MANGALORE UNIVERSITY

MASTER OF SCIENCE
IN
COMPUTER SCIENCE

23CSP201: PRINCIPLES OF DATA SCIENCE LAB

SUBMITTED
BY

II SEMESTER MSC
Computer Science Students

SUBMITTED
TO

Dr. H.L. Shashirekha
Department of Computer Science

Lecturers, In-charge:

1.

2.

Mangalore University

Dept. of Post-Graduate Studies and Research in Computer Science

Mangalagangothri - 574199

PRINCIPLES OF DATA SCIENCE LAB PROGRAMS

1. Write a python program to

i. read multiple files from single folder

ii. read multiple files from multiple folders

```
#program to read multiple files from single folder
import os

path = os.getcwd()

for file in os.listdir(path):
    if file.endswith(".txt"):
        file_path = os.path.join(path, file)
        print(file)
        with open(file_path, 'r') as f:
            print(f.read())
```

Output:

```
file1.txt
Hello world!!!
file2.txt
Principles of Data Science
file3.txt
Welcome
```

```
#program to read multiple files from multiple folders
import os

def read_text_files_from_folders(root_folder):
    for folder_name, subfolders, filenames in os.walk(root_folder):
        print("File name :: ",filenames)
        for filename in filenames:
            if filename.endswith('.txt'):
                file_path = os.path.join(folder_name, filename)
                try:
                    with open(file_path, 'r') as file:

                        print(folder_name)
                        print(filename)
                print(file.read())
            except Exception as e:
               print(f"Error reading file {file_path}: {e}")

root_folder = "F:\Sample"
texts = read_text_files_from_folders(root_folder)
```

Output:

```
File name ::  ['file1.txt', 'file2.txt', 'file3.txt']
F:\Sample\Sample_subfolder1
file1.txt
Hello world!!!
F:\Sample\Sample_subfolder1
file2.txt
Principles of Data Science
F:\Sample\Sample_subfolder1
file3.txt
Welcome
File name ::  ['file11.txt', 'sample.txt']
F:\Sample\Sample_subfolder3
file11.txt
Matplotlib is an amazing visualization library in Python for 2D plots of arrays.
It was introduced by John Hunter in the year 2002.
Matplotlib consists of several plots like line, bar, scatter, histogram, etc.
F:\Sample\Sample_subfolder3
sample.txt
Python is a dynamic, high-level, free open source, and interpreted programming language. It
supports object-oriented programming as well as procedural-oriented programming.
```

2. Implement the python program to find central tendency (mean, median, and mode) of data, with and without using built-in function on the data.

```
import numpy as np
import statistics

def find_mean(list1):
    total = 0
    for ele in list1:
        total += ele
    mean = total / n
    return mean

def find_median(list1):
    list1.sort()
    print("Sorted list elements are :: ",list1)
    if n % 2 == 0:
        median = (list1[n // 2] + list1[n // 2 - 1]) / 2
    else:
        median = list1[n // 2]
    return median

def find_mode(list1):
    unq_list = []
    for ele in list1:
        if ele not in unq_list:
            unq_list.append(ele)
    max_count = 0
    mode_list = []
    for ele in unq_list:
        currentCount = list1.count(ele)
        if currentCount > max_count:
            max_count = currentCount
            mode_list = [ele]
        elif currentCount == max_count:
            mode_list.append(ele)
    return mode_list
```

```
list1 = []
n = int(input("Enter number of elements :: "))
for i in range(n) :
   list1.append(int(input("Enter a number :: ")))

print("List elements are :: ",list1)
print("Mean (built-in) : ", np.mean(list1))
print("Mean (without built-in) :",find_mean(list1))
print("Median (built-in) : ", np.median(list1))
print("Median (without built-in) :",find_median(list1))
print("Mode (built-in) : ", statistics.multimode(list1))
print("Mode (without built-in) :",find_mode(list1))
```

Output:

```
Enter number of elements :: 7
Enter a number :: 1
Enter a number :: 1
Enter a number :: 2
Enter a number :: 5
Enter a number :: 2
Enter a number :: 8
Enter a number :: 4
List elements are ::  [1, 1, 2, 5, 2, 8, 4]
Mean (built-in) :  3.2857142857142856
Mean (without built-in) : 3.2857142857142856
Median (built-in) :  2.0
Sorted list elements are ::  [1, 1, 2, 2, 4, 5, 8]
Median (without built-in) : 2
Mode (built-in) :  [1, 2]
Mode (without built-in) : [1, 2]
```

3. Implement a program to perform measure of dispersion (range, variance, standard deviation, IQR), with and without using built-in function on the data.

```
import numpy as np
import math
from scipy import stats

def find_range(list1):
   maxElement = minElement = list1[0]
   for ele in list1:
      if ele > maxElement:
         maxElement = ele
      if ele < minElement:
         minElement = ele
   rangeValue = maxElement - minElement
   return rangeValue

def find_variance(list1):
   total = 0
   for ele in list1:
      total += ele
   mean = total / len(list1)
   sumValue = 0
   for i in list1:
      sumValue += (i - mean) ** 2
   variance = sumValue / len(list1)
   return variance

def find_sd(list1):
   sd = math.sqrt(find_variance(list1))
   return sd

def percentile_midpoint(data, percent):
   sorted_data = sorted(data)
   n = len(sorted_data)

   # Calculate the position of the percentile
   k = (n - 1) * percent
```

```
    f = int(k)  # floor value
    c = k - f   # fractional part

    if c == 0:
        # If k is an integer, return the exact value at that position
        return sorted_data[f]
    else:
        # Midpoint interpolation between the two closest ranks
        return (sorted_data[f] + sorted_data[f + 1]) / 2

def find_iqr_midpoint(data):
    Q1 = percentile_midpoint(data, 0.25)  # Calculate the 25th percentile
    Q3 = percentile_midpoint(data, 0.75)  # Calculate the 75th percentile
    IQR = Q3 - Q1  # Calculate the IQR
    return IQR

list1 = []
n=int(input("Enter the number of elements ::"))
for i in range(n):
    ele = int(input("Enter the elements ::"))
    list1.append(ele)
print("List elements are\n", list1)
print("The range(without built-in):: ", find_range(list1))
maximum = np.max(list1)
minimum = np.min(list1)
range_value = maximum – minimum
print("The range(with built-in):: ", range_value)
print("The variance(without built-in):: ", find_variance(list1))
print("The variance(with built-in):: ", np.var(list1))
print("The standard deviation(without built-in):: ", find_sd(list1))
print("The standard deviation(with built-in):: ", np.std(list1))
iqr = find_iqr_midpoint(list1)
print("Interquartile Range(without built-in function):", iqr)
IQR = stats.iqr(list1, interpolation='midpoint')
print("IQR(with built-in function):", IQR)
```

Output:

```
Enter the number of elements ::7
Enter the elements ::5
Enter the elements ::2
Enter the elements ::7
Enter the elements ::13
Enter the elements ::8
Enter the elements ::6
Enter the elements ::1
List elements are
 [5, 2, 7, 13, 8, 6, 1]
The range(without built-in)::  12
The range(with built-in)::  12
The variance(without built-in)::  13.714285714285714
The variance(with built-in)::  13.714285714285714
The standard deviation(without built-in)::  3.7032803990902057
The standard deviation(with built-in)::  3.7032803990902057
Interquartile Range(without built-in function): 4.0
IQR(with built-in function): 4.0
```

4. Write a program to perform text data pre-processing with and without using built-in functions.

```
#With built-in
import pandas as pd
import spacy
import string
import contractions
import re
import nltk
import emoji

from nltk import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from nltk.stem import PorterStemmer
from num2words import num2words

nltk.download('punkt')
nltk.download('stopwords')
nltk.download('whitespace')
nltk.download('wordnet')
nltk.download('omw-1.4')
stop_words = set(stopwords.words("english"))

data = pd.read_csv('Training.tsv',sep='\t')

data.head(10)
```

Output:

	tweet_id	text	label
0	1382343793341575169	@IrvineWelsh I don’t know about you Irvine but...	0
1	1377631738692796417	I bet money if i went n took a covid test righ...	0
2	1386448010029240326	@JamesMelville My wife received a POSITIVE Cov...	0

3	1361342676340211717	Out of the 180,000+ people who have had the tw...	0
4	1386757983254765569	My whole family is sick af and here I am now i...	0
5	1382001700853125122	@renfrew1962 @PeakePolly @J_Deliciouso I'm not...	0
6	1383272654212272136	Test came back positive, no surprise. I have c...	1
7	1374479299047084035	My Pawpaw has been in the hospital a few days....	0
8	1354020426620547072	@MattHancock 4 people I know had covid and rec...	0
9	1362671045136809985	I’m going to sound like I have lost my marbles...	1

```
data['label'].value_counts()
```

Output:

```
0   6266
1   1334
Name: label, dtype: int64
```

```
ps =PorterStemmer()
lemmatiser = WordNetLemmatizer()
english_stopwords = stopwords.words('english')
exclude = set(string.punctuation)
def preprocess(text):
  #text=demoji.findall(df['Text'])
  text = contractions.fix(text.lower(), slang=True)
  text = re.sub(r'\d+', lambda x: num2words(int(x.group(0))), text)
  #text= re.sub(r'\d+', '', text)
  text=re.sub(r'$', '', text)
  text= re.sub(r'”','', text )
  text=re.sub('<.*?>','',text)
  text=re.sub(r'http\S+', '', text)
  #text=emoji.demojize(text, delimiters=(" ", " "))
  text = ''.join(ch for ch in text if ch not in exclude)
  tokens = word_tokenize(text)
  #print("Tokens:", tokens)
  text = [t for t in tokens if t not in english_stopwords]
```

```
    text = " ".join(text)
    return text


import emoji
#import demoji
#demoji.download_codes()
def emo(text):
    temp=emoji.demojize(text,delimiters=(" "," "))
    temp=temp.replace("_","  ")
    return temp
data['emo']=data["text"].apply(lambda x:emo(x))
data["clean_text"]=data['emo'].apply(lambda X: preprocess(X))

data.head()
```

Output:

	tweet_id	text	label	emo	clean_text
0	1382343793341575169	@IrvineWelsh I don’t know about you Irvine but...	0	@IrvineWelsh I don’t know about you Irvine but...	irvinewelsh know irvine keep told covid exist ...
1	1377631738692796417	I bet money if i went n took a covid test righ...	0	I bet money if i went n took a covid test righ...	bet money went n took covid test right going t...
2	1386448010029240326	@JamesMelville My wife received a POSITIVE Cov...	0	@JamesMelville My wife received a POSITIVE Cov...	jamesmelville wife received positive covid tes...
3	1361342676340211717	Out of the 180,000+ people who have had the tw...	0	Out of the 180,000+ people who have had the tw...	one hundred eightyzero people two vaccine shot...
4	1386757983254765569	My whole family is sick af and here I am now i...	0	My whole family is sick af and here I am now i...	whole family sick af hospital heart palpitatio...

#Without built-in

```
import pandas as pd
data = pd.read_csv('Training.tsv',sep='\t')
data
```

Output:

	tweet_id	text	label
0	1382343793341575169	@IrvineWelsh I don’t know about you Irvine but...	0
1	1377631738692796417	I bet money if i went n took a covid test righ...	0
2	1386448010029240326	@JamesMelville My wife received a POSITIVE Cov...	0
3	1361342676340211717	Out of the 180,000+ people who have had the tw...	0
4	1386757983254765569	My whole family is sick af and here I am now i...	0
5	1382001700853125122	@renfrew1962 @PeakePolly @J_Deliciouso I'm not...	0
6	1383272654212272136	Test came back positive, no surprise. I have c...	1
7	1374479299047084035	My Pawpaw has been in the hospital a few days....	0
8	1354020426620547072	@MattHancock 4 people I know had covid and rec...	0
9	1362671045136809985	I’m going to sound like I have lost my marbles...	1

```
# Define English stopwords
english_stopwords = set(['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're",
"you've","you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself','she',
"she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their', 'such', 'no', 'nor',
'not','only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can','will', 'just', 'don', "don't", 'should',
"should've", 'now', 'd', 'll', 'm', 'o', 're', 've','wasn', "wasn't", 'weren', "weren't", 'won', "won't",
'wouldn', "wouldn't"])

def normalize_apostrophes(text):
   # Replace different representations of apostrophes with a single consistent representation
   text = text.replace("’", "'")  # Replace curly apostrophe with straight apostrophe
   return text
```

```
# Define function to preprocess text
def preprocess_text(text):
    # Lowercasing apostrophe
    text = to_lowercase(text)
    #Normalising
    text = normalize_apostrophes(text)
    # Removing Contractions
    text = remove_contraction(text)
    # Converting number to words
    text = convert_numbers_to_words(text)
    # Removing URLs
    text = remove_urls(text)
    # Removing special characters
    text = remove_special_characters(text)
   # Tokenization and removing stopwords
    tokens = text.split()
    tokens = [token for token in tokens if token not in english_stopwords]
   # Joining tokens
    text = ' '.join(tokens
    return text

def remove_contraction(text):
   # Define contractions
    contractions = {
        "ain't": "am not / is not / are not / has not / have not",
        "aren't": "are not",
        "can't": "cannot",
        "could've": "could have",
        "couldn't": "could not",
        "didn't": "did not",
        "doesn't": "does not",
        "don't": "do not",
        "hadn't": "had not",
       "hasn't": "has not",
        "haven't": "have not",
         "i'll": "i will",
        "i'll've": "i will have",
        "i'm": "i am",
        "i've": "i have",
        "isn't": "is not",
        "it'll": "it will",
        "it's": "it is / it has",
        "let's": "let us",
         "we've": "we have",
        "weren't": "were not",
        "what'll": "what will",
        "what're": "what are",
        "what's": "what is / what has",
         "you're": "you are",
        "you've": "you have"
    }
     # Expanding contractions
    for contraction, expansion in contractions.items():
        text = text.replace(contraction, expansion)
    return text

# Function to remove convert number to words
def convert_numbers_to_words(text):
 # Define a dictionary mapping numeric words to their corresponding words
    num_words = {
        '0': 'zero',
        '1': 'one',
        '2': 'two',
        '3': 'three',
        '4': 'four',
        '5': 'five',
        '6': 'six',
        '7': 'seven',
        '8': 'eight',
        '9': 'nine'
    }
     # Converting numbers to words
    for digit, word in num_words.items():
        text = text.replace(digit, word)
    return text
```

```
def to_lowercase(text):
    lowercase_text = ''
    for char in text:
        # Check if character is uppercase
        if 'A' <= char <= 'Z':
            # Convert uppercase to lowercase
            lowercase_text += chr(ord(char) + 32)
        else:
            lowercase_text += char
    return lowercase_text

# Function to remove special characters
def remove_special_characters(text):
    # Define special characters
    special_chars = {'!', '"', '#', '$', '%', '&', "'", '(', ')', '*', '+', ',', '-', '.', '/', ':', ';', '<', '=',
                     '>', '?', '@', '[', '\\', ']', '^', '_', '`', '{', '|', '}', '~'}
    return ''.join(char for char in text if char not in special_chars)

# Function to remove URLs
def remove_urls(text):
    # Split text into words
    words = text.split()
    # Filter out words that do not start with 'http' or 'https'
    filtered_words = [word for word in words if not (word.startswith('http://') or
word.startswith('https://'))]
    # Join the filtered words back into a string
    return ' '.join(filtered_words)

import emoji
#import demoji
#demoji.download_codes()
def emo(text):
    temp=emoji.demojize(text,delimiters=(" "," "))
    temp=temp.replace("_","  ")
    return temp

data['emo']=data["text"].apply(lambda x:emo(x))
```

```
data["clean_text"]=data['emo'].apply(lambda X: preprocess_text(X))
```

```
data.head()
```

Output:

	tweet_id	text	label	emo	clean_text
0	1382343793341575169	@IrvineWelsh I don’t know about you Irvine but...	0	@IrvineWelsh I don’t know about you Irvine but...	irvinewelsh know irvine keep told covid exist ...
1	1377631738692796417	I bet money if i went n took a covid test righ...	0	I bet money if i went n took a covid test righ...	bet money went n took covid test right imma te...
2	1386448010029240326	@JamesMelville My wife received a POSITIVE Cov...	0	@JamesMelville My wife received a POSITIVE Cov...	jamesmelville wife received positive covid tes...
3	1361342676340211717	Out of the 180,000+ people who have had the tw...	0	Out of the 180,000+ people who have had the tw...	oneeightzerozerozerozero people two vaccine sh...
4	1386757983254765569	My whole family is sick af and here I am now i...	0	My whole family is sick af and here I am now i...	whole family sick af hospital heart palpitatio...

5. Write a program to perform Numeric data per-processing with and without using built-in functions.

```
#With built-in
import numpy as np
import pandas as pd

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

Output:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6.0	148	72.0	35.0	0.0	33.6	0.627	50	1
1	1.0	85	66.0	29.0	0.0	26.6	0.351	31	0
2	8.0	183	64.0	0.0	0.0	23.3	0.672	32	1
3	1.0	89	66.0	23.0	94.0	28.1	0.167	21	0
4	NaN	137	40.0	35.0	168.0	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
764	10.0	101	76.0	48.0	180.0	32.9	0.171	63	0
765	2.0	122	70.0	27.0	0.0	36.8	0.340	27	0
766	5.0	121	72.0	23.0	112.0	26.2	0.245	30	0
767	1.0	126	60.0	0.0	0.0	30.1	0.349	47	1
768	1.0	93	70.0	31.0	0.0	30.4	0.315	23	0

```
df.shape
```

Output:

(769, 9)

```
df.describe()
```

Output:

	Pregnancies	**Glucose**	**BloodPressure**	**SkinThickness**	**Insulin**	**BMI**	**DiabetesPedigreeFunction**	**Age**	**Outcome**
count	768.000000	769.000000	768.000000	768.000000	768.000000	769.000000	769.000000	769.000000	769.000000
mean	3.846354	120.847854	69.191406	20.574219	79.799479	31.985566	0.471719	33.237971	0.348505
std	3.368283	31.978003	19.194430	15.937859	115.244002	7.881425	0.331142	11.752850	0.476807
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	63.500000	0.000000	0.000000	27.300000	0.244000	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.371000	29.000000	0.000000
75%	6.000000	140.000000	80.000000	32.000000	127.250000	36.600000	0.626000	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

```
#seperate features and class label
features = df.iloc[:, :-1]
class_label = df.iloc[:, -1]

def find_duplicates(data):
    duplicate_rows = data[data.duplicated()]
    return duplicate_rows

duplicate_values = find_duplicates(features)
```

```
print("Duplicates values:")
print(duplicate_values)
```

Output:

```
Duplicates values:
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
6          1.0       85           66.0           29.0      0.0  26.6

   DiabetesPedigreeFunction  Age
6                     0.351   31
```

```
def remove_duplicates(data):
    unique_data = data.drop_duplicates()
    return unique_data
features = remove_duplicates(features)
print("Data after removing duplicates :")
print(features)
```

Output:

```
Data after removing duplicates :
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6.0      148           72.0           35.0      0.0  33.6
1            1.0       85           66.0           29.0      0.0  26.6
2            8.0      183           64.0            0.0      0.0  23.3
3            1.0       89           66.0           23.0     94.0  28.1
4            NaN      137           40.0           35.0    168.0  43.1
..           ...      ...            ...            ...      ...   ...
764         10.0      101           76.0           48.0    180.0  32.9
765          2.0      122           70.0           27.0      0.0  36.8
766          5.0      121           72.0           23.0    112.0  26.2
767          1.0      126           60.0            0.0      0.0  30.1
768          1.0       93           70.0           31.0      0.0  30.4

     DiabetesPedigreeFunction  Age
0                       0.627   50
1                       0.351   31
2                       0.672   32
3                       0.167   21
```

```
4                    2.288   33
..                    ...  ...
764                   0.171   63
765                   0.340   27
766                   0.245   30
767                   0.349   47
768                   0.315   23
```

```
def find_number_of_missing_values(data):
    missing_values = data.isnull().sum()

    # Filter out columns with missing values
    missing_values = missing_values[missing_values > 0]
    return missing_values

missing_values = find_number_of_missing_values(features)
# Print columns with missing values and their respective counts
print("Columns with missing values:")
print(missing_values)
```

Output:

```
Columns with missing values:
Pregnancies      1
BloodPressure    1
SkinThickness    1
Insulin          1
dtype: int64
```

```
# 1. Handling missing values
def handle_missing_values(data, strategy='mean'):
   if strategy == 'mean':
      return data.fillna(data.mean())

   elif strategy == 'max':
      return data.fillna(data.max())

   elif strategy == 'min':
      return data.fillna(data.min())

   elif strategy == 'zero':
      return data.fillna(0)

   elif strategy == 'drop':
      return data.dropna()

features = handle_missing_values(features)
print("Data after handling missing values:")
print(features)
```

Output:

```
Data after handling missing values:
    Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0      6.000000     148           72.0           35.0      0.0  33.6
1      1.000000      85           66.0           29.0      0.0  26.6
2      8.000000     183           64.0            0.0      0.0  23.3
3      1.000000      89           66.0           23.0     94.0  28.1
4      3.850065     137           40.0           35.0    168.0  43.1
..          ...      ...           ...            ...      ...   ...
764   10.000000     101           76.0           48.0    180.0  32.9
765    2.000000     122           70.0           27.0      0.0  36.8
766    5.000000     121           72.0           23.0    112.0  26.2
767    1.000000     126           60.0            0.0      0.0  30.1
768    1.000000      93           70.0           31.0      0.0  30.4

    DiabetesPedigreeFunction  Age
0                      0.627   50
```

```
1                0.351   31
2                0.672   32
3                0.167   21
4                2.288   33
..                ...  ...
764               0.171   63
765               0.340   27
766               0.245   30
767               0.349   47
768               0.315   23
```

#Without built-in

```
import numpy as np
import pandas as pd

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

Output:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6.0	148	72.0	35.0	0.0	33.6	0.627	50	1
1	1.0	85	66.0	29.0	0.0	26.6	0.351	31	0
2	8.0	183	64.0	0.0	0.0	23.3	0.672	32	1
3	1.0	89	66.0	23.0	94.0	28.1	0.167	21	0
4	NaN	137	40.0	35.0	168.0	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
764	10.0	101	76.0	48.0	180.0	32.9	0.171	63	0
765	2.0	122	70.0	27.0	0.0	36.8	0.340	27	0
766	5.0	121	72.0	23.0	112.0	26.2	0.245	30	0

767	1.0	126	60.0	0.0	0.0	30.1	0.349	47	1
768	1.0	93	70.0	31.0	0.0	30.4	0.315	23	0

df.shape

Output:

(770, 9)

df.describe()

Output:

	Pregnancies	**Glucose**	**BloodPressure**	**SkinThickness**	**Insulin**	**BMI**	**DiabetesPedigreeFunction**	**Age**	**Outcome**
count	768.000000	769.000000	768.000000	768.000000	768.000000	769.000000	769.000000	769.000000	769.000000
mean	3.846354	120.847854	69.191406	20.574219	79.799479	31.985566	0.471719	33.237971	0.348505
std	3.368283	31.978003	19.194430	15.937859	115.244002	7.881425	0.331142	11.752850	0.476807
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	63.500000	0.000000	0.000000	27.300000	0.244000	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.371000	29.000000	0.000000
75%	6.000000	140.000000	80.000000	32.000000	127.250000	36.600000	0.626000	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

```
#seperate features and class label
features = df.iloc[:, :-1]
class_label = df.iloc[:, -1]

import pandas as pd
def find_duplicates(data):
    duplicate_rows = []
    seen_rows = set()  # To track rows that we have already seen
    # Iterate through each row in the DataFrame
    for index, row in data.iterrows():
        # Convert the row to a tuple to make it hashable
        row_tuple = tuple(row)
        # Check if this row tuple has already been seen
        if row_tuple in seen_rows:
            # Append the duplicate row as a Series object
            duplicate_rows.append(row)
        else:
            seen_rows.add(row_tuple)  # Add the row tuple to the set of seen rows

    # Create a DataFrame from the list of duplicate rows
    columns = data.columns
    duplicate_df = pd.DataFrame(duplicate_rows, columns=columns)
    return duplicate_df

duplicate_values = find_duplicates(features)
print("Duplicates values:")
print(duplicate_values)
```

Output:

```
Duplicates values:
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
5          6.0    148.0           72.0           35.0      0.0  33.6
7          1.0     85.0           66.0           29.0      0.0  26.6

   DiabetesPedigreeFunction   Age
5                     0.627  50.0
7                     0.351  31.0
```

```
def remove_duplicates(data):
    seen_rows = set()
    unique_data = []

    for index, row in data.iterrows():
        row_tuple = tuple(row)
        if row_tuple not in seen_rows:
            seen_rows.add(row_tuple)
            unique_data.append(row)

    # Convert list of rows back to DataFrame
    unique_data_df = pd.DataFrame(unique_data, columns=data.columns)
    return unique_data_df

features = remove_duplicates(features)
print("Data after removing duplicates :")
print(features)
```

Output:

```
Data after removing duplicates :
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6.0    148.0           72.0           35.0      0.0  33.6
1            1.0     85.0           66.0           29.0      0.0  26.6
2            8.0    183.0           64.0            0.0      0.0  23.3
3            1.0     89.0           66.0           23.0     94.0  28.1
4            NaN    137.0           40.0           35.0    168.0  43.1
..           ...      ...            ...            ...      ...   ...
765         10.0    101.0           76.0           48.0    180.0  32.9
766          2.0    122.0           70.0           27.0      0.0  36.8
767          5.0    121.0           72.0           23.0    112.0  26.2
768          1.0    126.0           60.0            0.0      0.0  30.1
769          1.0     93.0           70.0           31.0      0.0  30.4

     DiabetesPedigreeFunction   Age
0                       0.627  50.0
1                       0.351  31.0
2                       0.672  32.0
3                       0.167  21.0
```

```
4                 2.288  33.0
..                  ...   ...
765                 0.171  63.0
766                 0.340  27.0
767                 0.245  30.0
768                 0.349  47.0
769                 0.315  23.0
```

```
def find_number_of_missing_values(data):
    # Create an empty dictionary to store column names and their respective counts of missing values
    missing_values = {}

    # Iterate through each column in the DataFrame
    for column in data.columns:
        # Count the number of missing values in the column
        missing_count = sum(1 for value in data[column] if pd.isna(value))

        # If there are missing values in the column, add it to the dictionary
        if missing_count > 0:
            missing_values[column] = missing_count

    # Print columns with missing values and their respective counts
    print("Columns with missing values:")
    for column, count in missing_values.items():
        print(f"{column}: {count}")

missing_values = find_number_of_missing_values(features)
```

Output:

```
Columns with missing values:
Pregnancies: 1
BloodPressure: 1
SkinThickness: 1
Insulin: 1
```

```
import pandas as pd
import numpy as np

def handle_missing_values(data, strategy='mean'):
    num_cols = data.shape[1]  # Number of columns
    filled_data = data.copy()  # Create a copy to modify

    if strategy == 'mean':
        # Calculate column means
        col_means = [np.mean(data.iloc[:, col]) for col in range(num_cols)]

        # Replace NaN values with column means
        for col in range(num_cols):
            col_mean = col_means[col]
            for row in range(len(data)):
                if pd.isna(data.iloc[row, col]):
                    filled_data.iloc[row, col] = col_mean

    elif strategy == 'max':
        # Calculate column max values
        col_max = [np.max(data.iloc[:, col]) for col in range(num_cols)]

        # Replace NaN values with column max values
        for col in range(num_cols):
            col_max_value = col_max[col]
            for row in range(len(data)):
                if pd.isna(data.iloc[row, col]):
                    filled_data.iloc[row, col] = col_max_value

    elif strategy == 'min':
        # Calculate column min values
        col_min = [np.min(data.iloc[:, col]) for col in range(num_cols)]

        # Replace NaN values with column min values
        for col in range(num_cols):
            col_min_value = col_min[col]
            for row in range(len(data)):
                if pd.isna(data.iloc[row, col]):
                    filled_data.iloc[row, col] = col_min_value
    elif strategy == 'zero':
        # Replace NaN values with 0
        for col in range(num_cols):
            for row in range(len(data)):
                # Check if the value is NaN
                if pd.isna(data.iloc[row, col]):
                    # If it is NaN, replace it with 0
                    filled_data.iloc[row, col] = 0

    elif strategy == 'drop':
        # Drop rows with NaN values
        filled_data = data.dropna()

    return filled_data

features = handle_missing_values(features, 'max')
print("Data after handling missing values:")
print(features)
```

Output:

```
Data after handling missing values:
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6.0    148.0           72.0           35.0      0.0  33.6
1            1.0     85.0           66.0           29.0      0.0  26.6
2            8.0    183.0           64.0            0.0      0.0  23.3
3            1.0     89.0           66.0           23.0     94.0  28.1
4           17.0    137.0           40.0           35.0    168.0  43.1
..           ...      ...            ...            ...      ...   ...
765         10.0    101.0           76.0           48.0    180.0  32.9
766          2.0    122.0           70.0           27.0      0.0  36.8
767          5.0    121.0           72.0           23.0    112.0  26.2
768          1.0    126.0           60.0            0.0      0.0  30.1
769          1.0     93.0           70.0           31.0      0.0  30.4

     DiabetesPedigreeFunction   Age
0                       0.627  50.0
1                       0.351  31.0
```

```
2                  0.672  32.0
3                  0.167  21.0
4                  2.288  33.0
..                  ...   ...
765                  0.171  63.0
766                  0.340  27.0
767                  0.245  30.0
768                  0.349  47.0
769                  0.315  23.0
```

6.. Write a python program to read and display various kinds of data (image, text, and numeric) saved in different format using various python libraries.

Image

```
#code to read and display .png file
import cv2

image=cv2.imread('flower.png')
cv2.imshow("image", image)
cv2.waitKey(0)
cv2.destroyAllWindows()


#code to read and display .jpg file
import cv2
image=cv2.imread('dog.jpg')
cv2.imshow("image", image)
cv2.waitKey(0)
cv2.destroyAllWindows()

#code to read and display .gif file
import cv2
def show_gif(file_path):
    cap = cv2.VideoCapture(file_path)
    while True:
        ret, frame = cap.read()
        if not ret:
            break
        cv2.imshow('GIF Viewer', frame)
        if cv2.waitKey(100) & 0xFF == ord('q'):
            break
    cap.release()
    cv2.destroyAllWindows()

show_gif("moon.gif")
```

Numeric Data

```
#code to read and display .csv file
import pandas as pd
csvdata=pd.read_csv("headbrain.csv")
csvdata
```

Output:

	Gender	Age Range	Head Size(cm^3)	Brain Weight(grams)
0	1.0	1.0	4512	1530.0
1	NaN	1.0	3738	1297.0
2	1.0	1.0	4261	1335.0
3	1.0	1.0	3777	1282.0
4	1.0	1.0	4177	NaN
...	...	...	...	...
232	2.0	2.0	3214	1110.0
233	2.0	2.0	3394	1215.0
234	2.0	2.0	3233	1104.0
235	2.0	2.0	3352	1170.0
236	2.0	2.0	3391	1120.0

```
#code to read and display .tsv file
import pandas as pd
tsvdata=pd.read_csv("file.tsv", sep = "\t")
tsvdata
```

Output:

	0	50	5	881250949
0	0	172	5	881250949
1	0	133	1	881250949
2	196	242	3	881250949
3	186	302	3	891717742
4	22	377	1	878887116
...	...	...	...	...
99997	880	476	3	880175444
99998	716	204	5	879795543
99999	276	1090	1	874795795
100000	13	225	2	882399156

100001	12	203	3	879959583

```
#code to read and display excel
import pandas as pd
exceldata=pd.read_excel("exceldata.xlsx", names = ["Number 1" , "Number 2"])
exceldata
```

Output:

	Number 1	**Number 2**
0	5.5277	9.13020
1	8.5186	13.66200
2	7.0032	11.85400
3	5.8598	6.82330
4	8.3829	11.88600
...	...	...
91	5.8707	7.20290
92	5.3054	1.98690
93	8.2934	0.14454
94	13.3940	9.05510
95	5.4369	0.61705

Text Data

```
#code to read and display .txt file
import pandas as pd
txtdata= pd.read_csv("records.txt", sep=" ")
Txtdata
```

Output:

```
Python is a dynamic, high-level, free open source, and interpreted programming language. It
supports object-oriented programming as well as procedural-oriented programming.
```

```
#code to read and display json file
import json
with open("sample1-json.json", 'r') as f:
   json_ob = json.load(f)
print(json_ob)
```

Output:

```
{'fruit': 'Apple', 'size': 'Large', 'color': 'Red'}
```

7. Write a python program to read and display video and audio data.

```
#Audio
import librosa
from IPython.display import Audio

# Load audio file
audio_path = "sample-file-4.wav"
y, sr = librosa.load(audio_path)

# Play audio
Audio(data=y, rate=sr)

#Video
import cv2

# Path to the video file
video_path = "file_example.mp4"

# Open the video file
cap = cv2.VideoCapture(video_path)

# Check if the video opened successfully
if not cap.isOpened():
    print("Error: Could not open the video.")
else:
    # Create a flag to track window status
    window_open = True

    # Loop through each frame in the video
    while window_open:
        # Read a frame from the video
        ret, frame = cap.read()

        # If the frame was read successfully
        if ret:
            # Display the frame
            cv2.imshow('Video', frame)
            # Check for the 'q' key to quit
            if cv2.waitKey(25) & 0xFF == ord('q'):
                break
        else:
            # Break the loop if the video has ended
            break

        # Check if the window is still open
        if cv2.getWindowProperty('Video', cv2.WND_PROP_VISIBLE) < 1:
            window_open = False

    # Release the video capture object and close the window
    cap.release()
    cv2.destroyAllWindows()
```

8. Write a program to implement a Naive Bayes classifier for sample training dataset. Also plot the confusion matrix to evaluate the classifier's performance.

```
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
dataset = pd.read_csv('Social_Network_Ads.csv')
dataset
```

Output:

	User ID	**Gender**	**Age**	**EstimatedSalary**	**Purchased**
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
...	...	...	...	...	...
395	15691863	Female	46	41000	1
396	15706071	Male	51	23000	1
397	15654296	Female	50	20000	1
398	15755018	Male	36	33000	0
399	15594041	Female	49	36000	1

```
X = dataset.iloc[:, [2,3]].values
y = dataset.iloc[:,4].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB(priors=[0.4, 0.6], var_smoothing=1e-9)
classifier.fit(X_train, y_train)
```

Output:

```
GaussianNB(priors=[0.4, 0.6])
```

```
#changing hyperparameter values
from sklearn.naive_bayes import GaussianNB

# Example hyperparameter values
custom_priors = [0.3, 0.7]  # Custom priors for classes
custom_var_smoothing = 1e-8  # Custom var_smoothing value

# Initialize Gaussian Naive Bayes classifier with custom hyperparameters
classifier = GaussianNB(priors=custom_priors, var_smoothing=custom_var_smoothing)

# Assuming X_train and y_train are your training data
classifier.fit(X_train, y_train)
```

Output:

```
GaussianNB(priors=[0.3, 0.7], var_smoothing=1e-08)
```

```
y_pred = classifier.predict(X_test)

y_pred
```

Output:

```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1,
    1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,
    1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1,
    0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1,
    1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1], dtype=int64)
```

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)

cm
```

Output:

```
array([[53, 15],
       [ 1, 31]], dtype=int64)
```

```
from sklearn.metrics import accuracy_score
print("The accuracy score is:", accuracy_score(y_pred, y_test))
```

Output:

The accuracy score is: 0.84

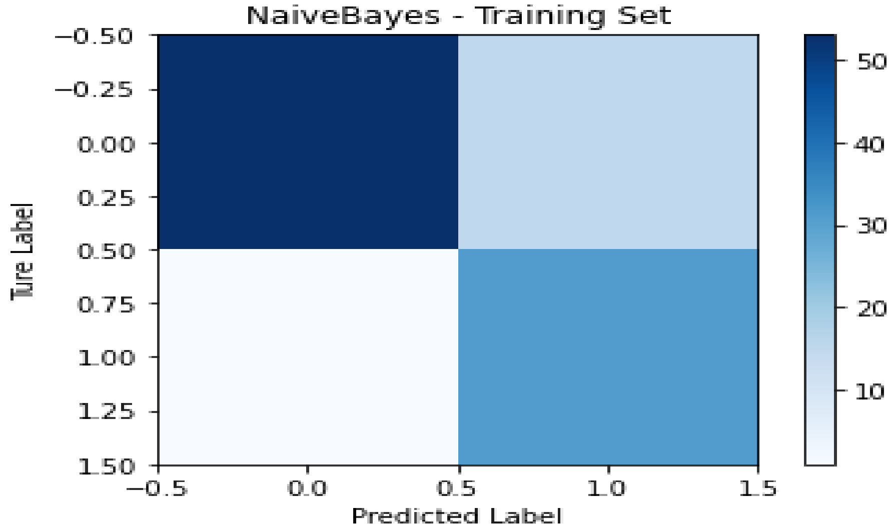
```
from sklearn.metrics import classification_report
print("classification_report:")
print( classification_report(y_pred, y_test))
```

Output:

```
classification_report:
              precision    recall  f1-score   support

           0       0.78      0.98      0.87        54
           1       0.97      0.67      0.79        46

    accuracy                           0.84       100
   macro avg       0.87      0.83      0.83       100
weighted avg       0.87      0.84      0.83       100
```

```
plt.imshow(cm,interpolation='nearest',cmap=plt.cm.Blues)
plt.title('NaiveBayes - Training Set')
plt.xlabel('Predicted Label')
plt.ylabel('Ture Label')
plt.colorbar()
plt.show()
```

Output:



9. Write a program to implement a Support Vector Machine (SVM) classifier sample training dataset. Fine-tune various hyperparameters and assess the classifier's performance on the dataset.

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

dataset = pd.read_csv('Social_Network_Ads.csv')
Dataset
```

Output:

User ID	Gender	Age	EstimatedSalary	Purchased	
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
...	...	...	...	...	...
395	15691863	Female	46	41000	1
396	15706071	Male	51	23000	1
397	15654296	Female	50	20000	1
398	15755018	Male	36	33000	0
399	15594041	Female	49	36000	1

```
# converting gender column to numeric
from sklearn.preprocessing import LabelEncoder
label_encoder=LabelEncoder()
dataset['Gender']=label_encoder.fit_transform(dataset['Gender'])
dataset['Gender'].unique()
```

Output:

```
array([1, 0])
```

```
# to include gender
X = dataset.iloc[:, [1, 3]].values
```

```
y = dataset.iloc[:,4].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)

# X = dataset.iloc[:, [2, 3]].values
# y = dataset.iloc[:,4].values
# X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)

# sc = StandardScaler()
# X_train = sc.fit_transform(X_train)
# X_test = sc.transform(X_test)

from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

from sklearn.svm import SVC
classifier = SVC(kernel='poly', random_state=0)
classifier.fit(X_train, y_train)
```

Output:

```
SVC(kernel='poly', random_state=0)
```

```
y_pred = classifier.predict(X_test)

y_pred
```

Output:

```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1,
    0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1,
    0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1], dtype=int64)
```

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

```
cm
```

Output:

```
array([[66,  2],
       [19, 13]], dtype=int64)
```

```
from sklearn.metrics import accuracy_score
print("The accuracy score is:", accuracy_score(y_pred, y_test))
```

Output:

```
The accuracy score is: 0.79
```

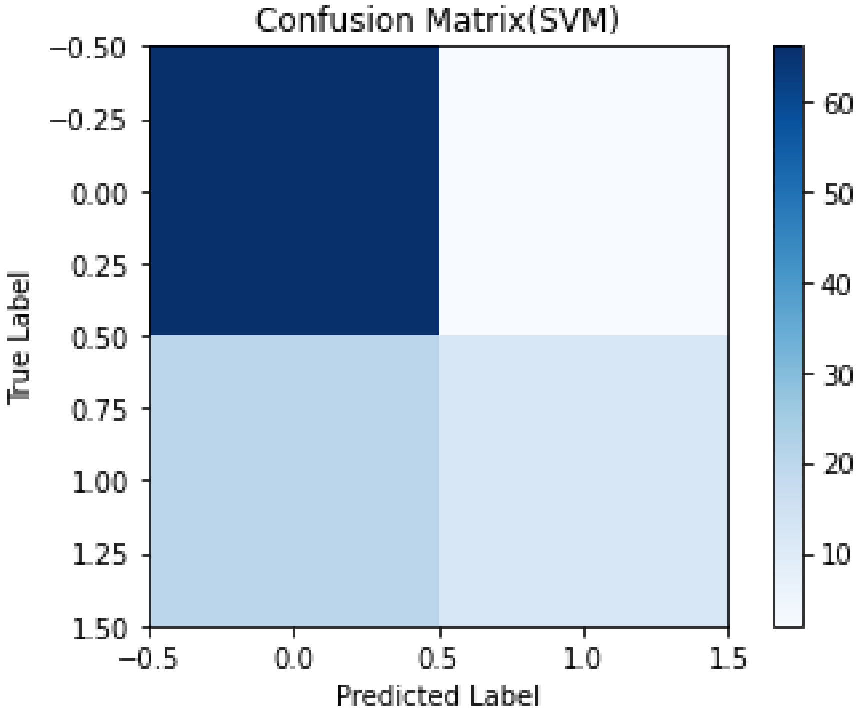
```
from sklearn.metrics import classification_report
print("classification_report:")
print( classification_report(y_pred, y_test))
```

Output:

```
classification_report:
              precision    recall  f1-score   support

           0       0.97      0.78      0.86        85
           1       0.41      0.87      0.55        15

    accuracy                           0.79       100
   macro avg       0.69      0.82      0.71       100
weighted avg       0.89      0.79      0.82       100
```

```
plt.imshow(cm,interpolation='nearest',cmap=plt.cm.Blues)
plt.title('Confusion Matrix(SVM)')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.colorbar()

# Show the plot
plt.show()
```

Output:


Confusion Matrix(SVM)
True Label
Predicted Label
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
1.50
−0.5
0.0
0.5
1.0
1.5
60
50
40
30
20
10

10. Write a program to implement Decision Tree classifier. Experiment with different hyperparameters to evaluate and optimize the classifier's performance.

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

dataset = pd.read_csv('Social_Network_Ads.csv')
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:,4].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)

sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

dataset
```

Output:

	UserID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
...	...	...	...	...	...
395	15691863	Female	46	41000	1
396	15706071	Male	51	23000	1
397	15654296	Female	50	20000	1
398	15755018	Male	36	33000	0
399	15594041	Female	49	36000	1

```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion='gini', random_state=0)
classifier.fit(X_train, y_train)
```

Output:

```
DecisionTreeClassifier(random_state=0)
```

```
y_pred = classifier.predict(X_test)

y_pred
```

Output:

```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1,
       0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1,
       1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1], dtype=int64)
```

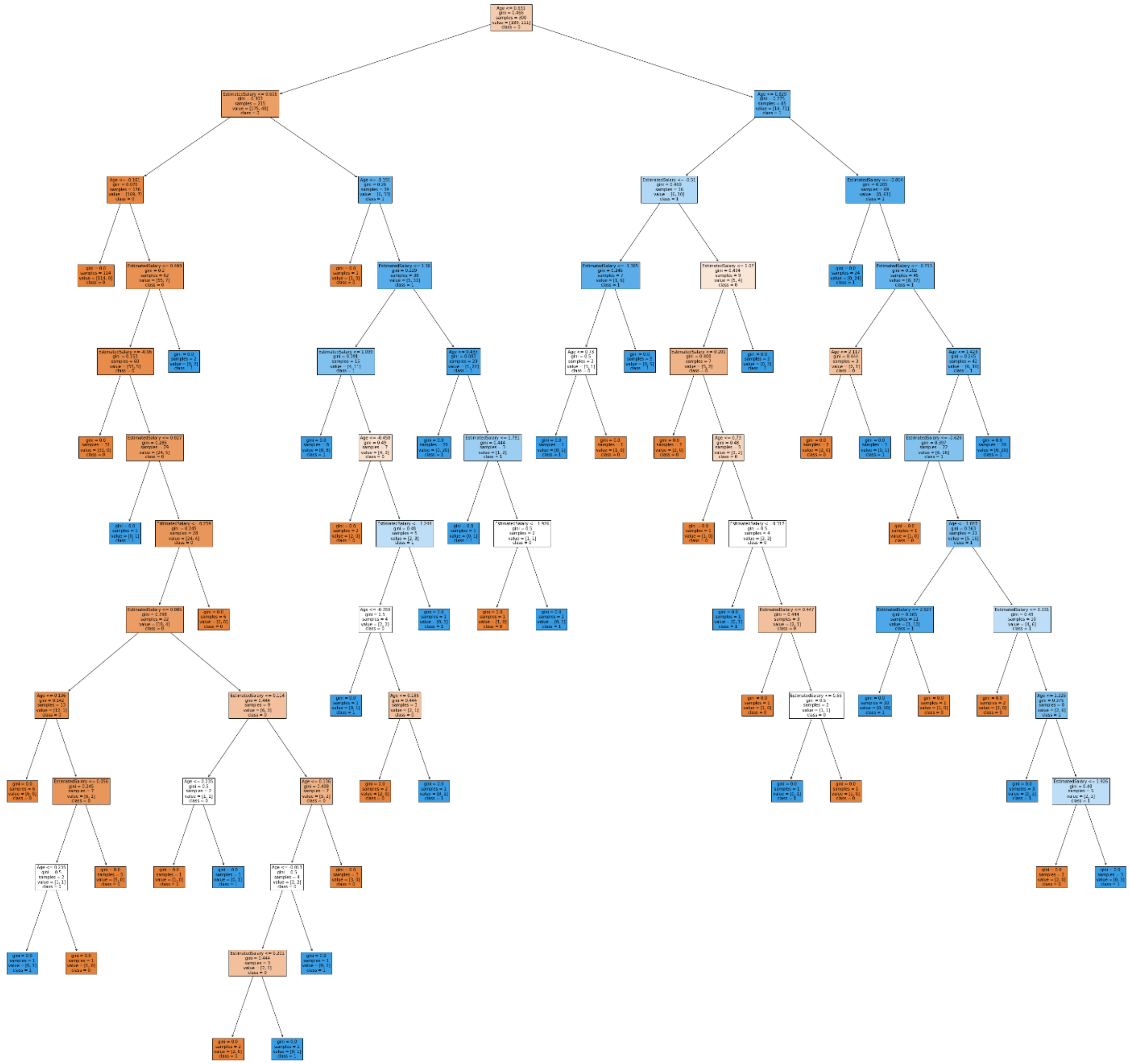
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)

cm
```

Output:

```
array([[62,  6],
       [ 4, 28]], dtype=int64)
```

```
from sklearn.tree import plot_tree
plt.figure(figsize=(50,50))
plot_tree(classifier, feature_names=['Age', 'EstimatedSalary'], class_names=['0', '1'],
filled=True)
plt.show()
```

Output:


Age <= 0.631
gini = 0.466
samples = 300
value = [189, 111]
class = 0
Age <= 0.829
gini = 0.275
samples = 85
value = [14, 71]
class = 1
EstimatedSalary <= -0.51
gini = 0.469
samples = 16
value = [6, 10]
class = 1

11. Write a program to implement K-means clustering. Using data visualization (Scatter Plot) technique to illustrate the clustering.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
dataset = pd.read_csv('Mall_Customers.csv')
dataset
```

Output:

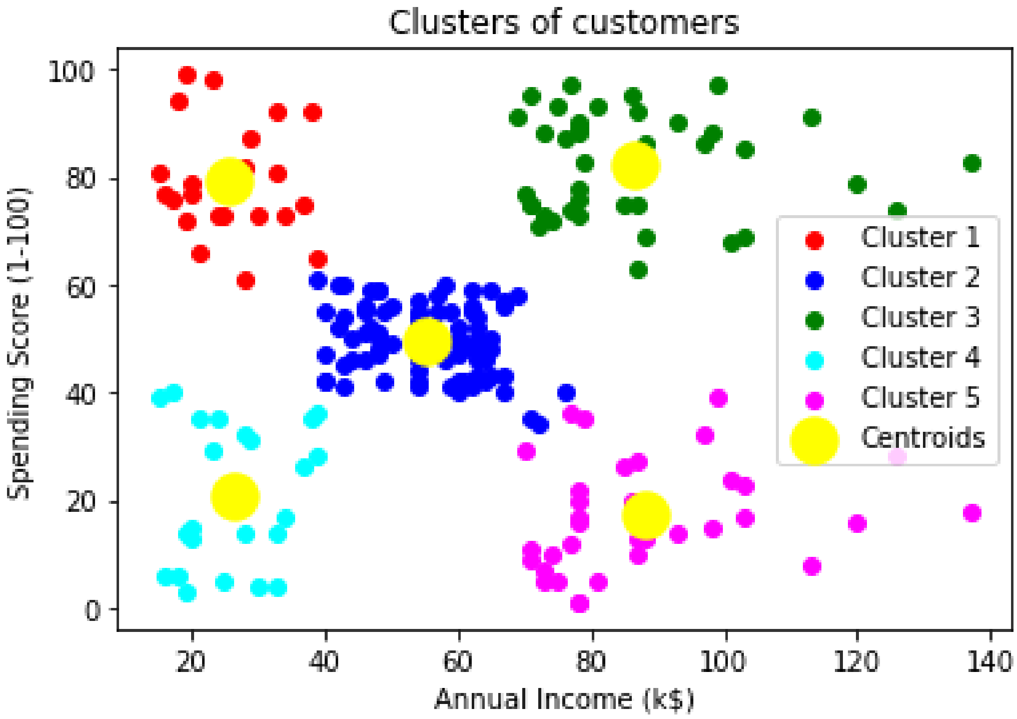
	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40
...	...	...	...	...	...
195	196	Female	35	120	79
196	197	Female	45	126	28
197	198	Male	32	126	74
198	199	Male	32	137	18
199	200	Male	30	137	83

200 rows × 5 columns

```
X = dataset.iloc[:, [3, 4]].values

# Fitting K-Means to the dataset
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=5, init='k-means++', random_state=42)
y_kmeans = kmeans.fit_predict(X)

# Visualising the clusters
plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], c = 'red', label = 'Cluster 1')
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], c = 'blue', label = 'Cluster 2')
plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], c = 'green', label = 'Cluster 3')
plt.scatter(X[y_kmeans == 3, 0], X[y_kmeans == 3, 1], c = 'cyan', label = 'Cluster 4')
plt.scatter(X[y_kmeans == 4, 0], X[y_kmeans == 4, 1], c = 'magenta', label = 'Cluster 5')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 300, c = 'yellow',
label = 'Centroids')
```

```
plt.title('Clusters of customers')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
```

Output:



12. Write a program to implement hierarchical clustering algorithm. Using data visualization (Scatter Plot) technique to illustrate the clustering.

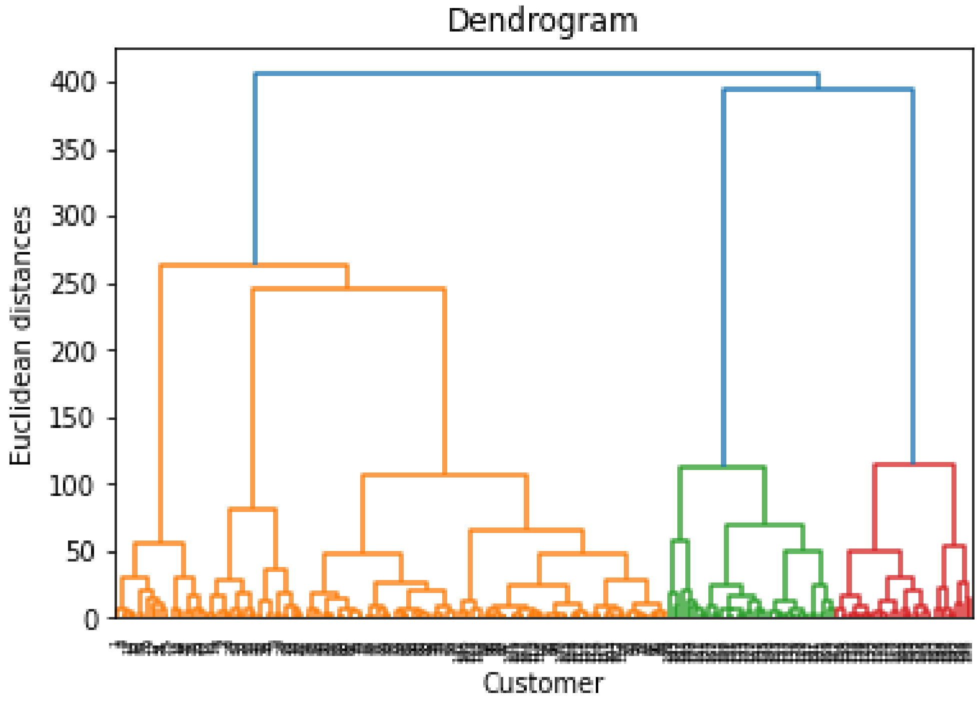
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
dataset = pd.read_csv("Mall_Customers.csv")
dataset.head()
```

Output:

	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

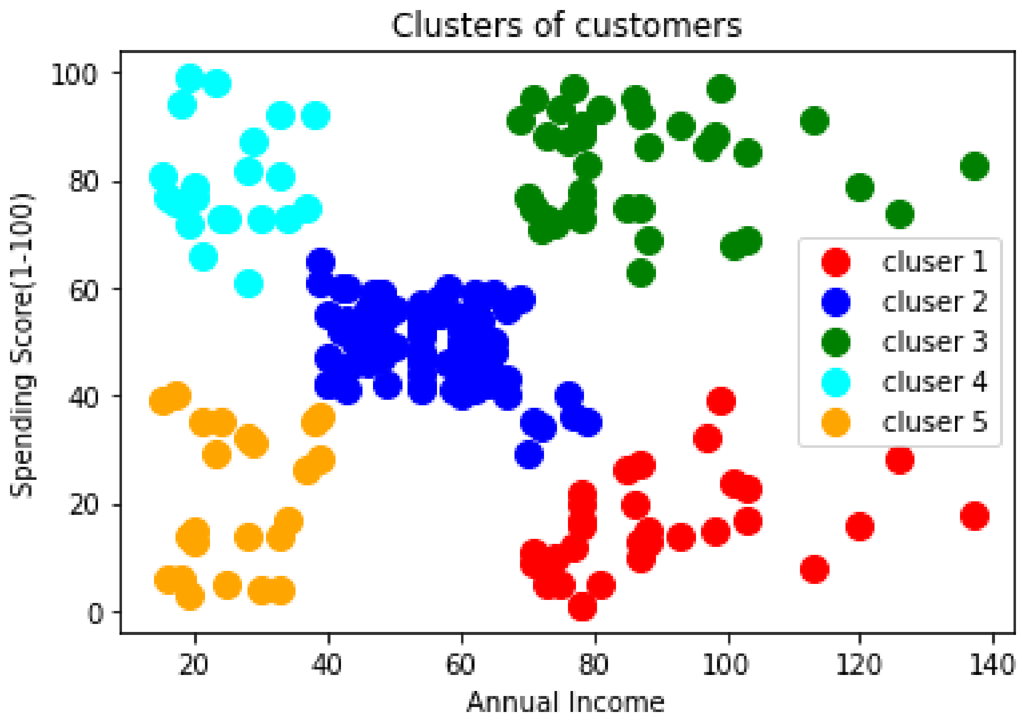
```
x = dataset.iloc[:,[3,4]].values

import scipy.cluster.hierarchy as sch
dendrogram = sch.dendrogram(sch.linkage(x, method = 'ward'))
plt.title("Dendrogram")
plt.xlabel("Customer")
plt.ylabel("Euclidean distances")
plt.show()
```

Output:



```
from sklearn.cluster import AgglomerativeClustering
hc = AgglomerativeClustering(n_clusters=5, linkage = 'ward')
y_hc = hc.fit_predict(x)
plt.scatter(x[y_hc == 0, 0], x[y_hc == 0, 1], s = 100, c = "red", label = "cluser 1")
plt.scatter(x[y_hc == 1, 0], x[y_hc == 1, 1], s = 100, c = "blue", label = "cluser 2")
plt.scatter(x[y_hc == 2, 0], x[y_hc == 2, 1], s = 100, c = "green", label = "cluser 3")
plt.scatter(x[y_hc == 3, 0], x[y_hc == 3, 1], s = 100, c = "cyan", label = "cluser 4")
plt.scatter(x[y_hc == 4, 0], x[y_hc == 4, 1], s = 100, c = "orange", label = "cluser 5")
plt.title("Clusters of customers")
plt.xlabel("Annual Income")
plt.ylabel("Spending Score(1-100)")
plt.legend()
plt.show()
```

Output:


Clusters of customers
100
80
60
40
20
0
Spending Score(1-100)
20
40
60
80
100
120
140
Annual Income
cluser 1
cluser 2
cluser 3
cluser 4
cluser 5

13. Implement density-based clustering using a suitable dataset. Explore the DBSCAN algorithm and visualize the data.

```
import pandas as pd
from sklearn.cluster import DBSCAN
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs

data = pd.read_csv('blobs.csv')
data
```

Output:

	0	1
0	8.622185	1.935796
1	-4.736710	-7.970958
2	9.621222	0.925423
3	6.162095	-0.273254
4	8.697488	-1.057452
0	8.622185	1.935796
...	...	...
995	8.993880	2.203768
996	-5.082768	-9.644539
997	-6.252268	-8.412482
998	-5.479154	-10.536955
999	6.120559	0.968963

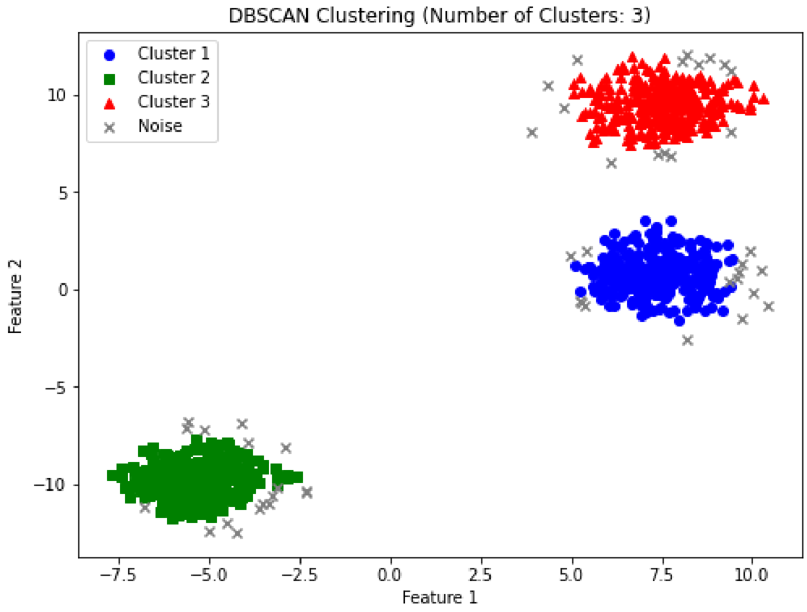
```
# Extract the features (assuming your CSV file has columns 'Feature1' and 'Feature2')
X = data.iloc[:,[0,1]].values
X
```

Output:

```
array([[  8.62218539,   1.93579579],
       [ -4.73670958,  -7.97095765],
       [  9.62122205,   0.92542315],
       ...,
       [ -6.2522678 ,  -8.412482  ],
       [ -5.479154  , -10.53695547],
       [  6.12055883,   0.96896287]])
```

```
# DBSCAN clustering
db = DBSCAN(eps=0.5, min_samples=5)
y_db = db.fit_predict(X)

# Number of clusters in labels, ignoring noise if present (-1)
n_clusters_ = len(set(y_db)) - (1 if -1 in y_db else 0)

# Plot the clusters
plt.figure(figsize=(8, 6))
plt.scatter(X[y_db == 0][:, 0], X[y_db == 0][:, 1], c='blue', marker='o', label='Cluster 1')
plt.scatter(X[y_db == 1][:, 0], X[y_db == 1][:, 1], c='green', marker='s', label='Cluster 2')
plt.scatter(X[y_db == 2][:, 0], X[y_db == 2][:, 1], c='red', marker='^', label='Cluster 3')
plt.scatter(X[y_db == -1][:, 0], X[y_db == -1][:, 1], c='gray', marker='x', label='Noise')
plt.legend(loc='best')
plt.title(f"DBSCAN Clustering (Number of Clusters: {n_clusters_})")
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.show()
```

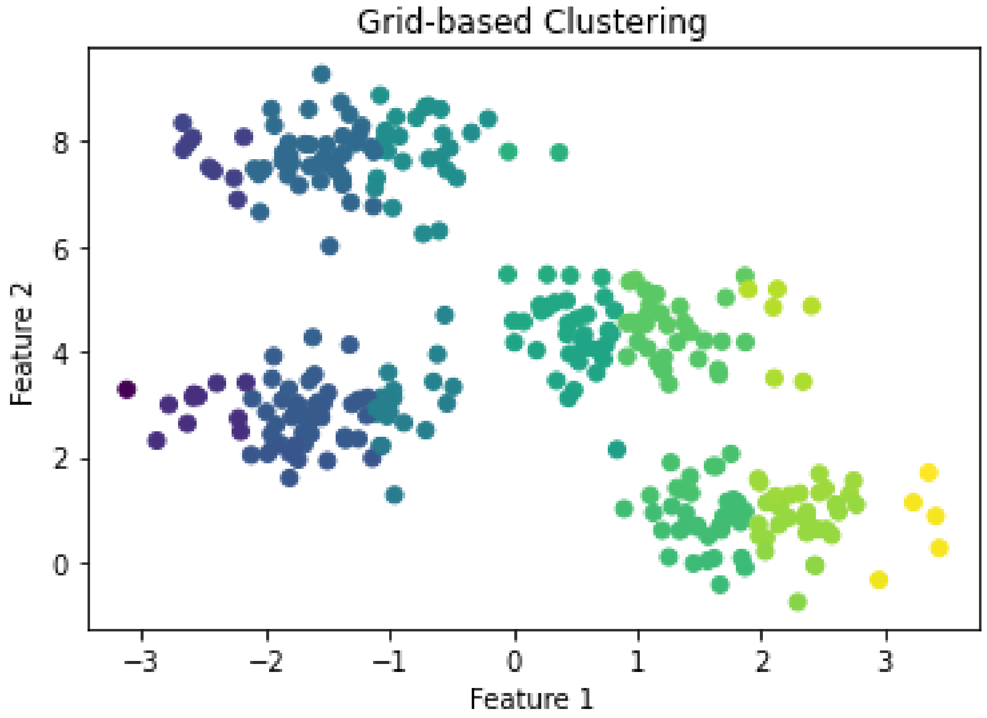
Output:



14. Write a program to implement grid-based clustering using a suitable dataset. Visualize the data using scatter plot.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs

# Generate synthetic data
data, _ = make_blobs(n_samples=300, centers=4, cluster_std=.60, random_state=0)

# Set the grid size (you can adjust this based on your data distribution)
grid_size = 1.0

# Get the minimum and maximum values for x and y coordinates
x_min, x_max = data[:, 0].min(), data[:, 0].max()
y_min, y_max = data[:, 1].min(), data[:, 1].max()

# Create a grid by defining intervals using the minimum and maximum values
x_grid = np.arange(x_min, x_max + grid_size, grid_size)
y_grid = np.arange(y_min, y_max + grid_size, grid_size)

# Initialize labels array with zeros
labels = np.zeros(data.shape[0], dtype=int)

# Assign each data point to a grid cell based on its coordinates
for i, point in enumerate(data):
    x, y = point
    x_label = np.searchsorted(x_grid, x) - 1
    y_label = np.searchsorted(y_grid, y) - 1
    labels[i] = x_label * len(y_grid) + y_label

# Visualize the clusters
plt.scatter(data[:, 0], data[:, 1], c=labels, cmap='viridis')
plt.title('Grid-based Clustering')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.show()
```

Output:



15. Write a program to perform linear regression using
i. Single variable
Ii. Multiple variable

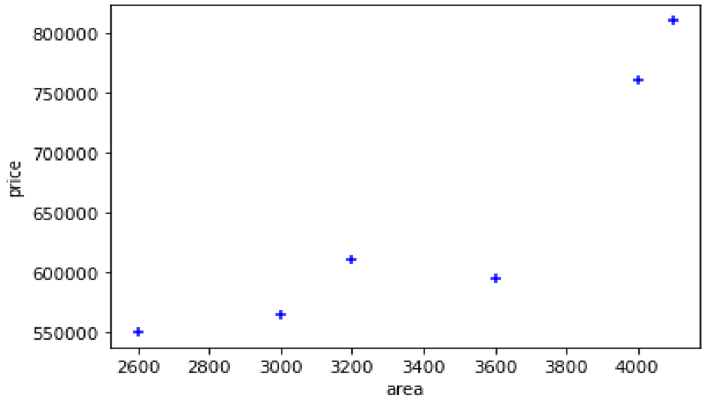
```
#Using single variable
import pandas as pd
from sklearn import linear_model
import matplotlib.pyplot as plt
df = pd.read_csv("homeprices.csv")
df
```

Output:

	Area	price
0	2600	550000
1	3000	565000
2	3200	610000
3	3600	595000
4	4000	760000
5	4100	810000

```
plt.xlabel('area')
plt.ylabel('price')
plt.scatter(df.area, df.price, color = 'blue', marker = '+')
```

Output:



```
reg = linear_model.LinearRegression()
reg.fit(df[['area']], df.price)
reg.coef_
```

Output:

```
array([167.30954677])
```

```
reg.intercept_
```

Output:

```
76692.3818707813
```

```
reg.predict([[6800]])
```

Output:

```
array([1214397.29990357])
```

```
#Using multiple variable

import pandas as pd
from sklearn import linear_model
import matplotlib.pyplot as plt
df = pd.read_csv("homeprices_multiple.csv")
df
```

Output:

	area	bedrooms	age	price
0	2600	3.0	20	550000
1	3000	4.0	15	565000
2	3200	NaN	18	610000
3	3600	3.0	30	595000
4	4000	5.0	8	760000
5	4100	6.0	8	810000

```
df.bedrooms.median()
```

Output:

```
4.0
```

```
df.bedrooms = df.bedrooms.fillna(df.bedrooms.median())
df
```

Output:

	area	bedrooms	age	price
0	2600	3.0	20	550000
1	3000	4.0	15	565000
2	3200	4.0	18	610000
3	3600	3.0	30	595000
4	4000	5.0	8	760000
5	4100	6.0	8	810000

```
x = df.iloc[:, [0,1,2]].values
y = df.iloc[:, 3].values
reg = linear_model.LinearRegression()
reg.fit(x, y)
```

Output:

```
LinearRegression()
```

```
reg.coef_
```

Output:

```
array([  112.06244194, 23388.88007794, -3231.71790863])
```

```
reg.intercept_
```

Output:

```
221323.00186540408
```

```
reg.predict([[2600, 3, 30]])
```

Output:

```
array([485900.45388978])
```

16. Write a program to implement chi-square test for feature selection to train SVM classifier using suitable dataset.

```
import pandas as pd
# Load dataset
data = pd.read_csv('fruit_data_with_colours.csv')
data.head(5)
fruit_label = 'fruit_label'
fruit_subtype = 'fruit_subtype'
fruit_name = 'fruit_name'
# Drop non-numeric columns if necessary and extract features (X) and target (y)
X = data.drop([fruit_label, fruit_subtype, fruit_name], axis=1)  # Features
y = data[fruit_label]

from sklearn.feature_selection import SelectKBest, chi2
k_selected_features = 4 # Adjust this value based on how many top features you want to select
chi2_selector = SelectKBest(chi2, k=k_selected_features)
X_selected = chi2_selector.fit_transform(X, y)

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_selected, y, test_size=0.2,random_state=42)

# Step 3: Train the SVM classifier
from sklearn.svm import SVC
svm_classifier = SVC(kernel='linear')
svm_classifier.fit(X_train, y_train)
```

```
SVC(kernel='linear')
```

```
# Step 4: Evaluate the SVM classifier
from sklearn.metrics import accuracy_score, classification_report
y_pred = svm_classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)
print("Accuracy:", accuracy)
print("Classification Report:")
print(report)
```

Output:

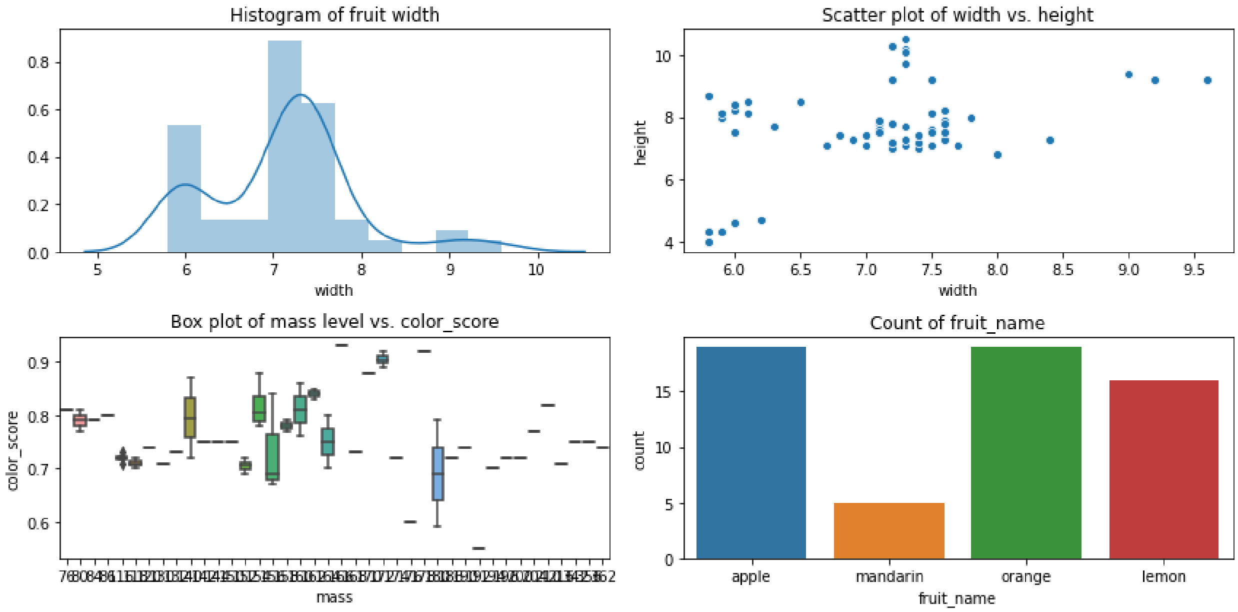
```
Accuracy: 0.75
Classification Report:
              precision    recall  f1-score   support

           1       0.67      0.67      0.67         3
           2       1.00      1.00      1.00         2
           3       0.33      0.50      0.40         2
           4       1.00      0.80      0.89         5

    accuracy                           0.75        12
   macro avg       0.75      0.74      0.74        12
weighted avg       0.81      0.75      0.77        12
```

17. Implement a program to perform various data visualization techniques on sample dataset.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load dataset (replace with your dataset loading code)
data = pd.read_csv('fruit_data_with_colours.csv')

# Display the first few rows of the dataset
print("First few rows of the dataset:")
print(data.head())

plt.figure(figsize=(12, 6))

# Plot 1: Histogram
plt.subplot(2, 2, 1)
sns.distplot(data['width'], bins=10, kde=True)
plt.title('Histogram of fruit width')

# Plot 2: Scatter plot
plt.subplot(2, 2, 2)
sns.scatterplot(x='width', y='height', data=data)
plt.title('Scatter plot of width vs. height')

# Plot 3: Box plot
plt.subplot(2, 2, 3)
sns.boxplot(x='mass', y='color_score', data=data)
plt.title('Box plot of mass level vs. color_score')

# Plot 4: Count plot
plt.subplot(2, 2, 4)
sns.countplot(x='fruit_name', data=data)
plt.title('Count of fruit_name')

# Adjust layout
plt.tight_layout()
```

Output:

```
First few rows of the dataset:
   fruit_label fruit_name fruit_subtype  mass  width  height  color_score
0            1      apple  granny_smith   192    8.4     7.3         0.55
1            1      apple  granny_smith   180    8.0     6.8         0.59
2            1      apple  granny_smith   176    7.4     7.2         0.60
3            2   mandarin      mandarin    86    6.2     4.7         0.80
4            2   mandarin      mandarin    84    6.0     4.6         0.79
```



18. Implement a program to perform attribute selection measures.

```
import pandas as pd
import numpy as np

def entropy(data):
    values, counts = np.unique(data, return_counts=True)
    probs = counts / len(data)
    return -np.sum(probs * np.log2(probs))

def information_gain(data, attribute_index):
    total_entropy = entropy(data[:, -1])
    values, counts = np.unique(data[:, attribute_index], return_counts=True)
    weighted_entropy = sum((counts[i] / len(data)) * entropy(data[data[:, attribute_index] ==
values[i], -1]) for i in range(len(values)))
    return total_entropy - weighted_entropy

def gain_ratio(data, attribute_index):
    # Information Gain
    ig = information_gain(data, attribute_index)
    # Calculate Intrinsic Value
    values, counts = np.unique(data[:, attribute_index], return_counts=True)
    total_instances = len(data)
    intrinsic_value = -np.sum((counts / total_instances) * np.log2(counts / total_instances))
    return ig / intrinsic_value if intrinsic_value != 0 else 0

def gini_index(data):
    # Calculate the Gini index of a dataset
    class_labels = data[:, -1]
    total_instances = len(class_labels)
    label_counts = np.unique(class_labels, return_counts=True)[1]
    label_probabilities = label_counts / total_instances
    gini = 1 - np.sum(label_probabilities**2)
    return gini

def gini_index_attribute(data, attribute_index):
    # Calculate the Gini index of an attribute in a dataset
    attribute_values = np.unique(data[:, attribute_index])
    total_instances = len(data)
    gini_attribute = 0
    for value in attribute_values:
        subset = data[data[:, attribute_index] == value]
        subset_instances = len(subset)
        gini_subset = gini_index(subset)
        gini_attribute += (subset_instances / total_instances) * gini_subset
    return gini_attribute

# Load CSV file
df = pd.read_csv("Buys_Computer.csv")
data = df.values

print("Dataset loaded successfully:")
print(df)

while True:
    print("\n1. Information Gain\n2. Gain Ratio\n3. Gini Index\n4. Exit")
    ch = input("Enter your choice: ")

    if ch == "1":
        try:
            attribute_index = int(input(f"Enter the index of the attribute (0 to {data.shape[1] - 2}) for
which you want to calculate Information Gain: "))
            if 0 <= attribute_index < data.shape[1] - 1:
                ig = information_gain(data, attribute_index)
                print(f"Information Gain for attribute {attribute_index}: {ig}")
            else:
                print(f"Invalid attribute index. Please enter a number between 0 and {data.shape[1] -
2}.")
        except ValueError:
            print("Invalid input. Please enter a valid integer for the attribute index.")
        except Exception as e:
            print(f"An error occurred: {e}")

    elif ch == "2":
        try:
            attribute_index = int(input(f"Enter the index of the attribute (0 to {data.shape[1] - 2}) for
which you want to calculate Gain Ratio: "))
            if 0 <= attribute_index < data.shape[1] - 1:
                # Calculating Gain Ratio for the specified attribute
                gain_ratio_attr = gain_ratio(data, attribute_index)
                print(f"Gain Ratio for attribute {attribute_index}: {gain_ratio_attr}")
            else:
                print(f"Invalid attribute index. Please enter a number between 0 and {data.shape[1] -
2}.")
        except ValueError:
            print("Invalid input. Please enter a valid integer for the attribute index.")
        except Exception as e:
            print(f"An error occurred: {e}")

    elif ch == "3":
        try:
            attribute_index = int(input(f"Enter the index of the attribute (0 to {data.shape[1] - 2}) for
which you want to calculate Gini Index: "))
            if 0 <= attribute_index < data.shape[1] - 1:
                # Calculating Gini index for the specified attribute
                gini_attr = gini_index_attribute(data, attribute_index)
                print(f"Gini index for attribute {attribute_index}: {gini_attr}")
            else:
                print(f"Invalid attribute index. Please enter a number between 0 and {data.shape[1] -
2}.")
        except ValueError:
            print("Invalid input. Please enter a valid integer for the attribute index.")
        except Exception as e:
            print(f"An error occurred: {e}")
    elif ch == "4":
        break
    else:
        print("Invalid choice")
```

Output:

```
Dataset loaded successfully:
   age  income student credit_rating buys_computer
0      <=30    high      no          fair            no
1      <=30    high      no     excellent            no
2   31...40    high      no          fair           yes
3       >40  medium      no          fair           yes
4       >40     low     yes          fair           yes
5       >40     low     yes     excellent            no
6   31...40     low     yes     excellent           yes
7      <=30  medium      no          fair            no
8      <=30     low     yes          fair           yes
9       >40  medium     yes          fair           yes
10     <=30  medium     yes     excellent           yes
11  31...40  medium      no     excellent           yes
12  31...40    high     yes          fair           yes
13      >40  medium      no     excellent            no

1. Information Gain
2. Gain Ratio
3. Gini Index
4. Exit
Enter your choice: 1
Enter the index of the attribute (0 to 3) for which you want to calculate Information Gain: 1
Information Gain for attribute 1: 0.02922256565895487

1. Information Gain
2. Gain Ratio
3. Gini Index
4. Exit
Enter your choice: 1
Enter the index of the attribute (0 to 3) for which you want to calculate Information Gain: 3
Information Gain for attribute 3: 0.04812703040826949

1. Information Gain
2. Gain Ratio
3. Gini Index
4. Exit
Enter your choice: 3
Enter the index of the attribute (0 to 3) for which you want to calculate Gini Index: 0
```

```
Gini index for attribute 0: 0.34285714285714286

1. Information Gain
2. Gain Ratio
3. Gini Index
4. Exit
Enter your choice: 2
Enter the index of the attribute (0 to 3) for which you want to calculate Gain Ratio: 2
Gain Ratio for attribute 2: 0.15183550136234159

1. Information Gain
2. Gain Ratio
3. Gini Index
4. Exit
Enter your choice: 4
```

19. Implement a program to perform different distance measures.

```
import pandas as pd
import numpy as np

# Function to calculate Euclidean distance
def euclidean_distance(instance1, instance2):
    return np.linalg.norm(instance1 - instance2)

# Function to calculate Manhattan distance
def manhattan_distance(instance1, instance2):
    return np.sum(np.abs(instance1 - instance2))

# Function to calculate Cosine similarity
def cosine_similarity(instance1, instance2):
    dot_product = np.dot(instance1, instance2)
    norm1 = np.linalg.norm(instance1)
    norm2 = np.linalg.norm(instance2)
    return dot_product / (norm1 * norm2)

# Load CSV file
file_path = 'homeprices_multiple.csv' #input("Enter the path to the CSV file: ")
df = pd.read_csv(file_path)

# Print loaded dataset
print("Dataset loaded successfully:")
print(df)

# Mapping of distance measure names to functions
distance_measures = {
    "1": ("Euclidean", euclidean_distance),
    "2": ("Manhattan", manhattan_distance),
    "3": ("Cosine Similarity", cosine_similarity)
}

# Print distance measure options
print("\nSelect a distance measure:")
for key, (measure_name, _) in distance_measures.items():
    print(f"{key}. {measure_name}")

# Accept user input for selecting distance measure
selected_measure_name = input("Enter the index or name of the distance measure: ")

# Validate the selected measure
if selected_measure_name in distance_measures:
    selected_measure = distance_measures[selected_measure_name][1]  # Get the function corresponding to the selected measure
    selected_measure_name = distance_measures[selected_measure_name][0]  # Get the name of the selected measure
else:
    print("Invalid distance measure selection. Please choose from the available options.")
    exit()

# Input indices of two instances
index1 = int(input(f"Enter index of the first instance (0 to {len(df)-1}): "))
index2 = int(input(f"Enter index of the second instance (0 to {len(df)-1}): "))

# Validate indices
if 0 <= index1 < len(df) and 0 <= index2 < len(df):
    instance1 = df.iloc[index1, :-1].values  # Exclude last column (assuming it's the target variable)
    instance2 = df.iloc[index2, :-1].values  # Exclude last column (assuming it's the target variable)

    # Calculate distance based on user's choice
    distance = selected_measure(instance1, instance2)
    print(f"{selected_measure_name} distance between instance {index1} and instance {index2}: {distance}")
else:
    print(f"Invalid indices. Please enter indices between 0 and {len(df)-1}.")
```

Output 1:

```
Dataset loaded successfully:
   area  bedrooms  age   price
0  2600       3.0   20  550000
1  3000       4.0   15  565000
2  3200       NaN   18  610000
3  3600       3.0   30  595000
4  4000       5.0    8  760000
5  4100       6.0    8  810000

Select a distance measure:
1. Euclidean
2. Manhattan
3. Cosine Similarity
Enter the index or name of the distance measure: 1
Enter index of the first instance (0 to 5): 5
Enter index of the second instance (0 to 5): 0
Euclidean distance between instance 5 and instance 0: 1500.0509991330296
```

Output 2:

```
Dataset loaded successfully:
   area  bedrooms  age   price
0  2600       3.0   20  550000
1  3000       4.0   15  565000
2  3200       NaN   18  610000
3  3600       3.0   30  595000
4  4000       5.0    8  760000
5  4100       6.0    8  810000

Select a distance measure:
1. Euclidean
2. Manhattan
3. Cosine Similarity
Enter the index or name of the distance measure: 2
Enter index of the first instance (0 to 5): 0
Enter index of the second instance (0 to 5): 1
Manhattan distance between instance 0 and instance 1: 406.0
```

20. Implement a LinearSVC classifier for text classification using TF-IDF features from char n-grams. Evaluate the performance of the model.

```
#With built-in
import pandas as pd
import spacy
import string
import contractions
import re
import nltk
import emoji
from nltk import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from nltk.stem import PorterStemmer
from num2words import num2words

nltk.download('punkt')
nltk.download('stopwords')
nltk.download('whitespace')
nltk.download('wordnet')
nltk.download('omw-1.4')
stop_words = set(stopwords.words("english"))

data = pd.read_csv('Training.tsv',sep='\t')

data.head(10)
```

Output:

	tweet_id	text	label
0	1382343793341575169	@IrvineWelsh I don’t know about you Irvine but...	0
1	1377631738692796417	I bet money if i went n took a covid test righ...	0

2	1386448010029240326	@JamesMelville My wife received a POSITIVE Cov...	0
3	1361342676340211717	Out of the 180,000+ people who have had the tw...	0
4	1386757983254765569	My whole family is sick af and here I am now i...	0
5	1382001700853125122	@renfrew1962 @PeakePolly @J_Deliciouso I'm not...	0
6	1383272654212272136	Test came back positive, no surprise. I have c...	1
7	1374479299047084035	My Pawpaw has been in the hospital a few days....	0
8	1354020426620547072	@MattHancock 4 people I know had covid and rec...	0
9	1362671045136809985	I’m going to sound like I have lost my marbles...	1

```
data['label'].value_counts()
```

Output:

```
0    6266
1    1334
Name: label, dtype: int64
```

```
ps =PorterStemmer()
lemmatiser = WordNetLemmatizer()
english_stopwords = stopwords.words('english')
exclude = set(string.punctuation)
def preprocess(text):
   #text=demoji.findall(df['Text'])
   text = contractions.fix(text.lower(), slang=True)
   text = re.sub(r'\d+', lambda x: num2words(int(x.group(0))), text)
   #text= re.sub(r'\d+', '', text)
   text=re.sub(r'$', '', text)
   text= re.sub(r'’','', text )
   text=re.sub('<.*?>','',text)
   text=re.sub(r'http\S+', '', text)
   #text=emoji.demojize(text, delimiters=(" ", " "))
```

```
    text = ''.join(ch for ch in text if ch not in exclude)
    tokens = word_tokenize(text)
    #print("Tokens:", tokens)
    text = [t for t in tokens if t not in english_stopwords]
    text = " ".join(text)
    return text

import emoji
#import demoji
#demoji.download_codes()
def emo(text):
    temp=emoji.demojize(text,delimiters=(" "," "))
    temp=temp.replace("_","  ")
    return temp
data['emo']=data["text"].apply(lambda x:emo(x))
data["clean_text"]=data['emo'].apply(lambda X: preprocess(X))

data.head()
```

Output:

	tweet_id	text	label	emo	clean_text
0	1382343793341575169	@IrvineWelsh I don’t know about you Irvine but...	0	@IrvineWelsh I don’t know about you Irvine but...	irvinewelsh know irvine keep told covid exist ...
1	1377631738692796417	I bet money if i went n took a covid test righ...	0	I bet money if i went n took a covid test righ...	bet money went n took covid test right going t...
2	1386448010029240326	@JamesMelville My wife received a POSITIVE Cov...	0	@JamesMelville My wife received a POSITIVE Cov...	jamesmelville wife received positive covid tes...
3	1361342676340211717	Out of the 180,000+ people who have had the tw...	0	Out of the 180,000+ people who have had the tw...	one hundred eightyzero people two vaccine shot...
4	1386757983254765569	My whole family is sick af and here I am now i...	0	My whole family is sick af and here I am now i...	whole family sick af hospital heart palpitatio...

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(data['clean_text'], data['label'], test_size=0.33,
random_state=42)
```

Feature Extraction: TF-IDF (char_wb)

```
Tfidf_vec1 = TfidfVectorizer(analyzer='char_wb', ngram_range=(1, 5), max_df=1.0,
min_df=1, max_features=5000)
count_train1 = Tfidf_vec1.fit(X_train)
train_features1 = Tfidf_vec1.transform(X_train)
test_features1 = Tfidf_vec1.transform(X_test)
```

Feature Extraction- (word) TFIDF

```
Tfidf_vec2 = TfidfVectorizer(analyzer='word', ngram_range=(1, 3), max_df=1.0, min_df=1,
max_features=5000)
count_train2 = Tfidf_vec2.fit(X_train)
train_features2= Tfidf_vec2.transform(X_train)
test_features2 = Tfidf_vec2.transform(X_test)
```

Feature Extraction - (word) CountVectorizer

```
count_vec3 = CountVectorizer(analyzer='word', ngram_range=(1, 3), max_df=1.0, min_df=1,
max_features=5000)
count_train3 = count_vec3.fit(X_train)
train_features3 = count_vec3.transform(X_train)
test_features3 = count_vec3.transform(X_test)
```

Feature Extraction - (char_wb) CountVectorizer

```
count_vec4 = CountVectorizer(analyzer='char_wb', ngram_range=(1, 5), max_df=1.0, min_df=1,
max_features=5000)
count_train4 = count_vec4.fit(X_train)
train_features4 = count_vec4.transform(X_train)
test_features4 = count_vec4.transform(X_test)
```

Model Building with SVM – LinearSVC

```
clf1 =LinearSVC(C=1.0, class_weight="balanced", max_iter=10000, random_state=123)
```

```
clf1.fit(train_features1, y_train)
y_pred1=clf1.predict(test_features1)
accuracy = accuracy_score(y_test, y_pred1)
print("Test Accuracy(Feature Extraction: TF-IDF (char_wb)):", round(accuracy*100, 4))
print("\n", classification_report(y_test, y_pred1))
```

Output:

```
Test Accuracy(Feature Extraction: TF-IDF (char_wb)): 79.1866

              precision    recall  f1-score   support

           0       0.92      0.81      0.86      2040
           1       0.46      0.70      0.56       468

    accuracy                           0.79      2508
   macro avg       0.69      0.76      0.71      2508
weighted avg       0.84      0.79      0.81      2508
```

```
clf1 =LinearSVC(C=1.0, class_weight="balanced", max_iter=10000, random_state=123)
clf1.fit(train_features2, y_train)
y_pred2=clf1.predict(test_features2)
accuracy = accuracy_score(y_test, y_pred2)
print("Test Accuracy(Feature Extraction- (word) TFIDF):", round(accuracy*100, 4))
print("\n", classification_report(y_test, y_pred2))
```

Output:

```
Test Accuracy(Feature Extraction- (word) TFIDF): 81.2998

              precision    recall  f1-score   support

           0       0.91      0.86      0.88      2040
           1       0.50      0.62      0.55       468

    accuracy                           0.81      2508
   macro avg       0.70      0.74      0.72      2508
weighted avg       0.83      0.81      0.82      2508
```

```
clf1=LinearSVC(C=1.0, class_weight="balanced", max_iter=10000,random_state=123)
clf1.fit(train_features3, y_train)
```

```
y_pred3 = clf1.predict(test_features3)
accuracy = accuracy_score(y_test, y_pred3)
print("Test Accuracy(Feature Extraction - (word) CountVectorizer):", round(accuracy*100, 4))
print("\n", classification_report(y_test, y_pred3))
```

Output:

```
Test Accuracy(Feature Extraction - (word) CountVectorizer): 80.2632

              precision    recall  f1-score   support

           0       0.89      0.87      0.88      2040
           1       0.47      0.51      0.49       468

    accuracy                           0.80      2508
   macro avg       0.68      0.69      0.69      2508
weighted avg       0.81      0.80      0.81      2508
```

```
clf1 = LinearSVC(C=1.0, class_weight="balanced", max_iter=10000, random_state=123)
clf1.fit(train_features4, y_train)
y_pred4 = clf1.predict(test_features4)
accuracy = accuracy_score(y_test, y_pred4)
print("Test AccuracyFeature Extraction - (char_wb) CountVectorizer:", round(accuracy*100, 4))
print("\n", classification_report(y_test, y_pred4))
```

Output:

```
Test AccuracyFeature Extraction - (char_wb) CountVectorizer: 79.7448

              precision    recall  f1-score   support

           0       0.88      0.86      0.87      2040
           1       0.46      0.50      0.48       468

    accuracy                           0.80      2508
   macro avg       0.67      0.68      0.68      2508
weighted avg       0.80      0.80      0.80      2508
```