Software Lab II
(MIDS & ANN)

LABORATORY MANUAL

Third Year (SEM - II)

(2024-2025)

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

GENBA SOPANRAO MOZE COLLEGE OF ENGINEERING,
Balewadi, Pune.

Software Lab II (Machine Intelligence for Data Science & ANN)

Course Code	Course Name	Teaching Scheme (Hrs./ Week)	Credits
318556	Software Laboratory II	4	1

Course Objectives:

1. Students will demonstrate proficiency with statistical analysis of data.
2. Students will execute statistical analyses with professional statistical software.
3. Students will apply data science concepts and methods to solve problems.

Course Outcomes:

On completion of the course, students will be able to–

CO1: Demonstrate proficiency with statistical analysis of data.

CO2: Use statistical analyses with professional statistical software.

CO3: Apply data science concepts and methods to solve problems.

Table of Contents

Group A

Lab Assignment No.	1
Title	Data Preprocessing
Roll No.	
Class	TE
Date of Completion	
Subject	Software lab 2
Assessor's Sign	

ASSIGNMENT No: 01

Title : Data Preprocessing

Aim : To perform Data Preprocessing Techniques.

Objective : To understand different data Preprocessing techniques in python.

Problem Statement: Access an open-source dataset "Titanic". Apply pre-processing techniques on the raw dataset.

Theory :
Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.

When creating a machine learning project, it is not always a case that we come across clean and formatted data and while doing any operation with data, it is mandatory to clean it and put it in a formatted way. So, for this, we use data preprocessing tasks.

Why do we need Data Preprocessing?

Real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data preprocessing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.

It involves below steps:

- **Getting the dataset**
- **Importing libraries**
- **Importing datasets**
- **Finding Missing Data**
- **Encoding Categorical Data**
- **Splitting dataset into training and test set**
- **Feature scaling**

Code:

1. Importing the libraries

```
import pandas as pd
```

2. Importing the dataset

```
df = pd.read_csv("train.csv")
```

3. Understanding the dataset

```
Df
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 12 columns

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

4. Filling the rows with missing values with the mean of the rows.

```
df['Age'] = df['Age'].fillna(df['Age'].mean())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          891 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

5. Dropping the columns with many missing data values and at the end rows with a few missing values.

```
df=df.drop(['Cabin'], axis=1)
            a()
df=df.dropna
Int64Index: 8
Data columns
 #   Column
---  ------
 0   Passenge
 1   Survived
 2   Pclass
 3   Name
 4   Sex
 5   Age
 6   SibSp
 7   Parch
 8   Ticket
 9   Fare
 10  Embarked
dtypes: float
memory usage:
```

Conclusion: Thus, we saw how data preprocessing is carried out.

Frequently asked questions :

Q. No	**Questions**	BT	CO

1		1	1
2		2	1
3		2	1
4		2	1
5		2	1

Guidelines for Students

The experiments should be completed and get checked by the concerned teacher in the lab on or before the date of submission. After which the experiment will not be signed. Every experiment must be included in the file in following format.

a) **Aim**: In this section write complete objective of the program you are going to make in the lab. This section specifies the complete description of the including problem analysis, input description, method used, fundamental concept and desired output format.
b) **Theory**: Write brief theory related to practical.
c) **Algorithm**: Write Algorithm for given task.
d) **Input**: Write input test data/ or program that are used to test program objective to see whether program is achieving the given objective or not.
e) **Output**: describe the results in few lines
f) **Conclusion**: Write complete conclusion whether what the student has learned from this experiment.
g) **Source Code**: Submit in the form of soft copies.

- **Marking criteria.**

  Experiment completion (Timely)
  Lab file (neatness and regularity)
  Viva (from time to time)
  Mock Practical Exam
  Exam (end term): Practical + Viva

- **Assessment Methodology**

  Timely completion of assignment- 2marks
  Program demonstration- 4 marks
  Viva-voce -2 marks
  Timely submission of journal- 2 marks

Lab Assignment No.	2

Title	Document Polarity Analysis
Roll No.	
Class	TE
Date of Completion	
Subject	Software lab 2
Assessor's Sign	

ASSIGNMENT No: 02

Title : Document Polarity analysis

Aim : To carry out document polarity analysis.

Objective : To understand how document polarity analysis is carried out.

Problem Statement: Write a program to recognize a document is positive or negative based on polarity words using suitable classification method.

Theory :
Polarity analysis, also known as sentiment analysis, is a natural language processing (NLP) task that involves determining the sentiment or polarity of a given document, such as a text message, customer review, or social media post. Classification techniques are commonly used for sentiment analysis, as they can effectively categorize text into positive, negative, or neutral sentiment classes. One popular classification technique for polarity analysis is the Support Vector Machine (SVM) algorithm. Here's an outline of how you can perform polarity analysis using SVM:

1. Data Preparation:

- Collect a dataset of labeled documents with their corresponding sentiment labels (positive, negative, neutral).

- Preprocess the text data by removing stopwords, punctuation, and converting text to lowercase.

- Tokenize the text by splitting it into individual words or n-grams (contiguous sequences of n words).

- Convert the tokenized text into numerical features using techniques like bag-of-words or TF-IDF (Term Frequency-Inverse Document Frequency).

2. Data Split:

- Split the labeled dataset into training and testing sets. The typical split is around 70-80% for training and 20-30% for testing.

- Randomize the data to ensure that the distribution of sentiment labels is balanced between the training and testing sets.

3. Feature Extraction:

- Apply the chosen feature extraction technique (e.g., bag-of-words or TF-IDF) to the training data to convert text documents into numerical feature vectors.

- Transform the testing data using the same feature extraction technique applied to the training data.

4. Training:

- Train an SVM classifier on the training data, using the extracted features and their corresponding sentiment labels.

- SVM models aim to find a hyperplane that maximally separates different sentiment classes.

5. Prediction:

- Use the trained SVM classifier to predict the sentiment of the test documents by inputting their corresponding feature vectors.

- The classifier assigns each test document to one of the sentiment classes based on the learned decision boundaries.

6. Evaluation:

- Compare the predicted sentiment labels with the ground truth labels of the test set to evaluate the performance of the classifier.

- Common evaluation metrics for sentiment analysis include accuracy, precision, recall, and F1-score.

7. Fine-tuning:

- If the performance of the classifier is not satisfactory, you can experiment with different variations:

  - Try different feature extraction techniques or adjust their parameters.

  - Explore other classification algorithms or ensemble methods.

  - Perform hyperparameter tuning for the SVM model (e.g., kernel type, regularization parameter).

8. Deployment:

- Once you are satisfied with the performance, you can deploy the trained SVM classifier to analyze sentiment in new, unseen documents.

Code:

1. Importing required libraries

```
import nltk
import random
nltk.download('movie_reviews')
```

2. Using the movie_reviews dataset from NLTK corpus

```
from nltk.corpus import movie_reviews
```

3. Loading the dataset

```
documents = [(list(movie_reviews.words(fileid)), category)
              for category in movie_reviews.categories()
              for fileid in movie_reviews.fileids(category)]
random.shuffle(documents)
```

4. Extracting features

```
all_words = nltk.FreqDist(w.lower() for w in movie_reviews.words())
word_features = list(all_words)[:2000]

def document_features(document):
    document_words = set(document)
    features = {}
    for word in word_features:
        features['contains({})'.format(word)] = (word in document_words)
    return features
```

5. Training Naive Bayes Classifier

```
featuresets = [(document_features(d), c) for (d,c) in documents]
train_set, test_set = featuresets[100:], featuresets[:100]
classifier = nltk.NaiveBayesClassifier.train(train_set)
```

6. Testing the classifier

```
print('Accuracy of Naive Bayes Classifier is',
      f'{nltk.classify.accuracy(classifier, test_set) * 100}%')
Accuracy of Naive Bayes Classifier is 92.0%
```

7. Listing most informative features

```
classifier.show_most_informative_features(5)
```

```
Most Informative Features
   contains(outstanding) = True              pos : neg    =     10.7 : 1.0
         contains(mulan) = True              pos : neg    =      9.1 : 1.0
        contains(wasted) = True              neg : pos    =      6.7 : 1.0
        contains(seagal) = True              neg : pos    =      6.6 : 1.0
   contains(wonderfully) = True              pos : neg    =      6.4 : 1.0
```

Frequently asked questions :

Q. No	**Questions**	BT	CO
1		1	1
2		2	1
3		2	1
4		2	1
5		2	1

Guidelines for Students

The experiments should be completed and get checked by the concerned teacher in the lab on or before the date of submission. After which the experiment will not be signed. Every experiment must be included in the file in following format.

h) **Aim**: In this section write complete objective of the program you are going to make in the lab. This section specifies the complete description of the including problem analysis, input description, method used, fundamental concept and desired output format.

i) **Theory**: Write brief theory related to practical.

j) **Algorithm**: Write Algorithm for given task.

k) **Input**: Write input test data/ or program that are used to test program objective to see whether program is achieving the given objective or not.

l) **Output**: describe the results in few lines

m) **Conclusion**: Write complete conclusion whether what the student has learned from this experiment.

n) **Source Code**: Submit in the form of soft copies.

- **Marking criteria.**

  Experiment completion (Timely)
  Lab file (neatness and regularity)
  Viva (from time to time)
  Mock Practical Exam
  Exam (end term): Practical + Viva

- **Assessment Methodology**

  Timely completion of assignment- 2marks
  Program demonstration- 4 marks
  Viva-voce -2 marks
  Timely submission of journal- 2 marks

Lab Assignment No.	3

Title	Abalone Dataset Prediction
Roll No.	
Class	TE
Date of Completion	
Subject	Software lab 2
Assessor's Sign	

ASSIGNMENT No: 03

Title : Abalone Dataset Prediction

Aim : To analyze the abalone dataset and to predict various parameters.

Objective : To understand how classification and regression works.

Problem Statement: Download Abalone dataset. (URL: http://archive.ics.uci.edu/ml/datasets/Abalone)
a) Predict the number of rings either as a continuous value or as a classification problem.
b) Predict the age of abalone from physical measurements using linear regression

Theory :
Regression and classification are two fundamental tasks in machine learning that involve different types of predictive modeling.

Regression:
Regression is a supervised learning task that focuses on predicting a continuous numerical value or a numeric output variable. In regression, the goal is to find a relationship between input variables (also called independent variables or features) and the output variable. The model learns from labeled training data, where each training example consists of a set of input variables and the corresponding target output. The model then tries to approximate the relationship between the inputs and the continuous output, allowing it to make predictions on new, unseen data.

Common regression algorithms include linear regression, polynomial regression, support vector regression (SVR), decision tree regression, random forest regression, and neural networks. The output of a regression model can be interpreted as a numeric value or a range of values, depending on the problem at hand. Regression is used in various applications such as predicting housing prices, stock market analysis, weather forecasting, and demand prediction.

Classification:
Classification, like regression, is a supervised learning task. However, the objective of classification is different. Classification is concerned with predicting a discrete class or category for each input instance. In other words, it assigns input variables to predefined classes or labels based on their features. The model learns from labeled training data, where each training example has a set of input variables and the corresponding class or label.

Classification algorithms aim to build a decision boundary or a model that can separate different classes in the input space. Some popular classification algorithms include logistic regression, support vector machines (SVM), decision trees, random forests, naive Bayes, and neural networks (specifically, with softmax activation in the output layer). The output of a classification model is a categorical variable, indicating the predicted class or label for a given input instance. Classification is widely used in various domains, such as email spam detection, sentiment analysis, image recognition, and medical diagnosis.

In summary, regression predicts a continuous numeric value, while classification predicts discrete class labels. Both tasks are important in machine learning and have numerous applications in different fields.

Abalone dataset description

Name / Data Type / Measurement Unit / Description

Sex / nominal / -- / M, F, and I (infant)

Length / continuous / mm / Longest shell measurement

Diameter / continuous / mm / perpendicular to length

Height / continuous / mm / with meat in shell

Whole weight / continuous / grams / whole abalone

Shucked weight / continuous / grams / weight of meat

Viscera weight / continuous / grams / gut weight (after bleeding)

Shell weight / continuous / grams / after being dried

Rings / integer / -- / +1.5 gives the age in years

Code:

1. Load the abalone dataset

```
from google.colab import drive
drive.mount('/content/drive')
import pandas as pd
abalone=pd.read_csv('/content/drive/My Drive/TE/abalone.csv')
```

2. Describing the dataset

```
abalone.head()
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7

```
abalone.shape
(4177, 9)

abalone.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4177 entries, 0 to 4176
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Sex             4177 non-null   object
 1   Length          4177 non-null   float64
 2   Diameter        4177 non-null   float64
 3   Height          4177 non-null   float64
 4   Whole weight    4177 non-null   float64
 5   Shucked weight  4177 non-null   float64
 6   Viscera weight  4177 non-null   float64
 7   Shell weight    4177 non-null   float64
 8   Rings           4177 non-null   int64
dtypes: float64(7), int64(1), object(1)
memory usage: 293.8+ KB
```

```
abalone.describe()
```

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
count	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000
mean	0.523992	0.407881	0.139516	0.828742	0.359367	0.180594	0.238831	9.933684
std	0.120093	0.099240	0.041827	0.490389	0.221963	0.109614	0.139203	3.224169
min	0.075000	0.055000	0.000000	0.002000	0.001000	0.000500	0.001500	1.000000
25%	0.450000	0.350000	0.115000	0.441500	0.186000	0.093500	0.130000	8.000000
50%	0.545000	0.425000	0.140000	0.799500	0.336000	0.171000	0.234000	9.000000
75%	0.615000	0.480000	0.165000	1.153000	0.502000	0.253000	0.329000	11.000000
max	0.815000	0.650000	1.130000	2.825500	1.488000	0.760000	1.005000	29.000000

3. Using label encoder, encode the “Sex” column, and split the dataset into train and test.

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
abalone['Sex'] = le.fit_transform(abalone['Sex'])
y = abalone.iloc[:, 8].values
x = abalone.iloc[:, range(0,8)].values
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25,random_state=0)
```

4. Perform Classification to classify the rings attribute.

```
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
```

```
classifier.fit(x_train, y_train)
y_pred= classifier.predict(x_test)
cm = confusion_matrix(y_test, y_pred)
print(cm)
print("Accuracy of classification(in %):", accuracy score(y_test, y_pred) * 100)
```

```
[[ 7  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 2  8  3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 4 13 17  4  2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 1  5 17 27  8  2  3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0 16 33 41 13  6  3  2  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  1  4 10 43 40 28  7  6  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  1  6 18 31 46 33 17  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  4  3 16 13 30 34 39  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  2  2 13 12 21 20 51  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  7 13 16 26 28  1  0  0  0  0  0  0  0  1  0  0  1  0  0  0]
 [ 0  0  0  0  5  6 11  8 20  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0]
 [ 0  0  0  0  1  9  8  6  5  1  0  0  0  0  0  0  0  1  0  0  0  0  1  0]
 [ 0  0  0  0  0  3  4  8  7  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  2  1  4  5  4  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  2  2  6  0  0  0  0  0  0  0  0  0  0  0  0  2  0  0]
 [ 0  0  0  0  0  1  0  2  3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  1  4  0  5  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  2  0  4  2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0]
 [ 0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  1  0  0  1  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  1  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  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]]
Accuracy of classification(in %): 26.02870813397129
```

5. To find the correlation between the attributes and the ring.

```
abalone.corr()
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
Sex	1.000000	-0.036066	-0.038874	-0.042077	-0.021391	-0.001373	-0.032067	-0.034854	-0.034627
Length	-0.036066	1.000000	0.986812	0.827554	0.925261	0.897914	0.903018	0.897706	0.556720
Diameter	-0.038874	0.986812	1.000000	0.833684	0.925452	0.893162	0.899724	0.905330	0.574660
Height	-0.042077	0.827554	0.833684	1.000000	0.819221	0.774972	0.798319	0.817338	0.557467
Whole weight	-0.021391	0.925261	0.925452	0.819221	1.000000	0.969405	0.966375	0.955355	0.540390
Shucked weight	-0.001373	0.897914	0.893162	0.774972	0.969405	1.000000	0.931961	0.882617	0.420884
Viscera weight	-0.032067	0.903018	0.899724	0.798319	0.966375	0.931961	1.000000	0.907656	0.503819
Shell weight	-0.034854	0.897706	0.905330	0.817338	0.955355	0.882617	0.907656	1.000000	0.627574
Rings	-0.034627	0.556720	0.574660	0.557467	0.540390	0.420884	0.503819	0.627574	1.000000

6. Using linear regression to predict the number of rings

```
from sklearn.linear_model import LinearRegression
regressor= LinearRegression()
regressor.fit(x_train, y_train)
y_pred= regressor.predict(x_test)
print('Train Score: ', regressor.score(x_train, y_train))
```

```
print('Test Score: ', regressor.score(x_test, y_test))
import sklearn.metrics as sm
print("Mean absolute error =", round(sm.mean_absolute_error(y_test, y_pred), 2))
print("Mean squared error =", round(sm.mean_squared_error(y_test, y_pred), 2))
print("Median absolute error =", round(sm.median_absolute_error(y_test, y_pred), 2))
print("Explain variance score =", round(sm.explained_variance_score(y_test, y_pred), 2))
print("R2 score =", round(sm.r2_score(y_test, y_pred), 2))
```

```
Train Score:  0.524458229544834
Test Score:  0.5354158501894077
Mean absolute error = 1.6
Mean squared error = 4.88
Median absolute error = 1.18
Explain variance score = 0.54
R2 score = 0.54
```

7. Create a new attribute Age which is calculated as rings+1.5.

```
abalone['age']=abalone['Rings']+1.5
abalone.head()
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings	age
0	2	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15	16.5
1	2	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7	8.5
2	0	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9	10.5
3	2	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10	11.5
4	1	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7	8.5

8. Finding the correlation between Age and other attributes.

```
abalone.corr()
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings	age
Sex	1.000000	-0.036066	-0.038874	-0.042077	-0.021391	-0.001373	-0.032067	-0.034854	-0.034627	-0.034627
Length	-0.036066	1.000000	0.986812	0.827554	0.925261	0.897914	0.903018	0.897706	0.556720	0.556720
Diameter	-0.038874	0.986812	1.000000	0.833684	0.925452	0.893162	0.899724	0.905330	0.574660	0.574660
Height	-0.042077	0.827554	0.833684	1.000000	0.819221	0.774972	0.798319	0.817338	0.557467	0.557467
Whole weight	-0.021391	0.925261	0.925452	0.819221	1.000000	0.969405	0.966375	0.955355	0.540390	0.540390
Shucked weight	-0.001373	0.897914	0.893162	0.774972	0.969405	1.000000	0.931961	0.882617	0.420884	0.420884
Viscera weight	-0.032067	0.903018	0.899724	0.798319	0.966375	0.931961	1.000000	0.907656	0.503819	0.503819
Shell weight	-0.034854	0.897706	0.905330	0.817338	0.955355	0.882617	0.907656	1.000000	0.627574	0.627574
Rings	-0.034627	0.556720	0.574660	0.557467	0.540390	0.420884	0.503819	0.627574	1.000000	1.000000
age	-0.034627	0.556720	0.574660	0.557467	0.540390	0.420884	0.503819	0.627574	1.000000	1.000000

9. Creating a linear regression model to predict the age of the abalone.

```
y = abalone.iloc[:, 9].values
```

```
x = abalone.iloc[:, range(0,8)].values
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25,random_state=0)
regressor_age= LinearRegression()
regressor.fit(x_train, y_train)
y_pred= regressor.predict(x_test)
print('Train Score: ', regressor.score(x_train, y_train))
print('Test Score: ', regressor.score(x_test, y_test))
print("Mean absolute error =", round(sm.mean_absolute_error(y_test, y_pred), 2))
print("Mean squared error =", round(sm.mean_squared_error(y_test, y_pred), 2))
print("Median absolute error =", round(sm.median_absolute_error(y_test, y_pred), 2))
print("Explain variance score =", round(sm.explained_variance_score(y_test, y_pred), 2))
print("R2 score =", round(sm.r2_score(y_test, y_pred), 2))
```

```
Train Score:  0.524458229544834
Test Score:  0.5354158501894077
Mean absolute error = 1.6
Mean squared error = 4.88
Median absolute error = 1.18
Explain variance score = 0.54
R2 score = 0.54
```

10. Displaying the predicted values of the age column.

```
y_pred
array([14.60451425, 11.16747548, 11.85605247, ..., 11.45962005, 14.09111443,
13.68516586])
```

Conclusion: Thus, we have used classification and regression to predict the rings and age of the abalone.

Frequently asked questions :

Q. No	**Questions**	BT	CO
1		1	1
2		2	1
3		2	1
4		2	1
5		2	1

Guidelines for Students

The experiments should be completed and get checked by the concerned teacher in the lab on or before the date of submission. After which the experiment will not be signed. Every experiment must be included in the file in following format.

o) **Aim**: In this section write complete objective of the program you are going to make in the lab. This section specifies the complete description of the including problem analysis, input description, method used,

fundamental concept and desired output format.

p) **Theory**: Write brief theory related to practical.
q) **Algorithm**: Write Algorithm for given task.
r) **Input**: Write input test data/ or program that are used to test program objective to see whether program is achieving the given objective or not.
s) **Output**: describe the results in few lines
t) **Conclusion**: Write complete conclusion whether what the student has learned from this experiment.
u) **Source Code**: Submit in the form of soft copies.

- **Marking criteria.**

  Experiment completion (Timely)
  Lab file (neatness and regularity)
  Viva (from time to time)
  Mock Practical Exam
  Exam (end term): Practical + Viva

- **Assessment Methodology**

  Timely completion of assignment- 2marks
  Program demonstration- 4 marks
  Viva-voce -2 marks
  Timely submission of journal- 2 marks

Lab Assignment No.	4
Title	Clustering of points
Roll No.	
Class	TE
Date of Completion	
Subject	Software lab 2

Assessor's Sign	

ASSIGNMENT No: 04

Title : Clustering

Aim : To carry out clustering of given points.

Objective : To understand how clustering is carried out.

Problem Statement: We have given a collection of 8 points.
P1=[0.1,0.6]
P2=[0.15,0.71]
P3=[0.08,0.9]
P4=[0.16, 0.85]
P5=[0.2,0.3]
P6=[0.25,0.5]
P7=[0.24,0.1]

P8=[0.3,0.2]
Perform the k-mean clustering with initial centroids as m1=P1 =Cluster#1=C1 and m2=P8=cluster#2=C2.
Answer the following :
1] Which cluster does P6 belong to?
2] What is the population of cluster around m2?
3] What is updated value of m1 and m2?

Theory :
Clustering is a technique used in machine learning and data analysis to group similar data points together based on their characteristics or features. The goal of clustering is to discover inherent structures or patterns within a dataset without prior knowledge of the groups.

One popular clustering algorithm is k-means clustering. The k-means algorithm aims to partition a given dataset into k clusters, where each data point belongs to the cluster with the nearest mean or centroid. The centroids represent the center of each cluster, and the algorithm iteratively adjusts them to minimize the sum of squared distances between data points and their assigned centroids.

Here's how k-means clustering works:

- Initialization: Randomly select k data points from the dataset as initial centroids or use some other initialization method. These centroids represent the initial cluster centers.
- Assignment: Assign each data point to the nearest centroid based on the Euclidean distance or any other distance metric. This step creates initial clusters.
- Update: Recalculate the centroids of each cluster by taking the mean of all the data points assigned to that cluster.
- Repeat: Repeat steps 2 and 3 until convergence or a specified number of iterations is reached. Convergence occurs when the centroids no longer change significantly or when the assignments stabilize.
- Output: The algorithm outputs the final cluster assignments, where each data point belongs to the cluster associated with the nearest centroid.

It's important to note that k-means clustering can converge to local optima since the initial centroids are chosen randomly. Multiple runs with different initializations can help mitigate this issue. Additionally, determining the optimal value of k (the number of clusters) can be challenging and often requires domain knowledge or evaluation metrics.

K-means clustering has several advantages, including simplicity, scalability, and efficiency for large datasets. However, it also has limitations. It assumes that clusters are spherical, equally sized, and have similar densities, which may not hold in all cases. It is sensitive to the initial centroids and outliers, and it may converge to suboptimal solutions.

Overall, k-means clustering is a widely used technique for exploratory data analysis, customer segmentation, image compression, and other applications where grouping similar data points is essential.

Code:

```
import numpy as np

# Define the points and initial centroids
P1 = np.array([0.1, 0.6])
P2 = np.array([0.15, 0.71])
P3 = np.array([0.08, 0.9])
P4 = np.array([0.16, 0.85])
P5 = np.array([0.2, 0.3])
P6 = np.array([0.25, 0.5])
P7 = np.array([0.24, 0.1])
P8 = np.array([0.3, 0.2])

m1 = P1
m2 = P8

# Define the maximum number of iterations
max_iter = 100

# Perform k-means clustering
for i in range(max_iter):
    # Assign each point to the closest centroid
    C1 = []
    C2 = []
    for point in [P1, P2, P3, P4, P5, P6, P7, P8]:
        dist1 = np.linalg.norm(point - m1)
        dist2 = np.linalg.norm(point - m2)
        if dist1 < dist2:
            C1.append(point)
        else:
            C2.append(point)

    # Recalculate the centroids
    m1_new = np.mean(C1, axis=0)
    m2_new = np.mean(C2, axis=0)

    # Check for convergence
    if np.allclose(m1, m1_new) and np.allclose(m2, m2_new):
        break

    # Update the centroids
    m1 = m1_new
    m2 = m2_new

# Print the final clusters
print("Cluster 1:", C1)
print("Length: ", len(C1))
```

```
print("Cluster 2:", C2)
print("Length: ", len(C2))
```

Output:

```
Cluster 1: [array([0.1, 0.6]), array([0.15, 0.71]), array([0.08, 0.9 ]), array([0.16,
0.85]), array([0.25, 0.5 ])]
Length:  5
Cluster 2: [array([0.2, 0.3]), array([0.24, 0.1 ]), array([0.3, 0.2])]
Length:  3
```

Code Explanation:

The code starts by importing the numpy library to perform vector operations and calculations. Next, it defines the 8 points P1 to P8, and initializes the initial centroids m1 and m2 to P1 and P8 respectively. It also sets the maximum number of iterations max_iter to 100. This is the maximum number of times that the algorithm will iterate before it stops.

The code then enters a loop that performs k-means clustering. It repeats the following steps until convergence is reached or until the maximum number of iterations is exceeded:

a. It assigns each point to the closest centroid (m1 or m2) by calculating the Euclidean distance between each point and each centroid.

b. It creates two lists, C1 and C2, to hold the points that are assigned to each centroid.

c. It calculates the new centroids by taking the mean of the points in each cluster.

d. It checks for convergence by using the np.allclose function to compare the old centroids (m1 and m2) with the new centroids (m1_new and m2_new).

e. If convergence is reached, the loop is broken.

f. If convergence is not reached, the old centroids are updated to the new centroids.

Finally, the code prints the final clusters C1 and C2.

Frequently asked questions :

Q. No	**Questions**	BT	CO
1		1	1
2		2	1
3		2	1
4		2	1

Guidelines for Students

The experiments should be completed and get checked by the concerned teacher in the lab on or before the date of submission. After which the experiment will not be signed. Every experiment must be included in the file in following format.

v) **Aim**: In this section write complete objective of the program you are going to make in the lab. This section specifies the complete description of the including problem analysis, input description, method used, fundamental concept and desired output format.

w) **Theory**: Write brief theory related to practical.

x) **Algorithm**: Write Algorithm for given task.

y) **Input**: Write input test data/ or program that are used to test program objective to see whether program is achieving the given objective or not.

z) **Output**: describe the results in few lines

aa) **Conclusion**: Write complete conclusion whether what the student has learned from this experiment.

ab) **Source Code**: Submit in the form of soft copies.

- **Marking criteria.**

  Experiment completion (Timely)
  Lab file (neatness and regularity)
  Viva (from time to time)
  Mock Practical Exam
  Exam (end term): Practical + Viva

- **Assessment Methodology**

  Timely completion of assignment- 2marks
  Program demonstration- 4 marks
  Viva-voce -2 marks
  Timely submission of journal- 2 marks

Group B

Lab Assignment No.	1
Title	Activation Functions
Roll No.	
Class	TE
Date of Completion	
Subject	Software lab 2
Assessor's Sign	

ASSIGNMENT No: 01

Title: Activation Functions

Aim: To write a program to scheme a few activation functions that are used in neural networks.

Objective: To understand various types of activation functions by plotting graphs of these functions.

Problem Statement: Write a program to scheme a few activation functions that are used in neural networks.

Theory:
- **Activation functions**: The main function of the activation function is to get the output node. This is also known as Tran4sfer function.

-Uses of activation functions:
● It is used to get the output layer of the neural network.

● It is used to determine the output of a neural network like yes or no. It maps the resulting values in between 0 to 1 or -1 to 1 etc. (depending upon the function).

- Types of Activation functions:

There are 2 types of activation functions: -

a) Linear Activation Functions.

b) Non-Linear Activation Functions.

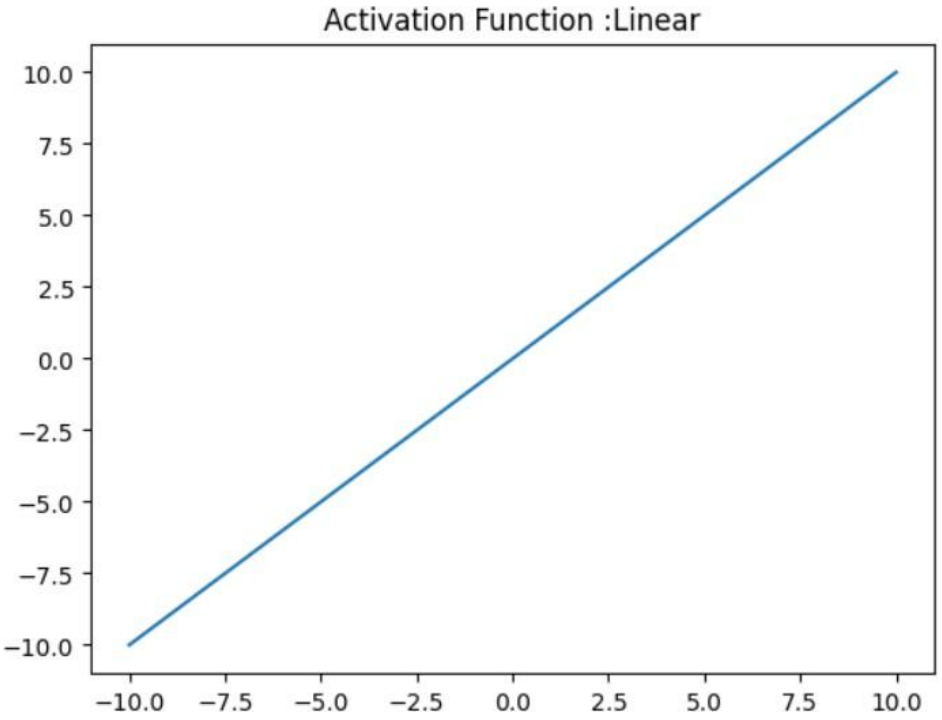
a) Linear Activation Functions:

This is where the activation is proportional to the input. The function doesn't do anything to the weighted sum of the input, it simply splits out the value it was given.

Equation: $f(x) = x$

Range: $(-\infty$ to $\infty)$

Graph:



b) Non-Linear Activation Functions:

Nonlinear functions are known to be the most used activation functions. It makes it easy for neural network models to adapt with the variety of data and to differentiate between the outcomes.

-Terminologies needed to understand the nonlinear functions:

i) **Derivative and Differentia**l: Change in y-axis w.r.t change in x-axis. It is also known as slope.

ii) **Monotonic Functions**: A function which is either entirely non-increasing or non-decreasing.

- Following are various types of nonlinear activation functions:

a) **<u>Sigmoid or Logistic Activation Function:</u>**

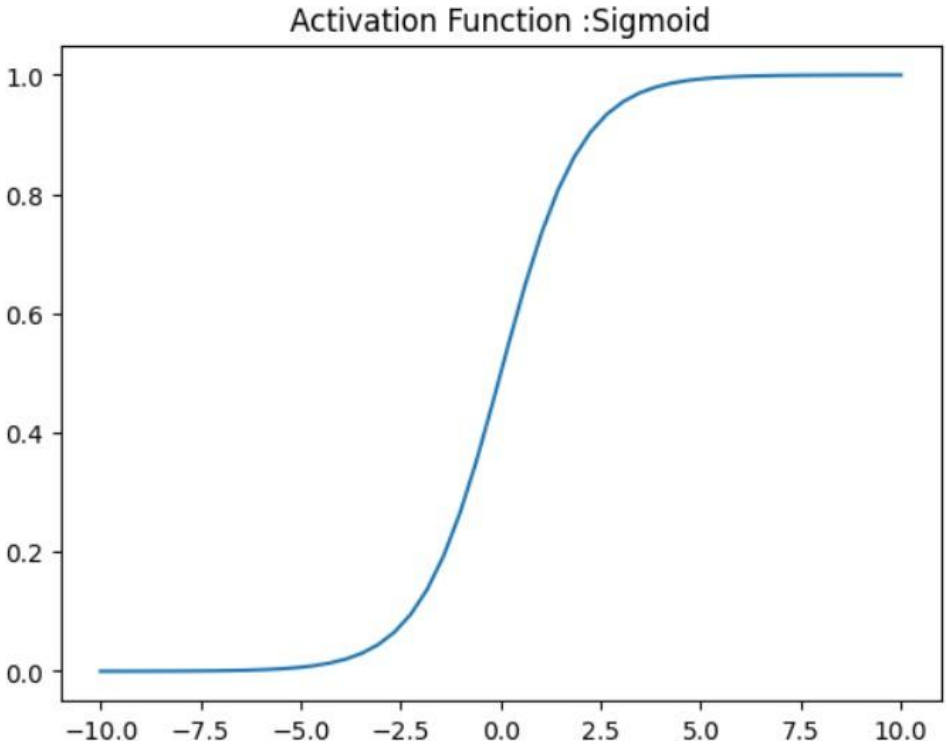
The sigmoid exists between 0 to 1. Therefore, it is especially used for models where we have to predict probability, as probability always exists between 0 and 1.

Equation:

$$S(x) = \frac{1}{1 + e^{-x}}$$

Range: (0 to 1)

Graph:



b) **SoftMax Function:**

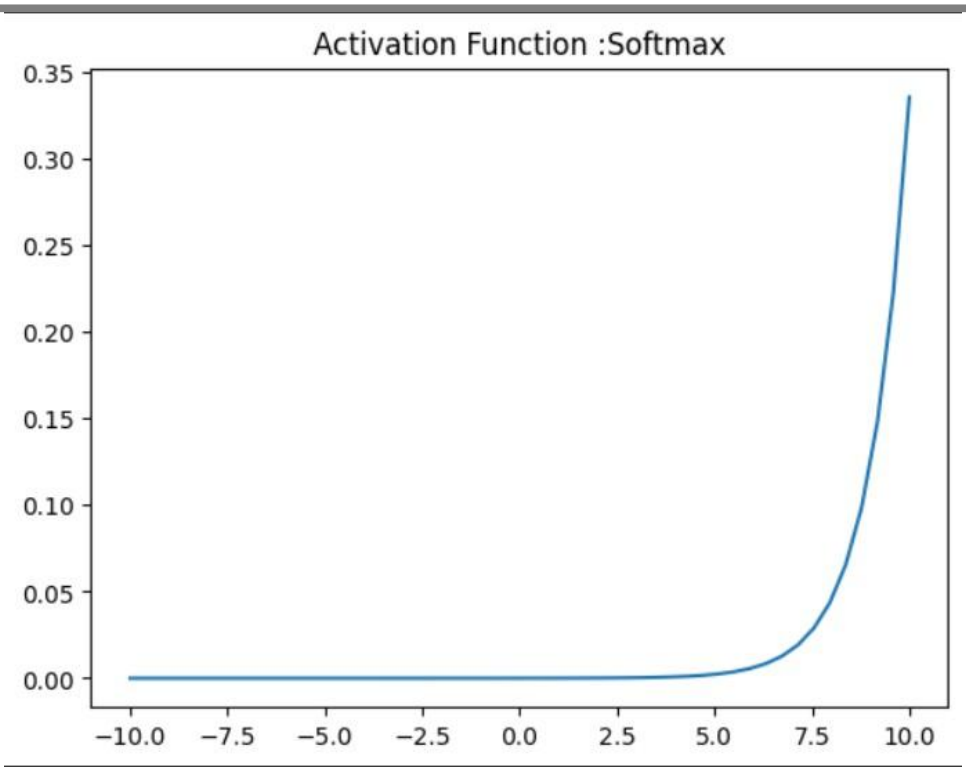
The SoftMax Function is like the sigmoid function, but the key difference is that SoftMax is used for multi-class classification.

Equation:

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Range: (0 to 1)

Graph:



c) **Tanh or hyperbolic activation function:**

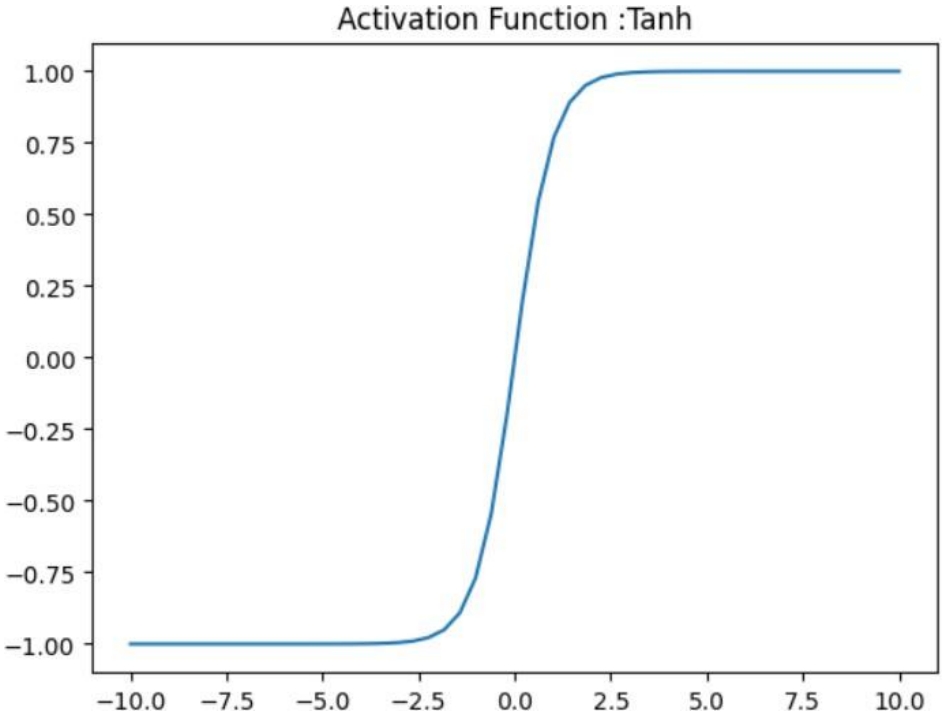
The shape of tanh is just like the sigmoid function, but its range is from (-1 to 1). The advantage of this function is that it maps negative inputs strongly. This function is differentiable and monotonic in nature.

Equation:

$$f(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$$

Range: (-1 to 1)

Graph:

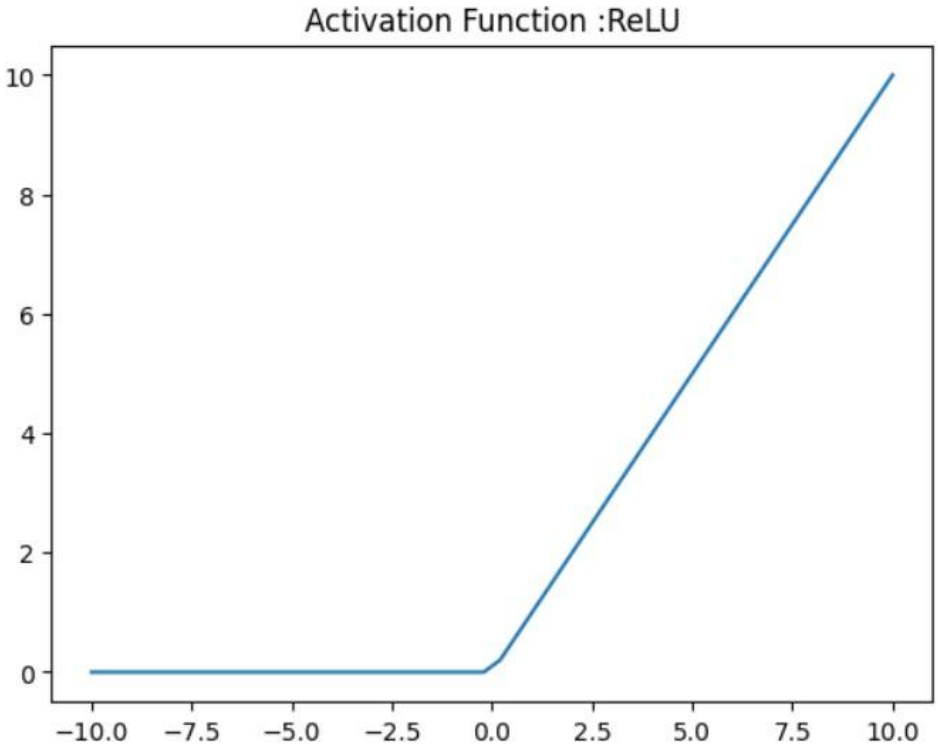


d) **ReLU (Rectified Linear Unit) activation function**:

The ReLU function is half rectified from bottom. f(x) is 0 when x is less than 0 and f(x) is equal to x when x is greater than or equal to 0. The function and its derivative are both monotonic.

Function: f(x) = max(0, z)

Graph:



Code:

```
import matplotlib.pyplot as plt
import numpy as np

class AF():
    def linear(x):
     return x


    def sigmoid(x):
     return 1/(1+np.exp(-x))

    def tanh(x):
     return np.tanh(x)

    def RELU(x):
        x1=[]
        for i in x:
            if i<0:
                x1.append(0)
            else:
                x1.append(i)
```

```
        return x1


    def softmax(x):
     return np.exp(x) / np.sum(np.exp(x), axis=0)

while True:
    n = int(input("Enter\n 0. to Cancel\n 1. Linear\n 2. Sigmoid\n 3.Tanh\n 4.ReLU\n 5.Softmax\n
"))
    if n==0:
        break
    if n==1:
        x = np.linspace(-10, 10)
        plt.plot(x, AF.linear(x))
        plt.title('Activation Function :Linear')
        plt.show()
    elif n==2:
        x = np.linspace(-10, 10)
        plt.plot(x, AF.sigmoid(x))
        plt.title('Activation Function :Sigmoid')
        plt.show()
    elif n==3:
        x = np.linspace(-10, 10)
        plt.plot(x, AF.tanh(x))
        plt.title('Activation Function :Tanh')
        plt.show()
    elif n==4:
        x = np.linspace(-10, 10)
        plt.plot(x, AF.RELU(x))
        plt.title('Activation Function :ReLU')
        plt.show()
    elif n==5:
        x = np.linspace(-10, 10)
        plt.plot(x, AF.softmax(x))
        plt.title('Activation Function :Softmax')
        plt.show()
```

Conclusion: We have learned about the activation functions.

Frequently asked questions :

Q. No	**Questions**	BT	CO
1		1	1
2		2	1
3		2	1
4		2	1
5		2	1

Guidelines for Students

The experiments should be completed and get checked by the concerned teacher in the lab on or before the date of submission. After which the experiment will not be signed. Every experiment must be included in the file in following format.

ac) **Aim**: In this section write the complete objective of the program you are going to make in the lab. This section specifies the complete description of the including problem analysis, input description, method used, fundamental concept and desired output format.

ad) **Theory**: Write brief theory related to practical.

ae) **Algorithm**: Write Algorithm for given task.

af) **Input**: Write input test data/ or program that are used to test program objective to see whether program is achieving the given objective or not.

ag) **Output**: describe the results in few lines

ah) **Conclusion**: Write a complete conclusion whether what the student has learned from this experiment.

ii) **Source Code**: Submit in the form of soft copies.

- **Marking criteria.**

  Experiment completion (Timely)
  Lab file (neatness and regularity)
  Viva (from time to time)
  Mock Practical Exam
  Exam (end term): Practical + Viva

- **Assessment Methodology**

  Timely completion of assignment- 2marks
  Program demonstration- 4 marks
  Viva-voce -2 marks
  Timely submission of journal- 2 marks

Lab Assignment No.	2
Title	Back Propagation
Roll No.	
Class	TE
Date of Completion	
Subject	Software lab 2
Assessor's Sign	

ASSIGNMENT No: 02

Title: Back Propagation

Aim: To write a program to show a back propagation network for XOR function with binary inputs and outputs.

Objective: Understanding how backpropagation works in neural networks.

Problem Statement: Write a program to show a back propagation network for XOR function with binary inputs and outputs..

Theory:

- **Backpropagation**: In an artificial neural network, the values of weights and biases are randomly initialized. Due to random initialization, the neural network probably has errors in giving the correct output. We need to reduce error values as much as possible. So, for reducing these error values, we need a mechanism that can compare the desired output of the neural network with the network's output that consists of errors and adjusts its weights and biases such that it gets closer to the desired output after each iteration. For this, we train the network such that it back propagates and updates the weights and biases. This is the concept of the back propagation algorithm.

- **Features of Backpropagation**:

● It is a gradient descent method as used the case of simple propagation perceptron network with differentiable unit

● It is different from other networks ub respect to the process by which the weights are calculated during the learning period of the network

● Training is done in 3 steps:

1) The feed forward of input training pattern.

2) The calculation and back propagation of error.

3) Updation of the weight.

- Woking of back propagation:

The following are the steps taken in back-propagation algorithm

1) Step 1: Input X, arrive through the preconnected path.

2) Step 2: The inputs in the method, moduled using tree weights W, weights are usually chosen at random.

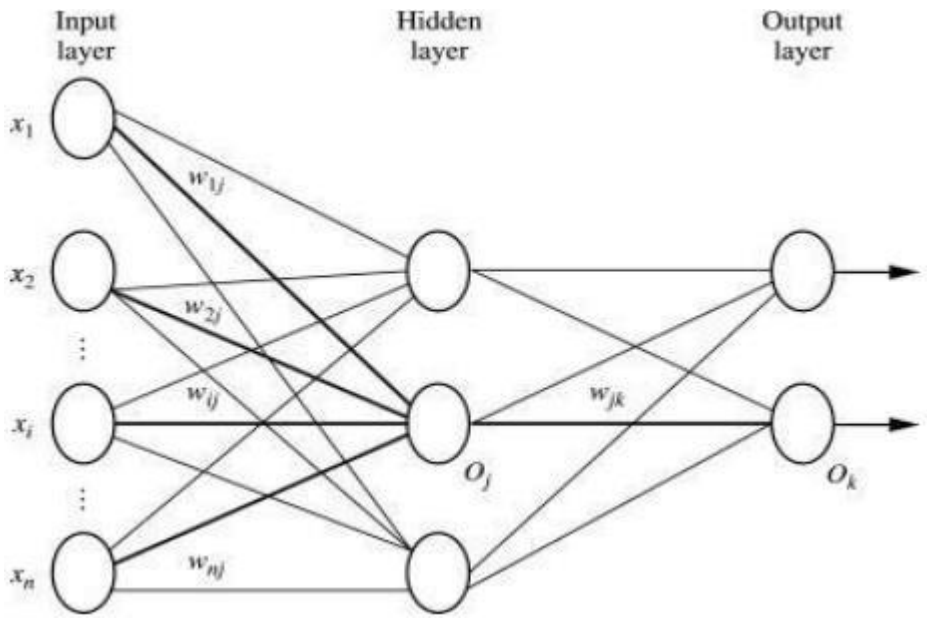
3) Step 3: Calculate the output of each neuron from the input layer to the hidden layer to the output layer.

4) Step 4: Calculate the error in the output. Error = Actual output - Desired output

5) Step 5: From the output layer, go back to the hidden layer to adjust the weights to reduce the error.

6) Step 6: Repeat the process until the desired output is achieved.

- **Diagram**:



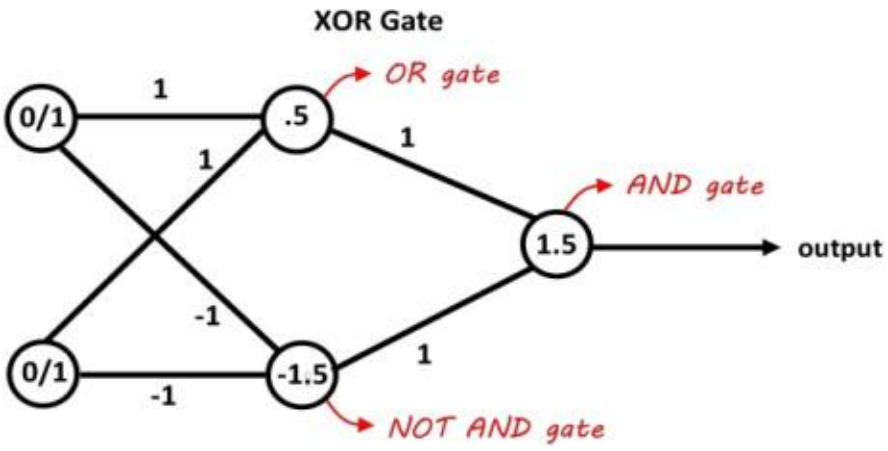
- **XOR Problem**:

1) The XOR problem with neural networks can be achieved by using multi-perceptron or a neural network.

2) During the forward propagation through the neural networks, the weights et updated through to the corresponding layer and XOR logic gets executed

3) For solving the problem, it is necessary to use multiple neurons in an architecture with certain weights and appropriate activation function.

4) The following is the diagram:



- Binary XOR:

1) The binary XOR has 2 inputs and 1 output.

2) It is like the ADD operation which takes the argument and produces one result.

3) It will produce 1 as an output if either of the inputs is 1.

4) It will produce 0 as an output if both inputs are 0 or 1.

Input		output
A	B	C = A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

Code:

```
import numpy as np


# Define the sigmoid activation function and its derivative
def sigmoid(x):
    return 1 / (1 + np.exp(-x))


def sigmoid_derivative(x):
    return sigmoid(x) * (1 - sigmoid(x))
```

```
# Define the XOR function
def xor(inputs):
    return np.array([int(inputs[0] != inputs[1])])


# Define the input and target data
input_data = np.array([[0, 0], [0, 1], [1, 0], [1, 1],[0, 0], [0, 1], [1, 0], [1, 1]])
target_data = np.array([[0], [1], [1], [0], [0], [1], [1], [0]])


# Define the neural network architecture
input_size = 2
hidden_size = 8
output_size = 1


# Initialize the weights with random values
hidden_weights = np.random.uniform(size=(input_size, hidden_size))
output_weights = np.random.uniform(size=(hidden_size, output_size))


# Define the learning rate and number of epochs
learning_rate = 0.1
epochs = 100000


# Train the neural network using backpropagation
for epoch in range(epochs):
    # Forward propagation
    hidden_layer = sigmoid(np.dot(input_data, hidden_weights))
    output_layer = sigmoid(np.dot(hidden_layer, output_weights))

    # Backward propagation
    output_error = target_data - output_layer
    output_delta = output_error * sigmoid_derivative(output_layer)

    hidden_error = output_delta.dot(output_weights.T)
    hidden_delta = hidden_error * sigmoid_derivative(hidden_layer)

    # Update the weights
    output_weights += hidden_layer.T.dot(output_delta) * learning_rate
    hidden_weights += input_data.T.dot(hidden_delta) * learning_rate


# Test the neural network on new input data
```

```
test_input = np.array([[1, 0], [0, 1], [1, 1], [0, 0]])
for i in range(len(test_input)):
    prediction = sigmoid(np.dot(sigmoid(np.dot(test_input[i], hidden_weights)), output_weights))
    print(f"Input: {test_input[i]} Output: {prediction.round()} Target: {xor(test_input[i])}")
```

Conclusion: Hence we have performed backpropagation for XOR problem with binary input and output.

Frequently asked questions :

Q. No	**Questions**	BT	CO
1		1	1
2		2	1
3		2	1
4		2	1
5		2	1

Guidelines for Students

The experiments should be completed and get checked by the concerned teacher in the lab on or before the date of submission. After which the experiment will not be signed. Every experiment must be included in the file in following format.

aj) **Aim**: In this section write the complete objective of the program you are going to make in the lab. This section specifies the complete description of the including problem analysis, input description, method used, fundamental concept and desired output format.

ak) **Theory**: Write brief theory related to practical.

al) **Algorithm**: Write Algorithm for given task.

am) **Input**: Write input test data/ or program that are used to test program objective to see whether program is achieving the given objective or not.

an) **Output**: describe the results in few lines

ao) **Conclusion**: Write a complete conclusion whether what the student has learned from this experiment.

ap) **Source Code**: Submit in the form of soft copies.

- **Marking criteria.**

  Experiment completion (Timely)
  Lab file (neatness and regularity)
  Viva (from time to time)
  Mock Practical Exam
  Exam (end term): Practical + Viva

- **Assessment Methodology**

  Timely completion of assignment- 2marks
  Program demonstration- 4 marks
  Viva-voce -2 marks
  Timely submission of journal- 2 marks

Lab Assignment No.	3
Title	Adaptive Resonance Theory
Roll No.	
Class	TE
Date of Completion	
Subject	Software lab 2
Assessor's Sign	

ASSIGNMENT No: 03

Title: Adaptive Resonance Theory

Aim: To write a program to demonstrate Adaptive Resonance Theory.

Objective: We will learn to implement ART.

Problem Statement: Write a program to demonstrate Adaptive Resonance Theory.

Theory:

- **Adaptive Resonance Theory**: The Adaptive Resonance Theory models are neural networks which perform clustering with a varied cluster size. The term "adaptive" and "resonance" used in this suggests that they are open to new learning (i.e., adaptive) without discarding the previous or the old information (i.e., resonance).

- **Types of ART**:

● **ART1**: It is the simplest and the basic ART architecture. It is capable of clustering binary input values.

● **ART2**: It is an extension of ART1 that is capable of clustering continuous-valued input data.

● **Fuzzy ART**: It is the augmentation of fuzzy logic and ART.

● **ARTMAP**: It is a supervised form of ART learning where one ART learns based on the previous ART module.

● **FARTMAP**: This is a supervised ART architecture with Fuzzy logic included.

- **Basics of Adaptive Resonance Theory**:

1. The basic ART system is an unsupervised learning model. It typically consists of a comparison layer (F1) and a recognition layer (F2) composed of neurons, a vigilance parameter (threshold of recognition), and a reset module.

2. The F1 layer accepts the inputs and performs some processing and transfers it to the F2 layer that best matches the classification factor.

3. There exist two sets of weighted interconnections for controlling the degree of similarity between the units in the F1 and the F2 layer.

4. The F2 layer is a competitive layer.

5. The cluster unit with the large net input becomes the candidate to learn the input pattern first and the rest F2 units

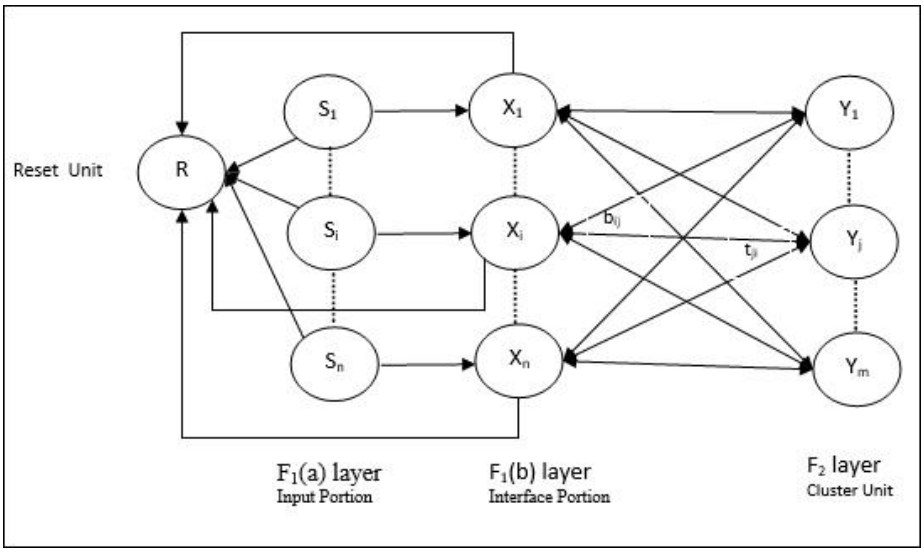
are ignored.

6. The reset unit makes the decision whether the cluster unit is allowed to learn the input pattern depending on how similar its top-down weight vector is to the input vector and to the decision.

7. This is called the vigilance test.

8. Thus we can say that the vigilance parameter helps to incorporate new memories or new information. Higher vigilance produces more detailed memories, lower vigilance produces more general memories.

- Diagram:



- **Advantages**:

1) It exhibits stability and is not disturbed by a wide variety of inputs provided to its network.

2) It can be used for various fields such as target recognition, medical diagnosis, signature verification, clustering web users, etc.

3) It has got advantages over competitive learning. The competitive learning lacks the capability to add new clusters when deemed necessary

- **Disadvantage**:

1) Some ART networks are inconsistent (like the Fuzzy ART and ART1) as they depend upon the order in which training data is, or upon the learning rate.

Code:

```
# -- coding: utf-8 --
"""
Created on Sat Apr 22 17:10:52 2023

@author: anish
```

```
"""

import numpy as np

inputs = [[0, 0, 0, 1], [0, 1, 0, 1], [0, 0, 1, 1], [1, 0, 0, 0]]

roh = 0.4
m = 3

buw = 1/(1+len(inputs[0]))
tdw = 1

buwarr = np.zeros((len(inputs[0]), m))

buwarr[buwarr == 0] = buw

tdwarr = np.zeros((m, len(inputs[0])))

tdwarr[tdwarr == 0] = tdw

index = 0

print("Input -> Cluster")
while True:

    if index <=len(inputs)-1:
        norm_s = sum(inputs[index])
        oplayer = []

        oplayer.sort()
        for column in buwarr.T:
            oplayer.append(sum(inputs[index]*column))

        j = np.where(oplayer == max(oplayer))[0][0]

        x = inputs[index]*tdwarr[j]

        norm_x = sum(x)

        reset = norm_x/norm_s

        if reset >= roh:
            tdwarr[j] = tdwarr[j] * inputs[index]
            buwarr.T[j] = tdwarr[j] / (0.5+sum(tdwarr[j]))
            print(inputs[index], "->", j)
            index += 1

    else:
        break
```

Conclusion: Hence we have performed Adaptive Resonance Theory.

Frequently asked questions :

Q. No	**Questions**	BT	CO
1		1	1
2		2	1
3		2	1
4		2	1
5		2	1

Guidelines for Students

The experiments should be completed and get checked by the concerned teacher in the lab on or before the date of submission. After which the experiment will not be signed. Every experiment must be included in the file in following format.

aq) **Aim**: In this section write the complete objective of the program you are going to make in the lab. This section specifies the complete description of the including problem analysis, input description, method used, fundamental concept and desired output format.

ar) **Theory**: Write brief theory related to practical.

as) **Algorithm**: Write Algorithm for given task.

at) **Input**: Write input test data/ or program that are used to test program objective to see whether program is achieving the given objective or not.

au) **Output**: describe the results in few lines

vv) **Conclusion**: Write a complete conclusion whether what the student has learned from this experiment.

ww) **Source Code**: Submit in the form of soft copies.

- **Marking criteria.**

  Experiment completion (Timely)
  Lab file (neatness and regularity)
  Viva (from time to time)
  Mock Practical Exam
  Exam (end term): Practical + Viva

- **Assessment Methodology**

Timely completion of assignment- 2marks
Program demonstration- 4 marks
Viva-voce -2 marks
Timely submission of journal- 2 marks

Lab Assignment No.	4
Title	Perceptron Learning Rule
Roll No.	
Class	TE
Date of Completion	
Subject	Software lab 2
Assessor's Sign	

ASSIGNMENT No: 04

Title: Perceptron Learning Rule

Aim: To write a program to demonstrate the perceptron learning rule with its decision region using python. Give the output in graphical form.

Objective: We will learn to implement perceptron learning law in python.

Problem Statement: Write a program to demonstrate the perceptron learning rule with its decision region using python. Give the output in graphical form.

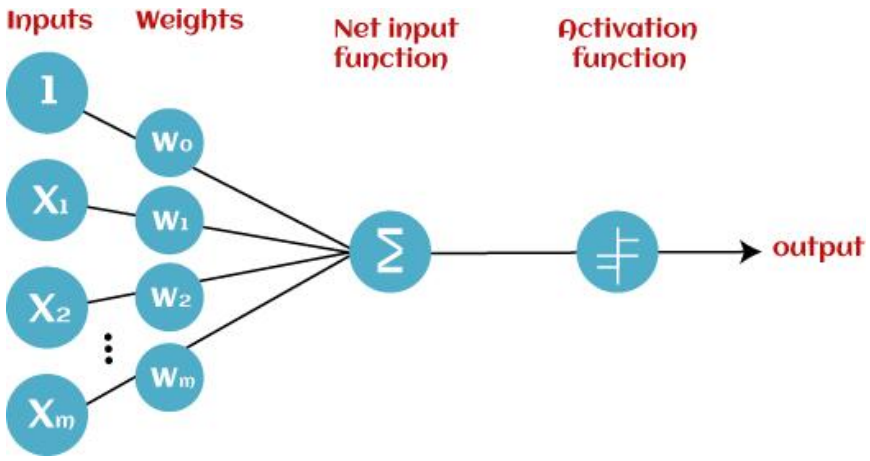
Theory:

- **Perceptron:**

Perceptron is a Machine Learning algorithm for supervised learning of various binary classification tasks. Further Perceptron is also understood as an Artificial Neuron or neural network unit that helps to detect certain input data computations in business intelligence.

- Perceptron is also understood as ANN unit that helps to detect certain input data consumption and computations in business intelligence.

- Perceptron is basically a machine learning algorithm for supervised learning of various binary classification tasks. It is also treated as one of the best and simplest types of ANN.



- Components of Perceptron:

● **Input Node or Input Layer**: This is a primary component of perceptron which accepts the initial data into the system for further processing. Each input node contains a real numerical value.

● **Weight and Bias**: Weight parameter represents the strength of the connection between units. This is another most important parameter of the perceptron component. Weight is directly proportional to the strength of associated input neurons, deciding the output.

● **Activation Function**: These are the final and important component that help to determine whether the neuron will fire or not. Activation function helps in getting the output layer.

- <u>Types of Perceptron</u>:

1. **Single Layer**: This can learn only linearly separable patterns.

2. **Multi-Layer**: This can learn about 2 or more layers having a greater processing power.

- <u>Characteristics of Perceptron</u>:

1. Perceptron is a machine learning algorithm for supervised learning of binary classifiers.

2. In Perceptron, the weight coefficient is automatically learned.

3. Initially, weights are multiplied with input features, and the decision is made whether the neuron is fired or not.

4. The activation function applies a step rule to check whether the weight function is greater than zero.

5. The linear decision boundary is drawn, enabling the distinction between the two linearly separable classes +1 and -1.

6. If the added sum of all input values is more than the threshold value, it must have an output signal; otherwise, no output will be shown.

- Limitations of Perceptron:

1) The output of a perceptron can only be a binary number (0 or 1) due to the hard limit transfer function.

2) Perceptron can only be used to classify the linearly separable sets of input vectors. If input vectors are non-linear, it is not easy to classify them properly.

- Algorithm:

Criteria	Input	Weight
Artists is Good	**x1** = 0 or 1	**w1** = 0.7
Weather is Good	**x2** = 0 or 1	**w2** = 0.6
Friend will Come	**x3** = 0 or 1	**w3** = 0.5
Food is Served	**x4** = 0 or 1	**w4** = 0.3
Alcohol is Served	**x5** = 0 or 1	**w5** = 0.4

- Set a threshold value
- Multiply all inputs with its weights
- Sum all the results
- Activate the output

1. Set a threshold value:
● Threshold = 1.5

2. Multiply all inputs with its weights:
● x1 * w1 = 1 * 0.7 = 0.7
● x2 * w2 = 0 * 0.6 = 0
● x3 * w3 = 1 * 0.5 = 0.5
● x4 * w4 = 0 * 0.3 = 0
● x5 * w5 = 1 * 0.4 = 0.4

3. Sum all the results:
● 0.7 + 0 + 0.5 + 0 + 0.4 = 1.6 (The Weighted Sum)

4. Activate the Output:
● Return true if the sum > 1.5 ("Yes I will go to the Concert")

Code:

```
import numpy as np


# for plotting the graphs
import matplotlib.pyplot as plt
```

```
# for implementing perceptron model
from sklearn.linear_model import Perceptron


x1 = [1, 0, 0, 1]
x2 = [1, 0, 1, 0]


x = [[1,1], [0,0], [0, 1], [1, 0]]
y = [1, 0, 0, 0]


plt.figure(figsize=(3, 3), dpi=80)
plt.xlabel("x1")
plt.ylabel("x2")
plt.scatter(x1, x2, c = y)


clf = Perceptron(max_iter=100).fit(x, y)

print("Classes of the model : ",clf.classes_)
print("Intercept of the decision boundary : ",clf.intercept_)
print("Coefficients of the decision boundary : ",clf.coef_)

ymin, ymax = -1,2
w = clf.coef_[0]
a = -w[0] / w[1]
xx = np.linspace(ymin, ymax)
yy = a * xx - (clf.intercept_[0]) / w[1]


# plotting the decision boundary
plt.figure(figsize=(4, 4))
ax = plt.axes()
ax.scatter(x1, x2, c = y)
plt.plot(xx, yy, 'k-')
ax.set_xlabel('X1')
ax.set_ylabel('X2')
plt.show()
```

Conclusion: Hence we have performed Adaptive Resonance Theory.

Frequently asked questions :

Q. No	**Questions**	BT	CO
1		1	1
2		2	1
3		2	1
4		2	1

5		2	1

Guidelines for Students

The experiments should be completed and get checked by the concerned teacher in the lab on or before the date of submission. After which the experiment will not be signed. Every experiment must be included in the file in following format.

xx) **Aim**: In this section write complete objective of the program you are going to make in the lab. This section specifies the complete description of the including problem analysis, input description, method used, fundamental concept and desired output format.

ay) **Theory**: Write brief theory related to practical.

az) **Algorithm**: Write Algorithm for given task.

aaa) **Input**: Write input test data/ or program that are used to test program objective to see whether program is achieving the given objective or not.

bbb) **Output**: describe the results in few lines

ccc) **Conclusion**: Write complete conclusion whether what the student has learned from this experiment.

ddd) **Source Code**: Submit in the form of soft copies.

- **Marking criteria.**
  Experiment completion (Timely)
  Lab file (neatness and regularity)
  Viva (from time to time)
  Mock Practical Exam
  Exam (end term): Practical + Viva

- **Assessment Methodology**
  Timely completion of assignment- 2marks
  Program demonstration- 4 marks
  Viva-voce -2 marks
  Timely submission of journal- 2 marks