Experiment 1:- Basic Plots Using Matplotlib Package

1. Aim: -

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

2. Software Required:-

- Operating System: Windows 10
- Software Required: Jupyter Notebook

3. Pre-Requisites:-

- Basics of Python Programming,
- Concept of Matplotlib Package

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.

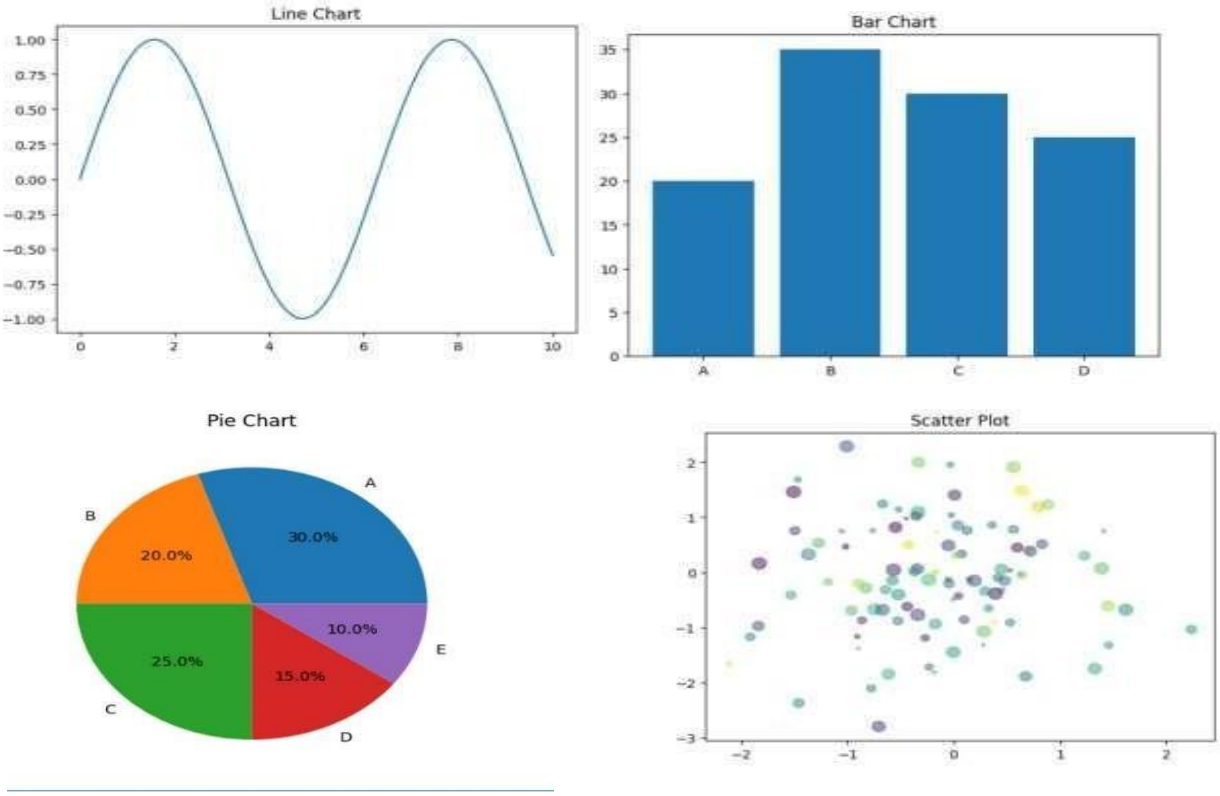
5. Program:-

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

# Generate some data for plotting
x = np.linspace(0, 10, 100)
y = np.sin(x)
plt.figure()
plt.plot(x,y)
plt.title("Line Chart")
categories=['A','B','C','D'] values=[20,35,30,25] plt.figure()
plt.bar(categories,values) plt.title("Bar Chart")
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")
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")
```

```
plt.show()
```

6. Result:-

Output:-



7. Experimental Question:-

1. What is Matplotlib?
2. What is Matplotlib, and why is it used in Python?
3. What is the purpose of a line plot in data visualization?
4. How does a bar plot differ from a line plot?
5. What is the use of a scatter plot?

Experiment 2:- File Operations on Excel Dataset

1. Aim: -

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

2. Software Required:-

- Operating System: Windows 10
- Software Required: Jupyter Notebook

3. Pre-Requisites:-

- Basics of Python Programming,
- Basic Knowledge of Excel

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.

5. Program:-

```
import pandas as pd
df=pd.read_excel('data.xlsx')
print("First few rows")
print(df.head())
print("\n Summary statistics:")
print(df.describe())
filtered_data=df[df['Age']>30]
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")
```

6. Result:-

First few rows

	Age	Gender	Education Level	Job Title	Years of Experience	\
0	32.0	Male	Bachelor's	Software Engineer	5.0	
1	28.0	Female	Master's	Data Analyst	3.0	
2	45.0	Male	PhD	Senior Manager	15.0	
3	36.0	Female	Bachelor's	Sales Associate	7.0	
4	52.0	Male	Master's	Director	20.0	

	Salary
0	90000.0
1	65000.0
2	150000.0
3	60000.0
4	200000.0

Summary statistics:

	Age	Years of Experience	Salary
count	373.000000	373.000000	373.000000
mean	37.431635	10.030831	100577.345845
std	7.069073	6.557007	48240.013482
min	23.000000	0.000000	350.000000
25%	31.000000	4.000000	55000.000000
50%	36.000000	9.000000	95000.000000
75%	44.000000	15.000000	140000.000000
max	53.000000	25.000000	250000.000000

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

```
Sorted data(by Salary):
      Age Gender Education Level                         Job Title  \
30   50.0   Male      Bachelor's                               CEO
83   52.0   Male             PhD           Chief Technology Officer
105  44.0   Male             PhD                 Chief Data Officer
4    52.0   Male        Master's                           Director
53   47.0   Male        Master's                      VP of Finance
..    ...    ...             ...                               ...
97   26.0   Male      Bachelor's          Junior Software Developer
82   25.0   Male      Bachelor's              Sales Representative
259  29.0   Male      Bachelor's  Junior Business Operations Analyst
172   NaN    NaN             NaN                               NaN
260   NaN    NaN             NaN                               NaN

     Years of Experience    Salary
30                  25.0  250000.0
83                  24.0  250000.0
105                 16.0  220000.0
4                   20.0  200000.0
53                  19.0  200000.0
..                   ...       ...
97                   1.0   35000.0
82                   0.0   30000.0
259                  1.5     350.0
172                  NaN       NaN
260                  NaN       NaN

[375 rows x 6 columns]


Data with new column(Bonus)
      Age  Gender Education Level                 Job Title  \
0    32.0    Male      Bachelor's         Software Engineer
1    28.0  Female        Master's              Data Analyst
2    45.0    Male             PhD             Senior Manager
3    36.0  Female      Bachelor's           Sales Associate
4    52.0    Male        Master's                  Director
..    ...     ...             ...                       ...
370  35.0  Female      Bachelor's  Senior Marketing Analyst
371  43.0    Male        Master's    Director of Operations
372  29.0  Female      Bachelor's    Junior Project Manager
373  34.0    Male      Bachelor's  Senior Operations Coordinator
374  44.0  Female             PhD   Senior Business Analyst

     Years of Experience    Salary    Bonus
0                    5.0   90000.0   9000.0
1                    3.0   65000.0   6500.0
2                   15.0  150000.0  15000.0
3                    7.0   60000.0   6000.0
4                   20.0  200000.0  20000.0
..                   ...       ...     ...
370                  8.0   85000.0   8500.0
371                 19.0  170000.0  17000.0
372                  2.0   40000.0   4000.0
373                  7.0   90000.0   9000.0
374                 15.0  150000.0  15000.0

[375 rows x 7 columns]

Data written to output.xlsx
```

7. Experimental Question:-

1. What method is used to save the workbook to a specific file?
2. How do you add data to an Excel sheet in Python?
3. How can you read data from an Excel sheet in Python?
4. How do you update an existing value in an Excel sheet?
5. How do you load an existing Excel workbook?

Experiment 3:- Array operations using the Numpy package

1. Aim: -

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

2. Software Required:-

- Operating System: Windows 10
- Software Required: Jupyter Notebook

3. Pre-Requisites:-

- Basics of Python Programming,
- Basic Math Knowledge

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.

5. Program:-

```
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)
```

6. Result:-

Output:-

```
[[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:

Median of Array 1:
3.5

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

Standard Deviation of Array 1:
1.707825127659933
```

7. Experimental Question:-

1. What is NumPy, and why is it used in Python?
2. How do you install the NumPy package in Python?
3. How do you create arrays in NumPy?
4. How do you create a one-dimensional array in NumPy?
5. How do you perform addition, subtraction, and multiplication of two arrays in NumPy?

Experiment 4:- Back propagation Algorithm

1. Aim: -

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

2. Software Required:-

- Operating System: Windows 10
- Software Required: Jupyter Notebook

3. Pre-Requisites:-

- Basics of Python Programming
- Basic Mathematics

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.

5. Program:-

```
import numpy as np
x=np.array(([2,9],[1,9],[3,6]),dtype=float)
y=np.array(([92],[86],[89]),dtype=float)
x=x/np.amax(x,axis=0)
y=y/100

def sigmoid(x):
return 1/(1+np.exp(-x))
def derivation_sigmoid(x):
return x*(1-x)
epoch=5000
lr=0.1
inputlayer_neurons=2
hiddenlayer_neurons=3
outputlayer_neurons=1

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))
 for i in range(epoch):
 hinp1=np.dot(x,wb)
 hinp=hinp1+bb hlayer_act=sigmoid(hinp) outinp1=np.dot(hlayer_act,wout)
 outinp=outinp1+bout output=sigmoid(outinp)


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
wout+=hlayer_act.T.dot(d_output)*lr
wb+=x.T.dot(d_output)*lr


print("Inpput:\n" +str(x))
print("Actual:\n"+str(y))
print("Predicted:\n",output)
```

6. Result:-

Output:-

```
Input:
[[0.66666667  1.          ]
[0.33333333    1.          ]
[1.              0.66666667  ]]
Actual:
[[0.92
[0.86]
[0.89]]


Predicted:

[[0.89184048]

[0.88433366]

[0.89399225]]
```

7. Experimental Question:-

1. What is Back propagation in Neural Networks?
2. How does back propagation help in adjusting weights in neural networks?
3. What are the key components of an Artificial Neural Network?
4. What is the purpose of learning rate in back propagation?
5. What are the layers in an artificial neural network, and what are their functions?

Experiment 5:-Linear Regression operation

1. Aim: -

Demonstrate Linear Regression operation using python programming.

2. Software Required:-

- Operating System: Windows 10
- Software Required: Jupyter Notebook

3. Pre-Requisites:-

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

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. Program:-

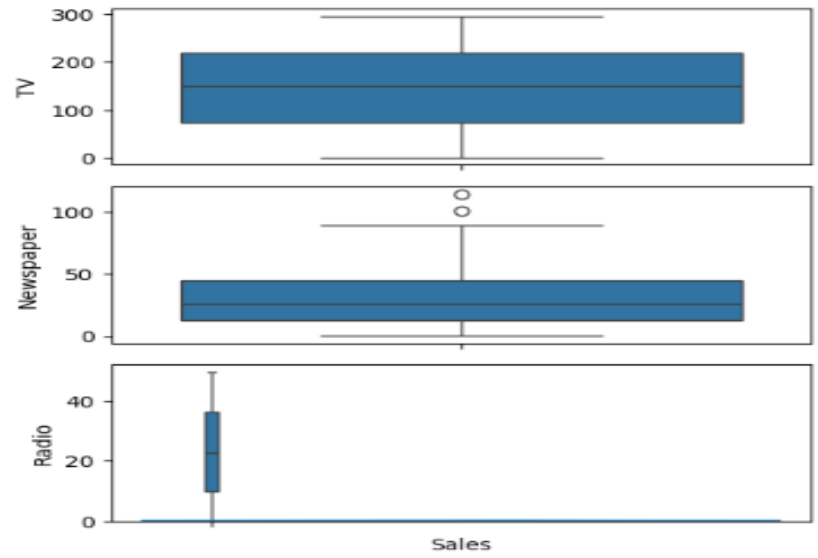
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt import seaborn as sns
dataset = pd.read_csv('advertising.csv')
dataset.head(10)
dataset.shape dataset.isna().sum()
dataset.duplicated().any()
fig, axs = plt.subplots(3, figsize = (5,5))
plt1 = sns.boxplot(dataset['TV'], ax = axs[0])
plt2 = sns.boxplot(dataset['Newspaper'], ax = axs[1])
plt3 = sns.boxplot(dataset['Radio'], ax = axs[2])
plt.tight_layout()
sns.distplot(dataset['Sales']);


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

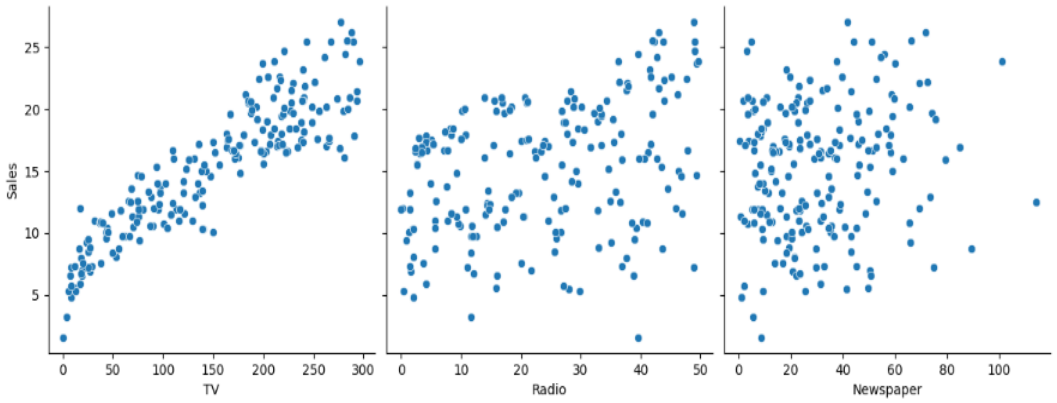
```
plt.show()
sns.heatmap(dataset.corr(), annot = True)
 plt.show()
from sklearn.model_selection  import train_test_split
from sklearn.linear_model  import LinearRegression
from sklearn import metrics
x= dataset[['TV']]
y= dataset['Sales']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 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()
#Prediction of Test and Training set result
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})
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))
```

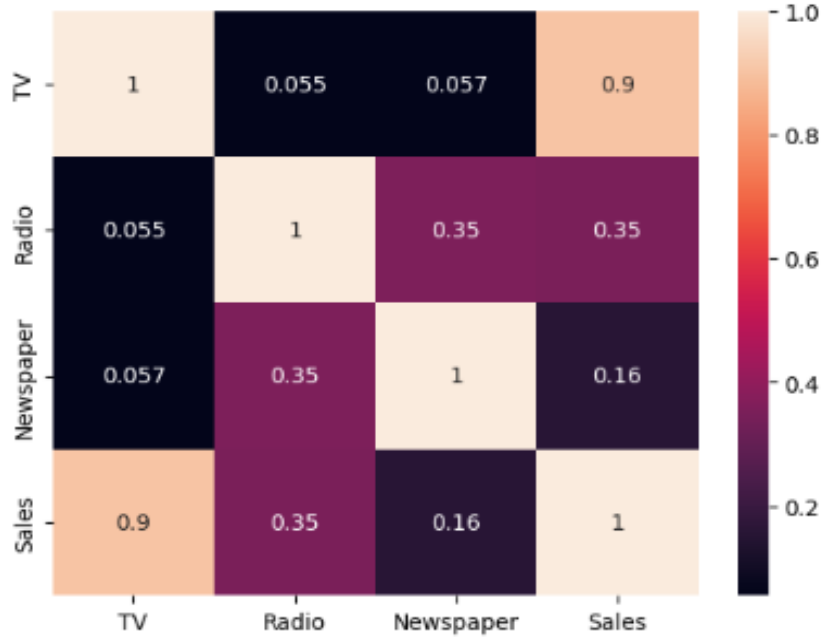
6. Result:-

Output:-

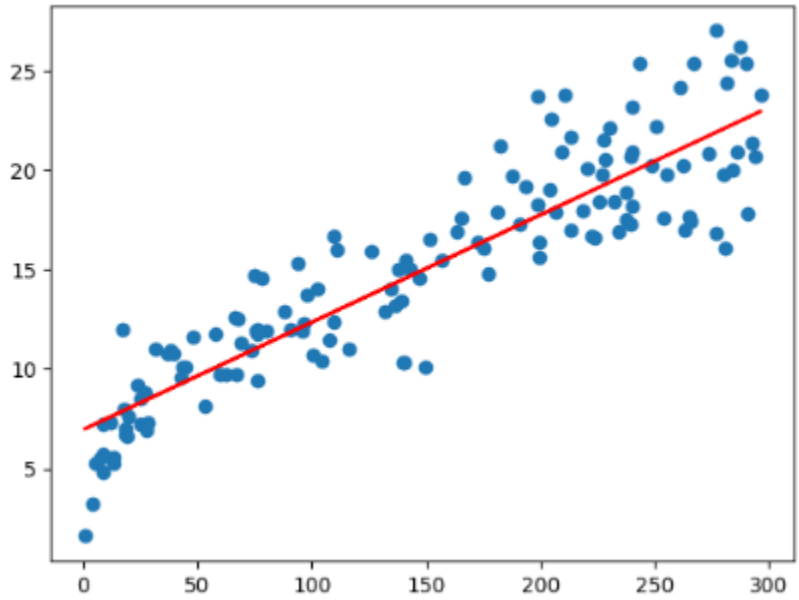
```
sns.distplot(dataset['Sales'])
```







```
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

7. Experimental Question:-

1. What is Linear Regression?
2. What is the output of the `predict()` function?
3. How is the Linear Regression model used in the program?
4. How is the R-squared value interpreted in Linear Regression?
5. What is the output of the `predict()` function?

Experiment 6:- Logistic Regression Classifier

1. Aim: -

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

2. Software Required:-

- Operating System: Windows 10
- Software Required: Jupyter Notebook

3. Pre-Requisites:-

- Basics of Python Programming

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.

5. Program:-

```
# 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='auto')
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()
```

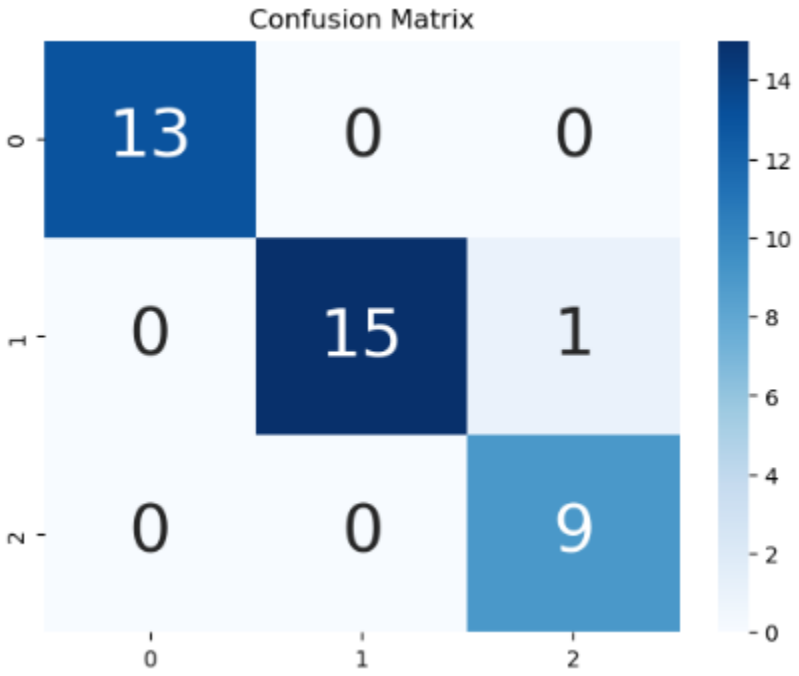
6. Result:-

Output:-

```
<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]]
```



7. Experimental Question:-

1. What is Logistic Regression, and how does it work?
2. What type of problems is Logistic Regression used for?
3. What are the features of the Iris dataset?
4. How many species of Iris are there in the dataset?
5. What could happen if you train and test on the same dataset?

Experiment 7:- Data Manipulation operations

1. Aim: -

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

2. Software Required:-

- Operating System: Windows 10
- Software Required: Jupyter Notebook

3. Pre-Requisites:-

- Basics of Python Programming

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: **Series** (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.).

5. Program:-

```
import pandas as pd

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)



name_age=df[['Name','Age']]

print("Original DataFrame")

print(df) name_age=df[['Name','Age']]

print("Name and Age columns")

print(name_age)

filtered_df=df[df['Country']=='USA']

print("\nfiltered DataFrame(Country='USA')")

print(filtered_df)
```

```
sorted_df=df.sort_values("Salary",ascending=False)

print("\nsorted DataFrame(by ssalary in descending order)")

print(sorted_df)

average_Salary=df['Salary'].mean()


print("\nAverage salary",average_Salary)
df['Experience']=[3,6,4,8,5]

print("\nDataFrame with added experience")

print(df)

df.loc[df['Name']=='Emma','Salary']=65000

print("\nDataFrame with updating emma salary")

print(df)


df.drop('Experience',axis=1) print("\nDataFrame after deleting the column ")

print(df)
```

6. Result:-

Original Data Frame

	Name	Age	Country	Salary
a.	John	25	USA	50000
b.	Emma	30	Canada	60000
c.	Sant	28	India	70000
d.	Lisa	32	UK	80000
e.	Tom	27	Australia	65000

filtered DataFrame(Country='USA')

	Name	Age	Country	Salary
0	John	25	USA	50000

sorted DataFrame(by salary 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	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

7. Experimental Question:-

1. What is Pandas and why is it important in data analysis?
2. What are the key features of the Pandas library in Python?
3. What types of data does Pandas handle?
4. How do you create a DataFrame in Pandas?
5. What are the default sorting options in Pandas?

Experiment 8:- MapReduce Program

1. Aim: -

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

2. Software Required:-

- Operating System: Windows 10
- Software Required: Jupyter Notebook

3. Pre-Requisites:-

- Basics of Python Programming,
- Basic Programming Skills

4. Introduction:-

Data manipulation operations involve techniques used to process and transform data to make it more useful for analysis and decision-making.

5. Program:-

```
 import csv
from mrjob.job import MRJob
class MRStudentGrades (MRJob):
# Mapper function: Process input data and compute average marks def mapper(self,_, line):
# Using csv.reader to parse the CSV line
      reader = csv.reader([line])
      for data in reader:
         # Extract student name and marks (assuming marks are integers)
         name = data[0].strip()
         marks = list(map(int, data[1:])) # Convert marks to integers
         # Calculate the average of the marks
         average=sum(marks)/ len(marks)
# Emit student name as key, and average marks as value yield name, average

   # Reducer function: Assign grades based on average marks
   def reducer(self, key, values):
    # There will be only one average value per student
      average = list(values)[0]
    # Determine the grade based on average marks
      if average >= 90:
         grade = 'A+'
      elif average >= 80:
         grade = 'A'
```

```
        elif average >= 70:
grade = 'B+' elif average >= 60:
            grade = 'B'
        elif average >= 50:
            grade = 'C+'
        elif average >= 40:
            grade = 'C'
        else:
            grade = 'F'
        # Yield student name and their grade
```

Pending pgm

6. Result:-

7. Experimental Question:-

1. What is MapReduce, and how does it work?
2. What are the roles of the "Map" and "Reduce" functions in MapReduce?
3. Why is MapReduce useful in processing large datasets?
4. What would the **Reduce function** do in the same job?
5. What would the output of the `map()` function look like in this scenario?