AD3301 DATA EXPLORATORY AND VISUALIZATION

(R2021)

SYLLABUS

COURSE OBJECTIVES

- To outline an overview of exploratory data analysis.
- To implement data visualization using Matplotlib.
- To perform univariate data exploration and analysis.
- To apply bivariate data exploration and analysis.
- To use Data exploration and visualization techniques for multivariate and time series data.

LIST OF EXPERIMENTS

1. Install the data Analysis and Visualization tool: R/ Python /Tableau Public/ Power BI.
2. Perform exploratory data analysis (EDA) on with datasets like email data set. Export all your emails as a dataset, import them inside a pandas data frame, visualize them and get different insights from the data.
3. Working with Numpy arrays, Pandas data frames , Basic plots using Matplotlib.
4. Explore various variable and row filters in R for cleaning data. Apply various plot features in R on sample data sets and visualize.
5. Perform Time Series Analysis and apply the various visualization techniques.
6. Perform Data Analysis and representation on a Map using various Map data sets with Mouse Rollover effect, user interaction, etc..
7. Build cartographic visualization for multiple datasets involving various countries of the world; states and districts in India etc.
8. Perform EDA on Wine Quality Data Set.
9. Use a case study on a data set and apply the various EDA and visualization techniques and present an analysis report.

TOTAL: 30 PERIODS

COURSE OUTCOMES

Upon Completion of the course, the students will be able to:

CO1: Understand the fundamentals of exploratory data analysis.

CO2: Implement the data visualization using Matplotlib.

CO3: Perform univariate data exploration and analysis.

CO4: Apply bivariate data exploration and analysis.

CO5: Use Data exploration and visualization techniques for multivariate and time series data

CO-PO Mapping Matrix:

Course Outcome	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	3	1	3	3	-	-	-	-	2	3	3	3	2	2	2
CO2	2	2	2	1	1	-	-	-	3	2	3	1	3	1	3
CO3	2	1	2	1	1	-	-	-	3	2	1	2	2	2	1
CO4	2	2	2	1	-	-	-	-	1	2	1	3	1	3	2
CO5	2	1	1	2	1	-	-	-	3	2	1	2	2	2	3
AVG.	**2**	**1**	**2**	**2**	**1**	**-**	**-**	**-**	**2**	**2**	**2**	**2**	**2**	**2**	**2**

EVALUATION OF LAB REPORT:

Evaluation Parameters	Marks Allotted	Marks Awarded
Aim & Hardware / Software Required:	10	
Design & Circuit Diagram / Algorithm & Flowchart:	30	
Observation & Calculation / Program:	30	
Graph / Output & Result:	20	
Viva Voce:	10	
Total:	100	

AD3301 DATA EXPLORATORY AND VISUALIZATION

Exp No: 1	Installation of Python
Date:	

AIM

To Install the Data Analysis and Visualization tool: Python.

INTRODUCTION TO PYTHON

Python is a popular programming language for data analysis and visualization due to its extensive libraries and packages designed for these tasks. One of the most common libraries used for these purposes is Matplotlib, along with other libraries like NumPy, Pandas, and Seaborn. In this guide, I'll provide an introduction to Python for data analysis and visualization and outline the installation steps for these essential libraries.

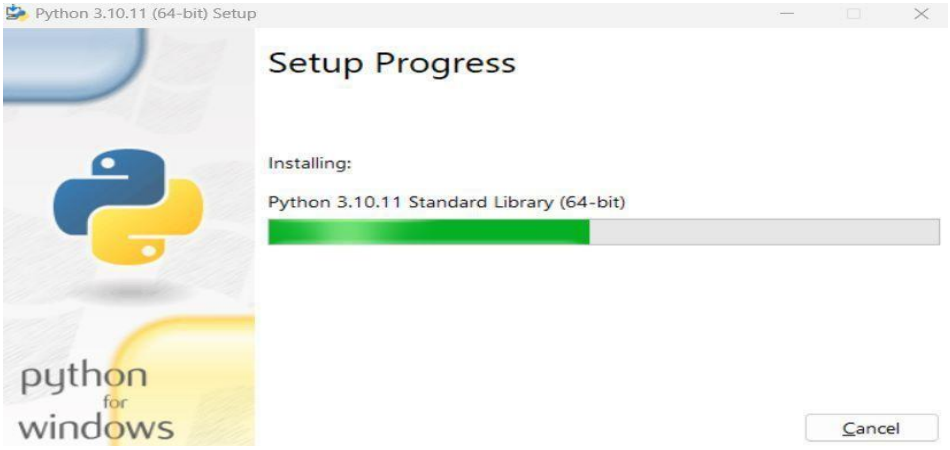
INSTALLATION STEPS OF PYTHON IDE

The step-by-step process to install the Python IDE was provided with images in the below content. To download Python on your system, you can use the following steps

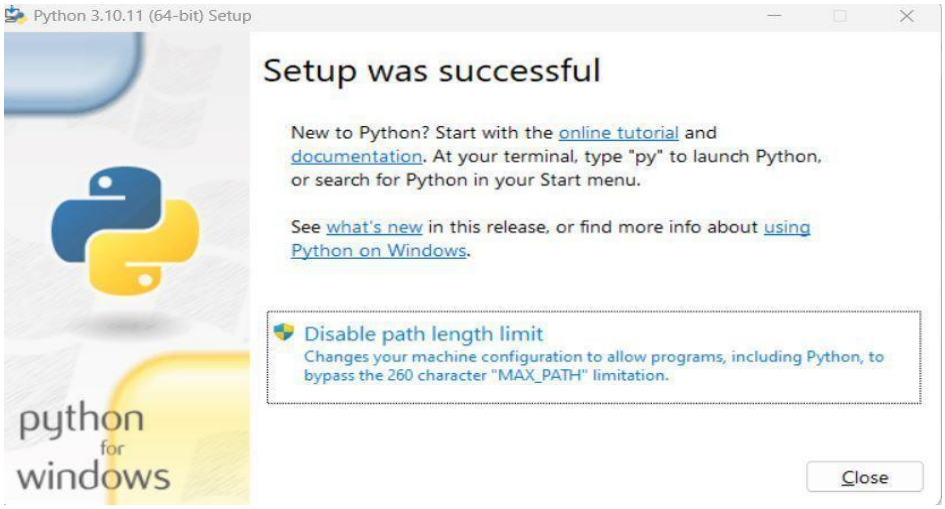
Step1: Visit the official page for Python https://www.python.org/downloads/ on the Windows operating system. Locate a reliable version of Python 3, preferably version 3.10.11, which was used in testing this tutorial. Choose the correct link for your device from the options provided: either Windows installer (64-bit) or Windows installer (32-bit) and proceed to download the executable file

Step2: Once you have downloaded the installer, open the .exe file, such as python-3.10.11- amd64.exe, by double-clicking it to launch the Python installer. Choose the option to Install the launcher for all users by checking the corresponding checkbox, so that all users of the computer can access the Python launcher application. Enable users to run Python from the command line by checking the Add python.exe to PATH checkbox.

After Clicking the Install Now Button the setup will start installing Python on your Windows system. You will see a window like this



Step3: After completing the setup. Python will be installed on your Windows system. You will see a successful message.



Step4: Close the window after successful installation of Python. You can check if the installation of Python was successful by using either the command line or the Integrated Development Environment (IDLE), which you may have installed. To access the command line, click on the Start menu and type "cmd" in the search bar. Then click on Command Prompt.

python –version

```
C:\Users\ashub>python --version
Python 3.10.11
```

You can also check the version of Python by opening the IDLE application. Go to Start and enter IDLE in the search bar and then click the IDLE app, for example, IDLE (Python 3.10.11 64-bit). If you can see the Python IDLE window then you are successfully able to download and installed Python on Windows.

test.py

```
1  my_file = open("C:/Documents/Python/binaryf.dll", "wb+")
2  message = "Hello Python"
3  file_encode = message.encode("ASCII")
4  my_file.write(file_encode)
5  my_file.seek(0)
6  bdata = my_file.read()
7  print("Binary Data:", bdata)
8  ntext = bdata.decode("ASCII")
9  print("Normal data:", ntext)
10
```

Installation Steps of libraries:

Before you start, you need to install Python on your computer. You can download the latest Python installer for your operating system from the official Python website (https://www.python.org/).

Once Python is installed, follow these steps to set up the necessary libraries:

1.Install NumPy and Pandas:

Open your command prompt or terminal and use the following commands to install NumPy and Pandas using pip (Python's package manager):

```bash
pip install numpy
pip install pandas
```

2.Install Matplotlib:

Use the following command to install Matplotlib:

```bash
pip install matplotlib
```

3.Install Seaborn:

To install Seaborn, use the following command:

```bash
pip install seaborn
```

4.Verify Installation:

After the installation is complete, you can verify it by opening a Python shell and importing the libraries:

```
```python
import numpy
import pandas
import matplotlib
import seaborn # If you installed Seaborn
```
```

5.IDE or Text Editor:

You can use an Integrated Development Environment (IDE) like Jupyter Notebook, Anaconda, or a text editor like Visual Studio Code to write and run your data analysis and visualization code.

Now you're ready to start using Python for data analysis and visualization. You can import your data, manipulate it using Pandas, and create visualizations with Matplotlib or Seaborn to gain insights and present your findings effectively.

RESULT

Thus, the Installation of Python has been successfully installed and explored.

Exp No: 2	Perform Exploratory Data Analysis on Email Data Set
Date:	

AIM

To perform exploratory data analysis (EDA) on email data set to export all your emails as a dataset, import them inside a pandas data frame, visualize them and get different insights from the data.

ALGORITHM

Step 1: Load Data

- Read the email dataset from a CSV file into a Pandas DataFrame.
- Display the first few rows of the DataFrame and its information.

Step 2: Feature Selection

- Remove rows with missing values in the "text" column.
- Separate the features (words) from the target variable (spam).
- Use mutual information for feature selection, selecting the top 1500 features.
- Create a new DataFrame with the selected features.

Step 3: Train-Test Split

- Split the data into training and testing sets (80% train, 20% test).
- Display the shapes of the train and test sets.

Step 4: Train a Naive Bayes Model

- Create a Multinomial Naive Bayes model.
- Fit the model using the training data.
- Predict probabilities and labels for the test data.
- Calculate and display the accuracy of the model.

Step 5: Plot ROC-AUC Curve

- Compute the ROC curve and AUC score.
- Plot the ROC curve with AUC score.

Step 6: Plot Confusion Matrix

- Define example predicted and true labels.
- Create a confusion matrix using the true and predicted labels.
- Plot the confusion matrix using a heatmap.

PROGRAM

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_csv('D:/SIBIYA/DEV LAB/emails.csv')
print(df.head().info)

# Feature Selection
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import mutual_info_classif
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, roc_curve, auc, confusion_matrix
from sklearn.svm import SVC

# Assuming the column with words is named "text"
df = df.dropna(subset=["text"])

# Separate the features (words) and the target variable (spam)
X = df.drop(["Email No.", "Prediction"], axis=1)  # Exclude Email_no. and spam columns
y = df["Prediction"]

# Perform feature selection using mutual information
selector = SelectKBest(score_func=mutual_info_classif, k=1500)  # Select top 1500 features
X_selected = selector.fit_transform(X, y)

# Get the selected feature names
selected_feature_names = X.columns[selector.get_support()].tolist()

# Create a new dataframe with the selected features
df_selected = df[["Email No.", "Prediction"] + selected_feature_names]
# Print the shape of the new dataframe
print("New dataframe shape:", df_selected.shape)
# Perform train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Print the shape of the train and test sets
print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
print("y_train shape:", y_train.shape)
print("y_test shape:", y_test.shape)

# Fit the model on the training data
model = MultinomialNB()
model.fit(X_train, y_train)

# Predict probabilities for the test data
probs = model.predict_proba(X_test)
# Predict labels for the test data
predicted_labels = model.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, predicted_labels)
print("Accuracy:", accuracy)

# Plotting ROC-AUC Curve # Compute the ROC curve and AUC score
fpr, tpr, thresholds = roc_curve(y_test, probs[:, 1])
auc_score = auc(fpr, tpr)
print("AUC Score:", auc_score)

# Plot the ROC curve
plt.figure()
plt.plot(fpr, tpr, label='ROC curve (area = %0.2f)' % auc_score)
plt.plot([0, 1], [0, 1], 'k--')  # Diagonal line indicating random chance
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.show()

# Plotting Confusion Matrix # Example predicted labels
predicted_labels = np.array(['spam', 'ham', 'spam', 'ham', 'spam'])
```

```
# Example true labels
true_labels = np.array(['spam', 'ham', 'ham', 'ham', 'spam'])
# Define the classes and the order of the confusion matrix
classes = ['spam', 'ham']

# Create confusion matrix
cm = confusion_matrix(true_labels, predicted_labels, labels=classes)

# Plot confusion matrix
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=classes, yticklabels=classes)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()

# Using Support Vector Classifier (SVC) from scikit-learn # Example feature vectors (X) and labels (y)
X = np.array([[1, 2], [3, 4], [5, 6], [7, 8], [9, 10]])
y = np.array(['spam', 'ham', 'spam', 'ham', 'spam'])

# Create an instance of SVC classifier
model = SVC()
model.fit(X, y)

# Predict labels for the same data
predicted_labels = model.predict(X)

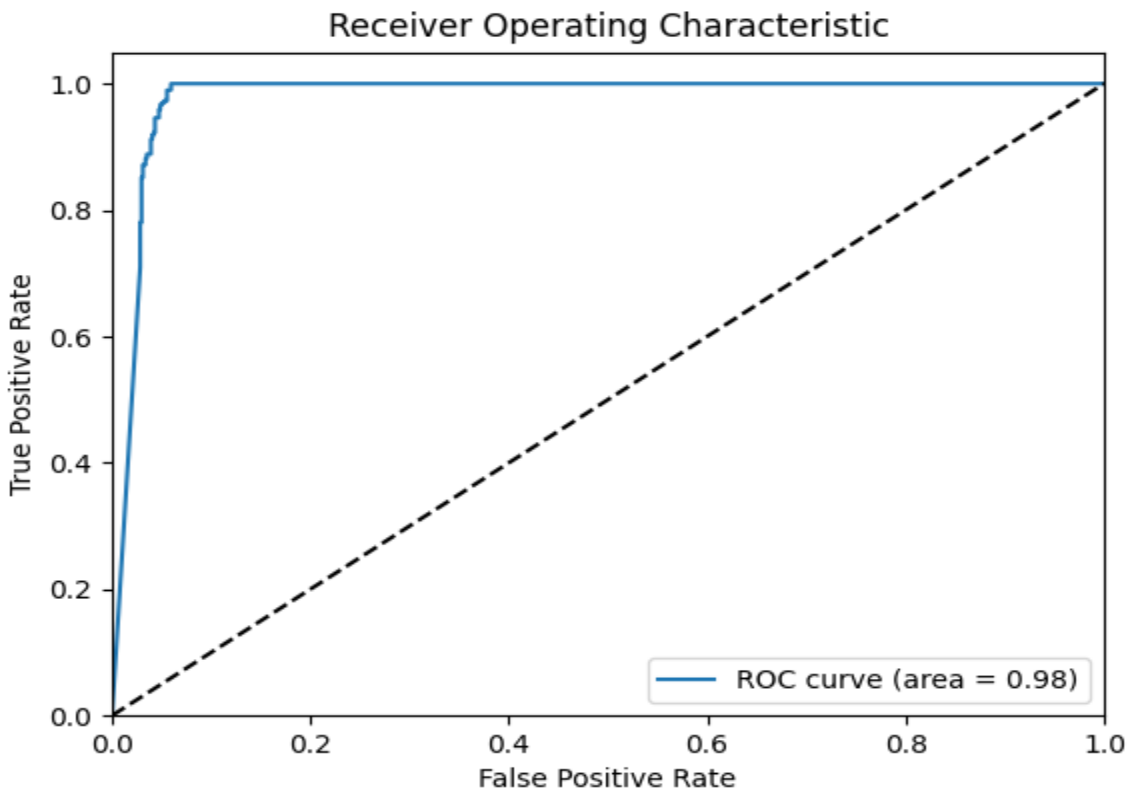
# Calculate accuracy
accuracy = accuracy_score(y, predicted_labels)
print("Accuracy:", accuracy)
# Confusion Matrix # Create confusion matrix
cm = confusion_matrix(y, predicted_labels)

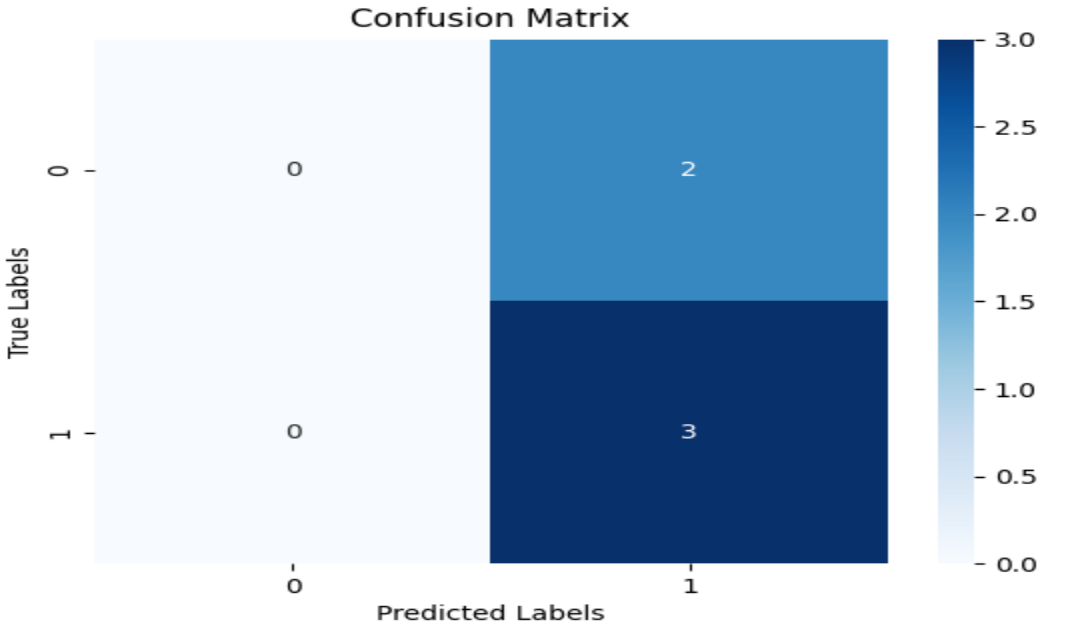
# Plot confusion matrix
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```

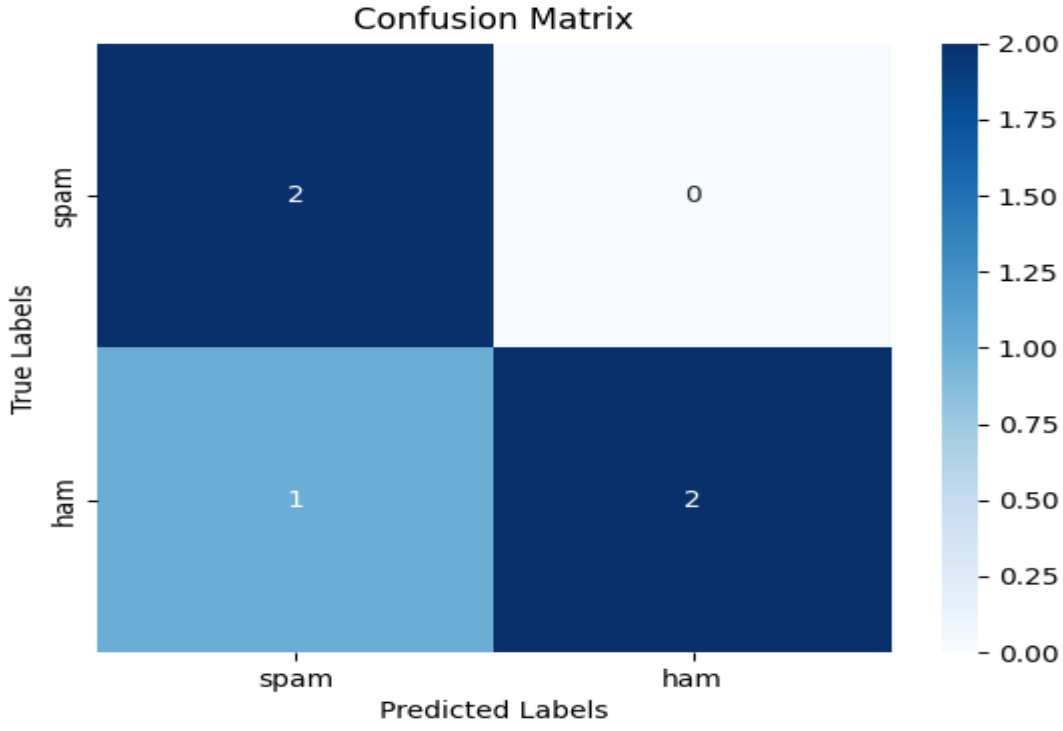
OUTPUT

```
<bound method DataFrame.info of   Email No.  the  to  ect  and  ...  military  allowing  ff  dry  Prediction
0   Email 1    0   0    1    0  ...         0         0   0    0           0
1   Email 2    8  13   24    6  ...         0         0   1    0           0
2   Email 3    0   0    1    0  ...         0         0   0    0           0
3   Email 4    0   5   22    0  ...         0         0   0    0           0
4   Email 5    7   6   17    1  ...         0         0   1    0           0

[5 rows x 3002 columns]>
New dataframe shape: (5172, 1502)
X_train shape: (4137, 3000)
X_test shape: (1035, 3000)
y_train shape: (4137,)
y_test shape: (1035,)
Accuracy: 0.9545893719806763
AUC Score: 0.9793548623047947
```







RESULT

Thus, the exploratory data analysis (EDA) on email data set to export all your emails as a dataset are executed and verified successfully.

Exp No: 3	Numpy arrays, Pandas data frames, Basic plots using Matplotlib
Date:	

AIM

To work with Numpy arrays, pandas data frames, Basic plots using Matplotlib.

Numpy Arrays

ALGORITHM

Step1: Start the program.

Step2: Import numpy module.

Step3: Print the basic characteristics and operations of array.

Step4: Stop the program.

PROGRAM

Write a NumPy program to create a null vector of size 10 and update sixth value to 11

```
import numpy as np
x = np.zeros(10)
print(x)
print("Update sixth value to 11")
x[6] = 11
print(x)
```

OUTPUT

```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Update sixth value to 11
[ 0.  0.  0.  0.  0.  0. 11.  0.  0.  0.]
```

Write a NumPy program to create a 3x3 matrix with values ranging from 2 to 10

```
import numpy as np
x =  np.arange(2, 11).reshape(3,3)
print(x)
```

OUTPUT

```
[[ 2  3  4]
 [ 5  6  7]
[ 8  9 10]]
```

Write a NumPy program to convert an array to a float type

```
import numpy as np
x= np.array([[12, 12], [2, 7], [25, 36]])
```

```
print("Original array elements:")
print(x)
print("Convert to float values :")
print(x.astype(float))
```

OUTPUT

```
Original array elements:
[[12 12]
 [ 2  7]
 [25 36]]
Convert to float values :
[[12. 12.]
 [ 2.  7.]
 [25. 36.]]
```

PANDAS

ALGORITHM

Step1: Start the program.

Step2: import numpy and pandas module.

Step3: Create a dataframe using the dictionary.

Step4: Print the output.

Step5: Stop the program.

PROGRAM

Write a Pandas program to get the powers of an array values element-wise.

```
import pandas as pd
import numpy as np
df = pd.DataFrame({'X':[78,85,96,80,86], 'Y':[84,94,89,83,86],'Z':[86,97,96,72,83]});
print("Original array")
print(df)
print("First array elements raised to powers from second array, element-wise:")
print(np.power(df, 2))
```

OUTPUT

```
Original array
   X  Y  Z
0 78 84 86
1 85 94 97
2 96 89 96
3 80 83 72
4 86 86 83
First array elements raised to powers from second array, element-wise:
    X    Y    Z
0 6084 7056 7396
1 7225 8836 9409
2 9216 7921 9216
3 6400 6889 5184
4 7396 7396 6889
```

Write a Pandas program to select the specified columns and rows from a given data frame.

Select 'name' and 'score' columns in rows 1, 3, 5, 6 from the following data frame. exam_data, 'score', 'attempts', 'qualify', labels.

```
import pandas as pd
```

```
import numpy as np
exam_data = {
   'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],
   'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
   'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
   'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
df = pd.DataFrame(exam_data, index=labels)
print("Select specific columns and rows:")
print(df.iloc[[1, 3, 5, 6], [1, 3]])
```

OUTPUT

```
Select specific columns and rows:
  score qualify
b   9.0      no
d   NaN      no
f  20.0     yes
g  14.5     yes
```

Write a Pandas program to count the number of rows and columns of a DataFrame. Sample Python dictionary data and list labels:

```
import pandas as pd
import numpy as np
exam_data  = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin',
'Jonas'],
      'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
      'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
      'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
df = pd.DataFrame(exam_data , index=labels)
total_rows=len(df.axes[0])
total_cols=len(df.axes[1])
print("Number of Rows: "+str(total_rows))
print("Number of Columns: "+str(total_cols))
```

OUTPUT

```
Number of Rows: 10
Number of Columns: 4
```

MATPLOTLIB

ALGORITHM

Step1: Start the program

Step2: import Matplotlib module

Step3: Create a Basic plots using Matplotlib

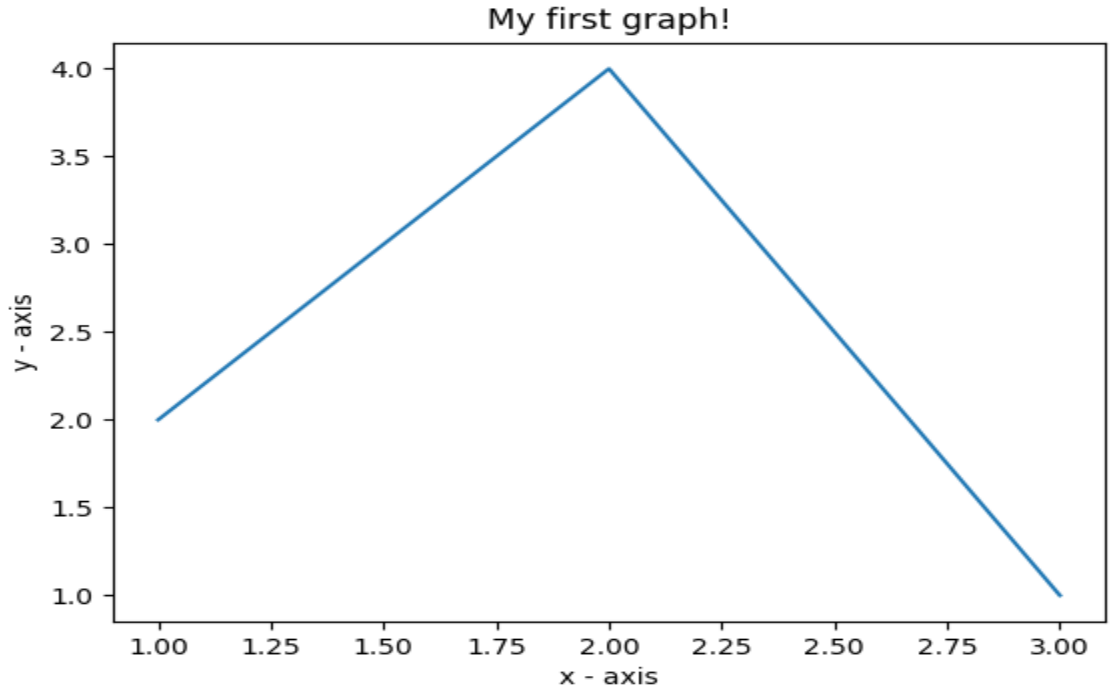
Step4: Print the output

Step5: Stop the program

PROGRAM

MATPLOTLIB-1

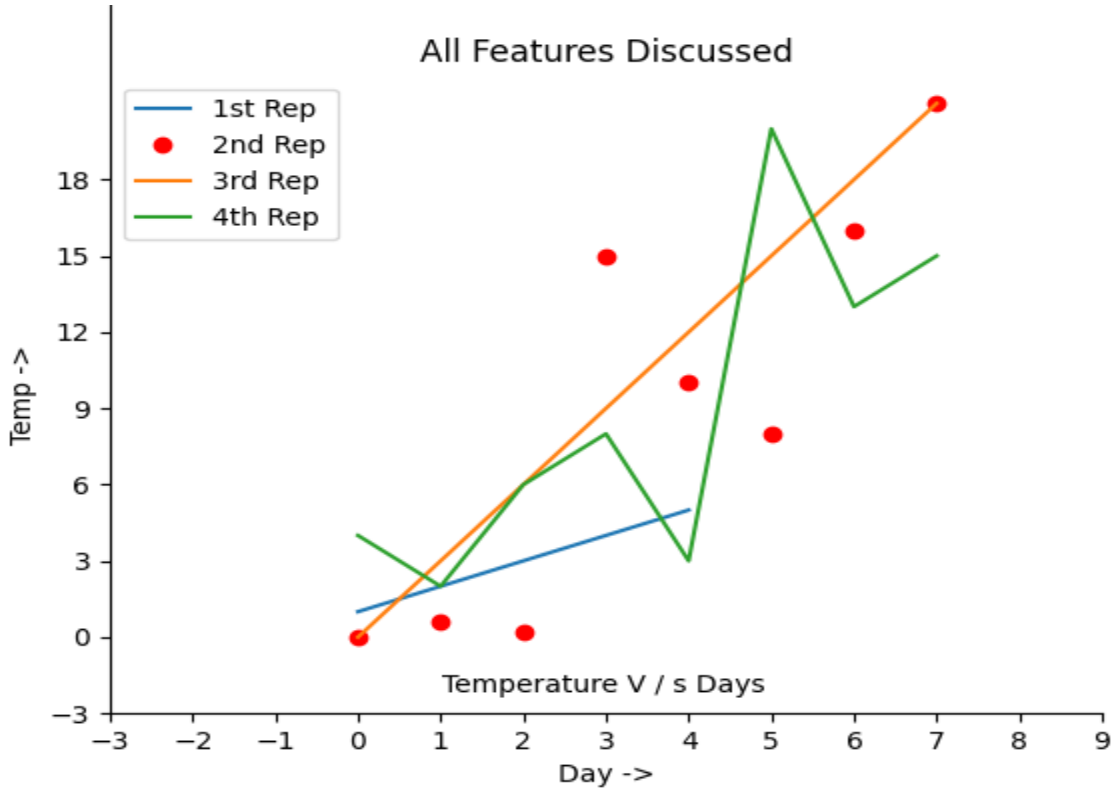
```
import matplotlib.pyplot as plt
x = [1,2,3]
y = [2,4,1]
plt.plot(x, y)
plt.xlabel('x - axis')
plt.ylabel('y - axis')
plt.title('My first graph!')
plt.show()
```



MATPLOTLIB-2

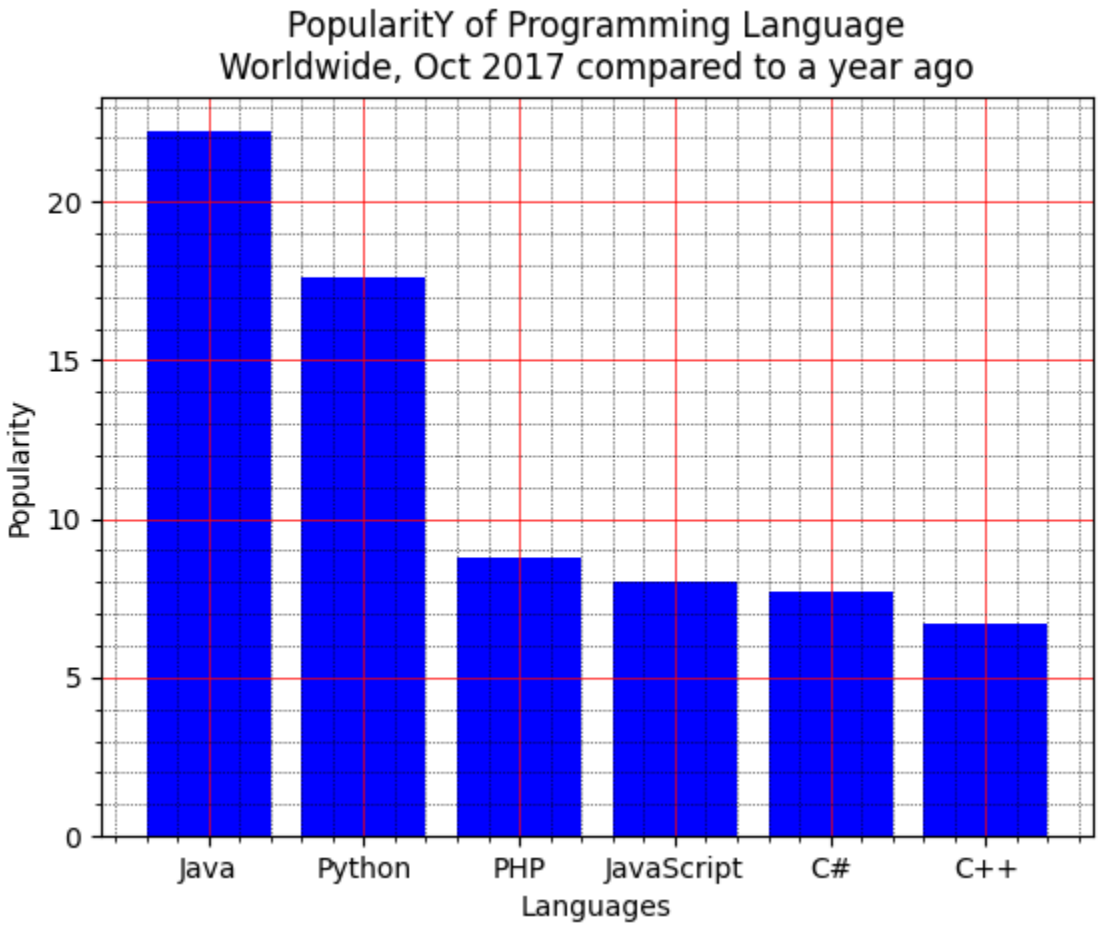
```
import matplotlib.pyplot as plt
a = [1, 2, 3, 4, 5]
b = [0, 0.6, 0.2, 15, 10, 8, 16, 21]
plt.plot(a)
# o is for circles and r is
# for red
plt.plot(b, "or")
plt.plot(list(range(0, 22, 3)))
# naming the x-axis
plt.xlabel('Day ->')
# naming the y-axis
plt.ylabel('Temp ->')
c = [4, 2, 6, 8, 3, 20, 13, 15]
plt.plot(c, label = '4th Rep')
# get current axes command
ax = plt.gca()
# get command over the individual
# boundary line of the graph body
ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)
# set the range or the bounds of
# the left boundary line to fixed range
ax.spines['left'].set_bounds(-3, 40)
# set the interval by which
# the x-axis set the marks
plt.xticks(list(range(-3, 10)))
# set the intervals by which y-axis
# set the marks
plt.yticks(list(range(-3, 20, 3)))
# legend denotes that what color
# signifies what
ax.legend(['1st Rep', '2nd Rep', '3rd Rep', '4th Rep'])
# annotate command helps to write
# ON THE GRAPH any text xy denotes
# the position on the graph
plt.annotate('Temperature V / s Days', xy = (1.01, -2.15))
# gives a title to the Graph
```

```
plt.title('All Features Discussed')
plt.show()
```



MATPLOTLIB-3

```
import matplotlib.pyplot as plt
x = ['Java', 'Python', 'PHP', 'JavaScript', 'C#', 'C++']
popularity = [22.2, 17.6, 8.8, 8, 7.7, 6.7]
x_pos = [i for i, _ in enumerate(x)]
plt.bar(x_pos, popularity, color='blue')
plt.xlabel("Languages")
plt.ylabel("Popularity")
plt.title("PopularitY of Programming Language\n" + "Worldwide, Oct 2017 compared to a year ago")
plt.xticks(x_pos, x)
# Turn on the grid
plt.minorticks_on()
plt.grid(which='major', linestyle='-', linewidth='0.5', color='red')
# Customize the minor grid
plt.grid(which='minor', linestyle=':', linewidth='0.5', color='black')
plt.show()
```



RESULT

Thus, the Numpy arrays, Pandas data frames, Basic plots using Matplotlib is executed and verified successfully.

Exp No: 4	Explore various variable and row filters in R for cleaning data.
Date:	

AIM

To Explore various variable and row filters in R for cleaning data. Apply various plot features in R on sample data sets and visualize.

ALGORITHM

Step 1: Data Loading and Exploration:

- Import necessary libraries (numpy, pandas, os, etc.).
- Load the email dataset from a CSV file.
- Explore the dataset's basic properties (e.g., shape and a sample email).

Step 2: Extract Email Fields:

- For each email message, extract specific fields (e.g., Date, Subject, X-Folder, X-From, X-To, and body) using email parsing.

Step 3: Data Transformation and Preprocessing:

- Convert the date field to a consistent format (e.g., "dd-mm-YYYY HH:MM:SS").
- Extract the last folder name from the 'X-Folder' field and convert it to lowercase.
- Replace empty values in 'Subject' and 'X-To' with NaN.
- Drop rows with missing values.

Step 4: Data Exploration and Visualization:

- Explore the dataset, e.g., the count of unique folders.
- Visualize the top 20 folders and top 20 email sender employees using bar plots.

Step 5: Data Cleaning:

- Drop unnecessary columns (e.g., 'file', 'message', 'date', 'X-From', 'X-To', 'employee').

Step 6: Save Cleaned Data:

- Save the cleaned dataset to a new CSV file.

PROGRAM

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
 for filename in filenames:
 print(os.path.join(dirname, filename))
import multiprocessing
import seaborn as sns
import email
import matplotlib.pyplot as plt
df = pd.read_csv("'D:/SIBIYA/DEV LAB/emails.csv")
print(df.head())
# get shape of the data
print(df.shape)
# a sample email
print(df.loc[1]['message'])
# transform the email into correct format
message = df.loc[1]['message']
e = email.message_from_string(message)
print(e.items())
# get date
print(e.get('Date'))
# show message body
print(e.get_payload())
# now we add those fields into our 'df' dataframe
def get_field(field, messages):
 column = []
 for message in messages:
 e = email.message_from_string(message)
 column.append(e.get(field))
 return column
df['date'] = get_field("Date", df['message'])
df['subject'] = get_field("Subject", df['message'])
df['X-Folder'] = get_field("X-Folder", df['message'])
df['X-From'] = get_field("X-From", df['message'])
df['X-To'] = get_field("X-To", df['message'])
print(df.head(3))
```

```
def body(messages):
 column = []
 for message in messages:
 e = email.message_from_string(message)
 column.append(e.get_payload())
 return column
df['body'] = body(df['message'])
print(df.head(3))
print(df['file'][:10])
def employee(file):
 column = []
 for string in file:
 column.append(string.split("/")[0])
 return column
df['employee'] = employee(df['file'])
print(df.head(3))
print("number of folders: ", df.shape[0])
print("number of unique folders: ", df['X-Folder'].unique().shape[0])
unique_emails = pd.DataFrame(df['X-Folder'].value_counts())
unique_emails.reset_index(inplace=True)
unique_emails.columns = ['folder_name', 'count']
# top 20 folders
print(unique_emails.iloc[:20,:])
plt.figure(figsize=(10,6))
sns.barplot(x='count', y='folder_name', data=unique_emails.iloc[:20, :], palette="Blues_d")
plt.title("Top 20 folders")
plt.xlabel("Count")
plt.ylabel("Folder_Name")
plt.show()
top_20 = pd.DataFrame(df['employee'].value_counts()[:20])
top_20.reset_index(inplace=True)
top_20.columns = ["Employee_name", "Counts"]
print(top_20)
plt.figure(figsize=(10,8))
sns.barplot(y="Employee_name", x="Counts", data=top_20, palette="Blues_d")
plt.title("Top 20 highest email sender employee")
plt.xlabel("Count")
plt.ylabel("Employee_name")
```

```
plt.show()
import datetime
from dateutil import parser
# this is sample example
x = parser.parse("Fri, 4 May 2001 13:51:00 -0700 (PDT)")
print(x.strftime("%d-%m-%Y %H:%M:%S"))
def change_type(dates):
 column = []
 for date in dates:
 column.append(parser.parse(date).strftime("%d-%m-%Y %H:%M:%S"))
 return column
df['date'] = change_type(df['date'])
print(df.head(2))
print(df['X-Folder'][0])
# we only want last folder name
print(df['X-Folder'][0].split("\\")[-1])
def preprocess_folder(folders):
 column = []
 for folder in folders:
 if (folder is None or folder == ""):
 column.append(np.nan)
 else:
 column.append(folder.split("\\")[-1].lower())
 return column
df['X-Folder'] = preprocess_folder(df['X-Folder'])
print(df.head(2))
# count unique folders
print("Unique Foldes: ", len(df['X-Folder'].unique()))
# view some of them
print(df['X-Folder'].unique()[0:20])
def replace_empty_with_nan(subject):
 column = []
 for val in subject:
 if (val == ""):
 column.append(np.nan)
 else:
 column.append(val)
 return column
```

```
df['subject'] = replace_empty_with_nan(df['subject'])
df['X-To'] = replace_empty_with_nan(df['X-To'])
# calculate percentage of missing values
miss = df.isnull().sum()
miss = miss[miss>0]
miss = miss / df.shape[0]
print(miss)
# drop missing value rows
print(df.dropna(axis=0, inplace=True))
print(df.isnull().sum(), df.shape)
cols_to_drop = ['file','message','date','X-From','X-To','employee']
print(df.head(3))
print(df.drop(cols_to_drop, axis=1, inplace=True))
print(df.head())
# save the data
df.to_csv("'D:/SIBIYA/DEV LAB/cleaned_data.csv", index=False)
```

OUTPUT

```
                  file                                        message
0    allen-p/_sent_mail/1.  Message-ID: <18782981.1075855378110.JavaMail.e...
1   allen-p/_sent_mail/10.  Message-ID: <15464986.1075855378456.JavaMail.e...
2  allen-p/_sent_mail/100.  Message-ID: <24216240.1075855687451.JavaMail.e...
3 allen-p/_sent_mail/1000.  Message-ID: <13505866.1075863688222.JavaMail.e...
4 allen-p/_sent_mail/1001.  Message-ID: <30922949.1075863688243.JavaMail.e...
(517401, 2)
Message-ID: <15464986.1075855378456.JavaMail.evans@thyme>
Date: Fri, 4 May 2001 13:51:00 -0700 (PDT)
From: phillip.allen@enron.com
To: john.lavorato@enron.com
Subject: Re:
Mime-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
X-From: Phillip K Allen
X-To: John J Lavorato <John J Lavorato/ENRON@enronXgate@ENRON>
X-cc:
X-bcc:
X-Folder: \Phillip_Allen_Jan2002_1\Allen, Phillip K.\'Sent Mail
X-Origin: Allen-P
X-FileName: pallen (Non-Privileged).pst

Traveling to have a business meeting takes the fun out of the trip.  Especially if you have to prepare a presentation.  I would suggest holding the business plan meetings here then ta

As far as the business meetings, I think it would be more productive to try and stimulate discussions across the different groups about what is working and what is not.  Too often the

My suggestion for where to go is Austin.  Play golf and rent a ski boat and jet ski's.  Flying somewhere takes too much time.
```

```
[('Message-ID', '<15464986.1075855378456.JavaMail.evans@thyme>'), ('Date', 'Fri, 4 May 2001 13:51:00 -0700 (PDT)'), ('From', 'phillip.allen@enron.com'), ('To', 'john.lavorato@enrc
Fri, 4 May 2001 13:51:00 -0700 (PDT)
Traveling to have a business meeting takes the fun out of the trip.  Especially if you have to prepare a presentation.  I would suggest holding the business plan meetings here the

As far as the business meetings, I think it would be more productive to try and stimulate discussions across the different groups about what is working and what is not.  Too ofter

My suggestion for where to go is Austin.  Play golf and rent a ski boat and jet ski's.  Flying somewhere takes too much time.

                  file  ...                                      X-To
0    allen-p/_sent_mail/1.  ...          Tim Belden <Tim Belden/Enron@EnronXGate>
1   allen-p/_sent_mail/10.  ...  John J Lavorato <John J Lavorato/ENRON@enronXg...
2  allen-p/_sent_mail/100.  ...                               Leah Van Arsdall

[3 rows x 7 columns]
                  file  ...                                      body
0    allen-p/_sent_mail/1.  ...                      Here is our forecast\n\n
1   allen-p/_sent_mail/10.  ...  Traveling to have a business meeting takes the...
2  allen-p/_sent_mail/100.  ...                  test successful.  way to go!!!

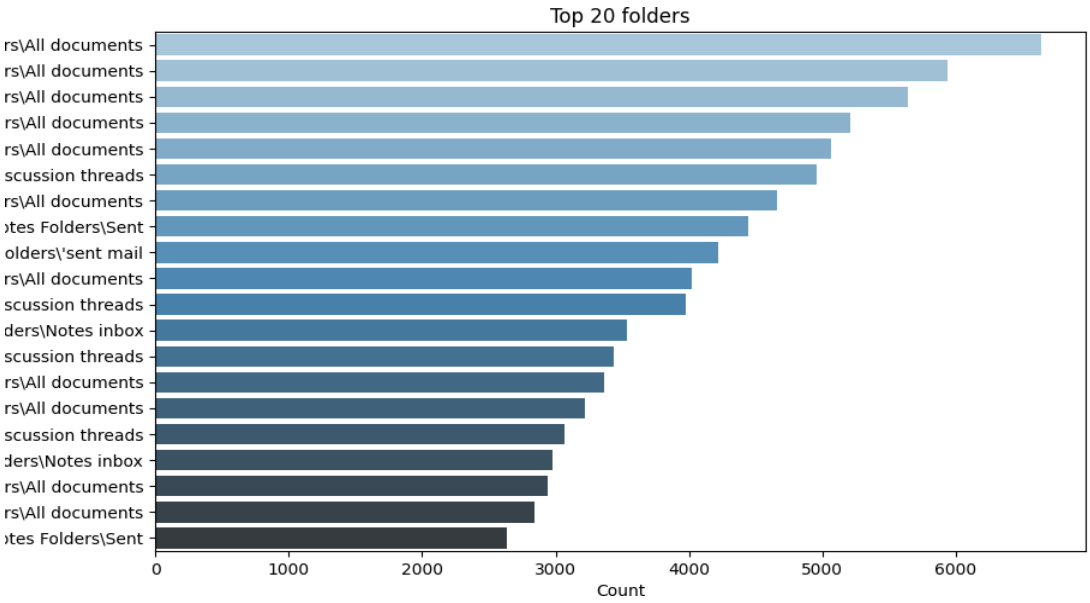
[3 rows x 8 columns]
0      allen-p/_sent_mail/1.
1     allen-p/_sent_mail/10.
2    allen-p/_sent_mail/100.
3   allen-p/_sent_mail/1000.
4   allen-p/_sent_mail/1001.
5   allen-p/_sent_mail/1002.
6   allen-p/_sent_mail/1003.
7   allen-p/_sent_mail/1004.
8    allen-p/_sent_mail/101.
9    allen-p/_sent_mail/102.
Name: file, dtype: object
                         file  ... employee
0    allen-p/_sent_mail/1.  ...  allen-p
1   allen-p/_sent_mail/10.  ...  allen-p
2  allen-p/_sent_mail/100.  ...  allen-p

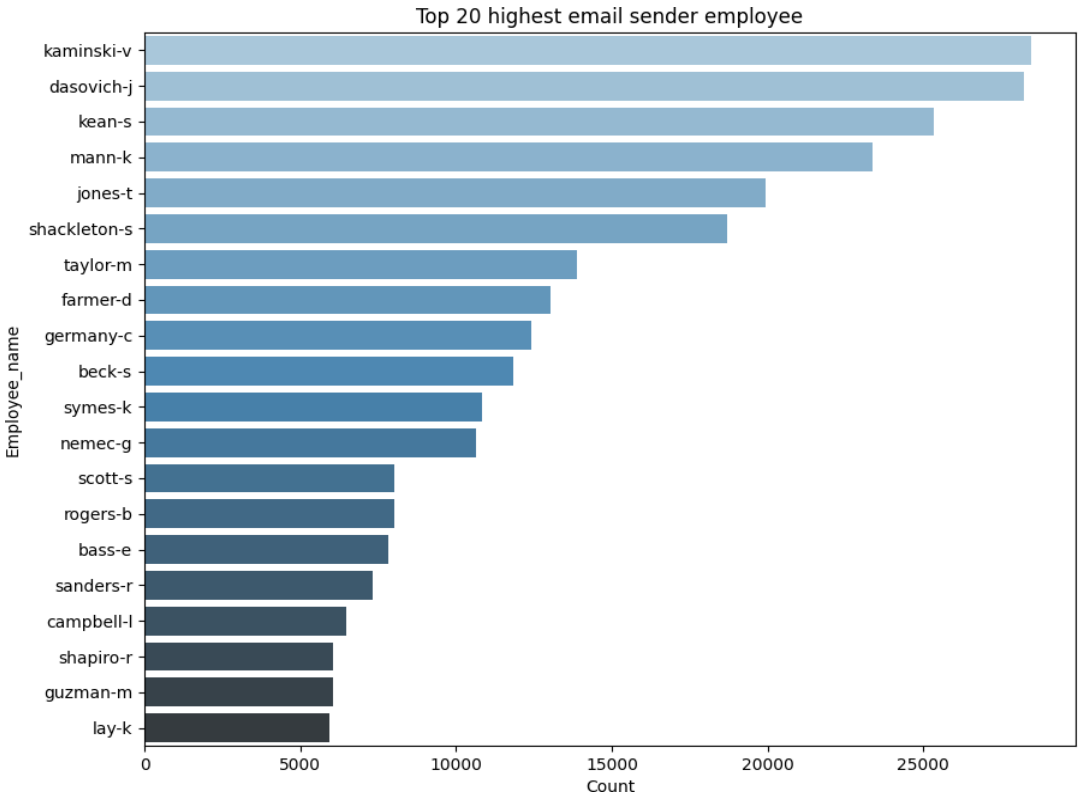
[3 rows x 9 columns]
number of folders:  517401
number of unique folders:  5336
                                                folder_name  count
0     \Kay_Mann_June2001_1\Notes Folders\All documents   6639
1     \Tanya_Jones_Dec2000\Notes Folders\All documents   5934
2   \Jeff_Dasovich_June2001\Notes Folders\All docu...   5637
3   \Sara_Shackleton_Dec2000_June2001_1\Notes Fold...   5211
4   \Vincent_Kaminski_Jun2001_1\Notes Folders\All ...   5066
5   \Kay_Mann_June2001_2\Notes Folders\Discussion ...   4956
6   \Jeff_Dasovich_Dec2000\Notes Folders\All docum...   4660
7              \Kay_Mann_June2001_3\Notes Folders\Sent   4440
8        \Kay_Mann_June2001_4\Notes Folders\'sent mail   4220
9   \Mark_Taylor _Dec_2000\Notes Folders\All docum...   4022
10  \Vincent_Kaminski_Jun2001_2\Notes Folders\Disc...   3980
11  \Jeff_Dasovich_June2001\Notes Folders\Notes inbox   3535
12  \Steven_Kean_June2001_4\Notes Folders\Discussi...   3434
13  \Tanya_Jones_June2001\Notes Folders\All documents   3365
14    \kate symes 6-27-02\Notes Folders\All documents   3221
15  \kate symes 6-27-02\Notes Folders\Discussion t...   3065
16     \Tanya_Jones_Dec2000\Notes Folders\Notes inbox   2975
17  \Sara_Shackleton_Dec2000_June2001_2\Notes Fold...   2944
18  \Darren_Farmer_Dec2000\Notes Folders\All docum...   2838
19             \Tanya_Jones_Dec2000\Notes Folders\Sent   2633
```

```
    Employee_name  Counts
0      kaminski-v   28465
1      dasovich-j   28234
2           kean-s   25351
3           mann-k   23381
4          jones-t   19950
5    shackleton-s   18687
6          taylor-m   13875
7         farmer-d   13032
8        germany-c   12436
9           beck-s   11830
10         symes-k   10827
11         nemec-g   10655
12         scott-s    8022
13        rogers-b    8009
14          bass-e    7823
15       sanders-r    7329
16     campbell-l     6490
17      shapiro-r     6071
18       guzman-m     6054
19           lay-k     5937
04-05-2001 13:51:00
                          file   ... employee
0    allen-p/_sent_mail/1.   ...   allen-p
1   allen-p/_sent_mail/10.   ...   allen-p

[2 rows x 9 columns]
\Phillip_Allen_Jan2002_1\Allen, Phillip K.\'Sent Mail
'Sent Mail
                          file   ... employee
0   allen-p/_sent_mail/1.  ...  allen-p
1  allen-p/_sent_mail/10.  ...  allen-p

[2 rows x 9 columns]
Unique Foldes:  1782
["'sent mail" 'all documents' 'contacts' 'deleted items'
 'discussion threads' 'inbox' 'notes inbox' 'sent items' 'sent' 'straw'
 '2000 conference' 'active international' 'avaya' 'bmc' 'bridge'
 'bristol babcock' 'colleen koenig' 'compaq' 'computer associates'
 'continental airlines']
subject     0.037083
X-Folder    0.000056
X-From      0.000056
X-To        0.017690
dtype: float64
None
file        0
message     0
date        0
subject     0
X-Folder    0
X-From      0
X-To        0
body        0
employee    0
dtype: int64 (489236, 9)
                         file  ... employee
1    allen-p/_sent_mail/10.  ...  allen-p
2   allen-p/_sent_mail/100.  ...  allen-p
4  allen-p/_sent_mail/1001.  ...  allen-p
```





RESULT

Thus, the python program is to Explore various variable and row filters in R for cleaning data. Apply various plot features in R on sample data sets and visualize has been done and executed successfully.

Exp No: 5	Perform Time Series Analysis and Apply the Visualization Techniques
Date:	

AIM

To perform Time Series Analysis and apply the various visualization techniques.

ALGORITHM

Step 1: Data Loading and Preparation:

- Import necessary libraries, including pandas, numpy, and matplotlib.

- Read a CSV file into a pandas DataFrame, specifying the "Date" column as the index and parsing dates.

- Print the first five rows of the dataset to inspect the data.

- Drop the 'Unnamed: 0' column from the DataFrame.

Step 2: Data Visualization:

- Plot the 'Volume' column as a line chart using matplotlib.

- Show the plot using `plt.show()`.

- Plot all columns in the DataFrame as separate subplots in a 4x4 grid.

- Show the subplots using `plt.show()`.

Step 3: Resampling and Monthly Averaging:

- Resample the time series data to a monthly frequency ('M') and calculate the mean for each month.

- Store the resampled data in a new DataFrame (e.g., `df_month`).

Step 4: Plotting with Subplots:

- Create a figure and axis for plotting.

- Plot a bar graph for the 'Volume' column for the year 2016 and beyond using the resampled data.

- Specify the width and alignment of the bars.

- Show the plot using `plt.show()`.

Step 5: Time Series Analysis and Moving Averages:

- Calculate the two-period difference of the 'Low' and 'High' columns and plot them separately.

- Calculate a rolling mean of the 'Open' column with a window size of 50 and plot it.

- Calculate the 'Change' column as the ratio of the 'Close' column to previous value and plot.

- Plot the 'Change' column for the year 2017.

- Show each plot using `plt.show()`.

PROGRAM

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from matplotlib import pyplot

from pandas import read_csv

import seaborn as sns

# reading the dataset using read_csv

df = pd.read_csv(r"D:\SIBIYA\DEV LAB\stock_data.csv", parse_dates=True, index_col="Date")

# displaying the first five rows of dataset

df.head()

print(df)

# Box Plot in Time Series

df.drop(columns='Unnamed: 0', inplace=True)

df['Date'] = pd.to_datetime(df['Date'])

# extract year from date column

df["Year"] = df["Date"].dt.year

# box plot grouped by year

sns.boxplot(data=df, x="Year", y="Open")

plt.show()

# Plotting Line plot for Time Series data.

df['Volume'].plot()

plt.show()

# plot all other columns using a subplot
```

```
df.plot(subplots=True, figsize=(4, 4))

plt.show()

df.Low.diff(2).plot(figsize=(6, 6))

plt.show()

# Finding the trend in the "Open" # column using moving average method

window_size = 50

rolling_mean = df['Open'].rolling \

   (window_size).mean()

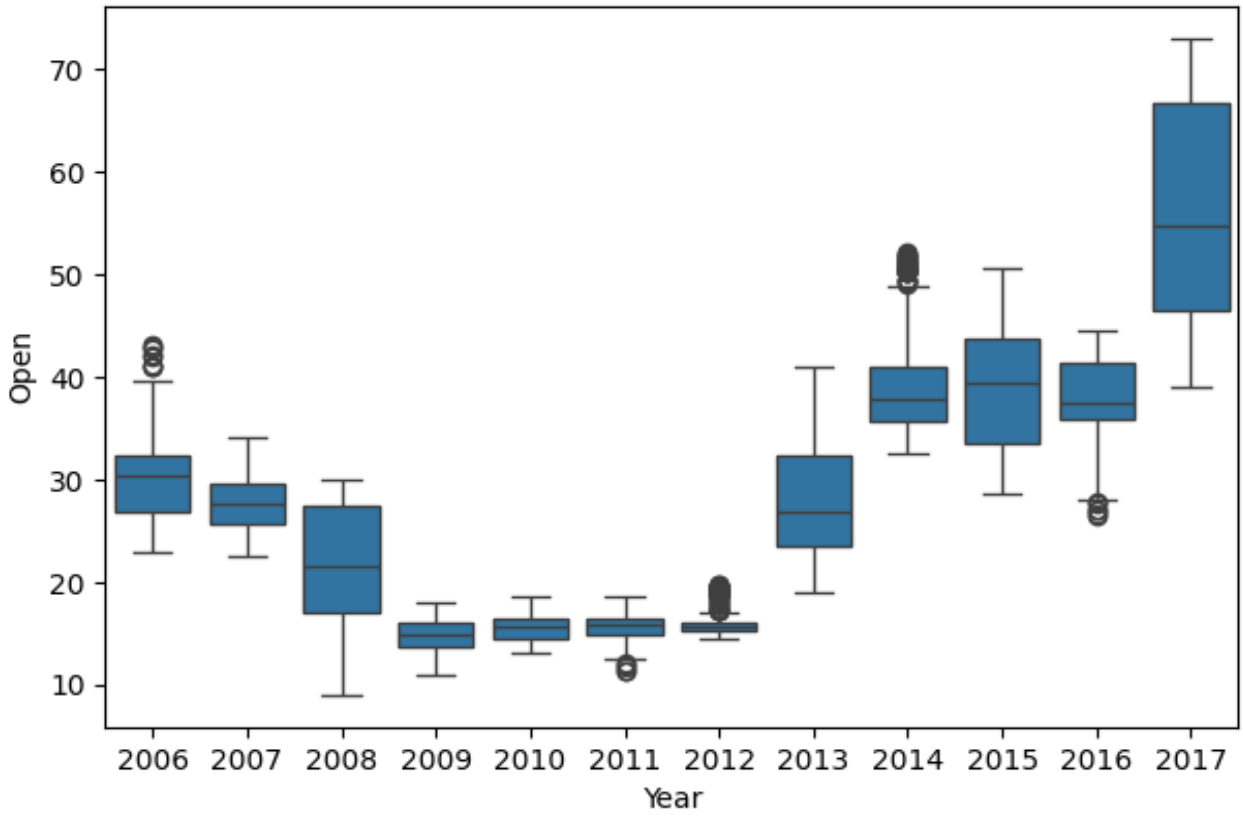
rolling_mean.plot()

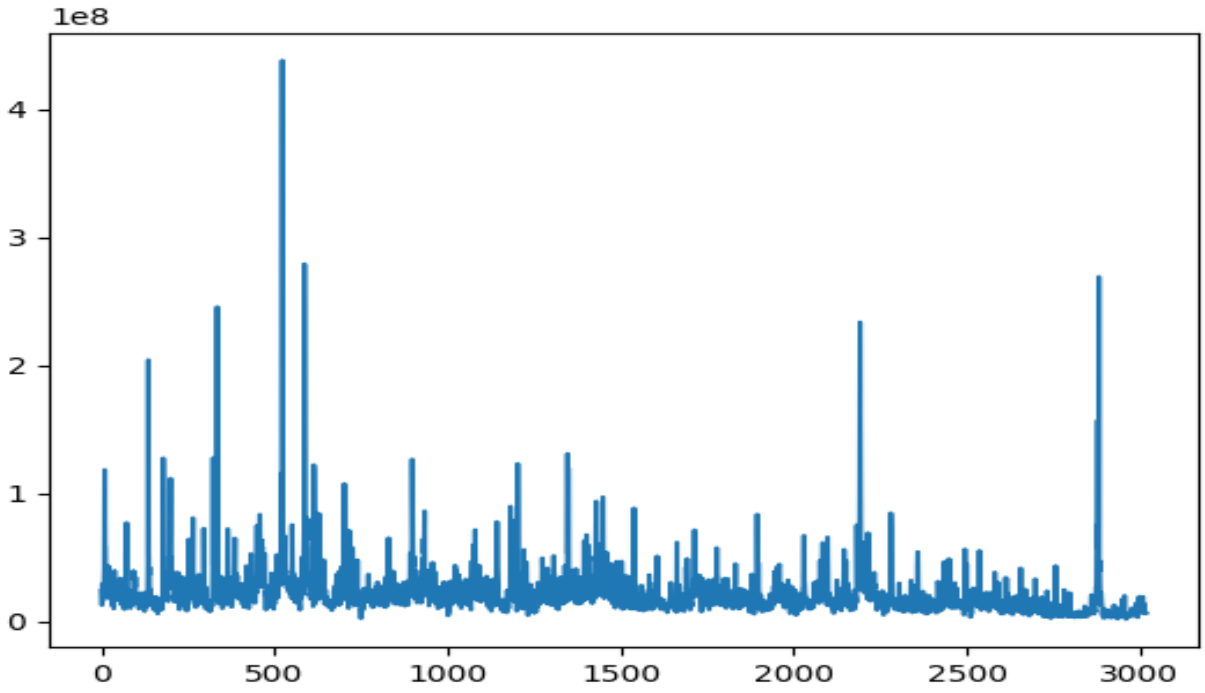
plt.show()
```

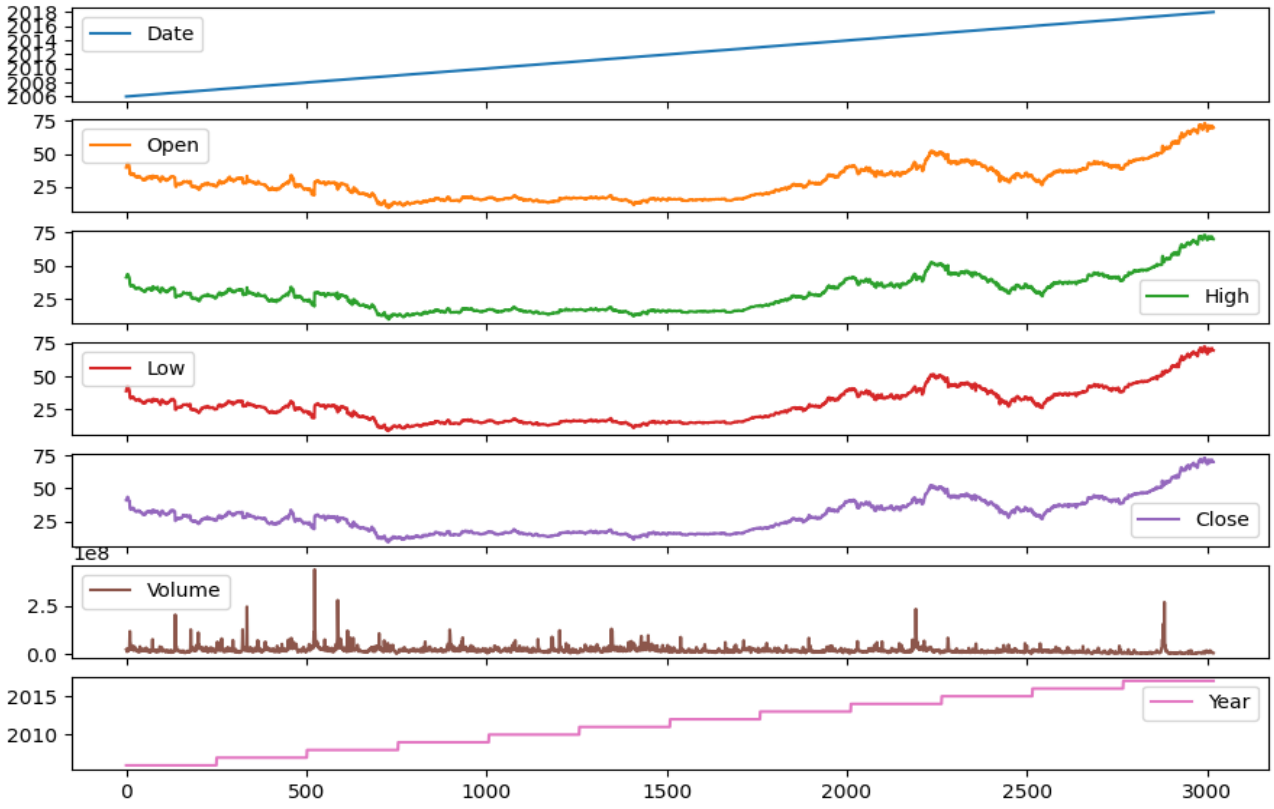
OUTPUT

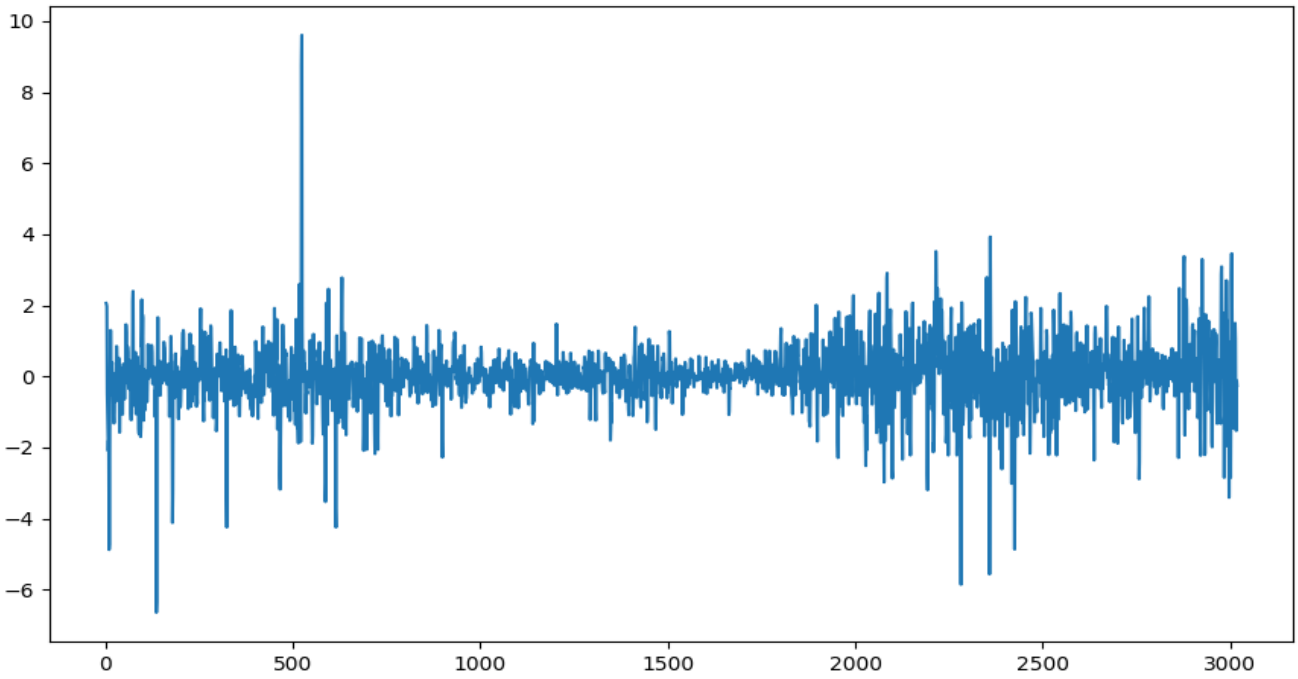
	Date	Unnamed: 0	Open	High	Low	Close	Volume	Name
0	1/3/2006	NaN	39.69	41.22	38.79	40.91	24232729	AABA
1	1/4/2006	NaN	41.22	41.90	40.77	40.97	20553479	AABA
2	1/5/2006	NaN	40.93	41.73	40.85	41.53	12829610	AABA
3	1/6/2006	NaN	42.88	43.57	42.80	43.21	29422828	AABA
4	1/9/2006	NaN	43.10	43.66	42.82	43.42	16268338	AABA
...	...	...	...	...	...	...	...	...
3014	12/22/2017	NaN	71.42	71.87	71.22	71.58	10979165	AABA
3015	12/26/2017	NaN	70.94	71.39	69.63	69.86	8542802	AABA
3016	12/27/2017	NaN	69.77	70.49	69.69	70.06	6345124	AABA
3017	12/28/2017	NaN	70.12	70.32	69.51	69.82	7556877	AABA
3018	12/29/2017	NaN	69.79	70.13	69.43	69.85	6613070	AABA

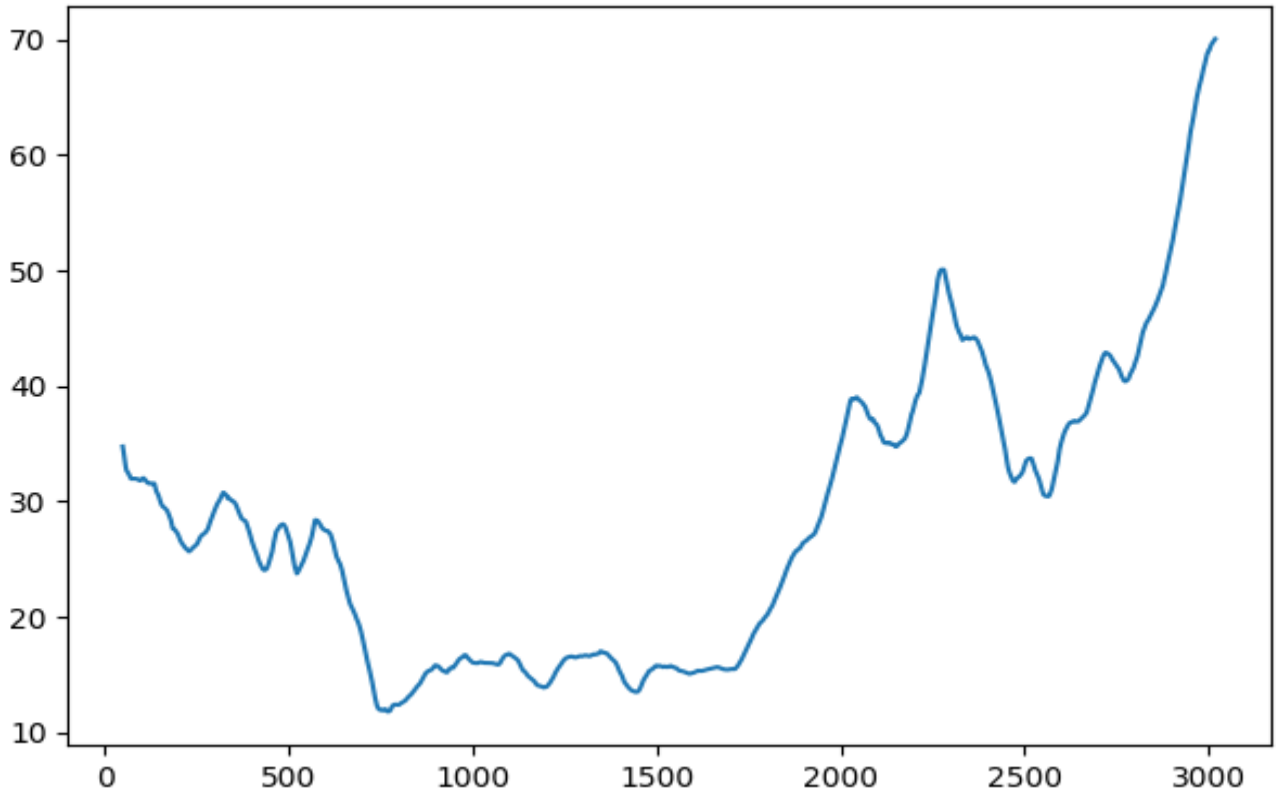
[3019 rows x 8 columns]


Open
70
60
50
40
30
20
10
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
Year


1e8
4
3
2
1
0
0
500
1000
1500
2000
2500
3000


Date
Open
High
Low
Close
1e8
Volume
Year
0
500
1000
1500
2000
2500
3000


10
8
6
4
2
0
−2
−4
−6
0
500
1000
1500
2000
2500
3000



RESULT

Thus, the Time Series Analysis and applying the various visualization techniques are executed and verified successfully.

Exp No: 6	Perform Data Analysis and Representation on a Map Data Sets
Date:	

AIM

To perform Data Analysis and representation on a Map using various Map data sets with Mouse Rollover effect, user interaction.

ALGORITHM

Step 1: Create a Map

- Create a folium map (m_1) with a specified location and zoom level.
- Save this map to an HTML file (map_1.html).

Step 2: Data Preparation

- Load crime data from a CSV file.
- Remove rows with missing location information.
- Filter the data to focus on major crimes in 2018.

Step 3: Create Maps with Markers

- Create a second map (m_2) using cartodbpositron tiles.
- Add markers to this map for daytime robberies.
- Save this map to an HTML file (map_2.html).

Step 4: Create a Map with Marker Clusters

- Create a third map (m_3) using cartodbpositron tiles.
- Add marker clusters to this map for daytime robberies.
- Save this map to an HTML file (map_3.html).

Step 5: Create a Map with Circles and Colors

- Create a fourth map (m_4) using cartodbpositron tiles.
- Define a function to determine circle colors based on the time of daytime robberies.

- Add circles with varying colors to represent the time of daytime robberies.

- Save this map to an HTML file (map_4.html).

Step 6: Create Heatmap and Choropleth Maps

- Create a fifth map (m_5) using cartodbpositron tiles.

- Add a heatmap layer to represent crime hotspots.

- Save this map to an HTML file (map_5.html).

- Load geographical boundary data for police districts.

- Calculate the number of crimes in each police district.

- Create a sixth map (m_6) using cartodbpositron tiles.

- Add a choropleth map based on the number of crimes in each police district.

- Save this map to an HTML file (map_6.html).

PROGRAM

```
import pandas as pd

import geopandas as gpd

import math

import matplotlib.pyplot as plt

import folium

from folium import Choropleth, Circle, Marker

from folium.plugins import HeatMap, MarkerCluster

# Create a map

m_1 = folium.Map(location=[42.32, -71.0589], tiles='openstreetmap', zoom_start=10)

# Display the map

m_1.save("D:/SIBIYA/DEV LAB/map_1.html")

# Load the data

crimes = pd.read_csv("D:/SIBIYA/DEV LAB/crime.csv", encoding='latin-1')

# Drop rows with missing locations

crimes.dropna(subset=['Lat', 'Long', 'DISTRICT'], inplace=True)

# Focus on major crimes in 2018

crimes = crimes[crimes.OFFENSE_CODE_GROUP.isin([

    'Larceny', 'Auto Theft', 'Robbery', 'Larceny From Motor Vehicle', 'Residential Burglary',

    'Simple Assault', 'Harassment', 'Ballistics', 'Aggravated Assault', 'Other Burglary',

    'Arson', 'Commercial Burglary', 'HOME INVASION', 'Homicide', 'Criminal Harassment',

    'Manslaughter'])]

crimes = crimes[crimes.YEAR >= 2018]

# Print the first five rows of the table
```

```
print(crimes.head())
daytime_robberies = crimes[((crimes.OFFENSE_CODE_GROUP == 'Robbery') & \
                    (crimes.HOUR.isin(range(9, 18))))]
# Create a map
m_2 = folium.Map(location=[42.32, -71.0589], tiles='cartodbpositron', zoom_start=13)
# Add points to the map
for idx, row in daytime_robberies.iterrows():
    Marker([row['Lat'], row['Long']]).add_to(m_2)
# Display the map
m_2.save("D:/SIBIYA/DEV LAB/map_2.html")
# Create the map
m_3 = folium.Map(location=[42.32, -71.0589], tiles='cartodbpositron', zoom_start=13)
# Add points to the map
mc = MarkerCluster()
for idx, row in daytime_robberies.iterrows():
    if not math.isnan(row['Long']) and not math.isnan(row['Lat']):
        mc.add_child(Marker([row['Lat'], row['Long']]))
m_3.add_child(mc)
# Display the map
m_3.save("D:/SIBIYA/DEV LAB/map_3.html")
# Create a base map
m_4 = folium.Map(location=[42.32, -71.0589], tiles='cartodbpositron', zoom_start=13)
def color_producer(val):
    if val <= 12:
```

```
        return 'forestgreen'
    else:
        return 'darkred'
# Add a bubble map to the base map
for i in range(0, len(daytime_robberies)):
    Circle(
        location=[daytime_robberies.iloc[i]['Lat'], daytime_robberies.iloc[i]['Long']],
        radius=20,
        color=color_producer(daytime_robberies.iloc[i]['HOUR'])).add_to(m_4)
# Display the map
m_4.save("D:/SIBIYA/DEV LAB/map_4.html")
```

OUTPUT

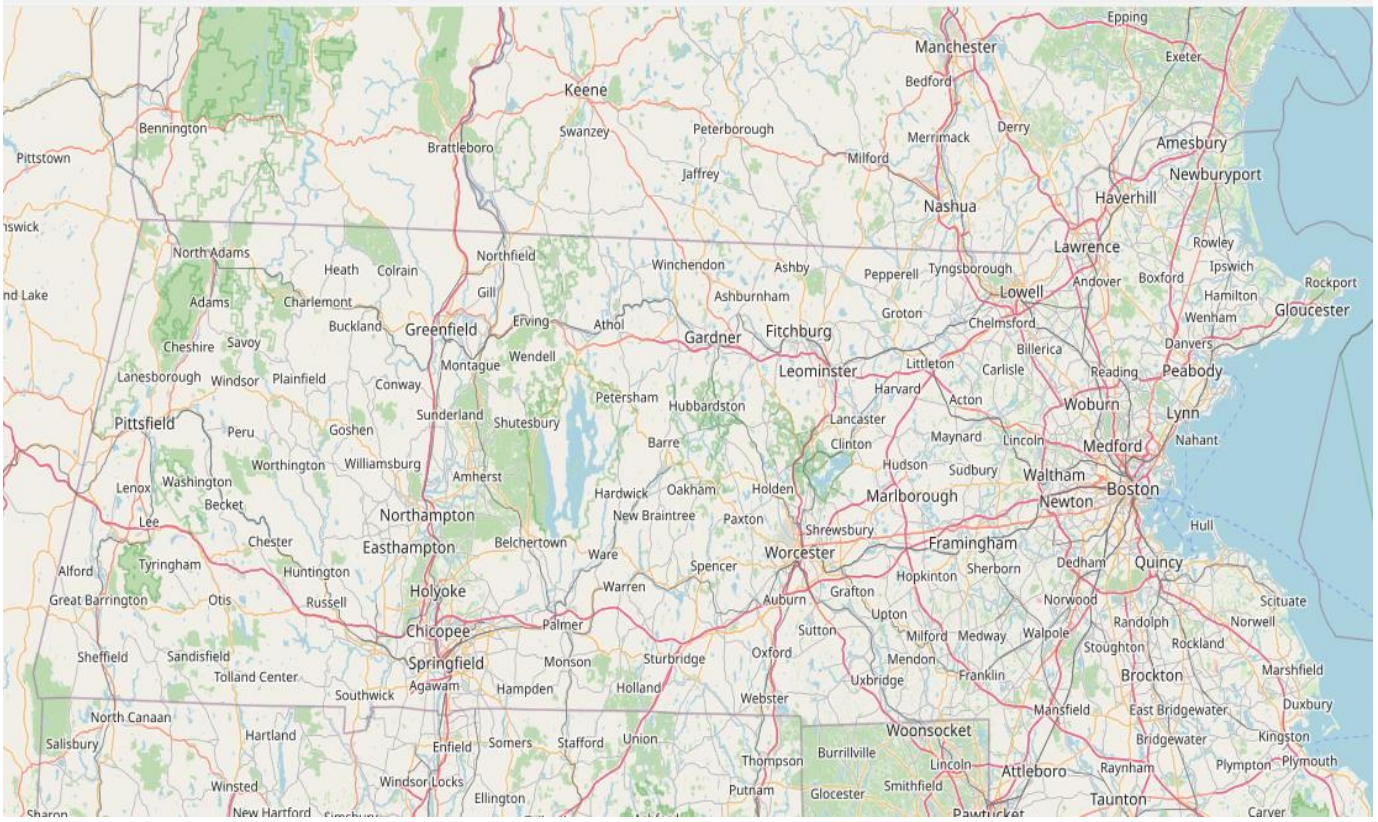
```
    INCIDENT_NUMBER  OFFENSE_CODE  ...       Long                    Location
0       I182070945           619  ... -71.139371  (42.35779134, -71.13937053)
6       I182070933           724  ... -71.082733  (42.30607218, -71.08273260)
8       I182070931           301  ... -71.070853  (42.33152148, -71.07085307)
19      I182070915           614  ... -71.068168  (42.32569490, -71.06816778)
24      I182070908           522  ... -71.093168  (42.33506218, -71.09316781)

[5 rows x 17 columns]
```

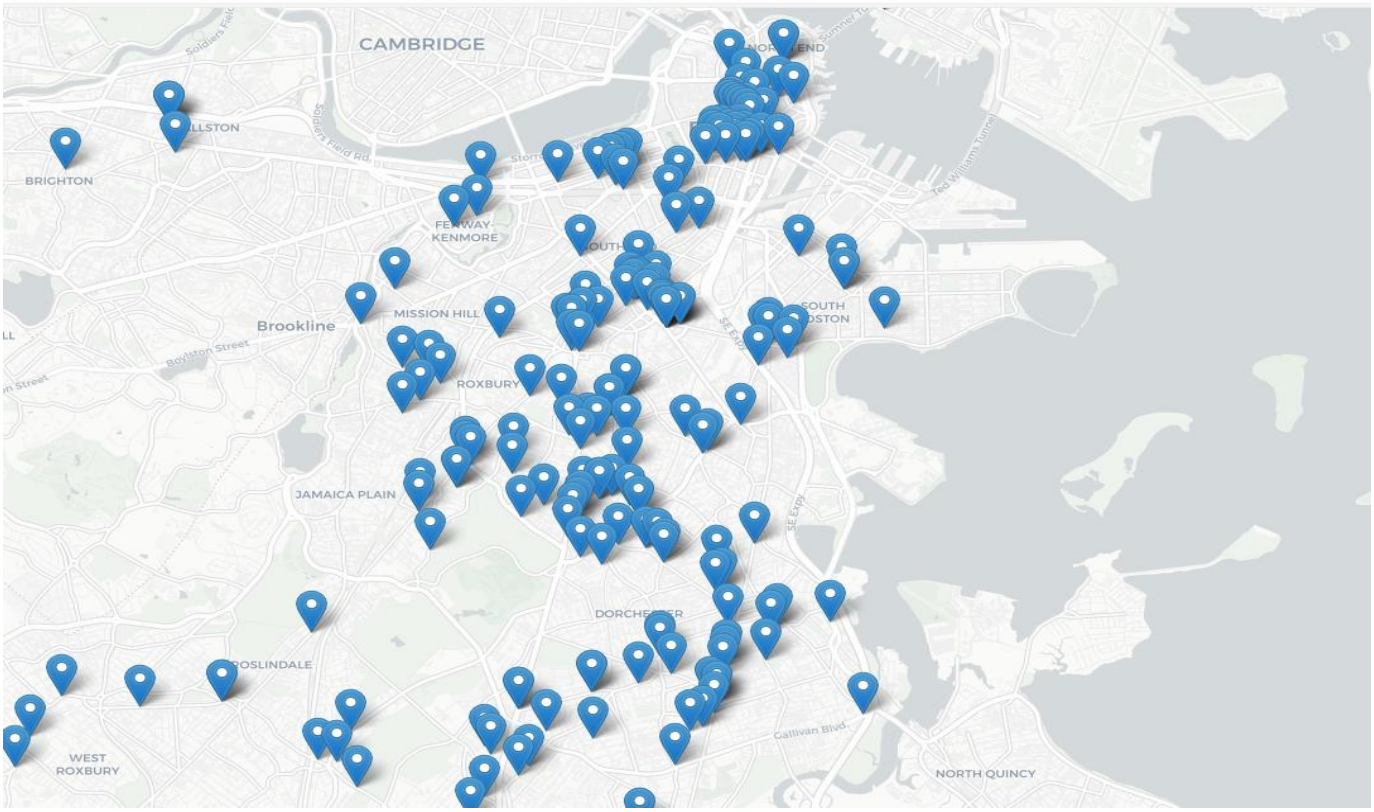
PC › New Volume (D:) › SIBIYA › MAP

Name	Date modified	Type	Size
map_1	12/12/2023 11:24 AM	Microsoft Edge H...	4 KB
map_2	12/12/2023 11:24 AM	Microsoft Edge H...	50 KB
map_3	12/12/2023 11:24 AM	Microsoft Edge H...	53 KB
map_4	12/12/2023 11:24 AM	Microsoft Edge H...	111 KB

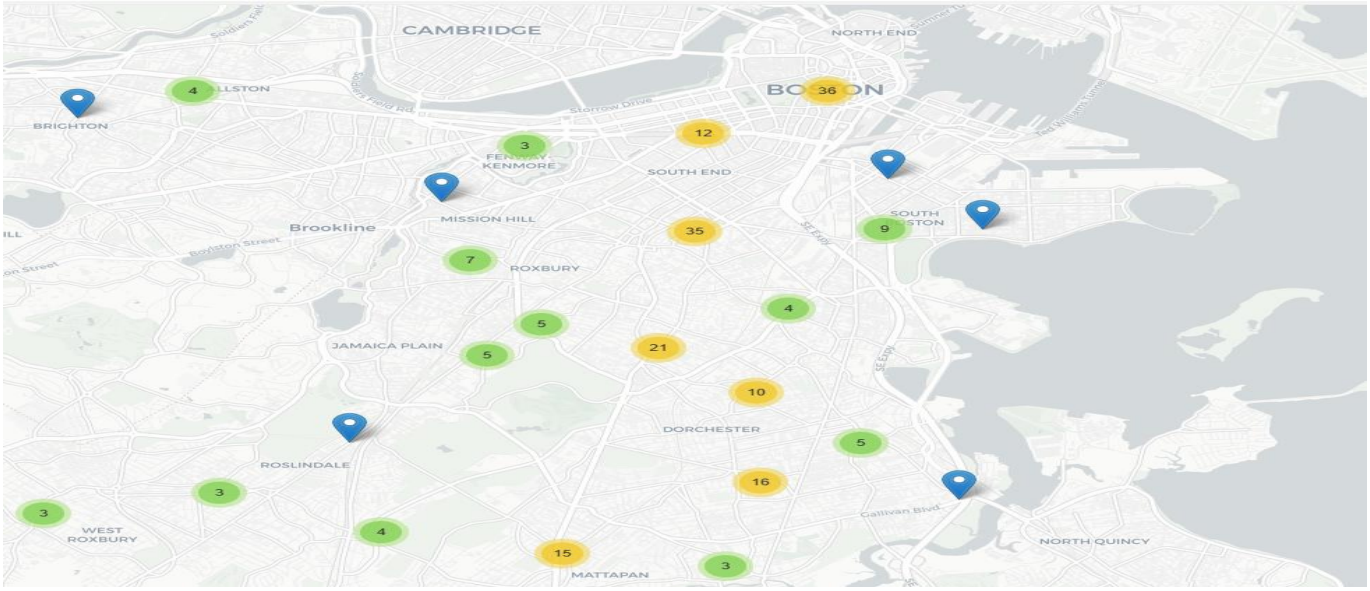
D:/SIBIYA/MAP/map_1.html

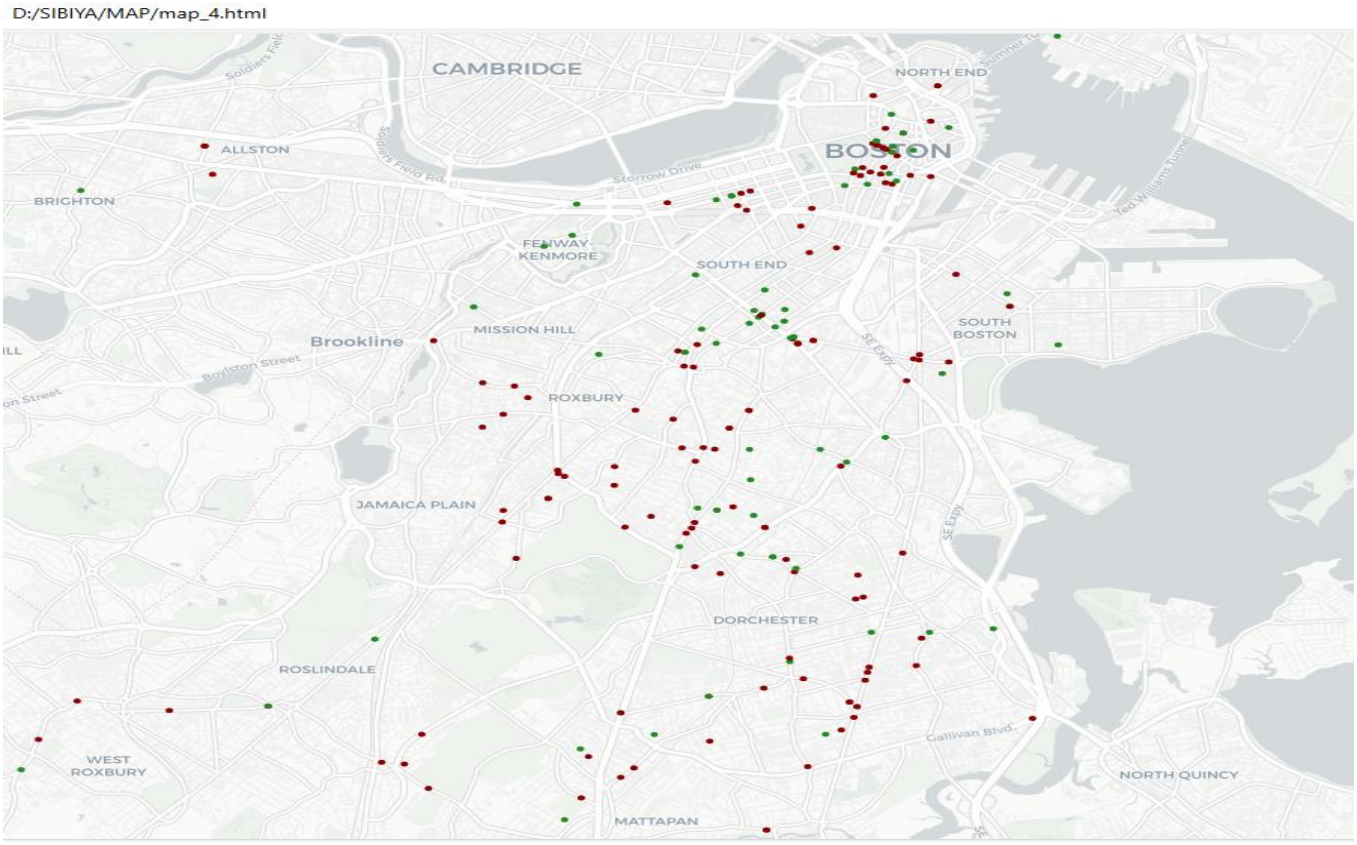


D:/SIBIYA/MAP/map_2.html



D:/SIBIYA/MAP/map_3.html





RESULT

Thus, the Data Analysis and representation on a Map using various Map data sets is executed and verified successfully.

Exp No: 7	Build Cartographic Visualization
Date:	

AIM

To build cartographic visualization for multiple datasets involving various countries of the world; states and districts in India.

ALGORITHM

Step 1: Data Preparation and Visualization Setup:

- Import necessary libraries including pandas, geopandas, folium, geodatasets, and matplotlib.
- Read volcano data from a CSV file into a pandas DataFrame.
- Select relevant columns from the DataFrame (Year, Name, Country, Latitude, Longitude, Type).
- Create point geometries using Latitude and Longitude columns.
- Create a GeoDataFrame with the selected columns and geometries.
- Load a world map using geopandas and set up a subplot for visualization.

Step 2: Visualization of Volcanoes on a World Map:

- Plot the world map as a base layer with reduced opacity.
- Overlay volcano data from the GeoDataFrame on the world map, color-coded by volcano type.
- Set a title for the map.

Step 3: Create Different Maps Using Folium:

- Create interactive maps using Folium and save them as HTML files with different tile layers.
- Maps are generated using Stamen Terrain, OpenStreetMap, and Stamen Toner tiles.

Step 4: Mark Volcano Locations on the Maps:

- Create a new Folium map with the "Stamen Terrain" tile layer for displaying volcano terrain.
- Iterate through the list of volcano coordinates and types from the GeoDataFrame.
- Assign a marker color based on the type of volcano.

- Add markers to the map for each volcano, with pop-up labels showing additional information.

Step 5: Heatmap Visualization:

- Create a new Folium map using "Cartodb dark_matter" tiles.

- Extract coordinates from the GeoDataFrame and store them in a list.

- Generate a heatmap layer using the extracted coordinates.

- Save the heatmap-enabled map as an HTML file.

PROGRAM

```
# Import Libraries

import pandas as pd

import geopandas

import folium

import geodatasets

import matplotlib.pyplot as plt

from folium import plugins

df1 = pd.read_csv("D:/SIBIYA/DEV LAB/volcano_data_2010.csv")

# Keep only relevant columns

df = df1.loc[:, ("Year", "Name", "Country", "Latitude", "Longitude", "Type")]

df.info()

# Create point geometries

geometry = geopandas.points_from_xy(df.Longitude, df.Latitude)

geo_df = geopandas.GeoDataFrame(

    df[["Year", "Name", "Country", "Latitude", "Longitude", "Type"]], geometry=geometry)

geo_df.head()

world = geopandas.read_file(geodatasets.get_path("naturalearth.land"))

df.Type.unique()

fig, ax = plt.subplots(figsize=(24, 18))

world.plot(ax=ax, alpha=0.4, color="grey")

geo_df.plot(column="Type", ax=ax, legend=True)

plt.title("Volcanoes")

# Stamen Terrain
```

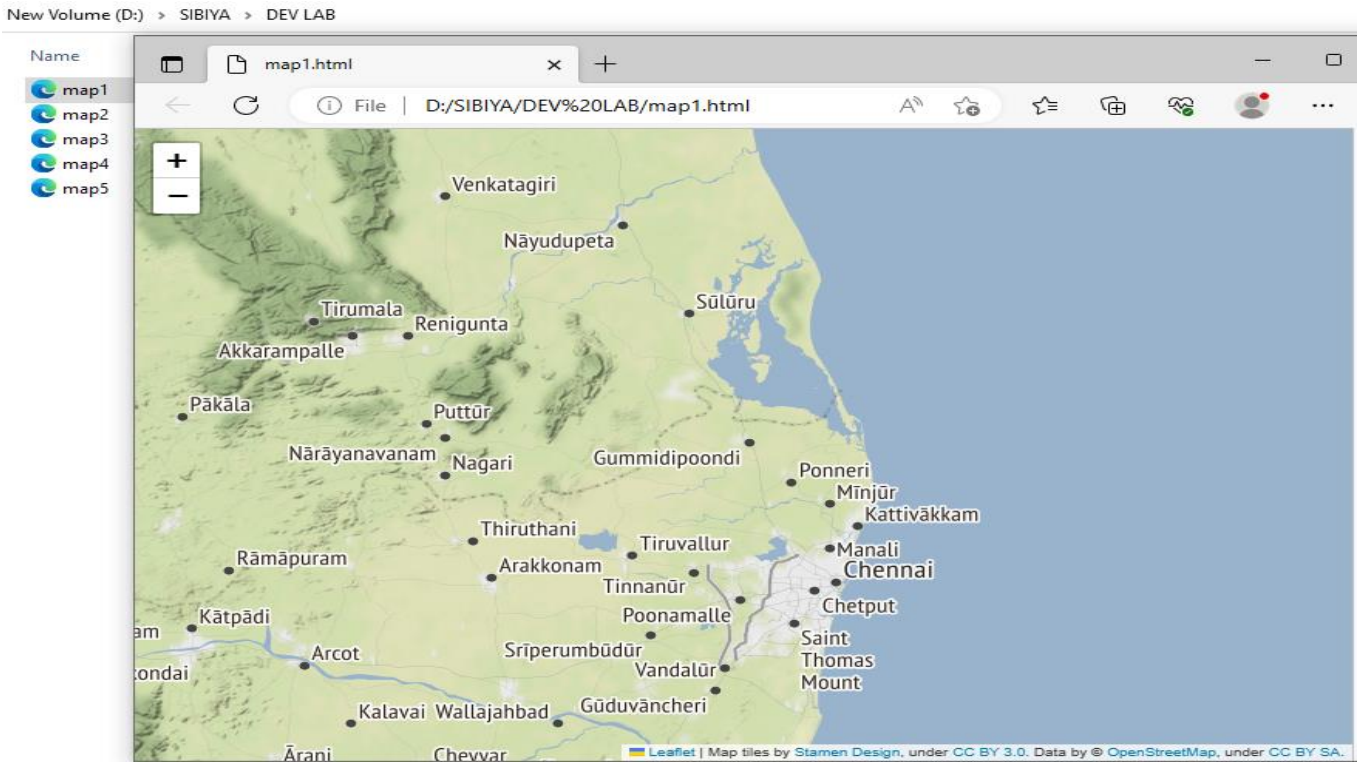
```
map = folium.Map(location=[13.406, 80.110], tiles="Stamen Terrain", zoom_start=9)
map.save("D:/SIBIYA/DEV LAB/map1.html")
# OpenStreetMap
map = folium.Map(location=[13.406, 80.110], tiles="OpenStreetMap", zoom_start=9)
map.save("D:/SIBIYA/DEV LAB/map2.html")
# Stamen Toner
map = folium.Map(location=[13.406, 80.110], tiles="Stamen Toner", zoom_start=9)
map.save("D:/SIBIYA/DEV LAB/map3.html")
# Use terrain map layer to see volcano terrain
map = folium.Map(location=[4, 10], tiles="Stamen Terrain", zoom_start=3)
# Create a geometry list from the GeoDataFrame
geo_df_list = [[point.xy[1][0], point.xy[0][0]] for point in geo_df.geometry]
# Iterate through list and add a marker for each volcano, color-coded by its type.
i = 0
for coordinates in geo_df_list:
    # assign a color marker for the type of volcano, Strato being the most common
    if geo_df.Type[i] == "Stratovolcano":
        type_color = "green"
    elif geo_df.Type[i] == "Complex volcano":
        type_color = "blue"
    elif geo_df.Type[i] == "Shield volcano":
        type_color = "orange"
    elif geo_df.Type[i] == "Lava dome":
        type_color = "pink"
```

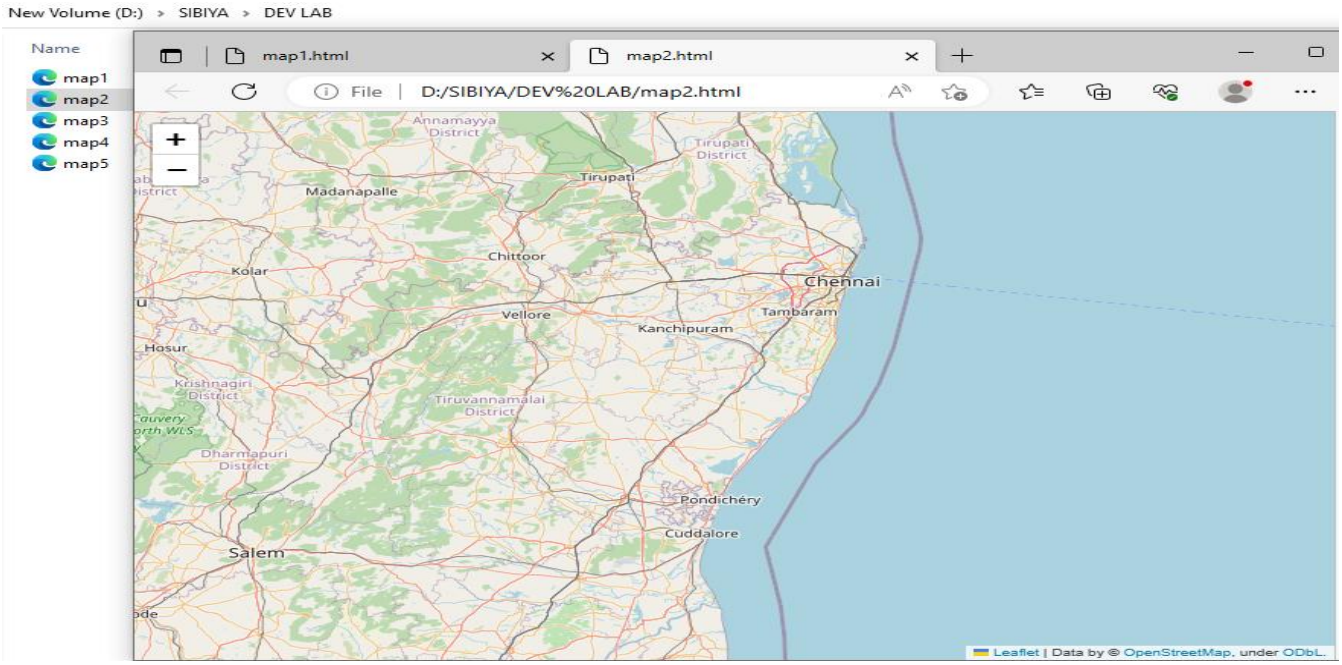
```
    else:
        type_color = "purple"
    # Place the markers with the popup labels and data
    map.add_child(
        folium.Marker(
            location=coordinates,
            popup="Year: "
            + str(geo_df.Year[i]) + "<br>"
            + "Name: "
            + str(geo_df.Name[i]) + "<br>"
            + "Country: "
            + str(geo_df.Country[i]) + "<br>"
            + "Type: "
            + str(geo_df.Type[i]) + "<br>"
            + "Coordinates: "
            + str(geo_df_list[i]),
            icon=folium.Icon(color="%s" % type_color),
        )    )
    i = i + 1
map.save("D:/SIBIYA/DEV LAB/map4.html")
map = folium.Map(location=[15, 30], tiles="Cartodb dark_matter", zoom_start=2)
heat_data = [[point.xy[1][0], point.xy[0][0]] for point in geo_df.geometry]
print(heat_data)
plugins.HeatMap(heat_data).add_to(map)
```

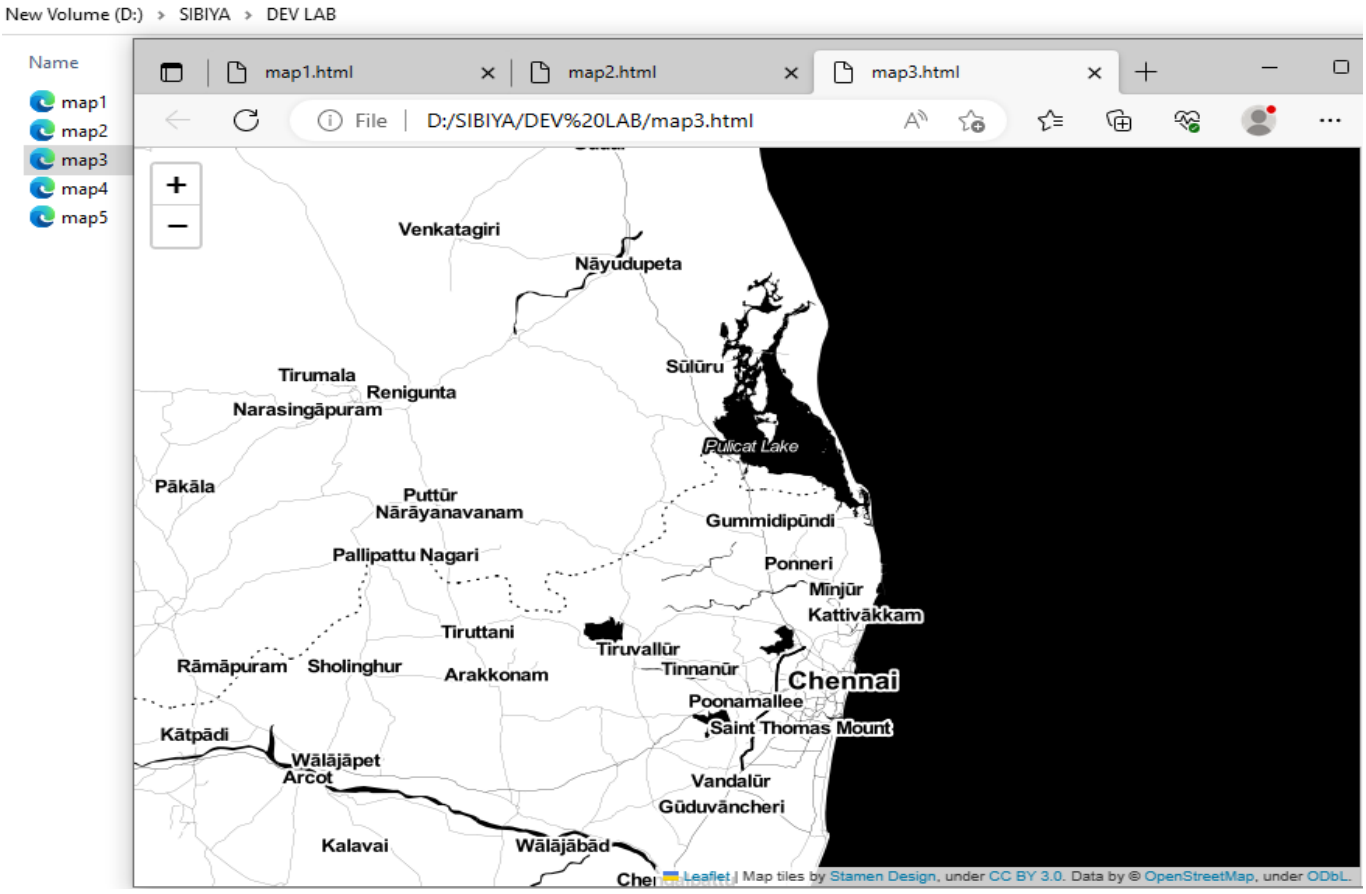
```
map.save("D:/SIBIYA/DEV LAB/map5.html")
```

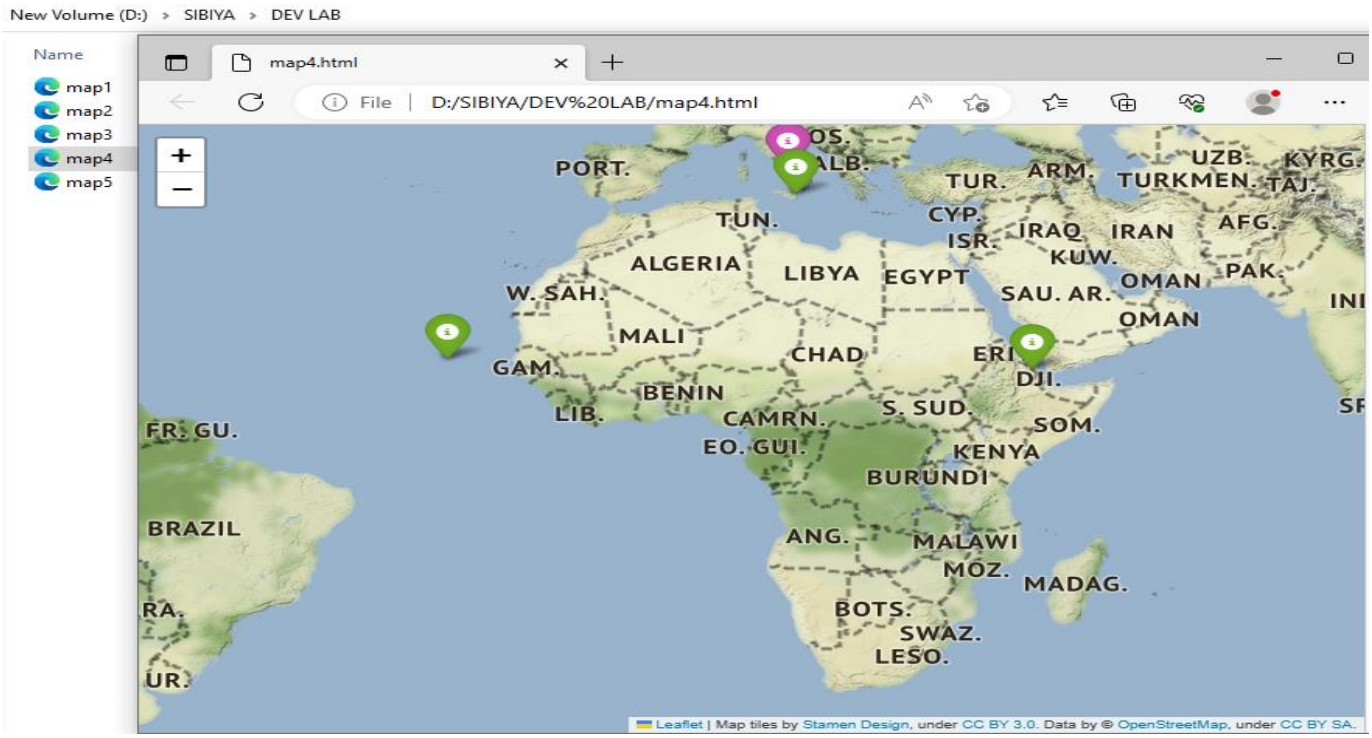
OUTPUT

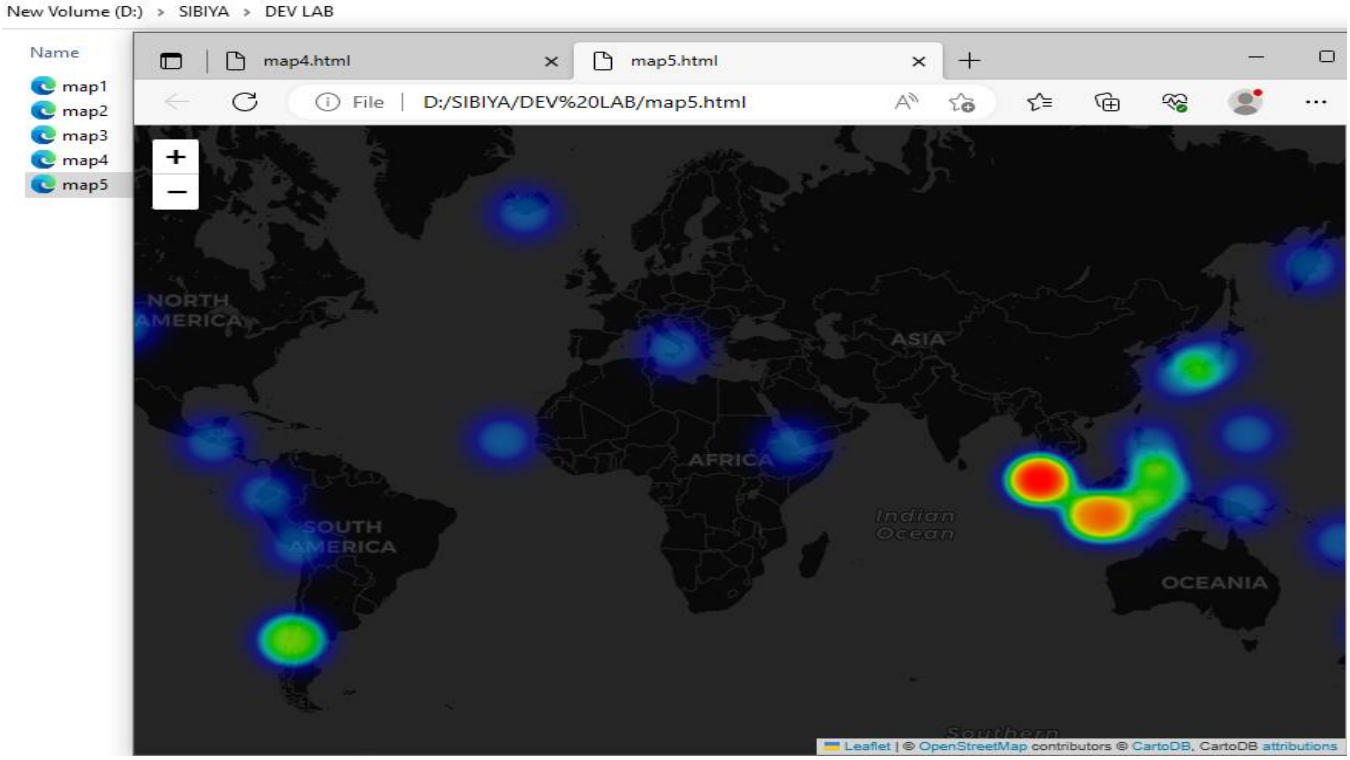
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 63 entries, 0 to 62
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Year       63 non-null     int64
 1   Name       63 non-null     object
 2   Country    63 non-null     object
 3   Latitude   63 non-null     float64
 4   Longitude  63 non-null     float64
 5   Type       63 non-null     object
dtypes: float64(2), int64(1), object(3)
memory usage: 3.1+ KB
[[-1.467, -78.442], [63.63, -19.62], [14.381, -90.601], [16.708, 145.78], [2.78, 125.48]
```




New Volume (D:) › SIBIYA › DEV LAB
Name
map1
map2
map3
map4
map5
map1.html
map2.html
File | D:/SIBIYA/DEV%20LAB/map2.html
Annamayya District
Tirupati District
Tirupati
Madanapalle
Chittoor
Kolar
Chennai
Vellore
Tambaram
Kanchipuram
Hosur
Krishnagiri District
Tiruvannamalai District
Dharmapuri District
Pondichéry
Cuddalore
Salem
Leaflet | Data by © OpenStreetMap, under ODbL.


New Volume (D:) › SIBIYA › DEV LAB
Name
map1
map2
map3
map4
map5
map1.html
map2.html
map3.html
File | D:/SIBIYA/DEV%20LAB/map3.html
Venkatagiri
Nāyudupeta
Sūlūru
Tirumala
Renigunta
Narasingāpuram
Pulicat Lake
Pākāla
Puttūr
Nārāyanavanam
Gummidipūndi
Pallipattu Nagari
Ponneri
Minjūr
Kattivākkam
Tiruttani
Tiruvallūr
Rāmāpuram
Sholinghur
Arakkonam
Tinnanūr
Chennai
Poonamallee
Saint Thomas Mount
Kātpādi
Wālājāpet
Arcot
Vandalūr
Gūduvāncheri
Kalavai
Wālājābād
Leaflet | Map tiles by Stamen Design, under CC BY 3.0. Data by © OpenStreetMap, under ODbL.





RESULT

Thus, the cartographic visualization for multiple datasets involving various countries of the world; states and districts in India is visualized and verified successfully.

Exp No: 8	Perform Exploratory Data Analysis on Wine Quality Data Set
Date:	

AIM

To perform EDA on Wine Quality Data Set.

ALGORITHM

Step 1: Import Libraries and Load Data

- Import the necessary Python libraries: `numpy`, `pandas`, `matplotlib`, `seaborn`, and various machine learning modules.

- Disable warnings for cleaner output.

- Load the wine quality dataset from the 'winequality.csv' file into a Pandas DataFrame (`df`).

Step 2: Data Exploration and Preprocessing

- Examine the data: Display the first few rows of the dataset, information about the dataset, and summary statistics.

- Check for missing values in the dataset, and if any are found, fill them with the mean of the respective column.

- Visualize the data with histograms and a bar plot.

- Generate a correlation heatmap to understand the relationships between variables.

- Remove the 'total sulfur dioxide' column from the dataset.

- Create a new binary column, 'best quality,' based on a condition (1 if quality > 5, else 0).

Step 3: Data Encoding and Splitting

- Encode the 'color' column as binary values (e.g., 'white' as 1, 'red' as 0) in the DataFrame.

- Define feature (independent variables) and target (dependent variable) variables:

- Features: Exclude 'quality' and 'best quality'.

- Target: 'best quality'.

- Split the dataset into training and testing sets (e.g., 80% training and 20% testing) using the `train_test_split` function.

Step 4: Data Normalization

- Normalize the feature data using Min-Max scaling, transforming values to the range [0, 1].

Step 5: Model Training and Evaluation

- Create a list of machine learning models (Logistic Regression, XGBoost Classifier, and Support Vector Classifier).

- Iterate through the list of models, fitting each model to the training data, and evaluating its

performance:

- Print the model name.

- Calculate and display training and validation accuracy using ROC AUC score.

Step 6: Visualization and Reporting

- Plot a confusion matrix for the XGBoost Classifier model to assess its classification performance.

- Generate a classification report for the XGBoost Classifier model, including precision, recall, F1-score, and support for each class (binary classification: 'best quality' as 0 or 1).

PROGRAM

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
import warnings


from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn import metrics
from sklearn.svm import SVC
from xgboost import XGBClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
warnings.filterwarnings('ignore')
df = pd.read_csv('D:/SIBIYA/DEV LAB/winequality.csv')
print(df.head())
print(df.info())
print(df.describe().T)
print(df.isnull().sum())
for col in df.columns:
    if df[col].isnull().sum() > 0:
        df[col] = df[col].fillna(df[col].mean())
print(df.isnull().sum().sum())
```

```
df.hist(bins=20, figsize=(10, 10))
plt.show()
plt.bar(df['quality'], df['alcohol'])
plt.xlabel('quality')
plt.ylabel('alcohol')
plt.show()
plt.figure(figsize=(12, 12))
sb.heatmap(df.corr() > 0.7, annot=True, cbar=False)
plt.show()
df = df.drop('total sulfur dioxide', axis=1)
df['best quality'] = [1 if x > 5 else 0 for x in df.quality]
print(df.replace({'white': 1, 'red': 0}, inplace=True))
features = df.drop(['quality', 'best quality'], axis=1)
target = df['best quality']
xtrain, xtest, ytrain, ytest = train_test_split(
    features, target, test_size=0.2, random_state=40)
print(xtrain.shape, xtest.shape)
norm = MinMaxScaler()
xtrain = norm.fit_transform(xtrain)
xtest = norm.transform(xtest)
models = [LogisticRegression(), XGBClassifier(), SVC(kernel='rbf')]
for i in range(3):
    models[i].fit(xtrain, ytrain)
    print(f'{models[i]} : ')
```

```
    print('Training Accuracy : ', metrics.roc_auc_score(ytrain, models[i].predict(xtrain)))
    print('Validation Accuracy : ', metrics.roc_auc_score(
        ytest, models[i].predict(xtest)))
    print()
metrics.plot_confusion_matrix(models[1], xtest, ytest)
plt.show()
print(metrics.classification_report(ytest, models[1].predict(xtest)))
```

OUTPUT

```
     type  fixed acidity  volatile acidity  ...  sulphates  alcohol  quality
0   white            7.0              0.27  ...       0.45      8.8        6
1   white            6.3              0.30  ...       0.49      9.5        6
2   white            8.1              0.28  ...       0.44     10.1        6
3   white            7.2              0.23  ...       0.40      9.9        6
4   white            7.2              0.23  ...       0.40      9.9        6

[5 rows x 13 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6497 entries, 0 to 6496
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   type                  6497 non-null   object
 1   fixed acidity         6487 non-null   float64
 2   volatile acidity      6489 non-null   float64
 3   citric acid           6494 non-null   float64
 4   residual sugar        6495 non-null   float64
 5   chlorides             6495 non-null   float64
 6   free sulfur dioxide   6497 non-null   float64
 7   total sulfur dioxide  6497 non-null   float64
 8   density               6497 non-null   float64
 9   pH                    6488 non-null   float64
 10  sulphates             6493 non-null   float64
 11  alcohol               6497 non-null   float64
 12  quality               6497 non-null   int64
dtypes: float64(11), int64(1), object(1)
memory usage: 660.0+ KB
None
```

```
                       count         mean   ...         75%         max
fixed acidity         6487.0     7.216579   ...     7.70000    15.90000
volatile acidity      6489.0     0.339691   ...     0.40000     1.58000
citric acid           6494.0     0.318722   ...     0.39000     1.66000
residual sugar        6495.0     5.444326   ...     8.10000    65.80000
chlorides             6495.0     0.056042   ...     0.06500     0.61100
free sulfur dioxide   6497.0    30.525319   ...    41.00000   289.00000
total sulfur dioxide  6497.0   115.744574   ...   156.00000   440.00000
density               6497.0     0.994697   ...     0.99699     1.03898
pH                    6488.0     3.218395   ...     3.32000     4.01000
sulphates             6493.0     0.531215   ...     0.60000     2.00000
alcohol               6497.0    10.491801   ...    11.30000    14.90000
quality               6497.0     5.818378   ...     6.00000     9.00000

[12 rows x 8 columns]
type                    0
fixed acidity          10
volatile acidity        8
citric acid             3
residual sugar          2
chlorides               2
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      9
sulphates               4
alcohol                 0
quality                 0
dtype: int64
0
None
(5197, 11) (1300, 11)
LogisticRegression() :
Training Accuracy :  0.7019709565048414
Validation Accuracy :  0.6937888865050418

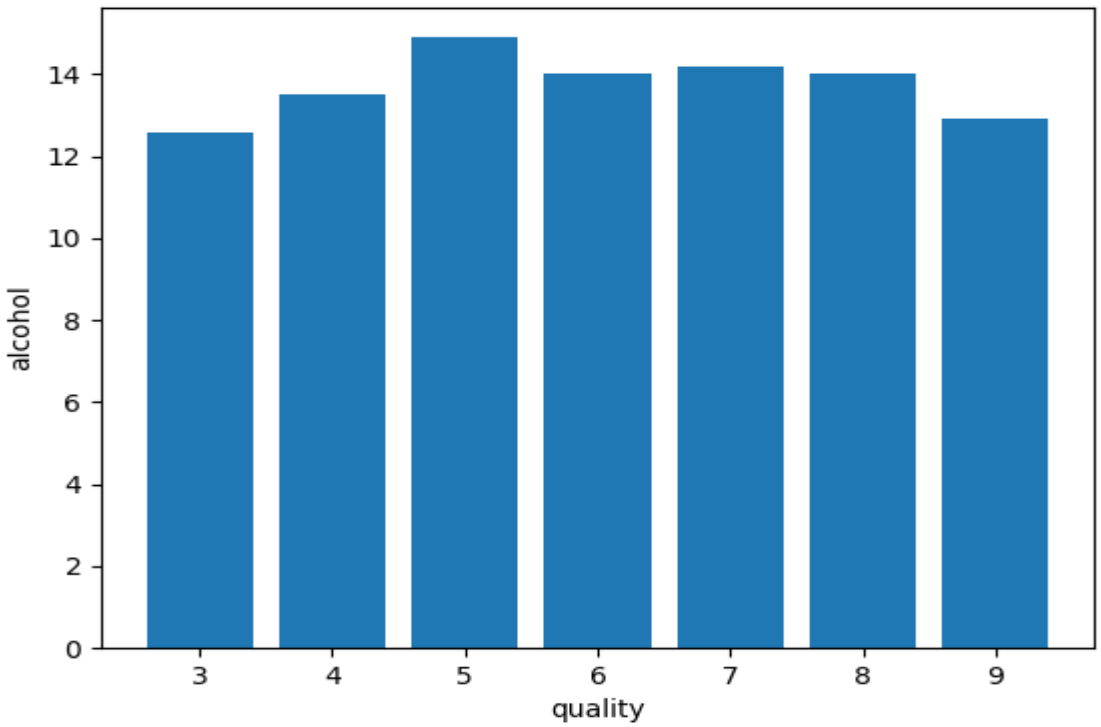
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=None, n_jobs=None,
              num_parallel_tree=None, random_state=None, ...) :
Training Accuracy :  0.9735567052182403
Validation Accuracy :  0.8050515421787681

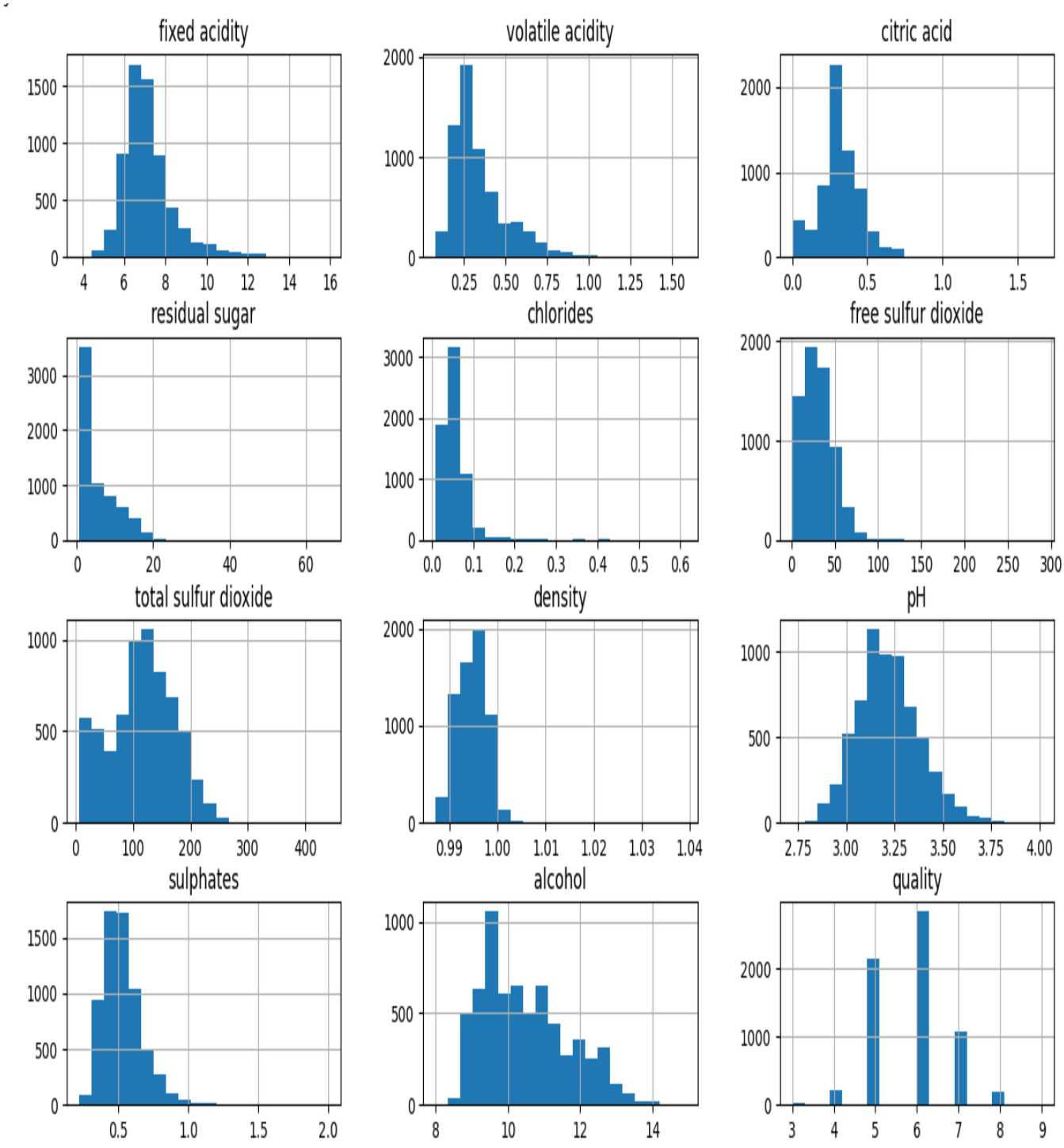
SVC() :
Training Accuracy :  0.7069199304892986
Validation Accuracy :  0.695796426272719

              precision    recall  f1-score   support

           0       0.78      0.73      0.75       474
           1       0.85      0.88      0.86       826

    accuracy                           0.83      1300
   macro avg       0.81      0.81      0.81      1300
weighted avg       0.82      0.83      0.82      1300
```


alcohol
14
12
10
8
6
4
2
0
3
4
5
6
7
8
9
quality



RESULT

Thus, the EDA on Wine Quality Data Set is performed and executed successfully.

Exp No: 9	Case Study
Date:	

AIM

Case study on a data set and apply the various EDA and visualization techniques and present an analysis report.

ALGORITHM

Step 1: Data Import and Preprocessing

- Import required libraries (`pandas`, `matplotlib.pyplot`).

- Read the CSV file ('violations.csv') into a DataFrame.

- Convert the 'Issue Date' column to a date data type.

- Print the initial number of rows in the dataset.

Step 2: Data Filtering

- Remove rows with invalid data based on multiple filtering conditions, such as date range, non-null values, valid codes, and specific values in columns.

- Print the number of rows remaining in the filtered dataset.

Step 3: Data Visualization

- Group data by 'Vehicle Year' and count parking violations for each year.

- Create a line plot to visualize the number of parking violations for each vehicle year.

Step 4: Data Analysis

- Group data for non-New York ('NY') registered vehicles by 'Violation Code' and find the top 5 violation codes with the most violations.

- Group data for 'HONDA' vehicles by 'Street Name' and find the street with the most violations.

- Create a subset for New York registered vehicles ('NY').

- Calculate the ratio of non-passenger plates to all plates, grouped by vehicle year, and replace null values with 0.

- Create a line plot to visualize the ratio of non-passenger plates to all plates for each vehicle year.

Step 5: Additional Analysis and Statistics

- Find the most common vehicle color for 'PAS' (passenger) plates.

- Find the most common vehicle color for 'COM' (commercial) plates.

- Calculate and print the number of unique 'Registration States' in the dataset.

- Calculate and print the average number of parking violations per 'Registration State'.

- Group data by 'Violation Code' and 'Plate Type' to count the most common plate type for each violation code.

- Calculate and display the number of parking violations in each 'Violation County' as a percentage of the total violations.

PROGRAM

```
import pandas as pd

import matplotlib.pyplot as plt

# Read the file and import all rows.

df = pd.read_csv('D:/SIBIYA/DEV LAB/violations.csv')

# Change the data type of the 'Issue Date' column to date.

df['Issue Date'] = pd.to_datetime(df['Issue Date'])

# Print out the number of rows imported from the file.

print('Number of Rows: ' + str(len(df)))

# Remove rows containing invalid data.

df = df[(df['Registration State'] != "99") & (df['Plate Type'] != "999") & (df['Issue Date'] >= '2020-04-01') &
(df['Issue Date'] <= '2020-11-30') & (df['Violation Code'] != 0) & (df['Vehicle Make'].notnull())

        & (df['Violation Time'].notnull()) & (df['Vehicle Year'] != 0) & (df['Vehicle Year'] <= 2020)]

# Print out the number of rows remaining in the dataset.

print('Number of Rows: ' + str(len(df)))

# Isolate the data to be used in the plot.

df_vehicle_year = df.groupby('Vehicle Year')['Summons Number'].count()

# Create a plot that shows the number of parking violations for each vehicle year.

plt.plot(df_vehicle_year)

plt.show()

df[df['Registration State'] != 'NY'].groupby('Violation Code')['Summons
Number'].count().nlargest(5).reset_index(name='Count')

df[df['Vehicle Make'] == 'HONDA'].groupby('Street Name')['Summons
Number'].count().nlargest(1).reset_index(name='Count')

# Subset for only rows where the Registration State is NY.
```

```
df_ny = df[df['Registration State'] == 'NY']

# Calculate the ratio of non-passenger plates to all plates, grouped by year.

df_ny_notpas = df_ny[df_ny['Plate Type'] != 'PAS'].groupby('Vehicle Year')['Summons Number'].count()

df_ny_all = df_ny.groupby('Vehicle Year')['Summons Number'].count()

ratio = df_ny_notpas / df_ny_all

# Replace nulls with 0.

ratio.fillna(0, inplace = True)

# Create and show plot.

plt.plot(ratio)

plt.show()

df[df['Plate Type'] == 'PAS'].groupby('Vehicle Color')['Summons Number'].count().nlargest(1).reset_index(name='Count')

df[df['Plate Type'] == 'COM'].groupby('Vehicle Color')['Summons Number'].count().nlargest(1).reset_index(name='Count')

print('Number of Registration States: ' + str(df['Registration State'].nunique()))

print('Average Number of Parking Violations per Registration State: ' +

    str(df.groupby('Registration State')['Summons Number'].count().mean()))

df.groupby('Violation Code')['Plate Type'].apply(lambda x: x.value_counts().head(1)).reset_index(name='Count').rename(columns={'level_1': 'Plate Type'})

# Count the number of parking violations in each county.

df_county = df.groupby('Violation County')['Summons Number'].count().reset_index(name='Percentage')

# Calculate the number of parking violations in each county as a percentage of all parking violations.

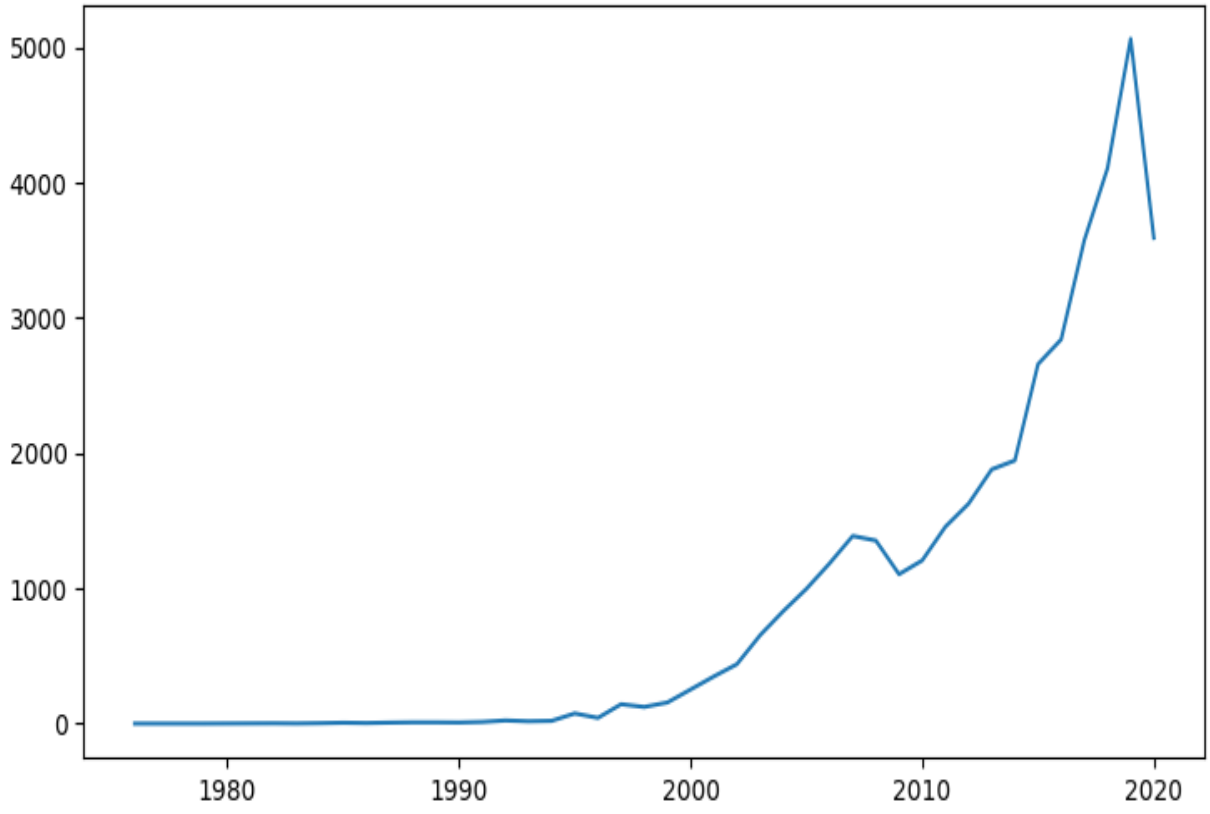
df_county['Percentage'] = df_county['Percentage'] / df_county['Percentage'].sum() * 100

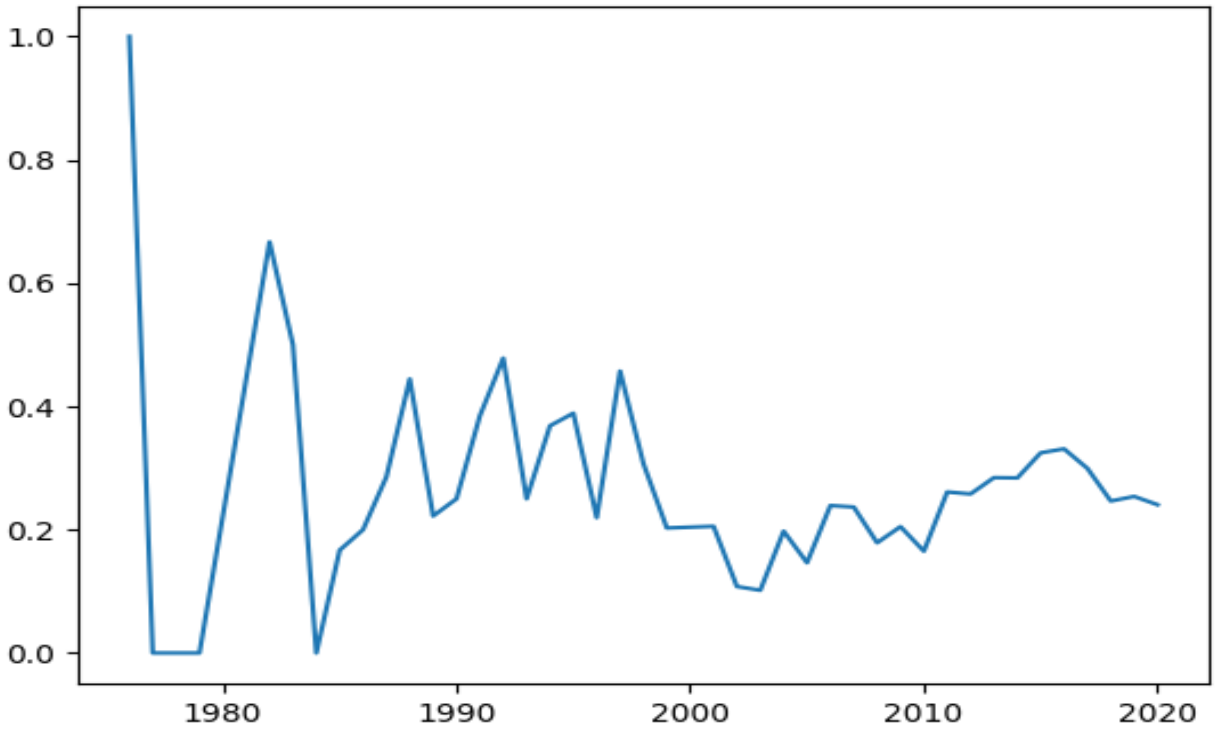
# Sort and display the resulting dataframe.

df_county.sort_values(by='Percentage', ascending=False).reset_index(drop=True)
```

OUTPUT

```
Number of Rows: 50000
Number of Rows: 38937
Number of Registration States: 45
Average Number of Parking Violations per Registration State: 865.2666666666667
```





RESULT

Thus, the data set and applying the various EDA and visualization techniques is studied.

Exp. No.: CB1	Perform EDA on Email Data set for spam Prediction using Random Forest Algorithm
Date:	

AIM

To Perform EDA on Email Data set for spam Prediction using Random Forest algorithm.

ALGORITHM

Step 1: Import Libraries and Load Data

- Import necessary libraries such as NumPy,Pandas,scikit-learn's`train_test_split`, `MultinomialNB`, `RandomForestClassifier`, and `accuracy_score`.

- Load the dataset from the specified CSV file into a Pandas DataFrame.

Step 2: Data Exploration

- Display the first 10 rows of the DataFrame using `df.head(10)`.

- Check for missing values in the DataFrame using `df.isnull().sum()`.

- Provide a summary of the DataFrame's statistics using `df.describe()`.

- Calculate the correlation matrix for numeric columns using `df.corr(numeric_only=True)` and store it in variable `c`.

Step 3: Data Preprocessing

- Extract the feature variables (X) by selecting columns from the 1st to 3000th (inclusive) using `df.iloc[:, 1:3001]`.

- Extract the target variable (Y) from the last column using `df.iloc[:, -1].values`.

Step 4: Data Splitting

- Split the data into training and testing sets using `train_test_split`. It assigns 75% of the data to the training set and 25% to the testing set. The resulting sets are `train_x`, `test_x`, `train_y`, and `test_y`.

Step 5: Model Training and Evaluation

- Create a Multinomial Naive Bayes classifier (`mnb`) with a specified alpha value and fit it to the training data.

- Make predictions on the test data using the trained Naive Bayes classifier (`mnb.predict(test_x)`).

- Calculate and print the accuracy score for the Naive Bayes model using `accuracy_score`.

- Create a Random Forest Classifier (`rfc`) with 100 decision trees and the 'gini' criterion, and fit it to the training data.

- Make predictions on the test data using the trained Random Forest classifier (`rfc.predict(test_x)`).

- Calculate and print the accuracy score for the Random Forest model using `accuracy_score`.

PROGRAM

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
import os
df = pd.read_csv("D:/SIBIYA/DEV LAB/emails.csv")
print(df.head(10))
print(df.isnull().sum())
print(df.describe())
c=df.corr(numeric_only = True)
print("Correlation",c)
X = df.iloc[:,1:3001]
print(X)
Y = df.iloc[:,-1].values
print(Y)
train_x,test_x,train_y,test_y = train_test_split(X,Y,test_size = 0.25)
mnb = MultinomialNB(alpha=1.9)
mnb.fit(train_x,train_y)
y_pred1 = mnb.predict(test_x)
print("Accuracy Score for Naive Bayes : ", accuracy_score(y_pred1,test_y))
rfc = RandomForestClassifier(n_estimators=100,criterion='gini')
```

```
rfc.fit(train_x,train_y)

y_pred3 = rfc.predict(test_x)

print("Accuracy Score of Random Forest Classifier : ", accuracy_score(y_pred3,test_y))
```

OUTPUT

```
   Email No.  the  to  ect  and  ...  military  allowing  ff  dry  Prediction
0    Email 1    0   0    1    0  ...         0         0   0    0           0
1    Email 2    8  13   24    6  ...         0         0   1    0           0
2    Email 3    0   0    1    0  ...         0         0   0    0           0
3    Email 4    0   5   22    0  ...         0         0   0    0           0
4    Email 5    7   6   17    1  ...         0         0   1    0           0
5    Email 6    4   5    1    4  ...         0         0   0    0           1
6    Email 7    5   3    1    3  ...         0         0   0    0           0
7    Email 8    0   2    2    3  ...         0         0   1    0           1
8    Email 9    2   2    3    0  ...         0         0   0    0           0
9   Email 10    4   4   35    0  ...         0         0   0    0           0

[10 rows x 3002 columns]
Email No.     0
the           0
to            0
ect           0
and           0
             ..
military      0
allowing      0
ff            0
dry           0
Prediction    0
Length: 3002, dtype: int64
                the           to  ...          dry   Prediction
count  5172.000000  5172.000000  ...  5172.000000  5172.000000
mean      6.640565     6.188128  ...     0.006961     0.290023
std      11.745009     9.534576  ...     0.098086     0.453817
min       0.000000     0.000000  ...     0.000000     0.000000
25%       0.000000     1.000000  ...     0.000000     0.000000
```

```
50%       3.000000     3.000000  ...     0.000000     0.000000
75%       8.000000     7.000000  ...     0.000000     1.000000
max     210.000000   132.000000  ...     4.000000     1.000000

[8 rows x 3001 columns]
Correlation                  the        to       ect  ...        ff       dry  Prediction
the         1.000000  0.852715  0.337249  ...  0.341878  0.051021   -0.004421
to          0.852715  1.000000  0.375480  ...  0.406666  0.071388    0.055277
ect         0.337249  0.375480  1.000000  ...  0.141460  0.002492   -0.120782
and         0.841200  0.825474  0.272863  ...  0.400225  0.042484    0.114364
for         0.784112  0.781971  0.369777  ...  0.301074  0.038126   -0.003101
...              ...       ...       ...  ...       ...       ...         ...
military    0.129466  0.091639 -0.007690  ...  0.049524  0.010835    0.064850
allowing    0.127019  0.120059  0.004368  ...  0.096212 -0.003995    0.011279
ff          0.341878  0.406666  0.141460  ...  1.000000  0.049690    0.135479
dry         0.051021  0.071388  0.002492  ...  0.049690  1.000000   -0.006260
Prediction -0.004421  0.055277 -0.120782  ...  0.135479 -0.006260    1.000000

[3001 rows x 3001 columns]
      the  to  ect  and  for  ...  infrastructure  military  allowing  ff  dry
0       0   0    1    0    0  ...               0         0         0   0    0
1       8  13   24    6    6  ...               0         0         0   1    0
2       0   0    1    0    0  ...               0         0         0   0    0
3       0   5   22    0    5  ...               0         0         0   0    0
4       7   6   17    1    5  ...               0         0         0   1    0
...   ...  ..  ...  ...  ...  ...             ...       ...       ...  ..  ...
5167    2   2    2    3    0  ...               0         0         0   0    0
5168   35  27   11    2    6  ...               0         0         0   1    0
5169    0   0    1    1    0  ...               0         0         0   0    0
5170    2   7    1    0    2  ...               0         0         0   1    0
5171   22  24    5    1    6  ...               0         0         0   0    0


[5172 rows x 3000 columns]
[0 0 0 ... 1 1 0]
Accuracy Score for Naive Bayes :  0.945862335653519
Accuracy Score of Random Forest Classifier :  0.9644238205723125

Process finished with exit code 0
```

RESULT

Thus, the python program is to Perform EDA on Email Data set for spam Prediction using RandomForest algorithm has been done and executed successfully.

Exp. No.: CB2	Perform EDA on India map dataset for finding most Populated cities
Date:	

AIM

To Perform EDA on India map dataset for finding most Populated cities.

ALGORITHM

Step 1: Import Necessary Packages

-Import the required Python packages, including pandas, numpy, and matplotlib, for data manipulation and visualization.

Step 2: Load and Explore Data

- Load the city data from a CSV file into a Pandas DataFrame.

- Display the first few rows of the DataFrame to get a sense of the data's structure.

Step 3: Create a Horizontal Bar Plot for the Number of Cities per State

- Group the data by the 'state_name' column and count the number of cities in each state.

- Sort the states in ascending order of city count.

- Create a horizontal bar plot to visualize the number of cities per state.

- Customize the plot with labels, grid lines, and a title.

Step 4: Extract Latitude and Longitude from 'location' Column

- Extract latitude and longitude information from the 'location' column in the DataFrame.

- Split the 'location' column into separate 'latitude' and 'longitude' columns.

Step 5: Create a Map with the Top 10 Populated Cities

- Sort the DataFrame by the 'population_total' column in descending order to find the top 10 populated cities.

- Create a Basemap with specified parameters for the map's size and location.

- Plot the map boundaries, coastlines, and country borders.

- Extract latitude, longitude, population, and city names for the top 10 cities.

- Calculate point sizes on the map based on population.

- Scatter plot the cities on the map with different point sizes and colors.

- Add city names as labels to the plot.

- Set the title for the map to "Top 10 Populated Cities in India.

PROGRAM

```
# importing packages

import pandas as pd

import numpy as np

from numpy import array

import matplotlib as mpl

# for plots

import matplotlib.pyplot as plt

from matplotlib import cm

from mpl_toolkits.basemap import Basemap

cities = pd.read_csv("D:/SIBIYA/datasets_557_1096_cities_r2.csv")

print(cities.head())

fig = plt.figure(figsize=(15, 20))

states = cities.groupby('state_name')['name_of_city'].count().sort_values(ascending=True)

print(states.plot(kind="barh", fontsize=20))

plt.grid(which='both', color='Black', linestyle='-')

plt.xlabel('No of cities taken for analysis', fontsize=20)

plt.show()

cities['latitude'] = cities['location'].apply(lambda x: x.split(',')[0])

cities['longitude'] = cities['location'].apply(lambda x: x.split(',')[1])

print("The Top 10 Cities sorted according to the Total Population (Descending Order)")

top_pop_cities = cities.sort_values(by='population_total', ascending=False)

top10_pop_cities = top_pop_cities.head()

print(top_pop_cities)
```

```
plt.subplots(figsize=(20, 15))
map1 = Basemap(width=1200000, height=900000, projection='lcc', resolution='l',
            llcrnrlon=67, llcrnrlat=5, urcrnrlon=99, urcrnrlat=37, lat_0=28, lon_0=77)
map1.drawmapboundary()
map1.drawcountries()
map1.drawcoastlines()
lg = array(top10_pop_cities['longitude'])
lt = array(top10_pop_cities['latitude'])
pt = array(top10_pop_cities['population_total'])
nc = array(top10_pop_cities['name_of_city'])
x, y = map1(lg, lt)
population_sizes = top10_pop_cities["population_total"].apply(lambda x: int(x / 5000))
plt.scatter(x, y, s=population_sizes, marker="o", c=population_sizes, cmap=cm.Dark2, alpha=0.7)
for ncs, xpt, ypt in zip(nc, x, y):
   plt.text(xpt + 60000, ypt + 30000, ncs, fontsize=10, fontweight='bold')
plt.title('Top 10 Populated Cities in India', fontsize=20)
plt.show()
```

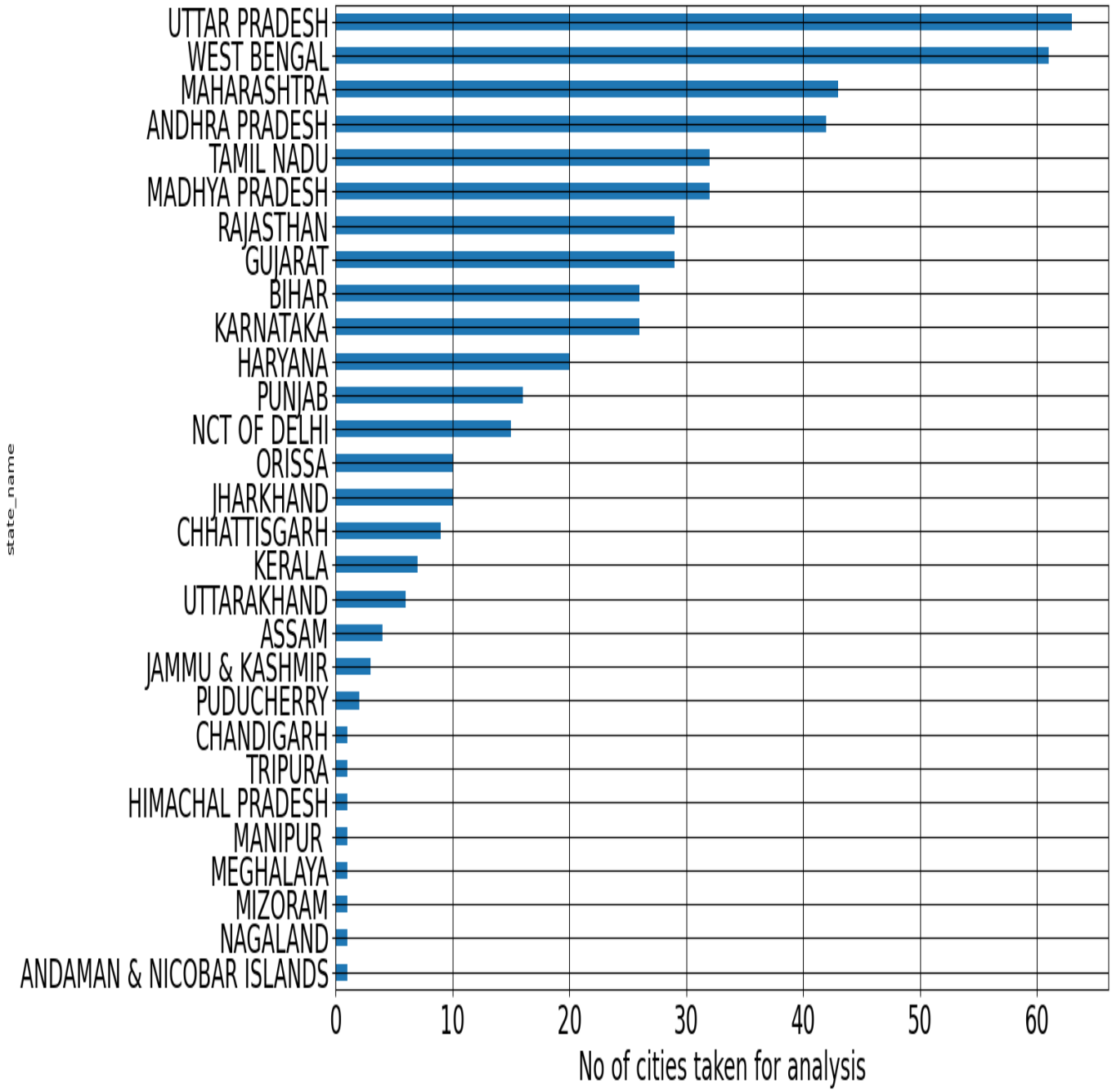
OUTPUT

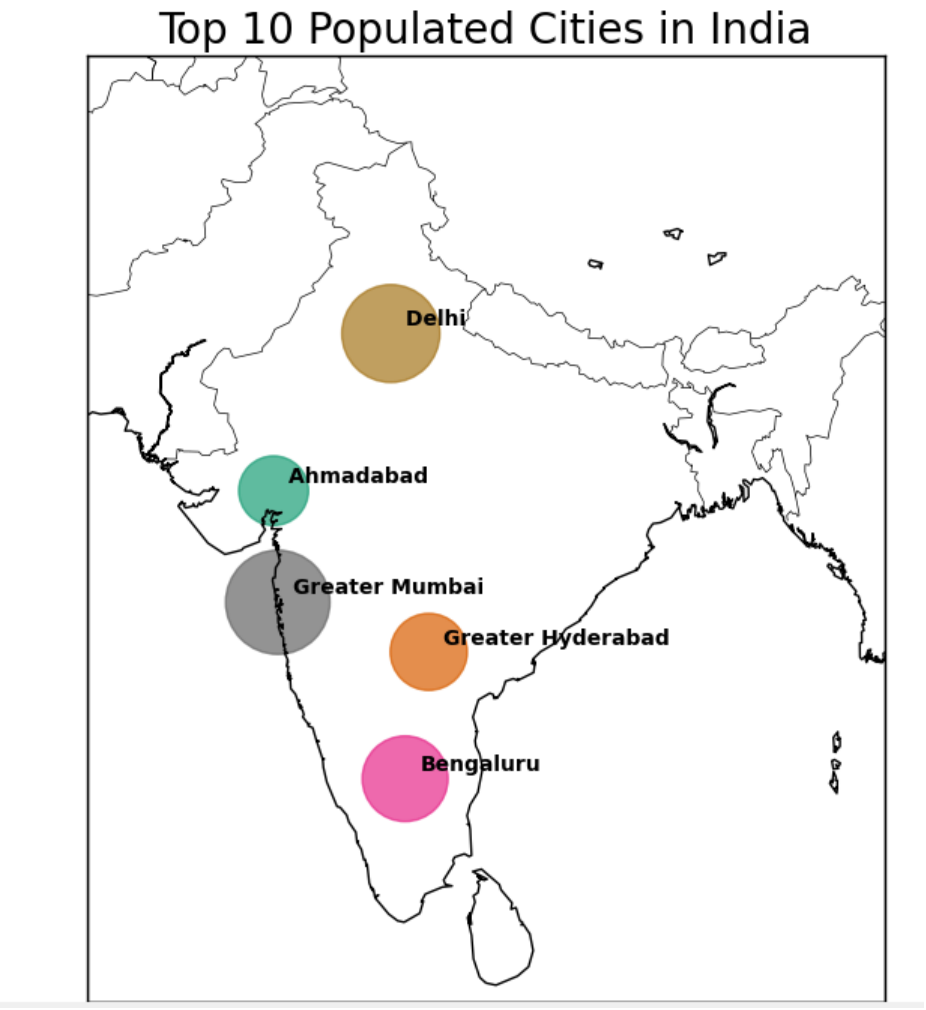
```
  name_of_city  state_code  ... male_graduates  female_graduates
0      Abohar           3  ...           8612              7675
1    Achalpur          27  ...           5269              3594
2    Adilabad          28  ...           6797              3768
3   Adityapur          20  ...          12189              7036
4       Adoni          28  ...           7871              4031

[5 rows x 22 columns]
Axes(0.125,0.11;0.775x0.77)
The Top 10 Cities sorted according to the Total Population (Descending Order)
          name_of_city  state_code  ...    latitude   longitude
185     Greater Mumbai          27  ...     19.0760     72.8777
141              Delhi           7  ...     28.7041     77.1025
72           Bengaluru          29  ...     12.9716     77.5946
184  Greater Hyderabad          28  ...     17.3850     78.4867
7            Ahmadabad          24  ...   23.022505  72.5713621
..                 ...         ...  ...         ...         ...
376        Port Blair          35  ...  11.6233774  92.7264828
136              Datia          23  ...  25.6653262  78.4609393
211        Hinganghat          27  ...  20.5505728  78.8411405
53           Banswara           8  ...  23.5461394  74.4349761
332              Nagda          23  ...  23.4454599  75.4169918

[493 rows x 24 columns]

Process finished with exit code 0
```


UTTAR PRADESH
WEST BENGAL
MAHARASHTRA
ANDHRA PRADESH
TAMIL NADU
MADHYA PRADESH
RAJASTHAN
GUJARAT
BIHAR
KARNATAKA
HARYANA
PUNJAB
NCT OF DELHI
ORISSA
JHARKHAND
CHHATTISGARH
KERALA
UTTARAKHAND
ASSAM
JAMMU & KASHMIR
PUDUCHERRY
CHANDIGARH
TRIPURA
HIMACHAL PRADESH
MANIPUR
MEGHALAYA
MIZORAM
NAGALAND
ANDAMAN & NICOBAR ISLANDS
state_name
0
10
20
30
40
50
60
No of cities taken for analysis



RESULT

Thus, the python program is to Perform EDA on India map dataset for finding most Populated cities has been done and executed successfully.