EXPERIMENT No. 01: Basic plots using Matplotlib package

1.1 Objective

Demonstrate all the basic plots using Matplotlib package and python programming.

1.2 Software Required

Windows 10, Jupyter Notebook.

1.3 Pre-Requisite

- Basics of Python Programming,
- Concept of Matplotlib Package

1.4 Introduction

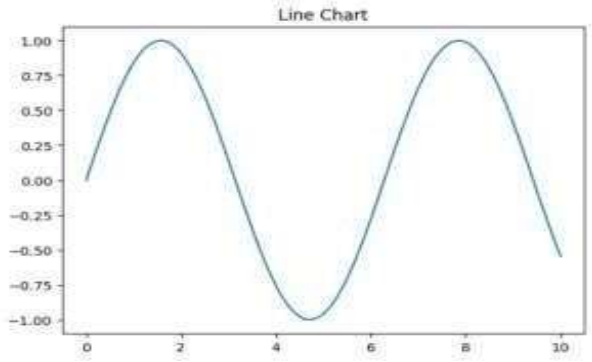
Matplotlib is a powerful and widely used Python library for creating static, animated, and interactive visualizations. It provides a flexible framework for generating a variety of plots and charts to help visualize data effectively.

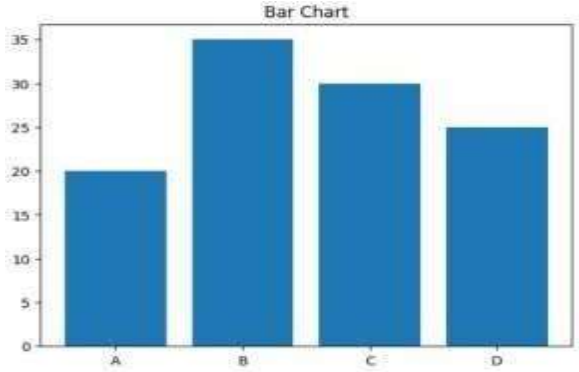
1.5 Procedure

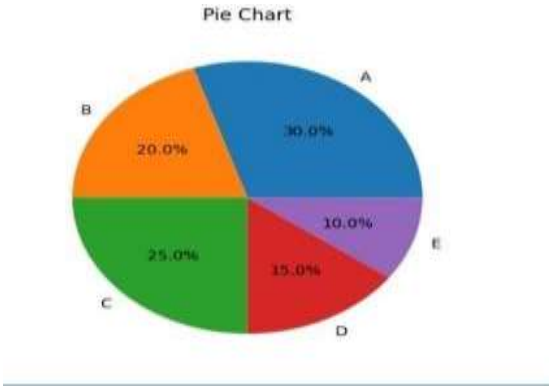
```
import matplotlib.pyplot as plt
import numpy as np
# Generate some data for plotting
x = np.linspace(0, 10, 100)
y = np.sin(x)
# Line Chart
plt.figure()
plt.plot(x, y)
```

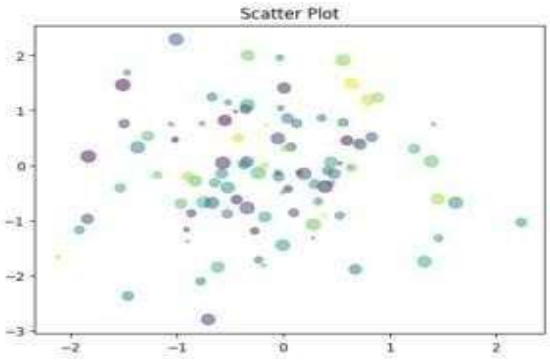
```
plt.title("Line Chart")
# Bar Chart
categories = ['A', 'B', 'C', 'D']
values = [20, 35, 30, 25]
plt.figure()
plt.bar(categories, values)
plt.title("Bar Chart")
# Scatter Plot
x = np.random.randn(100)
y = np.random.randn(100)
colors = np.random.rand(100)
sizes = 100 * np.random.rand(100)
plt.figure()
plt.scatter(x, y, c=colors, s=sizes, alpha=0.5)
plt.title("Scatter Plot")
# Pie Chart
sizes = [30, 20, 25, 15, 10]
labels = ['A', 'B', 'C', 'D', 'E']
plt.figure()
plt.pie(sizes, labels=labels, autopct="%1.1f%%")
plt.title("Pie Chart")
# Show all plots
plt.show()
```

1.6 Results









1.7 Pre-Requisite Questions

1. What is data visualization and why is it important?
2. List different types of plots supported by Matplotlib.
3. How do you install and import the Matplotlib library in Python?
4. What is the difference between plt.plot() and plt.bar()?
5. Explain how to label axes and add a title to a plot using Matplotlib.

5.1 Post-Experimental Questions

1. What are the different types of plots you implemented using Matplotlib?
2. How do you customize the appearance of a plot (e.g., title, axis labels, colors)?
3. What is the difference between plt.plot() and plt.scatter()?
4. How would you plot multiple data series on the same graph?

EXPERIMENT No. 02: File Operations on Excel Dataset

2.1 Objective

Implement a python program to perform File Operations on Excel Dataset.

2.2 Software Required

Windows 10, Jupyter Notebook.

2.3 Pre-Requisite

- Basics of Python Programming,
- Basic Knowledge of Excel

2.4 Introduction

File operations on Excel datasets involve various tasks such as reading, writing, and manipulating data stored in Excel files. Excel files are commonly used for storing structured data in rows and columns, making them suitable for organizing datasets.

2.5 Procedure

```
import pandas as pd
df=pd.read_csv('Salary_Data.csv')

print("First few rows")
print(df.head())

print("\n Summary statistics:")
print(df.describe())
```

```
filtered_data=df[df['Age']>30] # filter the df when condn is true
print("\n Filtered data(Age>30):")
print(filtered_data)

sorted_data=df.sort_values(by='Salary',ascending=False)
print("\nSorteddata(by Salary):")
print(sorted_data)

df['Bonus']=df['Salary']*0.1
print("\n Data with new column(Bonus)")
print(df)

df.to_excel('Output.xlsx',index=False)
print("\n Data written to output.xlsx")
```

2.6 Results

```
First few rows
   YearsExperience   Salary
0              1.1  39343.0
1              1.3  46205.0
2              1.5  37731.0
3              2.0  43525.0
4              2.2  39891.0


Summary statistics:
       YearsExperience         Salary
count        30.000000      30.000000
mean          5.313333   76003.000000
std           2.837888   27414.429785
min           1.100000   37731.000000
25%           3.200000   56720.750000
50%           4.700000   65237.000000
75%           7.700000  100544.750000
max          10.500000  122391.000000
```

```
Filtered data(Age>30):
      Age  Gender Education Level                  Job Title  \
0    32.0    Male      Bachelor's             Software Engineer
2    45.0    Male             PhD               Senior Manager
3    36.0  Female      Bachelor's                Sales Associate
4    52.0    Male        Master's                     Director
6    42.0  Female        Master's                Product Manager
..    ...     ...             ...                          ...
369  33.0    Male      Bachelor's       Junior Business Analyst
370  35.0  Female      Bachelor's      Senior Marketing Analyst
371  43.0    Male        Master's        Director of Operations
373  34.0    Male      Bachelor's  Senior Operations Coordinator
374  44.0  Female             PhD       Senior Business Analyst

     Years of Experience    Salary
0                    5.0   90000.0
2                   15.0  150000.0
3                    7.0   60000.0
4                   20.0  200000.0
6                   12.0  120000.0
..                   ...       ...
369                  4.0   60000.0
370                  8.0   85000.0
371                 19.0  170000.0
373                  7.0   90000.0
374                 15.0  150000.0
```

2.7 Pre-Requisite Questions

1. What are the common file formats used for storing data?
2. How do you read an Excel file in Python?
3. What is the purpose of the pandas.read_excel() function?
4. How do you write data to an Excel file using pandas?
5. What Python package is commonly used for Excel file manipulation?

2.8 Post- Experimental Questions

1. What libraries are required to handle Excel files in Python?
2. How do you read and write data to an Excel file using Python?
3. What challenges did you face while working with Excel file formats?
4. How do you handle missing or null values in Excel files programmatically?
5. What are the benefits of automating Excel file operations with Python?

EXPERIMENT No.03: Array operations using the Numpy package

3.1 Objectives

Write a python program to perform Array operations using the Numpy package

3.2 Software Required

Windows 10, Jupyter Notebook.

3.3 Pre-Requisite:

- Basics of Python Programming,
- Basic Math Knowledge.

3.4 Introduction:

Array operations using the Numpy package involve performing efficient and powerful computations on arrays, which are collections of elements (usually numbers) organized in a grid or matrix format. Numpy is a fundamental package in Python for scientific computing, and it provides a wide range of tools to create, manipulate, and perform mathematical operations on arrays.

3.5 Procedure:

```
import numpy as np
# Create two sample arrays
array1 = np.array([[1, 2, 3], [4, 5, 6]])
array2 = np.array([[7, 8, 9], [10, 11, 12]])
print("Array 1:")
print(array1)
```

```
print("\nArray 2:")
print(array2)

# Addition
array_addition = array1 + array2
print("\nAddition of Array 1 and Array 2:")
print(array_addition)

# Subtraction
array_subtraction = array1 - array2
print("\nSubtraction of Array 1 and Array 2:")
print(array_subtraction)

# Multiplication (element-wise)
array_multiplication = array1 * array2
print("\nElement-wise Multiplication of Array 1 and Array 2:")
print(array_multiplication)

# Transpose of Array 1
array_transpose = np.transpose(array1)
print("\nTranspose of Array 1:")
print(array_transpose)

# Reshape Array 1 from (2, 3) to (3, 2)
array_reshaped = array1.reshape(3, 2)
print("\nReshaped Array 1 (from 2x3 to 3x2):")
print(array_reshaped)
```

```
# Calculate Mean
mean_array1 = np.mean(array1)
print("\nMean of Array 1:")
print(mean_array1)

# Calculate Median
median_array1 = np.median(array1)
print("\nMedian of Array 1:")
print(median_array1)

# Calculate Min and Max
min_value = np.min(array1)
max_value = np.max(array1)
print("\nMin and Max of Array 1:")
print("Min:", min_value)
print("Max:", max_value)

# Calculate Standard Deviation
std_deviation = np.std(array1)
print("\nStandard Deviation of Array 1:")
print(std_deviation)
```

3.6 Results:

```
Array 1:
[[1 2 3]
 [4 5 6]]

Array 2:
[[ 7  8  9]
 [10 11 12]]
```

```
Addition of Array 1 and Array 2:
[[ 8 10 12]
 [14 16 18]]

Subtraction of Array 1 and Array 2:
[[-6 -6 -6]
 [-6 -6 -6]]

Element-wise Multiplication of Array 1 and Array 2:
[[ 7 16 27]
 [40 55 72]]

Transpose of Array 1:
[[1 4]
 [2 5]
 [3 6]]

Reshaped Array 1 (from 2x3 to 3x2):
[[1 2]
 [3 4]
 [5 6]]

Mean of Array 1:
3.5

Median of Array 1:
3.5

Min and Max of Array 1:
Min: 1
Max: 6

Standard Deviation of Array 1:
1.707825127659933
```

3.7 Pre – Experimentation Questions

1. What is a NumPy array and how is it different from a Python list?
2. How do you create a NumPy array?
3. List at least three functions provided by NumPy for array manipulation.
4. What is the use of array slicing and indexing in NumPy?

3.8 Post-Requisite Questions

1. What are the advantages of using Numpy arrays over Python lists?
2. Explain the difference between array slicing and indexing.
3. How do you perform mathematical operations on arrays?

EXPERIMENT No.04: Data Manipulation operations

4.1 Objective

Write a python program to perform Data Manipulation operations using Pandas package.

4.2 Software Required

Windows 10, Jupyter Notebook.

4.3 Pre-Requisite

Basics of Python Programming

4.4 Introduction

Pandas is one of the most widely used open-source libraries in Python for data manipulation and analysis. It provides two primary data structures: **S**eries (1-dimensional) and DataFrame (2-dimensional). Pandas allows users to efficiently manipulate, analyze, and visualize data, especially structured data such as tabular data (spreadsheets, SQL databases, CSV files, etc.).

4.5 Procedure

```
import pandas as pd
# Creating data frame
data={
'Name':['John','Emma','Sant','Lisa','Tom'],
'Age':[25,30,28,32,27],
'Country':['USA','Canada','India','UK','Australia'],
'Salary':[50000,60000,70000,80000,65000]
}
df=pd.DataFrame(data)
```

```
print("Original DataFrame")
print(df)
# selecting specific Coloumns and printing
name_age=df[['Name','Age']]
print("Name and Age columns")
print(name_age)


# filtering the rows based on the conditions
filtered_df=df[df['Country']=='USA']
print("\nfiltered DataFrame(Country='USA')")
print(filtered_df)

# Sorting Data by Salary in Descending Order
sorted_df=df.sort_values("Salary",ascending=False)
print("\nsorted DataFrame(by ssalary in descending order)")
print(sorted_df)

# Calculating the Average Salary
average_Salary=df['Salary'].mean()
print("\nAverage salary",average_Salary)


# Adding a New Column: Experience
df['Experience']=[3,6,4,8,5]
print("\nDataFrame with added experience")
print(df)


# Updating a Specific Row Value
df.loc[df['Name']=='Emma','Salary']=65000
print("\nDataFrame with updating emma salary")
print(df)
```

```
# Deleting a Column
df.drop('Experience',axis=1, inplace=True)
print("\nDataFrame after deleting the column ")
print(df)
```

4.6 Result

```
Original DataFrame
   Name  Age    Country  Salary
0  John   25        USA   50000
1  Emma   30     Canada   60000
2  Sant   28      India   70000
3  Lisa   32         UK   80000
4   Tom   27  Australia   65000
Name and Age columns
   Name  Age
0  John   25
1  Emma   30
2  Sant   28
3  Lisa   32
4   Tom   27


filtered DataFrame(Country='USA')
   Name  Age Country  Salary
0  John   25     USA   50000


sorted DataFrame(by ssalary in descending order)
   Name  Age    Country  Salary
3  Lisa   32         UK   80000
2  Sant   28      India   70000
4   Tom   27  Australia   65000
1  Emma   30     Canada   60000
0  John   25        USA   50000

 Average salary 65000.0


DataFrame with added experience
   Name  Age    Country  Salary  Experience
0  John   25        USA   50000           3
1  Emma   30     Canada   60000           6
2  Sant   28      India   70000           4
3  Lisa   32         UK   80000           8
4   Tom   27  Australia   65000           5


 DataFrame with updating emma salary
```

```
    Name  Age    Country  Salary  Experience
0   John   25        USA   50000           3
1   Emma   30     Canada   65000           6
2   Sant   28      India   70000           4
3   Lisa   32         UK   80000           8
4    Tom   27  Australia   65000           5


DataFrame after deleting the column
    Name  Age    Country  Salary
0   John   25        USA   50000
1   Emma   30     Canada   65000
2   Sant   28      India   70000
3   Lisa   32         UK   80000
4    Tom   27  Australia   65000
```

4.7 Pre-Requisite Questions

1. What are DataFrames and Series in pandas?
2. How do you load a dataset into a pandas DataFrame?
3. What is the difference between .loc[] and .iloc[]?
4. How do you filter rows based on a condition?
5. Explain how to handle missing data in pandas.

4.8 Post-Experimental Questions

1. What is the difference between a Series and a DataFrame in Pandas?
2. How do you filter rows based on column values?
3. What methods are used to handle missing data in a DataFrame?
4. How can you merge or concatenate DataFrames?
5. How do you summarize and analyze data using Pandas?

EXPERIMENT No.05: Linear Regression operation

5.1 Objective

Demonstrate Linear Regression operation using python programming

5.2 Software Required

Windows 10, Jupyter Notebook.

5.3 Pre-Requisite

- Basics of Python Programming,
- Basic Knowledge of Statistics
- Understanding of Linear Algebra

5.4 Introduction

Linear Regression is a simple and widely used machine learning algorithm that models the relationship between a dependent variable (target) and one or more independent variables (features). It aims to find the best-fitting straight line through the data points, which can be used to predict the target variable based on the input features.

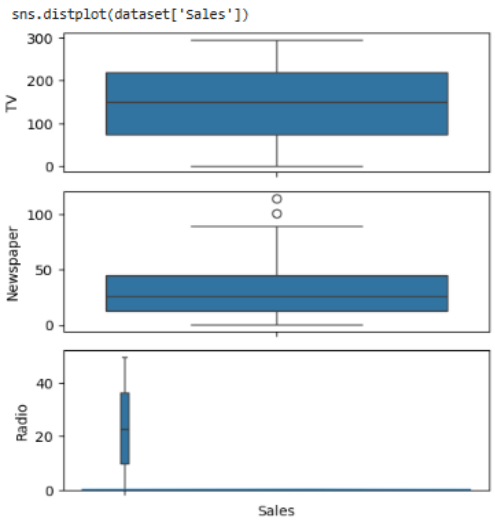
5.5 Procedure

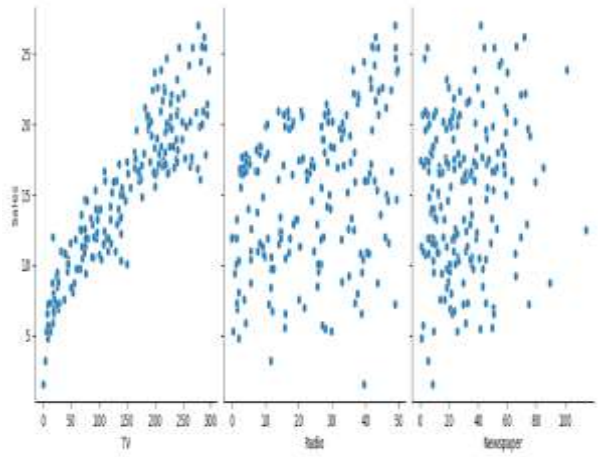
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
dataset = pd.read_csv('advertising.csv')
dataset.head(10)
```

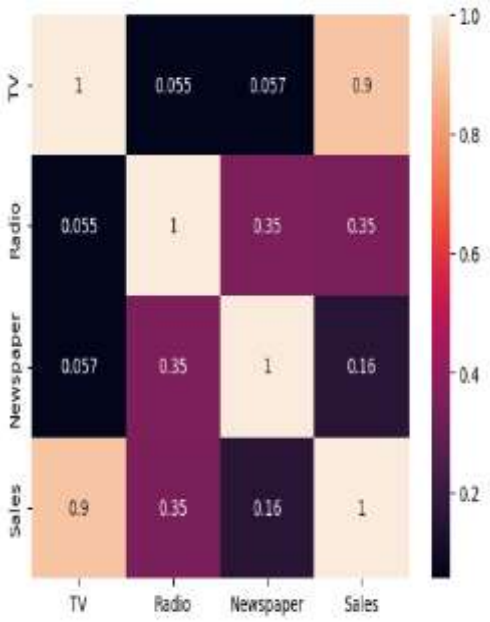
```
dataset.shape
dataset.isna().sum()
dataset.duplicated().any()

fig, axs = plt.subplots(3, figsize = (5,5))
plt1 = sns.boxplot(dataset['TV'], ax = axs[0])
plt2 = sns.boxplot(dataset['Newspaper'], ax = axs[1])
plt3 = sns.boxplot(dataset['Radio'], ax = axs[2])
plt.tight_layout()

sns.displot(dataset['Sales']);
sns.pairplot(dataset, x_vars=['TV', 'Radio', 'Newspaper'], y_vars='Sales',
height=4, aspect=1, kind='scatter')
sns.pairplot(dataset, x_vars=['TV', 'Radio', 'Newspaper'], y_vars='Sales',
kind='scatter')
plt.show()

sns.heatmap(dataset.corr(), annot = True)
plt.show()

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics

x= dataset[['TV']]
y= dataset['Sales']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3,
random_state = 100)
slr= LinearRegression()
slr.fit(x_train, y_train)
print('Intercept:', slr.intercept_)
print('Coefficient:', slr.coef_)
```

```
print('Regression Equation: Sales = 6.948 + 0.054 * TV')
plt.scatter(x_train, y_train)
plt.plot(x_train, 6.948 + 0.054*x_train, 'r')
plt.show()

y_pred_slr= slr.predict(x_test)
x_pred_slr= slr.predict(x_train)
print("Prediction for test set: {}".format(y_pred_slr))
slr_diff = pd.DataFrame({'Actual value': y_test, 'Predicted value': y_pred_slr})
print(slr_diff)

#Predict for any value
slr.predict([[56]])
# print the R-squared value for the model
from sklearn.metrics import accuracy_score
print('R squared value of the model: {:.2f}'.format(slr.score(x,y)*100))
```

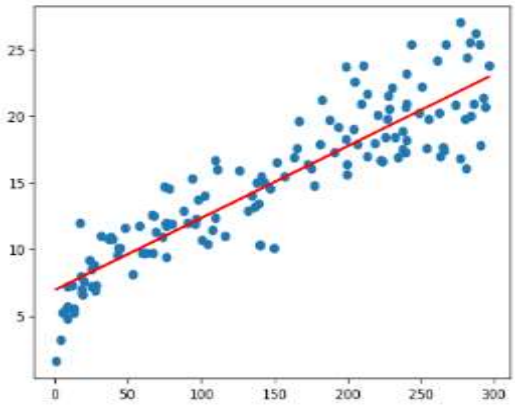
5.6 Results








```
Intercept: 6.948683200001357
Coefficient: [0.05454575]
Regression Equation: Sales = 6.948 + 0.054 * TV
```




```
Prediction for test set: [ 7.37414007 19.94148154 14.32326899 18.82329361 20.13239168 18.2287449
 14.54145201 17.72692398 18.75238413 18.77420243 13.34144544 19.46693349
 10.01415451 17.1923756  11.70507285 12.08689312 15.11418241 16.23237035
 15.8669138  13.1068987  18.65965635 14.00690363 17.60692332 16.60328147
 17.03419291 18.96511257 18.93783969 11.05597839 17.03419291 13.66326538
 10.6796127  10.71234015 13.5487193  17.22510305  9.67597085 13.52144643
 12.25053038 16.13418799 19.07965865 17.48692266 18.69783838 16.53237199
 15.92145955 18.86693021 13.5050827  11.84143724  7.87050642 20.51966653
 10.79961336  9.03233096 17.99419817 16.29237067 11.04506924 14.09963141
 18.44147334  9.3759692   7.88687015  8.34505447 17.72692398 11.62325422]
```

	Actual value	Predicted value
126	6.6	7.374140
104	20.7	19.941482
99	17.2	14.323269
92	19.4	18.823294
111	21.8	20.132392
167	17.2	18.228745
116	12.2	14.541452
96	16.7	17.726924
52	22.6	18.752384
69	22.3	18.774202
164	11.9	13.341445
124	19.7	19.466933
182	8.7	10.014155
154	20.6	17.192376
125	10.6	11.705073
196	14.0	12.086893

1.7 Pre-Requisite Questions

1. What is linear regression and what problem does it solve?
2. What are independent and dependent variables in regression?
3. How do you split data into training and testing sets in Python?
4. What metrics are used to evaluate a regression model?
5. Which library and functions are commonly used for linear regression in Python?

5.7 Post- Experimental Questions

1. What is the mathematical formula behind Linear Regression?
2. How do you train a Linear Regression model in Python?
3. What is the role of the cost function in Linear Regression?

EXPERIMENT No.06: Logistic Regression Classifier

6.1 Objective

Train a regularized logistic regression classifier on the in-build iris dataset using scikit-learn. Train the model and report the best classification accuracy.

6.2 Software Required

Windows 10, Jupyter Notebook.

6.3 Pre-Requisite

- Basics of Python Programming

6.4 Introduction

Logistic Regression is a statistical method used for binary classification, which means it predicts one of two possible outcomes based on input features. Despite its name, it is used for classification tasks rather than regression.

6.5 Procedure

```
 # Importing the necessary libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
# Importing the dataset
dataset = pd.read_csv('iris.csv')
dataset.describe()
dataset.info()

# Splitting the dataset into the Training set and Test set
```

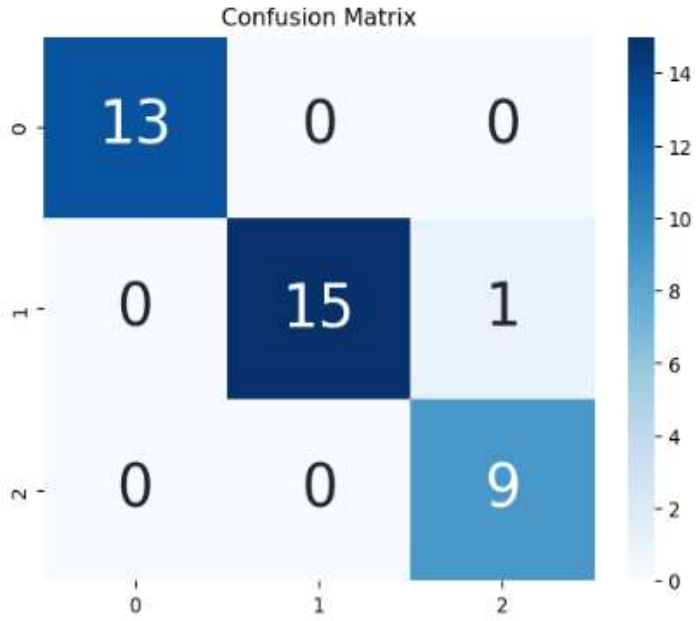
```
X = dataset.iloc[:, [0,1,2, 3]].values
y = dataset.iloc[:, 4].values
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
# Fitting Logistic Regression to the Training set
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0, solver='lbfgs',
multi_class='multinomial')

classifier.fit(X_train, y_train)
# Predicting the Test set results
y_pred = classifier.predict(X_test)
# Predict probabilities
probs_y=classifier.predict_proba(X_test)
probs_y = np.round(probs_y, 2)'

res = "{:<10} | {:<10} | {:<10} | {:<13} | {:<5}".format("y_test", "y_pred",
"Setosa(%)","versicolor(%)", "virginica(%)\n")
res += "-"*65+"\n"
res += "\n".join("{:<10} | {:<10} | {:<10} | {:<13} | {:<10}".format(x, y, a, b, c)
for x, y, a, b,c in zip(y_test, y_pred, probs_y[:,0], probs_y[:,1], probs_y[:,2]))
res += "\n"+"-"*65+"\n"
print(res)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
```

```
cm = confusion_matrix(y_test, y_pred)
print(cm)


# Plot confusion matrix
import seaborn as sns
import pandas as pd
# confusion matrix sns heatmap
## https://www.kaggle.com/agungor2/various-confusion-matrix-plots
ax = plt.axes()
df_cm = cm
sns.heatmap(df_cm, annot=True, annot_kws={"size": 30}, fmt='d',cmap="Blues", ax = ax )
ax.set_title('Confusion Matrix')
plt.show()
```

6.6 Results

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal.length  150 non-null    float64
 1   sepal.width   150 non-null    float64
 2   petal.length  150 non-null    float64
 3   petal.width   150 non-null    float64
 4   variety       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
y_test     | y_pred     | Setosa(%)  | versicolor(%) | virginica(%)
--------------------------------------------------------------------
Virginica  | Virginica  | 0.0        | 0.03          | 0.97
Versicolor | Versicolor | 0.01       | 0.95          | 0.04
Setosa     | Setosa     | 1.0        | 0.0           | 0.0
Virginica  | Virginica  | 0.0        | 0.08          | 0.92
Setosa     | Setosa     | 0.98       | 0.02          | 0.0
Virginica  | Virginica  | 0.0        | 0.01          | 0.99
```

```
[[13  0  0]
 [ 0 15  1]
 [ 0  0  9]]
```



6.7 Pre- Requisite Questions

1. What is logistic regression and how does it differ from linear regression?
2. What is regularization and why is it needed?
3. What is the significance of the iris dataset?
4. How do you train and test a model using scikit-learn?
5. What does model accuracy indicate in classification?

6.8 Post- Experimental Questions

1. What is the difference between Logistic and Linear Regression?
2. How do you split a dataset into training and testing sets?
3. What is regularization, and why is it important?
4. How is classification accuracy measured?
5. What are the limitations of logistic regression?

EXPERIMENT No.07: Back Propagation Algorithm

7.1 Objective

Build an Artificial Neural Network by implementing the Backpropagation algorithm and test the same using appropriate data sets.

7.2 Apparatus Required

Windows 10, Jupyter Notebook

7.3 Pre-Requisite

- Basics of Python Programming
- Basic Mathematics

7.4 Introduction

The Backpropagation Algorithm is a fundamental technique used in training artificial neural networks. It is an optimization method designed to minimize the error in the network's predictions by adjusting the weights of connections between neurons.

7.5 Procedure

```
import numpy as np

# Input data
x = np.array(([2, 9], [1, 9], [3, 6]), dtype=float)
y = np.array(([92], [86], [89]), dtype=float)

# Normalize inputs
x = x / np.amax(x, axis=0)
y = y / 100
```

```
# Activation functions
def sigmoid(x):
    return 1 / (1 + np.exp(-x))

def derivation_sigmoid(x):
    return x * (1 - x)

# Hyperparameters
epoch = 5000
lr = 0.1
inputlayer_neurons = 2
hiddenlayer_neurons = 3
outputlayer_neurons = 1

# Initialize weights and biases
wb = np.random.uniform(size=(inputlayer_neurons, hiddenlayer_neurons))
bb = np.random.uniform(size=(1, hiddenlayer_neurons))
wout = np.random.uniform(size=(hiddenlayer_neurons, outputlayer_neurons))
bout = np.random.uniform(size=(1, outputlayer_neurons))

# Training loop
for i in range(epoch):
    # Forward propagation
    hinp1 = np.dot(x, wb)
    hinp = hinp1 + bb
    hlayer_act = sigmoid(hinp)

    outinp1 = np.dot(hlayer_act, wout)
    outinp = outinp1 + bout
    output = sigmoid(outinp)
```

```
# Backpropagation
EO = y - output
outgrad = derivation_sigmoid(output)
d_output = EO * outgrad

EH = d_output.dot(wout.T)
hiddengrad = derivation_sigmoid(hlayer_act)
d_hiddenlayer = EH * hiddengrad

# Update weights and biases
wout += hlayer_act.T.dot(d_output) * lr
wb += x.T.dot(d_hiddenlayer) * lr
bout += np.sum(d_output, axis=0, keepdims=True) * lr
bb += np.sum(d_hiddenlayer, axis=0, keepdims=True) * lr

# Display results
 print("Input:\n", x)
 print("Actual:\n", y)
 print("Predicted:\n", output)
```

7 .6 Results

```
Input:
 [[0.66666667 1.]
 [0.33333333 1.]
 [1.0.66666667]]

Actual:
 [[0.92]
 [0.86]
 [0.89]]

Predicted:
 [[0.89078101]
 [0.8874631 ]
 [0.89151747]]
```

7.7 Pre-Requisite Questions

1. What is an artificial neural network?
2. What is the role of the activation function in neural networks?
3. Explain the forward and backward pass in training a neural network.
4. Which Python libraries can be used to build neural networks?

7.8 Post- Experimental Questions

1. What is the structure of a basic artificial neural network?
2. How does the backpropagation algorithm work?
3. What is the role of activation functions in a neural network?
4. How do you prevent overfitting in neural networks?

EXPERIMENT No.08: Develop a Map Reduce Program

.

8.1 Objective

Develop a map reduce program to find the grades of students in python.

8.2 Apparatus Required

Windows 10, Jupyter Notebook

8.3 Pre-Requisite

- Basics of Python Programming

8.4 Introduction

This Python program simulates a **MapReduce** approach to calculate and assign **letter grades** to students based on their test scores. It is inspired by the MapReduce programming model used in distributed computing, but implemented here in a simplified, local form using Python.

8.5 Procedure

```
from collections import defaultdict
# Sample data: List of student grades
data = [
    "1,85",
    "2,90",
    "1,75",
    "2,95",
    "3,80",
    "3,70"
```

```
]

# Mapper function
def mapper(data):
    for line in data:
        try:
            student_id, grade = line.split(',')
            yield student_id, float(grade)
        except ValueError:
            pass  # Skip lines that don't have the correct format

# Reducer function
def reducer(mapped_data):
    student_grades = defaultdict(list)

   # Collect grades for each student
    for student_id, grade in mapped_data:
        student_grades[student_id].append(grade)

 # Calculate average grades and assign letter grades
    average_grades = {}
    for student_id, grades in student_grades.items():
        average = sum(grades) / len(grades)

        # Determine the letter grade based on average marks
        if average >= 90:
            grade_letter = 'A+'
        elif average >= 80:
            grade_letter = 'A'
        elif average >= 70:
            grade_letter = 'B+'
        elif average >= 60:
```

```
            grade_letter = 'B'
        elif average >= 50:
            grade_letter = 'C+'
        elif average >= 40:
            grade_letter = 'C'
        else:
            grade_letter = 'F'

        # Store the average and the letter grade
        average_grades[student_id] = (average, grade_letter)

    return average_grades

# Run the MapReduce-like process
mapped_data = list(mapper(data))
average_grades = reducer(mapped_data)

# Display the results
for student_id, (avg, grade_letter) in average_grades.items():
    print(f"Student {student_id}: Average = {avg:.2f}, Grade = {grade_letter}")
```

8.6 Results

```
Student 1: Average = 80.00, Grade = A
Student 2: Average = 92.50, Grade = A+
Student 3: Average = 75.00, Grade = B+
```

8.7 Pre-Requisite Questions

1. What is the MapReduce programming model?
2. What is the role of the mapper and reducer functions?
3. How do you implement a simple MapReduce operation in Python?
4. What are some real-life examples of using MapReduce?
5. How does MapReduce handle large-scale data processing?

8.8 Post- Experimental Questions

1. What is the MapReduce programming model?
2. How is data processed in the Map and Reduce phases?
3. What are the advantages of using MapReduce for large datasets?
4. How do you simulate a MapReduce process in Python?
5. What challenges did you face while implementing the MapReduce program?