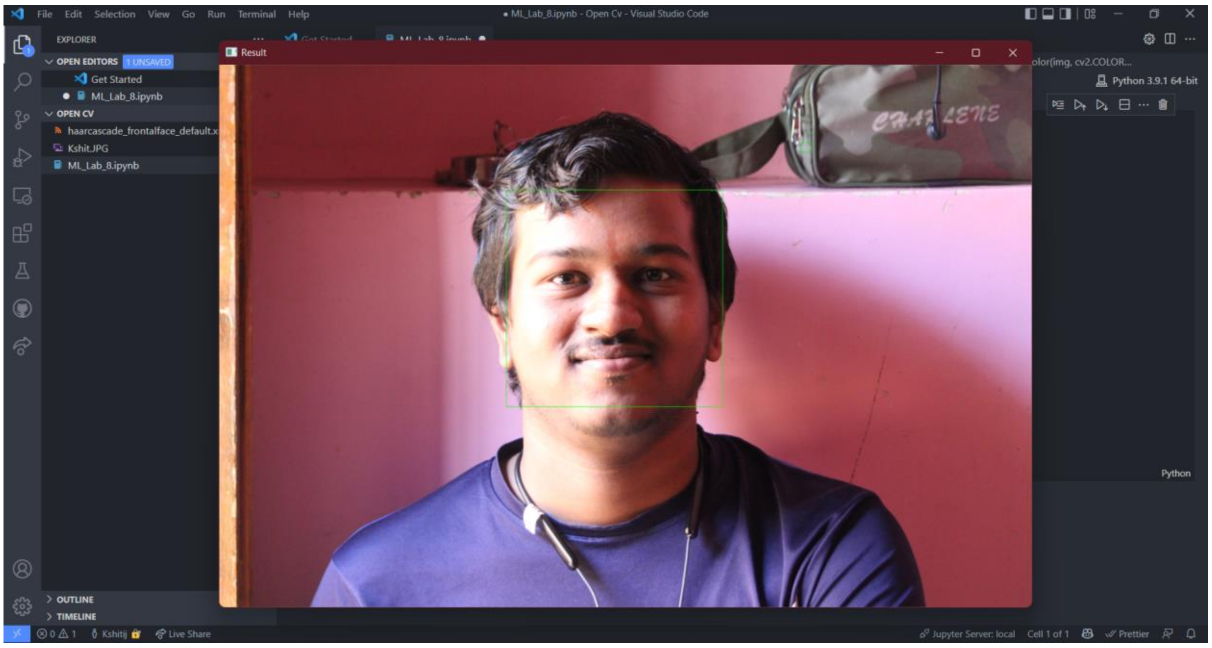
In [ ]:
```
faceHar = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

img = cv2.imread('./Kshit.JPG')

imgGray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

faces = faceHar.detectMultiScale(imgGray, 1.1, 4)
```

In [ ]:
```
for (x, y ,w, h) in faces:
    cv2.rectangle(img, (x, y), (x+h, y+w), (0, 255, 0), 2)

resized = cv2.resize(img, dim, interpolation = cv2.INTER_AREA)
```

In [ ]:
```
cv2.imshow("Result", resized)

cv2.waitKey(0)
```

Out[ ]:
```
-1
```

Output :


File Edit Selection View Go Run Terminal Help
● ML_Lab_8.ipynb - Open Cv - Visual Studio Code
EXPLORER
OPEN EDITORS 1 UNSAVED
Get Started
ML_Lab_8.ipynb
OPEN CV
haarcascade_frontalface_default.x
Kshit.JPG
ML_Lab_8.ipynb
Result
olor(img, cv2.COLOR...
Python 3.9.1 64-bit
Python
OUTLINE
TIMELINE
Kshitij
Live Share
Jupyter Server: local
Cell 1 of 1
Prettier