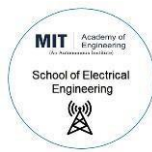


**Lab Manual**  
**Soft Computing**  
**(ET363)**  
**TY BTECH**

**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.

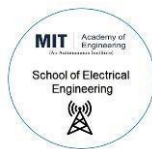
### **Program Educational Objectives (PEOs)**

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.

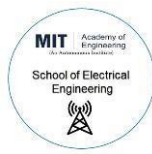
PO 9 Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

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.

---

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

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 tuning.	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 2022-23

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.

### Steps:

- 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

### Conclusion

### Preprocessing

```
[8] 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	

```
[9] database = database.dropna()
database.head()
```

	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	MALE
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	FEMALE
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	FEMALE
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	FEMALE
5	Adelie	Torgersen	39.3	20.6	190.0	3650.0	MALE

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

	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	sex
336	Gentoo	Biscoe	44.5	15.7	217.0	4875.0	.

```
[13] 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()
```

/usr/local/lib/python3.7/dist-packages/pandas/core/generic.py:6392: SettingWithCopyWarning:  
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/indexing.html#returning-a-view-versus-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)  
return self.\_update\_inplace(result)

	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	MALE
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	FEMALE
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	FEMALE
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	FEMALE
5	Adelie	Torgersen	39.3	20.6	190.0	3650.0	MALE

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

+ Code + Text

```
[15] database.head()
```

	new_species	new_island	new_culmen_length_mm	new_culmen_depth_mm	new_flipper_length	new_body_mass_g	new_sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	MALE
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	FEMALE
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	FEMALE
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	FEMALE
5	Adelie	Torgersen	39.3	20.6	190.0	3650.0	MALE

```
[16] database_new = database.drop(['new_island', 'new_culmen_length_mm', 'new_flipper_length'], axis = 1)
database_new.head()
```

	new_species	new_island	new_culmen_length_mm	new_culmen_depth_mm	new_flipper_length	new_body_mass_g	new_sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	MALE
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	FEMALE
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	FEMALE
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	FEMALE
5	Adelie	Torgersen	39.3	20.6	190.0	3650.0	MALE

```
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

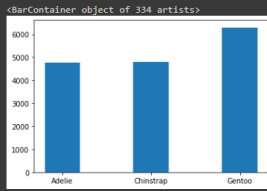
```
[18] database_new["islands"] = "Torgersen"
database_new.head()
```

	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



```
[19] import matplotlib.pyplot as plt  
import seaborn as sns
```

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



## Assessment Rubrics

Performance Area	Rating = 3	Rating = 2	Rating = 1	Rating= 0	Score
<b>Presentation of Markdown Cell</b>	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	
<b>Data Set</b>	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	
<b>ML Model Implementation</b>	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	
<b>Timely Submission</b>					1 Mark
<b>Total</b>					10

## 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

$$Y = \beta_0 + \beta_1 X \dots\dots\dots 1$$

#### 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.

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \dots\dots + \beta_n X_n \dots\dots\dots 2$$

### **Advantages of Linear Regression:**

1) Linear Regression is simple to implement. 2) Less 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

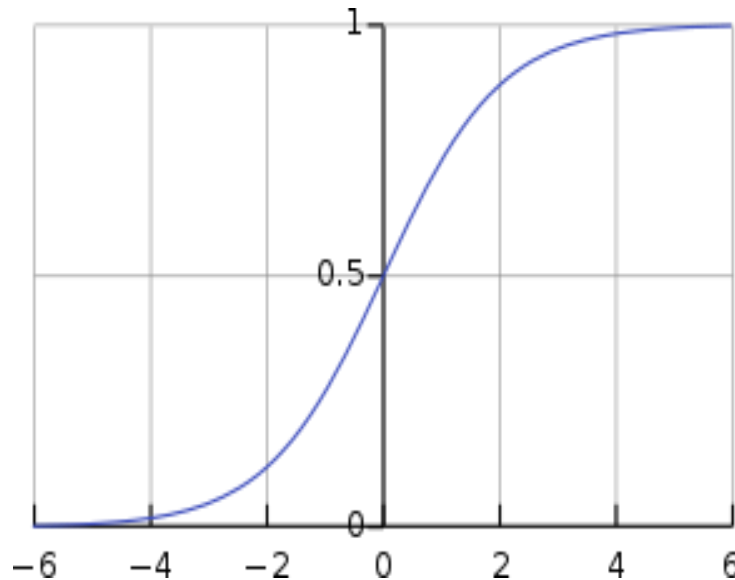
Want  $0 \leq h_{\theta}(x) \leq 1$

$$h_{\theta}(x) = g(\theta^T x)$$

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

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

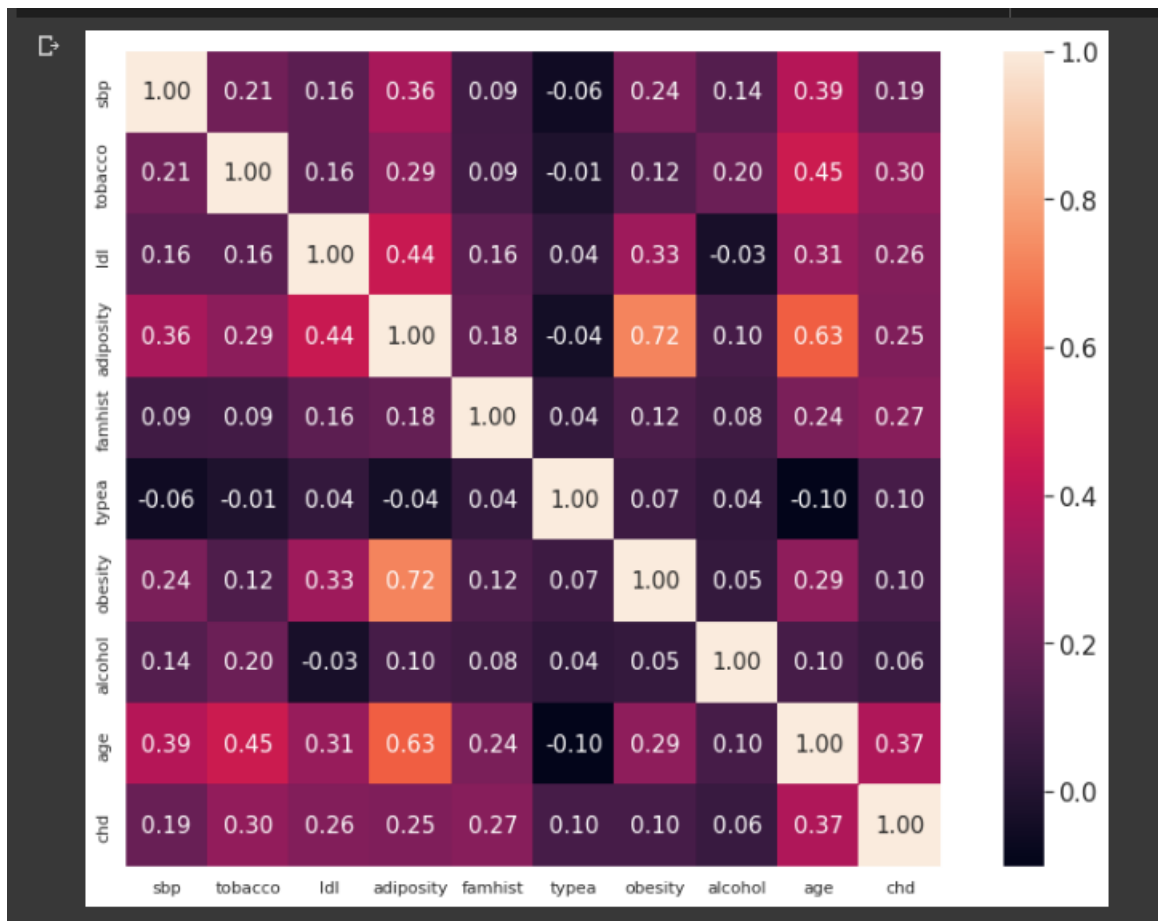
**Sigmoid function**



**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**



```
[10] # 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: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818: ConvergenceWarning: lbfgs failed to converge (status=1):
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,

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

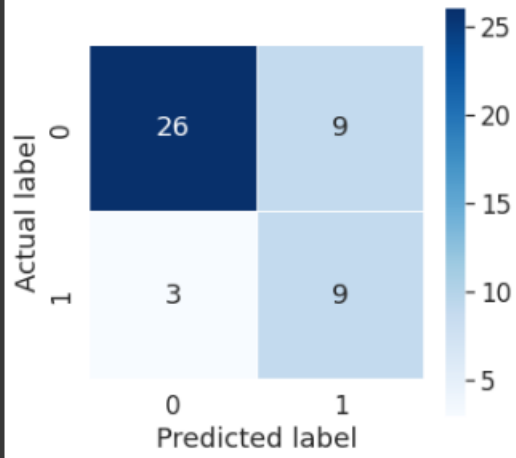
print ('Test Accuracy: %.2f' % model.score(X_test,y_test))

Training Accuracy: 0.74
Test Accuracy: 0.74
```

```
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')
```





## Assessment Rubrics

Performance Area	Rating = 3	Rating = 2	Rating = 1	Rating= 0	Score
<b>Presentation of Markdown Cell</b>	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	
<b>Data Set</b>	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	
<b>ML Model Implementation</b>	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	
<b>Timely Submission</b>					1 Mark
<b>Total</b>					10

Expt. No. 3

Date: \_\_\_\_\_

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

---

### 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**

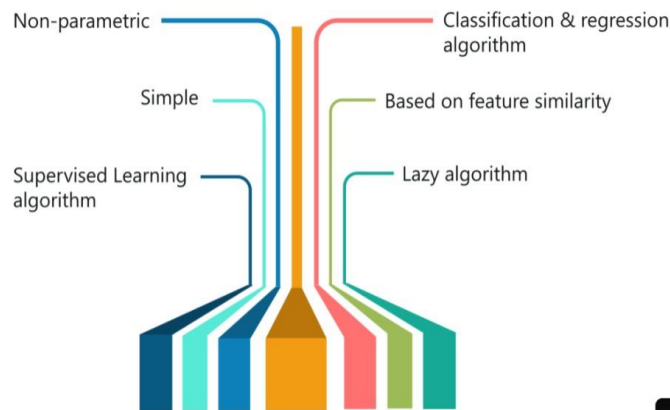
Random forests is an ensemble learning algorithm. The basic premise of the algorithm is that building a small decision-tree with few features is a computationally cheap process. If we can build many small, weak decision trees in parallel, we can then combine the trees to form a single, strong learner 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.

**Steps for Random Forest algorithm implementation:**

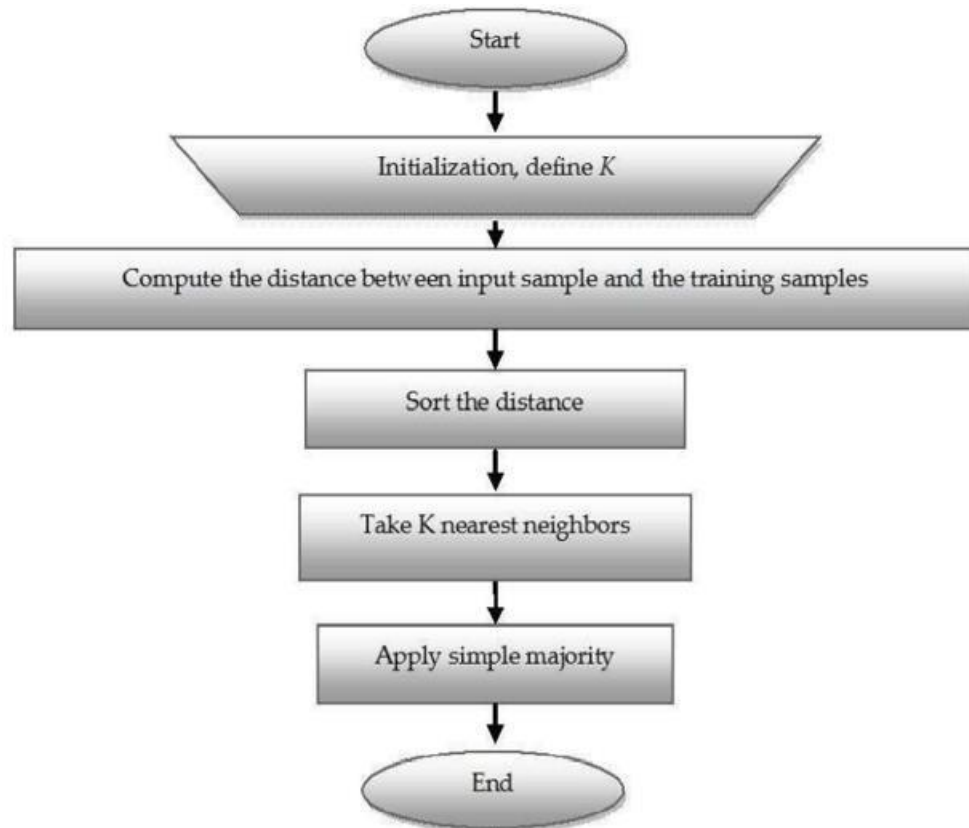
- Selection of appropriate data set as per the ML algorithm
- Data preprocessing
- ML Model Implementation with the applicable software
- Train the model with given dataset and test the model by training and test split
- Performance analysis of implemented ML model
- To improve the model performance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

**C. KNN**

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.



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.



**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. Naïve Bayes

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.

#### Applications

##### 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).

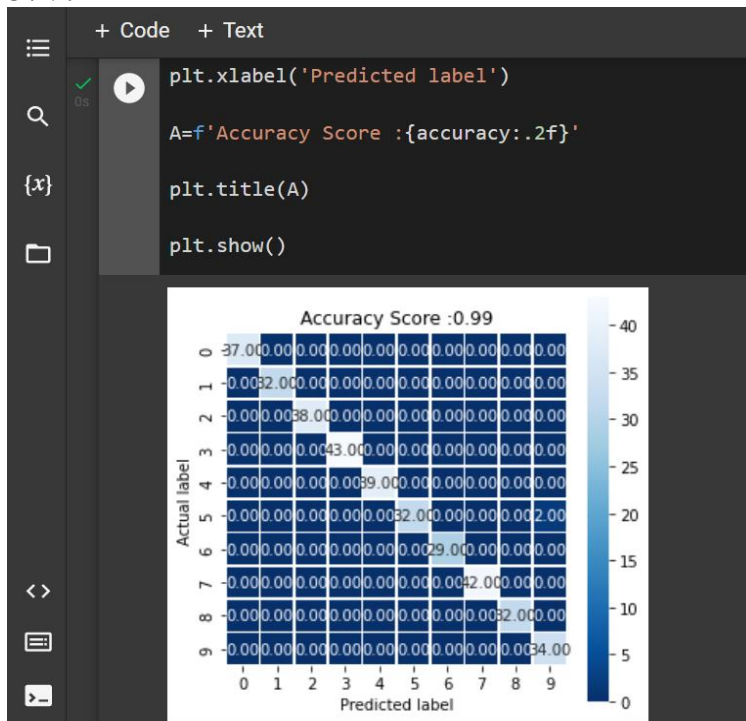
P(H) is called the prior probability, while P(H|E) is called the posterior probability. The factor that relates the two, P(H|E) / P(E), is called the likelihood ratio.

### Steps for algorithm implementation:

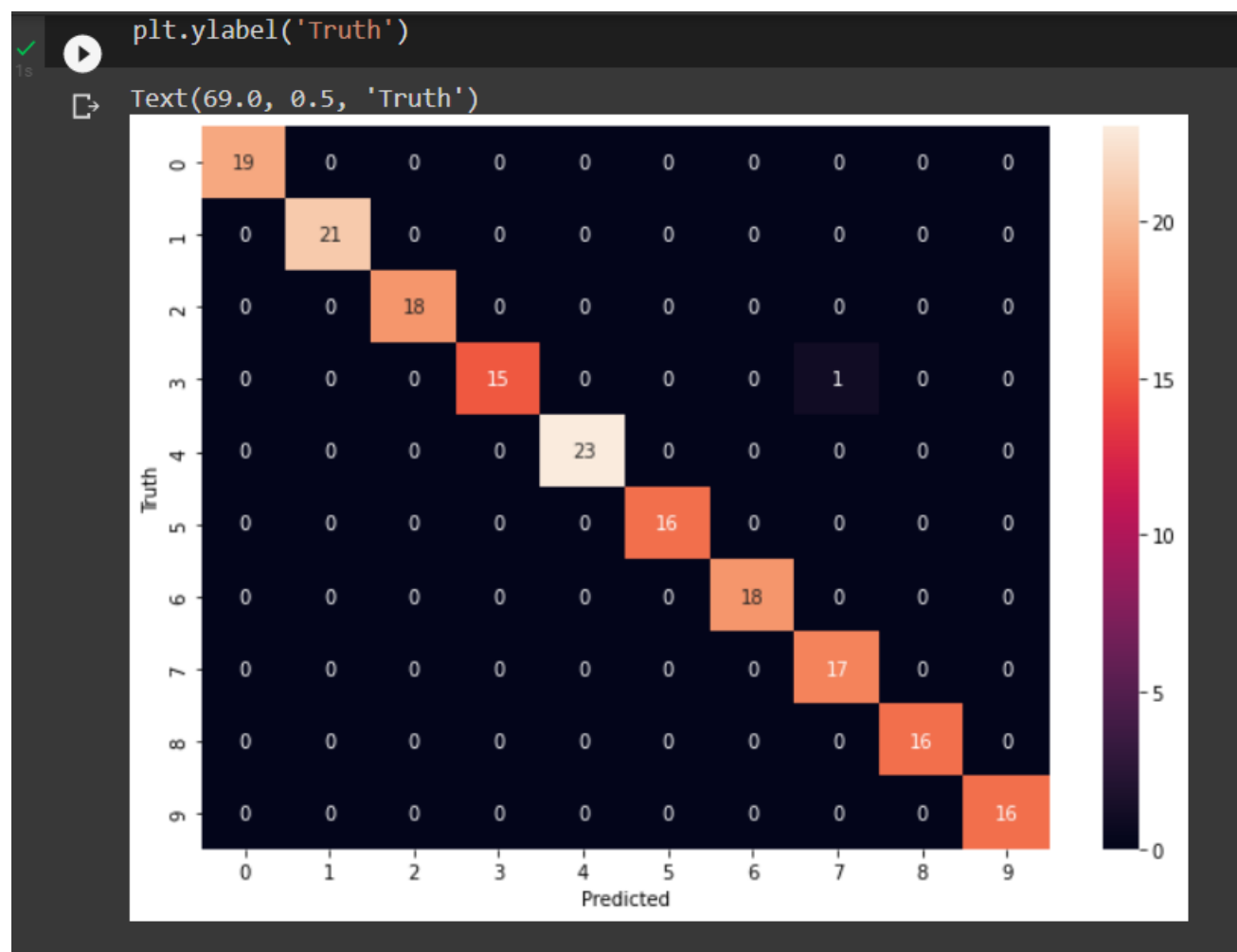
- Selection of appropriate data set as per the ML algorithm
- Data preprocessing
- ML Model Implementation with the applicable software
- Train the model with given dataset and test the model by training and test split
- Performance analysis of implemented ML model
- To improve the model performance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

## Conclusion

**SVM:**



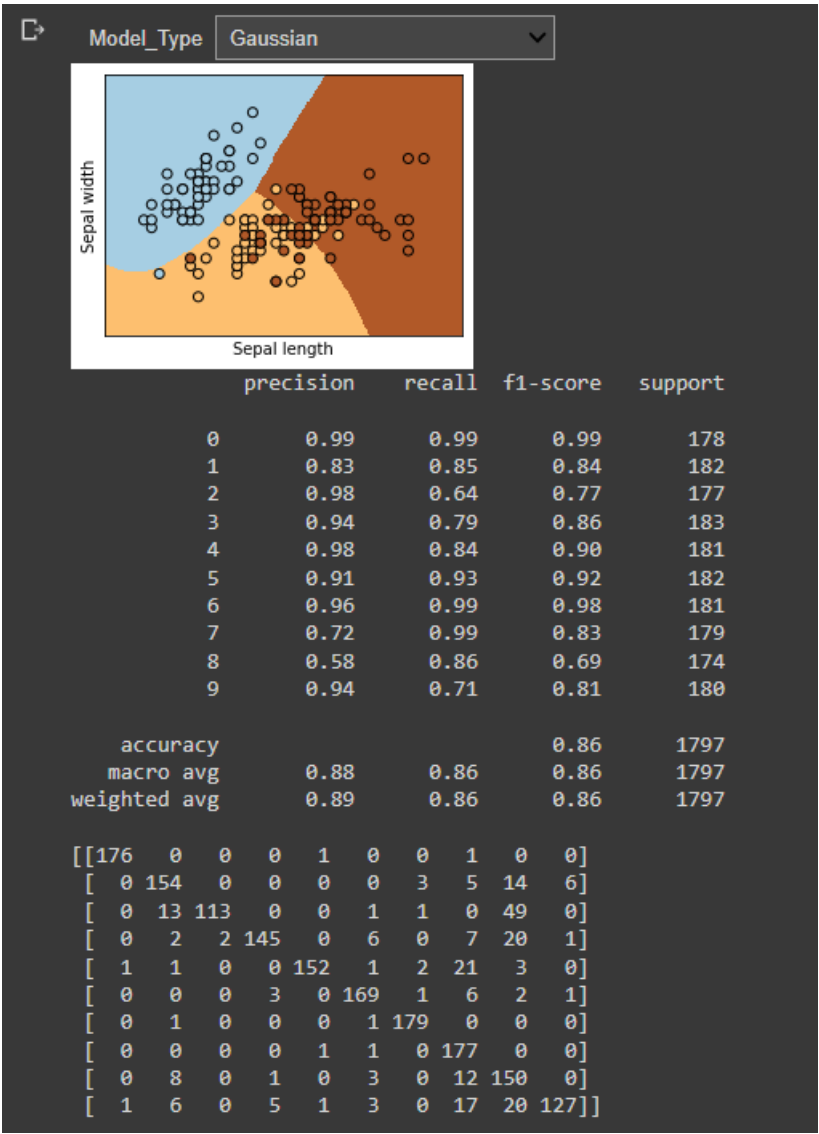
Random Forest :



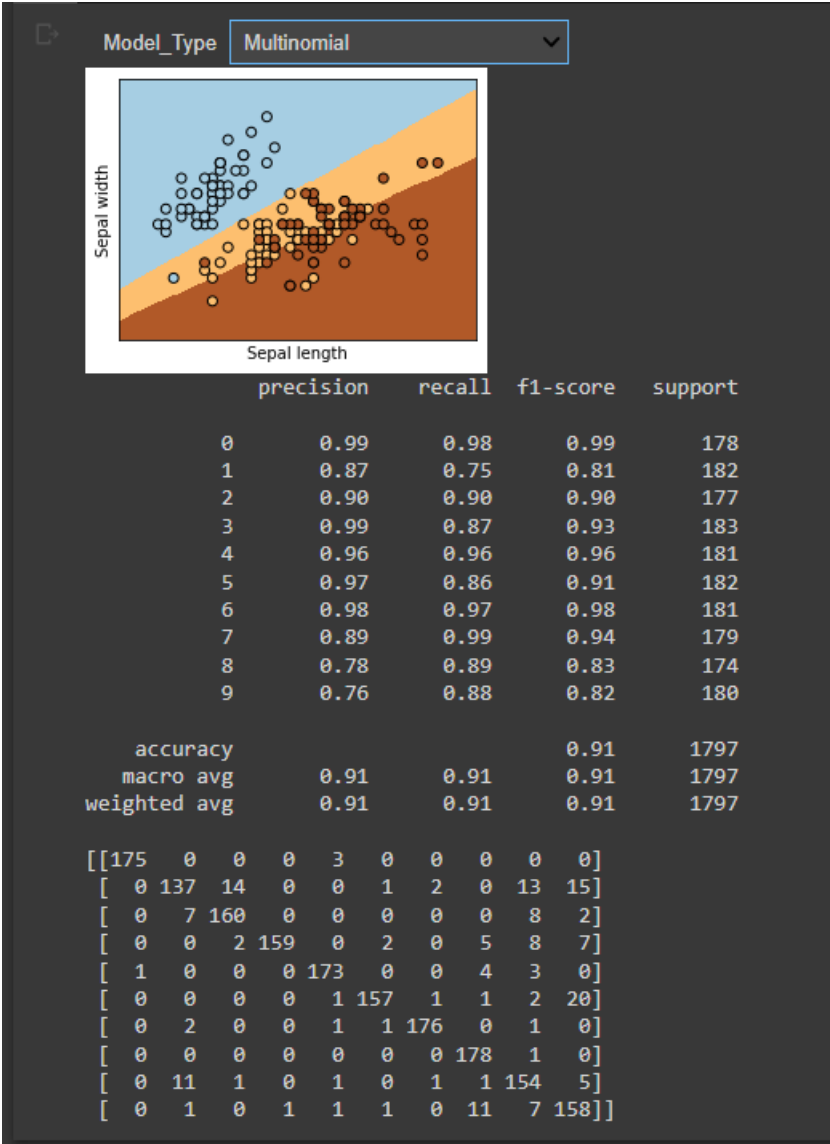
Naïve Bayes:

1. Gaussian

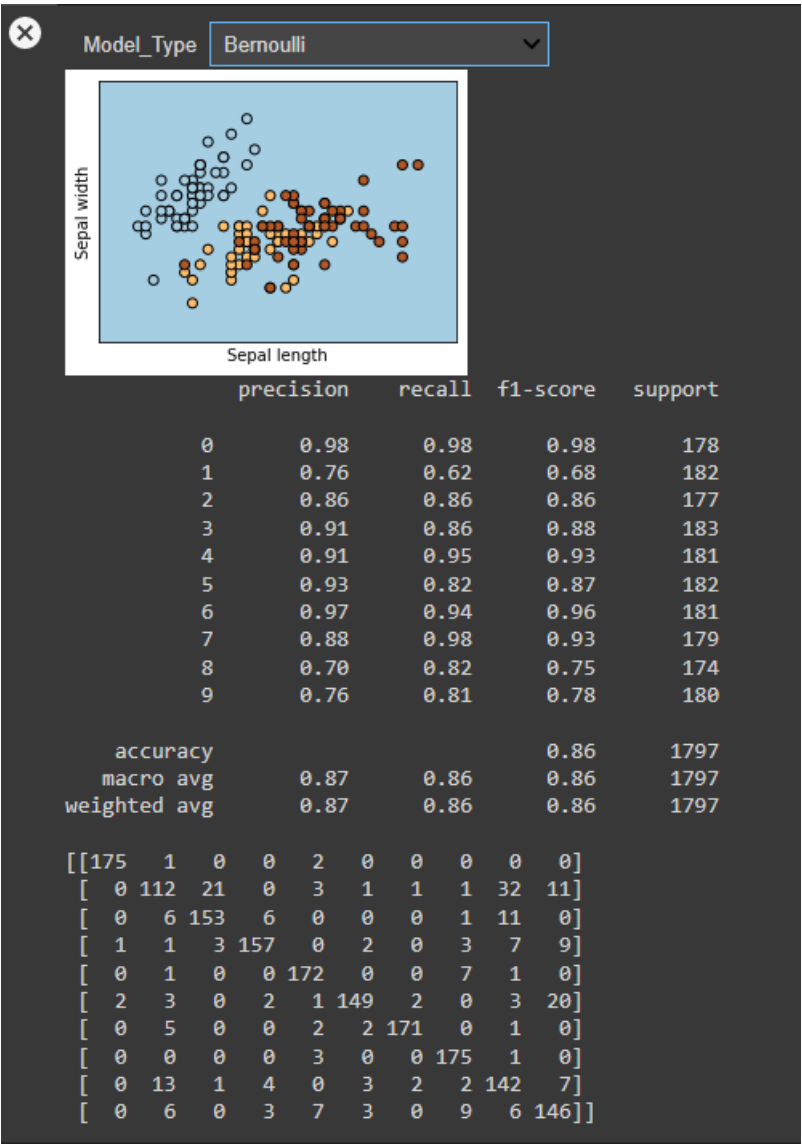




2. Multinomial



3. Bernoulli



## Assessment Rubrics

Performance Area	Rating = 3	Rating = 2	Rating = 1	Rating= 0	Score
<b>Presentation of Markdown Cell</b>	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	
<b>Data Set</b>	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	
<b>ML Model Implementation</b>	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	
<b>Timely Submission</b>					1 Mark
<b>Total</b>					10

## **Machine Learning for Image Classification**

---

### **Objectives:**

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

### **Software Requirement:**

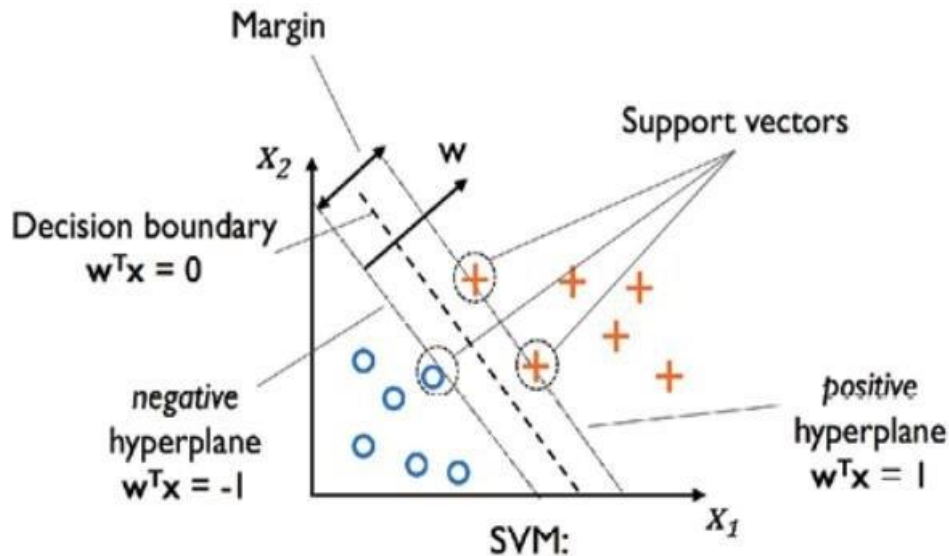
- Python jupyter notebook

### **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

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

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

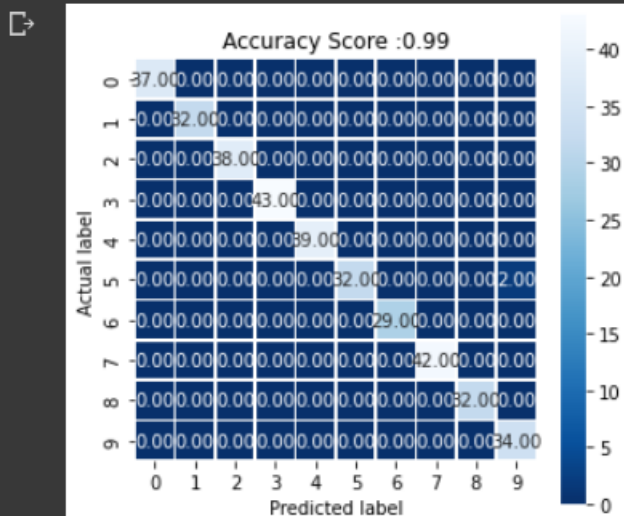
plt.ylabel('Actual label')

plt.xlabel('Predicted label')

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

plt.title(A)

plt.show()
```





## Assessment Rubrics

Performance Area	Rating = 3	Rating = 2	Rating = 1	Rating= 0	Score
<b>Presentation of Markdown Cell</b>	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	
<b>Data Set</b>	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	
<b>ML Model Implementation</b>	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	
<b>Timely Submission</b>					1 Mark
<b>Total</b>					10

## Implementation of Unsupervised Machine Learning

---

### Objectives:

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

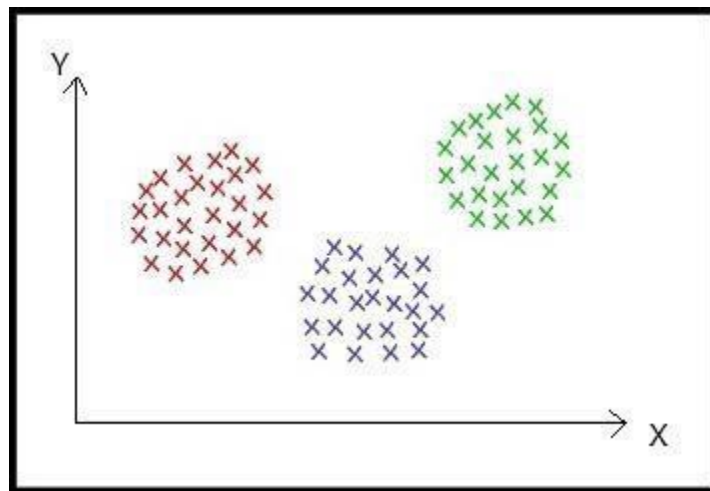
### Software Requirement:

- 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.

**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
<b>Presentation of Markdown Cell</b>	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	
<b>Data Set</b>	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	
<b>ML Model Implementation</b>	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	
<b>Timely Submission</b>					1 Mark
<b>Total</b>					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
<b>Defining the Application</b>	Student states the application clearly and identifies underlying issues.	Student adequately defines the application.	Student fails to define the application adequately.	
<b>Machine Learning model implementation after analyzing the application with proper dataset</b>	Data Set Visualization, Analysis and implementation is done properly.	Only Data Set Visualization or Analysis or Preprocessing is done.	Only Data Set Visualization done.	
<b>Presentation Quality</b>	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.	
<b>Timely Submission</b>				1 Mark
<b>Total</b>				10

## **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:

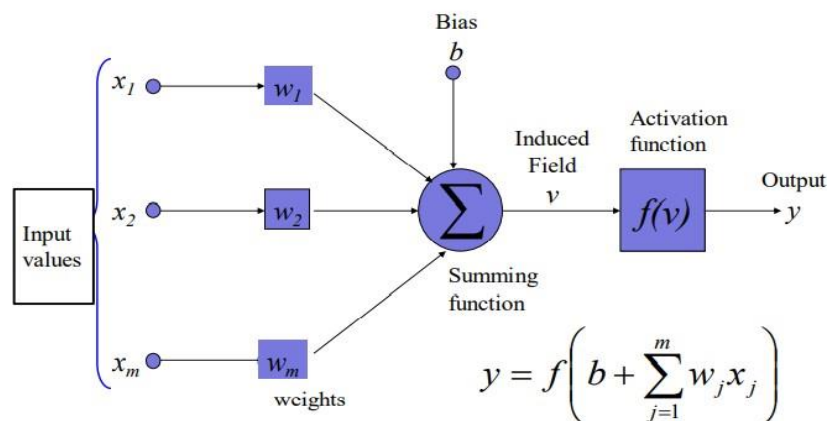
1. Neuron Model: The processing unit of the ANN which performs a linear combination of inputs.
2. Architecture: Just like the set of neurons in brain. The neurons are connected by links which have weight.
3. Learning Algorithm: Modifies the weight of links to model a specific task. The training relies 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  $j^{\text{th}}$  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
<b>Presentation of Markdown Cell</b>	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	
<b>Data Set</b>	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	
<b>ML Model Implementation</b>	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	
<b>Timely Submission</b>					1 Mark
<b>Total</b>					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
<b>Presentation of Markdown Cell</b>	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	
<b>Data Set</b>	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	
<b>ML Model Implementation</b>	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	
<b>Timely Submission</b>					1 Mark
<b>Total</b>					10