MACHINE LEARNING LAB RECORD

Task 1:

How to create a dataframe using the pandas library.

AIM:

Creating a dataframe in Pandas from csv files.

Dataset Description and Methods:

1. CardioGoodFitness.csv is a dataset of the size 180 rows x 9 columns
2. Bank.csv is a dataset of size 11162 rows x 17 columns
3. Housing.csv is a dataset of size 20640 rows x 10 columns
4. Taxi-trip-duration.csv is a dataset of size 1458644 rows x 11 columns
5. Auto-mpg is a dataset of size 398 rows x 9 columns

Methods to create dataframes from csv files are:

1. pd.read_csv()
2. pd.read_table()
3. Using csv module

Source Code:

For CardioGoodFitness.csv dataset

```
import pandas as pd
import csv
df1=pd.read_csv('CardioGoodFitness.csv')
print(df1)

df2=pd.read_table('CardioGoodFitness.csv')
print(df2)

with open('CardioGoodFitness.csv') as csv_file:
    csv_reader= csv.reader(csv_file)
    df3=pd.DataFrame([csv_reader],index=None)
print(df3)
```

```
In [3]: runfile('C:/Users/mvsla/OneDrive/Desktop/ml/task_1_dataset1.py', wdir='C:/Users/mvsla/OneDrive/
Desktop/ml')
    Product  Age  Gender  Education  ... Usage  Fitness  Income  Miles
0     TM195   18    Male         14  ...     3        4   29562    112
1     TM195   19    Male         15  ...     2        3   31836     75
2     TM195   19  Female         14  ...     4        3   30699     66
3     TM195   19    Male         12  ...     3        3   32973     85
4     TM195   20    Male         13  ...     4        2   35247     47
..      ...  ...     ...        ...  ...   ...      ...     ...    ...
175   TM798   40    Male         21  ...     6        5   83416    200
176   TM798   42    Male         18  ...     5        4   89641    200
177   TM798   45    Male         16  ...     5        5   90886    160
178   TM798   47    Male         18  ...     4        5  104581    120
179   TM798   48    Male         18  ...     4        5   95508    180

[180 rows x 9 columns]
```

```
    Product,Age,Gender,Education,MaritalStatus,Usage,Fitness,Income,Miles
0                TM195,18,Male,14,Single,3,4,29562,112
1                 TM195,19,Male,15,Single,2,3,31836,75
2              TM195,19,Female,14,Partnered,4,3,30699,66
3                   TM195,19,Male,12,Single,3,3,32973,85
4                TM195,20,Male,13,Partnered,4,2,35247,47
..                                                   ...
175                TM798,40,Male,21,Single,6,5,83416,200
176                TM798,42,Male,18,Single,5,4,89641,200
177                TM798,45,Male,16,Single,5,5,90886,160
178              TM798,47,Male,18,Partnered,4,5,104581,120
179               TM798,48,Male,18,Partnered,4,5,95508,180

[180 rows x 1 columns]
                                                    0    ...
```

```
180
0  [Product, Age, Gender, Education, MaritalStatu...  ...  [TM798, 48, Male, 18, Partnered, 4, 5, 95508,
...

[1 rows x 181 columns]
```

For bank.csv dataset:

```python
import pandas as pd
import csv
df1=pd.read_csv('bank.csv')
print(df1)

df2=pd.read_table('bank.csv')
print(df2)

with open('bank.csv') as csv_file:
    csv_reader=csv.reader(csv_file)
    df3=pd.DataFrame([csv_reader],index=None)
print(df3)
```

```
In [4]: runfile('C:/Users/mvsla/OneDrive/Desktop/ml/task_1_dataset2.py', wdir='C:/Users/mvsla/OneDrive/
Desktop/ml')
       age          job  marital  education  ... pdays  previous poutcome deposit
0       59       admin.  married  secondary  ...    -1         0  unknown     yes
1       56       admin.  married  secondary  ...    -1         0  unknown     yes
2       41   technician  married  secondary  ...    -1         0  unknown     yes
3       55     services  married  secondary  ...    -1         0  unknown     yes
4       54       admin.  married   tertiary  ...    -1         0  unknown     yes
...    ...          ...      ...        ...  ...   ...       ...      ...     ...
11157   33  blue-collar   single    primary  ...    -1         0  unknown      no
11158   39     services  married  secondary  ...    -1         0  unknown      no
11159   32   technician   single  secondary  ...    -1         0  unknown      no
11160   43   technician  married  secondary  ...   172         5  failure      no
11161   34   technician  married  secondary  ...    -1         0  unknown      no

[11162 rows x 17 columns]
```

```
      age,job,marital,education,default,balance,housing,loan,contact,day,month,duration,campaign,pdays,pre
vious,poutcome,deposit
0      59,admin.,married,secondary,no,2343,yes,no,unk...
1      56,admin.,married,secondary,no,45,no,no,unknow...
2      41,technician,married,secondary,no,1270,yes,no...
3      55,services,married,secondary,no,2476,yes,no,u...
4      54,admin.,married,tertiary,no,184,no,no,unknow...
...                                                  ...
11157  33,blue-collar,single,primary,no,1,yes,no,cell...
11158  39,services,married,secondary,no,733,no,no,unk...
11159  32,technician,single,secondary,no,29,no,no,cel...
11160  43,technician,married,secondary,no,0,no,yes,ce...
11161  34,technician,married,secondary,no,0,no,no,cel...

[11162 rows x 1 columns]
                                             0      ...
```

```
                                             0      ...
11162
0  [age, job, marital, education, default, balanc...  ...  [34, technician, married, secondary, no, 0,
no...

[1 rows x 11163 columns]

In [5]:
```

For housing.csv dataset:

```python
import pandas as pd
import csv
df1=pd.read_csv('housing.csv')
print(df1)

df2=pd.read_table('housing.csv')
print(df2)

with open('housing.csv') as csv_file:
    csv_reader=csv.reader(csv_file)
    df3=pd.DataFrame([csv_file],index=None)
print(df3)
```

```
In [5]: runfile('C:/Users/mvsla/OneDrive/Desktop/ml/task_1_dataset3.py', wdir='C:/Users/mvsla/OneDrive/
Desktop/ml')
       longitude  latitude  ...  median_house_value  ocean_proximity
0        -122.23     37.88  ...            452600.0         NEAR BAY
1        -122.22     37.86  ...            358500.0         NEAR BAY
2        -122.24     37.85  ...            352100.0         NEAR BAY
3        -122.25     37.85  ...            341300.0         NEAR BAY
4        -122.25     37.85  ...            342200.0         NEAR BAY
...          ...       ...  ...                 ...              ...
20635    -121.09     39.48  ...             78100.0           INLAND
20636    -121.21     39.49  ...             77100.0           INLAND
20637    -121.22     39.43  ...             92300.0           INLAND
20638    -121.32     39.43  ...             84700.0           INLAND
20639    -121.24     39.37  ...             89400.0           INLAND

[20640 rows x 10 columns]
```

```
      longitude,latitude,housing_median_age,total_rooms,total_bedrooms,population,households,median_income
,median_house_value,ocean_proximity
0       -122.23,37.88,41.0,880.0,129.0,322.0,126.0,8.3...
1       -122.22,37.86,21.0,7099.0,1106.0,2401.0,1138.0...
2       -122.24,37.85,52.0,1467.0,190.0,496.0,177.0,7....
3       -122.25,37.85,52.0,1274.0,235.0,558.0,219.0,5....
4       -122.25,37.85,52.0,1627.0,280.0,565.0,259.0,3....
...                                                   ...
20635   -121.09,39.48,25.0,1665.0,374.0,845.0,330.0,1....
20636   -121.21,39.49,18.0,697.0,150.0,356.0,114.0,2.5...
20637   -121.22,39.43,17.0,2254.0,485.0,1007.0,433.0,1...
20638   -121.32,39.43,18.0,1860.0,409.0,741.0,349.0,1....
20639   -121.24,39.37,16.0,2785.0,616.0,1387.0,530.0,2...

[20640 rows x 1 columns]
                                                    0      ...
```

```
                                                    0      ...
20640
0  longitude,latitude,housing_median_age,total_ro...   ...
-121.24,39.37,16.0,2785.0,616.0,1387.0,530.0,2...

[1 rows x 20641 columns]

In [6]:
```

For taxi-trip-duration dataset (training split):

```python
import pandas as pd
import csv
df1=pd.read_csv('train.csv')
print(df1)

df2=pd.read_table('train.csv')
print(df2)

with open('train.csv') as csv_file:
    csv_reader= csv.reader(csv_file)
    df3=pd.DataFrame([csv_reader],index=None)
csv_file.close()
print(df3)
```

```
In [8]: runfile('C:/Users/mvsla/OneDrive/Desktop/ml/task_1_dataset5.py', wdir='C:/Users/mvsla/OneDrive/
Desktop/ml')
                id  vendor_id  ... store_and_fwd_flag trip_duration
0        id2875421          2  ...                  N           455
1        id2377394          1  ...                  N           663
2        id3858529          2  ...                  N          2124
3        id3504673          2  ...                  N           429
4        id2181028          2  ...                  N           435
...            ...        ...  ...                ...           ...
1458639  id2376096          2  ...                  N           778
1458640  id1049543          1  ...                  N           655
1458641  id2304944          2  ...                  N           764
1458642  id2714485          1  ...                  N           373
1458643  id1209952          1  ...                  N           198

[1458644 rows x 11 columns]
```

```
       id,vendor_id,pickup_datetime,dropoff_datetime,passenger_count,pickup_longitude,pickup_latitude,dro
poff_longitude,dropoff_latitude,store_and_fwd_flag,trip_duration
0        id2875421,2,2016-03-14 17:24:55,2016-03-14 17:...
1        id2377394,1,2016-06-12 00:43:35,2016-06-12 00:...
2        id3858529,2,2016-01-19 11:35:24,2016-01-19 12:...
3        id3504673,2,2016-04-06 19:32:31,2016-04-06 19:...
4        id2181028,2,2016-03-26 13:30:55,2016-03-26 13:...
...                                                    ...
1458639  id2376096,2,2016-04-08 13:31:04,2016-04-08 13:...
1458640  id1049543,1,2016-01-10 07:35:15,2016-01-10 07:...
1458641  id2304944,2,2016-04-22 06:57:41,2016-04-22 07:...
1458642  id2714485,1,2016-01-05 15:56:26,2016-01-05 16:...
1458643  id1209952,1,2016-04-05 14:44:25,2016-04-05 14:...

[1458644 rows x 1 columns]
                                          0       ...
```

```
1458644
0  [id, vendor_id, pickup_datetime, dropoff_datet...  ...  [id1209952, 1, 2016-04-05 14:44:25,
2016-04-05...

[1 rows x 1458645 columns]
```

For AutoMPG dataset:

```python
import pandas as pd
import csv
df1=pd.read_csv('auto-mpg.csv')
print(df1)

df2=pd.read_table('auto-mpg.csv')
print(df2)

with open('auto-mpg.csv') as csv_file:
    csv_reader=csv.reader(csv_file)
    df3=pd.DataFrame([csv_file],index=None)
print(df3)
```

```
In [10]: runfile('C:/Users/mvsla/OneDrive/Desktop/ml/task_1_dataset4.py', wdir='C:/Users/mvsla/OneDrive/
Desktop/ml')
      mpg  cylinders  ...  origin                   car name
0    18.0          8  ...       1  chevrolet chevelle malibu
1    15.0          8  ...       1          buick skylark 320
2    18.0          8  ...       1         plymouth satellite
3    16.0          8  ...       1              amc rebel sst
4    17.0          8  ...       1                ford torino
..    ...        ...  ...     ...                        ...
393  27.0          4  ...       1           ford mustang gl
394  44.0          4  ...       2                 vw pickup
395  32.0          4  ...       1             dodge rampage
396  28.0          4  ...       1               ford ranger
397  31.0          4  ...       1                chevy s-10

[398 rows x 9 columns]
```

```
    mpg,cylinders,displacement,horsepower,weight,acceleration,model year,origin,car name
0    18,8,307,130,3504,12,70,1,chevrolet chevelle m...
1        15,8,350,165,3693,11.5,70,1,buick skylark 320
2         18,8,318,150,3436,11,70,1,plymouth satellite
3              16,8,304,150,3433,12,70,1,amc rebel sst
4              17,8,302,140,3449,10.5,70,1,ford torino
..                                                 ...
393          27,4,140,86,2790,15.6,82,1,ford mustang gl
394                44,4,97,52,2130,24.6,82,2,vw pickup
395            32,4,135,84,2295,11.6,82,1,dodge rampage
396              28,4,120,79,2625,18.6,82,1,ford ranger
397               31,4,119,82,2720,19.4,82,1,chevy s-10

[398 rows x 1 columns]
```

```
                                                    0    ...                                                398
0  mpg,cylinders,displacement,horsepower,weight,a...  ...  31,4,119,82,2720,19.4,82,1,chevy s-10\n

[1 rows x 399 columns]
```

Observation:

1. Pd.read_csv() method is giving the dataframe with exact dimensions with proper indexing and the values are separated by spaces, giving the dataframe a matrix format for ease of analysis.
2. Pd.read_table() method is giving the dataframe with all the values in one column and the values are separated by commas.
3. Csv module and pd.DataFrame() is giving the transpose of the pd.read_table() and proper indexing is not present.

Conclusion:

pd.read_csv() method is more suitable amongst the other methods because it is having proper indexing and the dataframe produced by this method is easy to understand and interpret.

Task 2:

How to add a new column to a dataframe.

AIM: To add a new column to an existing dataframe

Methods and dataset description:

- AutoMPG dataset is a dataset about cars and the dataset is of the size 398 rows x 9 columns.
- There are 8 numerical variables and 1 categorical variable in the dataset. Adding a new column to the dataframe can be done in the following methods.
  - Df.insert()
  - Df.assign()

Source Code:

For AutoMPG dataset:

```
import pandas as pd
import random
df=pd.read_csv('auto-mpg.csv')
print(df)
lst1 = random.sample(range(500), len(df))
lst2= random.sample(range(399),len(df))
df.insert(5,'random_value',lst1,True)
print(df)
print('\n')
df=df.assign(new_col=lst2)
print(df)
print('\n')
```

```
In [11]: runfile('C:/Users/mvsla/OneDrive/Desktop/ml/task_2.py', wdir='C:/Users/mvsla/OneDrive/Desktop/
ml')
      mpg  cylinders  ...  origin                   car name
0    18.0          8  ...       1  chevrolet chevelle malibu
1    15.0          8  ...       1          buick skylark 320
2    18.0          8  ...       1         plymouth satellite
3    16.0          8  ...       1              amc rebel sst
4    17.0          8  ...       1                ford torino
..    ...        ...  ...     ...                        ...
393  27.0          4  ...       1           ford mustang gl
394  44.0          4  ...       2                 vw pickup
395  32.0          4  ...       1             dodge rampage
396  28.0          4  ...       1               ford ranger
397  31.0          4  ...       1                chevy s-10

[398 rows x 9 columns]
```

	mpg	cylinders	...	origin	car name
0	18.0	8	...	1	chevrolet chevelle malibu
1	15.0	8	...	1	buick skylark 320
2	18.0	8	...	1	plymouth satellite
3	16.0	8	...	1	amc rebel sst
4	17.0	8	...	1	ford torino
..	...	...	...	...	...
393	27.0	4	...	1	ford mustang gl
394	44.0	4	...	2	vw pickup
395	32.0	4	...	1	dodge rampage
396	28.0	4	...	1	ford ranger
397	31.0	4	...	1	chevy s-10

[398 rows x 10 columns]

	mpg	cylinders	displacement	...	origin	car name	new_col
0	18.0	8	307.0	...	1	chevrolet chevelle malibu	385
1	15.0	8	350.0	...	1	buick skylark 320	94
2	18.0	8	318.0	...	1	plymouth satellite	318
3	16.0	8	304.0	...	1	amc rebel sst	165
4	17.0	8	302.0	...	1	ford torino	336
..	...	...	...	...	...	...	...
393	27.0	4	140.0	...	1	ford mustang gl	291
394	44.0	4	97.0	...	2	vw pickup	382
395	32.0	4	135.0	...	1	dodge rampage	209
396	28.0	4	120.0	...	1	ford ranger	366
397	31.0	4	119.0	...	1	chevy s-10	363

[398 rows x 11 columns]

Observations:

- Pd.insert() method inserts a new column in the original dataframe.
- Pd.assign() method creates a new dataframe in the runtime with the new column and that dataframe must be assigned to a new variable.

Conclusion:

Both the methods can be used as per the requirement and the are quite easy to use.

Task 3:

How to handle missing values in a dataset

AIM: Handling missing values in a pandas dataframe

Methods and Dataset Description:

- Missing values can be a problem in the data analysis and machine learing model decision.
- So they must be handled in the proper way possible and it can be done in the following ways.
  - Df.dropna()
  - Df.fillna()
  - Df.replace()
  - Df.interpolate()
- This task has been done on 2 datasets taken from kaggle:
  - Employees.csv is a dataset of the size 1000 rows x 8 columns.
  - It has 6 categorical varaibles and 2 numerical variables.
  - AutoMPG.csv is a dataset of the size 398 rows x 9 columns.
  - There are 8 numerical variables and 1 categorical variable in the dataset.

Source code:

For employees.csv dataset.

```
import pandas as pd
df=pd.read_csv('employees.csv')
print(df)
print(df.dtypes=='object')
print(df.isnull().sum())
print('\n')

df['First Name'].fillna('No First Name',inplace=True)
print(df.isnull().sum())
print('\n')
df['Senior Management'].fillna(method='bfill',inplace=True)
print(df.isnull().sum())
print('\n')
df['Team'].fillna(method='pad',inplace=True)
print(df.isnull().sum())
print('\n')
df['Gender'].fillna('No Gender',inplace=True)
print(df.isnull().sum())
```

```
In [13]: runfile('C:/Users/mvsla/OneDrive/Desktop/ml/task_3_dataset1.py', wdir='C:/Users/mvsla/OneDrive/
Desktop/ml')
    First Name  Gender  ... Senior Management                Team
0      Douglas    Male  ...              True           Marketing
1       Thomas    Male  ...              True                 NaN
2        Maria  Female  ...             False             Finance
3        Jerry    Male  ...              True             Finance
4        Larry    Male  ...              True      Client Services
..         ...     ...  ...               ...                 ...
995      Henry     NaN  ...             False        Distribution
996    Phillip    Male  ...             False             Finance
997    Russell    Male  ...             False             Product
998      Larry    Male  ...             False  Business Development
999     Albert    Male  ...              True               Sales

[1000 rows x 8 columns]
```

```
First Name              True
Gender                  True
Start Date              True
Last Login Time         True
Salary                 False
Bonus %                False
Senior Management       True
Team                    True
dtype: bool
First Name             67
Gender                145
Start Date              0
Last Login Time         0
Salary                  0
Bonus %                 0
Senior Management      67
Team                   43
dtype: int64
```

```
First Name              0
Gender                145
Start Date              0
Last Login Time         0
Salary                  0
Bonus %                 0
Senior Management      67
Team                   43
dtype: int64
```

```
First Name              0
Gender                145
Start Date              0
Last Login Time         0
Salary                  0
Bonus %                 0
Senior Management       0
Team                   43
dtype: int64
```

```
First Name              0
Gender                145
Start Date              0
Last Login Time         0
Salary                  0
Bonus %                 0
Senior Management       0
Team                    0
dtype: int64


First Name            0
Gender                0
Start Date            0
Last Login Time       0
Salary                0
Bonus %               0
Senior Management     0
Team                  0
dtype: int64
```

For AutoMPG dataset:

```
import pandas as pd
df=pd.read_csv('auto-mpg.csv')
print(df)
print(df.dtypes=='object')
print(df.isnull().sum())
print('\n')
df=df.replace(to_replace='?',value=0)
d1={'horsepower':int}
df=df.astype(d1)
print(df.dtypes)
print(df.isnull().sum())
print(df)
```

```
In [16]: runfile('C:/Users/mvsla/OneDrive/Desktop/ml/task_3_dataset2.py', wdir='C:/Users/mvsla/OneDrive/
Desktop/ml')
      mpg  cylinders  ...  origin                   car name
0    18.0          8  ...       1  chevrolet chevelle malibu
1    15.0          8  ...       1          buick skylark 320
2    18.0          8  ...       1         plymouth satellite
3    16.0          8  ...       1              amc rebel sst
4    17.0          8  ...       1                ford torino
..    ...        ...  ...     ...                        ...
393  27.0          4  ...       1            ford mustang gl
394  44.0          4  ...       2                  vw pickup
395  32.0          4  ...       1              dodge rampage
396  28.0          4  ...       1                ford ranger
397  31.0          4  ...       1                 chevy s-10

[398 rows x 9 columns]
```

```
mpg             False
cylinders       False
displacement    False
horsepower       True
weight          False
acceleration    False
model year      False
origin          False
car name         True
dtype: bool
```

```
mpg             0
cylinders       0
displacement    0
horsepower      0
weight          0
acceleration    0
model year      0
origin          0
car name        0
dtype: int64
```

```
mpg             float64
cylinders         int64
displacement    float64
horsepower        int32
weight            int64
acceleration    float64
model year        int64
origin            int64
car name         object
dtype: object
```

```
mpg             0
cylinders       0
displacement    0
horsepower      0
weight          0
acceleration    0
model year      0
origin          0
car name        0
dtype: int64
```

```
      mpg  cylinders  ...  origin                   car name
0    18.0          8  ...       1  chevrolet chevelle malibu
1    15.0          8  ...       1          buick skylark 320
2    18.0          8  ...       1         plymouth satellite
3    16.0          8  ...       1              amc rebel sst
4    17.0          8  ...       1                ford torino
..    ...        ...  ...     ...                        ...
393  27.0          4  ...       1            ford mustang gl
394  44.0          4  ...       2                  vw pickup
395  32.0          4  ...       1              dodge rampage
396  28.0          4  ...       1                ford ranger
397  31.0          4  ...       1                 chevy s-10

[398 rows x 9 columns]
```

Observations:

- ❖ df.dropna() is a method which removes the records with null avlues and reduces the size of the dataset.
- ❖ df.fillna() is method which can fill the null values as per the method for one column or for many columns.
- ❖ Df.replace() method creates a new dataframe and needs a dictionary as the input and can even work with replacing not null but wrong entries in the dataframe.
- ❖ Df.interpolate() is not used in this particular task as it needs much data to understand the distribution and fill the null values accordingly.

Conclusion:

It is very important to understand the distribution and the type of the dataset to use the appropriate missing data handling mechanism.

Task 4:

How to calculate arithmetic mean, median, trimmed mean and weighted mean in a dataframe.

AIM : To calculate the arithmetic mean, median, trimmed mean and weighted mean to a dataset.

Methods and dataset description:

- ❖ We used two datasets taken from kaggle to caluculate the given measures of central tendency, which are AutoMPG and iris datsets.
- ❖ AutoMPG is a datset about cars which has a size of 398 columns x 9 rows with 8 numerical variables and 1 categorical variable, however we can consider model year and origin as the nominal variables.

- ❖ Iris dataset is a datset about three types of flowers with a size of 150 rows x 5 columns, with 4 numerical variables and 1 categorical variable.
- ❖ We used numpy, pandas and scipy libraries to complete the given task.

Source Code: For Iris Dataset:

```
import pandas as pd
import numpy as np
from scipy import stats
import random
df=pd.read_csv('iris.csv')
print(df)
print(df.dtypes=='object')
print(df.isnull().sum())
print('Mean of sepal_length is:', np.average(df.sepal_length))
print('Mean of sepal_width is:',np.average(df.sepal_width))
print('Mean of petal_length is:',np.average(df.petal_length))
print('Mean of petal_width is:', np.average(df.petal_width))
print('\n')
print('Median of sepal_length is:', np.median(df.sepal_length))
print('Median of sepal_width is:',np.median(df.sepal_width))
print('Median of petal_length is:',np.median(df.petal_length))
print('Median of petal_width is:', np.median(df.petal_width))
print('\n')
print('trimmed Mean of sepal_length is:', stats.trim_mean(df.sepal_length,0.25))
print('Trimmed Mean of sepal_width is:',stats.trim_mean(df.sepal_width,0.25))
print('Trimmed Mean of petal_length is:',stats.trim_mean(df.petal_length,0.25))
print('Trimmed Mean of petal_width is:', stats.trim_mean(df.petal_width,0.25))
lst1=random.randint(1,10)
df.insert(4,'Weights',lst1,True)
print('\n')
print('Weighted mean of sepal_length is:',np.average(df.sepal_length,weights=df.Weights))
print('Weighted mean of sepal_width is:',np.average(df.sepal_width,weights=df.Weights))
print('Weighted mean of petal_length is:',np.average(df.petal_length,weights=df.Weights))
print('Weighted mean of petal_width is:',np.average(df.petal_width,weights=df.Weights))
```

```
In [18]: runfile('C:/Users/mvsla/OneDrive/Desktop/ml/task_4_dataset1_mean.py',
wdir='C:/Users/mvsla/OneDrive/Desktop/ml')
     sepal_length  sepal_width  petal_length  petal_width    species
0             5.1          3.5           1.4          0.2     setosa
1             4.9          3.0           1.4          0.2     setosa
2             4.7          3.2           1.3          0.2     setosa
3             4.6          3.1           1.5          0.2     setosa
4             5.0          3.6           1.4          0.2     setosa
..            ...          ...           ...          ...        ...
145           6.7          3.0           5.2          2.3  virginica
146           6.3          2.5           5.0          1.9  virginica
147           6.5          3.0           5.2          2.0  virginica
148           6.2          3.4           5.4          2.3  virginica
149           5.9          3.0           5.1          1.8  virginica

[150 rows x 5 columns]
```

```
sepal_length    False
sepal_width     False
petal_length    False
petal_width     False
species          True
dtype: bool
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

```
Mean of sepal_length is: 5.843333333333334
Mean of sepal_width is: 3.0540000000000003
Mean of petal_length is: 3.758666666666666
Mean of petal_width is: 1.1986666666666668
```

```
Median of sepal_length is: 5.8
Median of sepal_width is: 3.0
Median of petal_length is: 4.35
Median of petal_width is: 1.3
```

```
trimmed Mean of sepal_length is: 5.802631578947367
Trimmed Mean of sepal_width is: 3.030263157894738
Trimmed Mean of petal_length is: 3.9342105263157894
Trimmed Mean of petal_width is: 1.230263157894737
```

```
Weighted mean of sepal_length is: 5.843333333333334
Weighted mean of sepal_width is: 3.0540000000000003
Weighted mean of petal_length is: 3.7586666666666657
Weighted mean of petal_width is: 1.1986666666666665
```

For AutoMPG dataset:

```python
import pandas as pd
import numpy as np
from scipy import stats
import random
df=pd.read_csv('auto-mpg.csv')
print(df)
print(df.dtypes=='object')
print(df.isnull().sum())
df=df.replace(to_replace='?',value=0)
d1={'horsepower':int}
df=df.astype(d1)
print(df.dtypes)

print('Mean of mpg is:', np.average(df.mpg))
print('Mean of cylinders is:',np.average(df.cylinders))
print('Mean of displacement is:',np.average(df.displacement))
print('Mean of horsepower is:', np.average(df.horsepower))
print('Mean of weight is:', np.average(df.weight))
print('Mean of acceleration is:', np.average(df.acceleration))
print('\n')
print('Median of mpg is:', np.median(df.mpg))
print('Median of cylinders is:',np.median(df.cylinders))
print('Median of displacement is:',np.median(df.displacement))
print('Median of weight is:', np.median(df.weight))
print('Median of acceleration is:', np.median(df.acceleration))
print('\n')
print('trimmed Mean of mpg is:', stats.trim_mean(df.mpg,0.25))
print('Trimmed Mean of cylinders is:',stats.trim_mean(df.cylinders,0.25))
print('Trimmed Mean of displacement is:',stats.trim_mean(df.displacement,0.25))
print('Trimmed Mean of horsepower is:', stats.trim_mean(df.horsepower,0.25))
print('Trimmed Mean of weight is:', stats.trim_mean(df.weight,0.25))
print('Trimmed Mean of acceleration is:', stats.trim_mean(df.acceleration,0.25))
lst1=random.randint(1,10)
df.insert(4,'Weights',lst1,True)
print('\n')
print('Weighted mean of mpg is:',np.average(df.mpg,weights=df.Weights))
print('Weighted mean of cylinders is:',np.average(df.cylinders,weights=df.Weights))
print('Weighted mean of displacement is:',np.average(df.displacement,weights=df.Weights))
print('Weighted mean of horsepower is:',np.average(df.horsepower,weights=df.Weights))
print('Weighted mean of weight is:',np.average(df.weight,weights=df.Weights))
print('Weighted mean of acceleration is:',np.average(df.acceleration,weights=df.Weights))
```

```
In [20]: runfile('C:/Users/mvsla/OneDrive/Desktop/ml/task_4_dataset2.py', wdir='C:/
Users/mvsla/OneDrive/Desktop/ml')
      mpg  cylinders  ...  origin                   car name
0    18.0          8  ...       1  chevrolet chevelle malibu
1    15.0          8  ...       1          buick skylark 320
2    18.0          8  ...       1         plymouth satellite
3    16.0          8  ...       1              amc rebel sst
4    17.0          8  ...       1                ford torino
..    ...        ...  ...     ...                        ...
393  27.0          4  ...       1           ford mustang gl
394  44.0          4  ...       2                 vw pickup
395  32.0          4  ...       1             dodge rampage
396  28.0          4  ...       1               ford ranger
397  31.0          4  ...       1                chevy s-10
```

```
[398 rows x 9 columns]
mpg             False
cylinders       False
displacement    False
horsepower       True
weight          False
acceleration    False
model year      False
origin          False
car name         True
dtype: bool
mpg             0
cylinders       0
displacement    0
horsepower      0
weight          0
acceleration    0
model year      0
origin          0
car name        0
dtype: int64
```

```
mpg             float64
cylinders         int64
displacement    float64
horsepower        int32
weight            int64
acceleration    float64
model year        int64
origin            int64
car name         object
dtype: object
```

```
Mean of mpg is: 23.514572864321607
Mean of cylinders is: 5.454773869346734
Mean of displacement is: 193.42587939698493
Mean of horsepower is: 102.89447236180905
Mean of weight is: 2970.424623115578
Mean of acceleration is: 15.568090452261307
```

```
Median of mpg is: 23.0
Median of cylinders is: 4.0
Median of displacement is: 148.5
Median of weight is: 2803.5
Median of acceleration is: 15.5
```

```
trimmed Mean of mpg is: 22.7675
Trimmed Mean of cylinders is: 4.935
Trimmed Mean of displacement is: 167.805
Trimmed Mean of horsepower is: 94.48
Trimmed Mean of weight is: 2839.855
Trimmed Mean of acceleration is: 15.453000000000001
```

```
Weighted mean of mpg is: 23.514572864321607
Weighted mean of cylinders is: 5.454773869346734
Weighted mean of displacement is: 193.42587939698493
Weighted mean of horsepower is: 102.89447236180905
Weighted mean of weight is: 2970.424623115578
Weighted mean of acceleration is: 15.568090452261307
```

Observations:

- ❖ Each measure of central tendency needed a separate library to execute and they are giving different results as per their formulae.
- ❖ All these measures can be applied only on numerical data.

Conclusion:

According to the requirement and as per the datatype, appropriate measure of central tendency must be applied for a given dataset in a dataframe.

Task 5:

How do you index your dataframe and select the desired data from the dataframe?

AIM: To index and select data from the dataframe.

Methods and dataset description:

- We are taking two datasets into consideration for applying the indexing and selecting techniques.
- The datasets are: nba-2.csv dataset and iris dataset.
- The nba dataset has 4 numerical variables and 4 categorical variables and it has 84 missing values in college and 11 missing values in salary column and is of the size 457 rows x 8 columns.
- The iris dataset has 150 rows x 5 columns and has no missing values. It has 4 numerical variables and 1 class variable.
- Indexing in a dataframe is nothing but the process of selecting the set of rows required for the analysis and training dataset preparation, especially in the case of supervised learning.
- The indexing can be done in 4 ways:
  - Index operator []
  - df.loc[]
  - df.iloc[]
  - df.ix[]

Source code and output:

For nba-dataset:

```
import pandas as pd
df=pd.read_csv('nba-2.csv',index_col='Name')
print(df)
print(df.dtypes=='object','\n\n')
print(df.isnull().sum(),'\n\n')
#indexing using dataframe index operator
print(df['Age','College','Salary']],'\n\n\n')
#indexing using loc[]
first=df.loc['Avery Bradley']
second=df.loc['R.J. Hunter']
print(first,'\n\n\n',second,'\n')
print(df.loc[['Avery Bradley','R.J. Hunter']])
print(df.loc[['Avery Bradley','R.J. Hunter'],['Team','Number','Position']])
print(df.loc[:,['Team','Number','Position']])
#indexing using iloc[]
row2=df.iloc[3]
print(row2,'\n\n')
print(df.iloc[[3,5,7]],'\n\n\n')
print(df.iloc[[3,4],[1,2]],'\n\n\n')
print(df.iloc[:,[1,2]],'\n\n')
#indexing using ix[]
#first=df.ix['Avery Bradley']
#second=df.ix[1]
#print(first,'\n\n',second)
print(df.head())
print(df.tail())
```

```
                          Team  Number  ...            College     Salary
Name                                    ...
Avery Bradley   Boston Celtics       0  ...              Texas  7730337.0
Jae Crowder     Boston Celtics      99  ...          Marquette  6796117.0
John Holland    Boston Celtics      30  ...  Boston University        NaN
R.J. Hunter     Boston Celtics      28  ...      Georgia State  1148640.0
Jonas Jerebko   Boston Celtics       8  ...                NaN  5000000.0
...                        ...     ...  ...                ...        ...
Trey Lyles           Utah Jazz      41  ...          Kentucky  2239800.0
Shelvin Mack         Utah Jazz       8  ...            Butler  2433333.0
Raul Neto            Utah Jazz      25  ...                NaN   900000.0
Tibor Pleiss         Utah Jazz      21  ...                NaN  2900000.0
Jeff Withey          Utah Jazz      24  ...            Kansas   947276.0

[457 rows x 8 columns]
```

```
Team          0
Number        0
Position      0
Age           0
Height        0
Weight        0
College      84
Salary       11
dtype: int64
```

```
Team          True
Number       False
Position      True
Age          False
Height        True
Weight       False
College       True
Salary       False
dtype: bool
```

```
                Age          College     Salary
Name
Avery Bradley    25            Texas  7730337.0
Jae Crowder      25        Marquette  6796117.0
John Holland     27  Boston University        NaN
R.J. Hunter      22      Georgia State  1148640.0
Jonas Jerebko    29              NaN  5000000.0
...             ...              ...        ...
Trey Lyles       20          Kentucky  2239800.0
Shelvin Mack     26            Butler  2433333.0
Raul Neto        24              NaN   900000.0
Tibor Pleiss     26              NaN  2900000.0
Jeff Withey      26            Kansas   947276.0

[457 rows x 3 columns]
```

```
Team         Boston Celtics
Number                    0
Position                 PG
Age                      25
Height                06-Feb
Weight                   180
College                Texas
Salary            7730337.0
Name: Avery Bradley, dtype: object
```

```
Team         Boston Celtics
Number                   28
Position                 SG
Age                      22
Height                06-May
Weight                   185
College       Georgia State
Salary           1148640.0
Name: R.J. Hunter, dtype: object
```

```
                          Team  Number  ...          College     Salary
Name                                    ...
Avery Bradley   Boston Celtics       0  ...            Texas  7730337.0
R.J. Hunter     Boston Celtics      28  ...  Georgia State  1148640.0

[2 rows x 8 columns]
```

```
                          Team  Number Position
Name
Avery Bradley   Boston Celtics       0       PG
Jae Crowder     Boston Celtics      99       SF
John Holland    Boston Celtics      30       SG
R.J. Hunter     Boston Celtics      28       SG
Jonas Jerebko   Boston Celtics       8       PF
...                        ...     ...      ...
Trey Lyles           Utah Jazz      41       PF
Shelvin Mack         Utah Jazz       8       PG
Raul Neto            Utah Jazz      25       PG
Tibor Pleiss         Utah Jazz      21        C
Jeff Withey          Utah Jazz      24        C

[457 rows x 3 columns]
```

```
Team         Boston Celtics
Number                   28
Position                 SG
Age                      22
Height                06-May
Weight                   185
College       Georgia State
Salary           1148640.0
Name: R.J. Hunter, dtype: object
```

```
                       Team  Number  ...        College      Salary
Name                                   ...
R.J. Hunter   Boston Celtics      28  ...  Georgia State   1148640.0
Amir Johnson  Boston Celtics      90  ...            NaN  12000000.0
Kelly Olynyk  Boston Celtics      41  ...        Gonzaga   2165160.0

[3 rows x 8 columns]
```

```
              Number Position
Name
R.J. Hunter       28       SG
Jonas Jerebko      8       PF
```

```
               Number Position
Name
Avery Bradley       0       PG
Jae Crowder        99       SF
John Holland       30       SG
R.J. Hunter        28       SG
Jonas Jerebko       8       PF
...               ...      ...
Trey Lyles         41       PF
Shelvin Mack        8       PG
Raul Neto          25       PG
Tibor Pleiss       21        C
Jeff Withey        24        C

[457 rows x 2 columns]
```

```
                       Team  Number  ...            College     Salary
Name                                   ...
Avery Bradley  Boston Celtics       0  ...              Texas  7730337.0
Jae Crowder    Boston Celtics      99  ...          Marquette  6796117.0
John Holland   Boston Celtics      30  ...  Boston University        NaN
R.J. Hunter    Boston Celtics      28  ...      Georgia State  1148640.0
Jonas Jerebko  Boston Celtics       8  ...                NaN  5000000.0

[5 rows x 8 columns]
                  Team  Number Position  ...  Weight   College     Salary
Name                                     ...
Trey Lyles    Utah Jazz      41       PF  ...     234  Kentucky  2239800.0
Shelvin Mack  Utah Jazz       8       PG  ...     203    Butler  2433333.0
Raul Neto     Utah Jazz      25       PG  ...     179       NaN   900000.0
Tibor Pleiss  Utah Jazz      21        C  ...     256       NaN  2900000.0
Jeff Withey   Utah Jazz      24        C  ...     231    Kansas   947276.0

[5 rows x 8 columns]
```

For Iris dataset:

```
import pandas as pd
df=pd.read_csv('iris.csv')
print(df)
print(df.dtypes=='object','\n\n')
print(df.isnull().sum(),'\n\n')
#using loc[]
x=df.loc[:,['sepal_length','sepal_width','petal_length','petal_width']]
y=df.loc[:,'species']
print(x)
print(y)
#using iloc[]
print('\n')
x1=df.iloc[:,:-1]
y1=df.iloc[:,-1]
print(x1)
print(y1)
```

```
Users/MVSTU/OneDrive/Desktop/ML')
     sepal_length  sepal_width  petal_length  petal_width    species
0             5.1          3.5           1.4          0.2     setosa
1             4.9          3.0           1.4          0.2     setosa
2             4.7          3.2           1.3          0.2     setosa
3             4.6          3.1           1.5          0.2     setosa
4             5.0          3.6           1.4          0.2     setosa
..            ...          ...           ...          ...        ...
145           6.7          3.0           5.2          2.3  virginica
146           6.3          2.5           5.0          1.9  virginica
147           6.5          3.0           5.2          2.0  virginica
148           6.2          3.4           5.4          2.3  virginica
149           5.9          3.0           5.1          1.8  virginica

[150 rows x 5 columns]
```

```
sepal_length    False
sepal_width     False
petal_length    False
petal_width     False
species          True
dtype: bool
```

```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

```
     sepal_length  sepal_width  petal_length  petal_width
0             5.1          3.5           1.4          0.2
1             4.9          3.0           1.4          0.2
2             4.7          3.2           1.3          0.2
3             4.6          3.1           1.5          0.2
4             5.0          3.6           1.4          0.2
..            ...          ...           ...          ...
145           6.7          3.0           5.2          2.3
146           6.3          2.5           5.0          1.9
147           6.5          3.0           5.2          2.0
148           6.2          3.4           5.4          2.3
149           5.9          3.0           5.1          1.8

[150 rows x 4 columns]
```

```
0         setosa
1         setosa
2         setosa
3         setosa
4         setosa
         ...
145    virginica
146    virginica
147    virginica
148    virginica
149    virginica
Name: species, Length: 150, dtype: object
```

```
     sepal_length  sepal_width  petal_length  petal_width
0             5.1          3.5           1.4          0.2
1             4.9          3.0           1.4          0.2
2             4.7          3.2           1.3          0.2
3             4.6          3.1           1.5          0.2
4             5.0          3.6           1.4          0.2
..            ...          ...           ...          ...
145           6.7          3.0           5.2          2.3
146           6.3          2.5           5.0          1.9
147           6.5          3.0           5.2          2.0
148           6.2          3.4           5.4          2.3
149           5.9          3.0           5.1          1.8

[150 rows x 4 columns]
```

```
0         setosa
1         setosa
2         setosa
3         setosa
4         setosa
         ...
145    virginica
146    virginica
147    virginica
148    virginica
149    virginica
Name: species, Length: 150, dtype: object
```

Observations:

- ❖ Index operator does the selection of columns based on the column labels, however we can select all the rows not a particular set of row values.
- ❖ df.loc[] method works based on column values and we can also provide the range of row values to be selected.
- ❖ df.iloc[] method works based on the column indexes and row indexes we can select the dataframes.
- ❖ df.ix[] method method can work based on both index values and labels, however this method has been depricated due to inconsistency.

Conclusion:

loc[] and iloc[] methods can be used as per the requirement and iloc[] is more preferable as the generalization to training set preparation for model fitting.

TASK 6: Perform Principal Component Analysis for dimensionality reduction

AIM : To perform principal component analysis and also identify the best number of principal components for a given dataset using the explained variance ratio and scree plot

Methods and dataset description:

- Principal component analysis the one of the most important dimensionality reduction technique for feature transformation.
- PCA reduces the size of the dataset from the n dimensional space to m dimansional space where m<n, without the loss of data.
- PCA converts the dependent features into independent features by applying the concept of Eigen vectors and Eigen values to the features in a dataset.
- The optimal number of features for the data analysis can be decided using evaluation metrics like 'Explained Variance Ratio' (EVR) and visualization metric like scree plot.
- PCA is performed on two datasets : wine dataset and iris dataset.

Source Code and output:

Source Code for wine dataset:

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

# importing or loading the dataset
dataset = pd.read_csv('wine.csv')
print(dataset.info())

# distributing the dataset into two components X and Y
X = dataset.iloc[:, :-1].values
Y = dataset.iloc[:, -1].values


from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,random_state=None

from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
X_train=sc.fit_transform(X_train)
X_test=sc.transform(X_test)
```

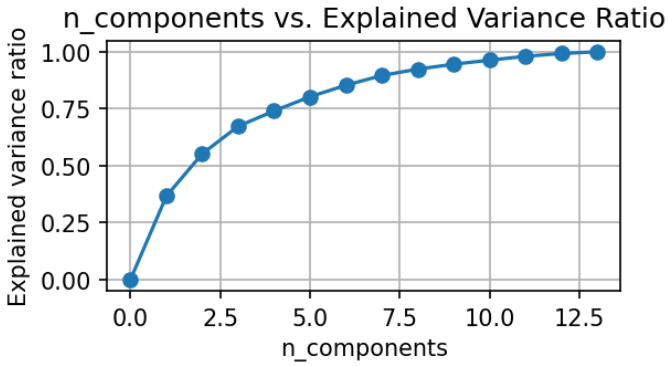
```
#explained variance ratio
from sklearn.decomposition import PCA
pca=PCA(n_components=3)
X_train_pca=pca.fit_transform(X_train)
X_test = pca.transform(X_test)

print(pca.components_)

print(sum(pca.explained_variance_ratio_))

nums = np.arange(14)
var_ratio = []
for num in nums:
  pca = PCA(n_components=num)
  pca.fit(X_train)
  var_ratio.append(np.sum(pca.explained_variance_ratio_))

plt.figure(figsize=(4,2),dpi=150)
plt.grid()
plt.plot(nums,var_ratio,marker='o')
plt.xlabel('n_components')
plt.ylabel('Explained variance ratio')
plt.title('n_components vs. Explained Variance Ratio')
plt.show()
```

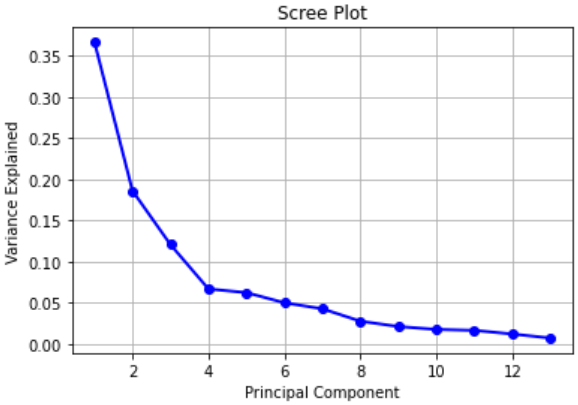
```
#scree plot
PC_values = np.arange(pca.n_components_) + 1
plt.grid()
plt.plot(PC_values, pca.explained_variance_ratio_, 'o-', linewidth=2, color='blue')
plt.title('Scree Plot')
plt.xlabel('Principal Component')
plt.ylabel('Variance Explained')
plt.show()
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 178 entries, 0 to 177
Data columns (total 14 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Alcohol               178 non-null    float64
 1   Malic_Acid            178 non-null    float64
 2   Ash                   178 non-null    float64
 3   Ash_Alcanity          178 non-null    float64
 4   Magnesium             178 non-null    int64
 5   Total_Phenols         178 non-null    float64
 6   Flavanoids            178 non-null    float64
 7   Nonflavanoid_Phenols  178 non-null    float64
 8   Proanthocyanins       178 non-null    float64
 9   Color_Intensity       178 non-null    float64
 10  Hue                   178 non-null    float64
 11  OD280                 178 non-null    float64
 12  Proline               178 non-null    int64
 13  Customer_Segment      178 non-null    int64
dtypes: float64(11), int64(3)
memory usage: 19.6 KB
None
```

```
[[-0.16484498  0.25435796 -0.02522561  0.21257657 -0.15877917 -0.40345525
  -0.42360664  0.29417206 -0.3078513   0.05798791 -0.29029427 -0.37100192
  -0.29713312]
 [-0.47956815 -0.20769407 -0.24803498  0.03614881 -0.31875337 -0.02989254
   0.03540574 -0.00987348 -0.0166148  -0.55171126  0.29462463  0.20912203
  -0.35355917]
 [-0.18152584  0.08877076  0.65047395  0.61251705  0.18771764  0.12400399
   0.12779098  0.19647242  0.12270532 -0.08978986  0.08723936  0.12958289
  -0.08835461]]
0.673067392407096
```





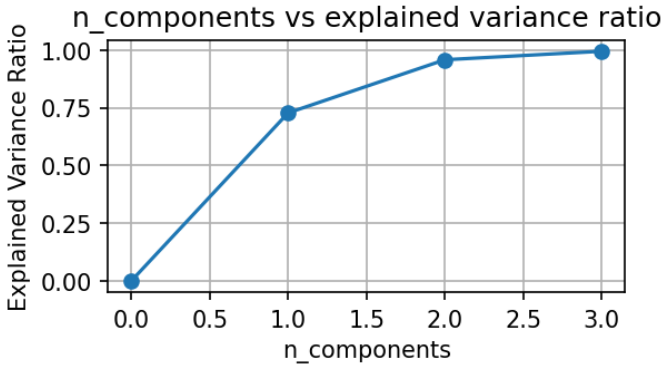
Source Code for iris dataset:

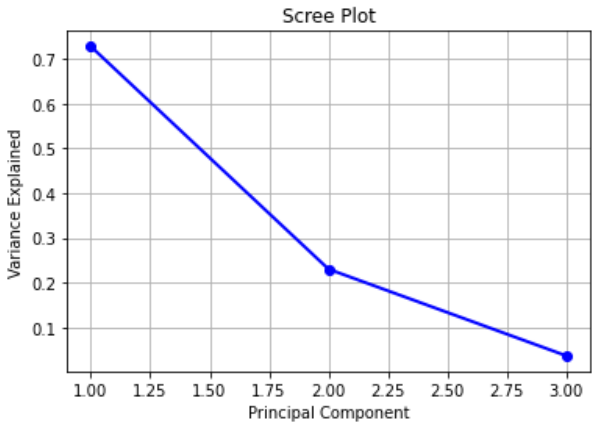
```python
#importing the libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

#import the dataset
df=pd.read_csv('iris.csv')
print(df.info())
X=df.iloc[:,:-1].values

#scaling the features
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
X=sc.fit_transform(X)

#principal component analysis
from sklearn.decomposition import PCA
pca=PCA(n_components=2)
X_pca=pca.fit_transform(X)
print(sum(pca.explained_variance_ratio_))

#explained variance ratio
nums=np.arange(4)
var_ratio=[]
for num in nums:
    pca=PCA(n_components=num)
    pca.fit(X)
    var_ratio.append(np.sum(pca.explained_variance_ratio_))

#display the graph
plt.figure(figsize=(4,2),dpi=150)
plt.grid()
plt.plot(nums,var_ratio,marker='o')
plt.xlabel('n_components')
plt.ylabel('Explained Variance Ratio')
plt.title('n_components vs explained variance ratio')
plt.show()
```

```
#scree plot
PC_values = np.arange(pca.n_components_) + 1
plt.grid()
plt.plot(PC_values, pca.explained_variance_ratio_, 'o-', linewidth=2, color='blue')
plt.title('Scree Plot')
plt.xlabel('Principal Component')
plt.ylabel('Variance Explained')
plt.show()
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
None
0.9580097536148198
```





Note: Source for scree plot is in the link given below
https://www.statology.org/scree-plot-python/

Observations:

- The optimal principal components for the wine dataset is 4 or 5 can be considered, as per the EVR and scree plot visualizations.
- The optimal principal components for the iris dataset is 2.

Conclusion:

We can say that the number of features in the wine dataset are reduced from 13 to 5 and for iris dataset, the number of features can be reduced from 4 to 2 features using Principal Component Analysis (PCA) and the machine learning algorithms can be implemented accordingly.

TASK 7: Performing Exploratory data analysis Using data visualizations and Statistics

AIM: To perform EDA on the given dataset and explore various types of data visualizations for univariate, bivariate and multivariate analysis.

Methods and dataset description:

- The red wine quality dataset is used to perform Exploratory Data Analysis.
- It has the dimensions as 1599 x 12 and has no missing values.
- It has 12 attributes as numeric and no categorical attributes.
- Exploratory Data analysis involves the process of exploring the data using statistical and data visulaization techniques to understand and improve the quality of the data.
- EDA also helps in understanding the type of machine learning algorithm is most suitable to the given dataset.

Source Code:

```
#importing the libraries
import numpy as np
import pandas as  pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings as wr
wr.filterwarnings('ignore')

#importing the dataset
df=pd.read_csv('winequality-red.csv')
print(df.head(),'\n')
print(df.shape,'\n')
print(df.dtypes=='object','\n\n')
print(df.info(),'\n')
print(df.describe(),'\n')
print(df.nunique())
print(df.isnull().sum(),'\n')
print(df.columns.tolist(),'\n')
```

```
#Multivariate analysis
#Heatmap using correlation matrix
plt.figure(figsize=(15,10))
sns.heatmap(df.corr(),annot=True,fmt='2f',cmap='Pastel2',linewidths=2)
plt.title('Correlation Heatmap')
plt.show()
```

```
#EXPLORATORY DATA ANALYSIS
#Univariate analysis

#creating a count plot
quality_counts=df['quality'].value_counts()
plt.figure(figsize=(8,6))
plt.bar(quality_counts.index,quality_counts,color='green')
plt.title('Count plot of quality')
plt.xlabel('count')
plt.ylabel('quality')
plt.show()

#creating the kernel density plots
sns.set_style('darkgrid')
numerical_columns=df.select_dtypes(include=['int64','float64']).columns
plt.figure(figsize=(14,len(numerical_columns)*3))
for idx, feature in enumerate(numerical_columns,1):
    plt.subplot(len(numerical_columns),2,idx)
    sns.histplot(df[feature],kde=True)
    plt.title(f"{feature}| skewness:{round(df[feature].skew(),2)}")

plt.tight_layout()
plt.show()

#creating a swarm plot
plt.figure(figsize=(10,8))
sns.swarmplot(x='quality',y='alcohol',data=df,palette='viridis')
plt.title('Swarm plot for quality and Alcohol')
plt.xlabel('Quality')
plt.ylabel('Alcohol')
plt.show()
```

```
#Bivariate analysis
#pair plots
sns.set_palette('pastel')
plt.figure(figsize=(10,6))
sns.pairplot(df)
plt.suptitle('Pair Plot Of dataframe')
plt.show()

#creating viloin plots
df['quality']=df['quality'].astype(str)
plt.figure(figsize=(10,8))
sns.violinplot(x="quality", y="alcohol", data=df, palette={
               '3': 'lightcoral', '4': 'lightblue', '5': 'lightgreen', '6': 'gold',
               '7':'lightskyblue', '8': 'lightpink'}, alpha=0.7)
plt.title('violin plot for quality and alcohol')
plt.xlabel('quality')
plt.ylabel('alcohol')
plt.show()

#creating a box plot
sns.boxplot(x='quality',y='alcohol',data=df)
```

Output:

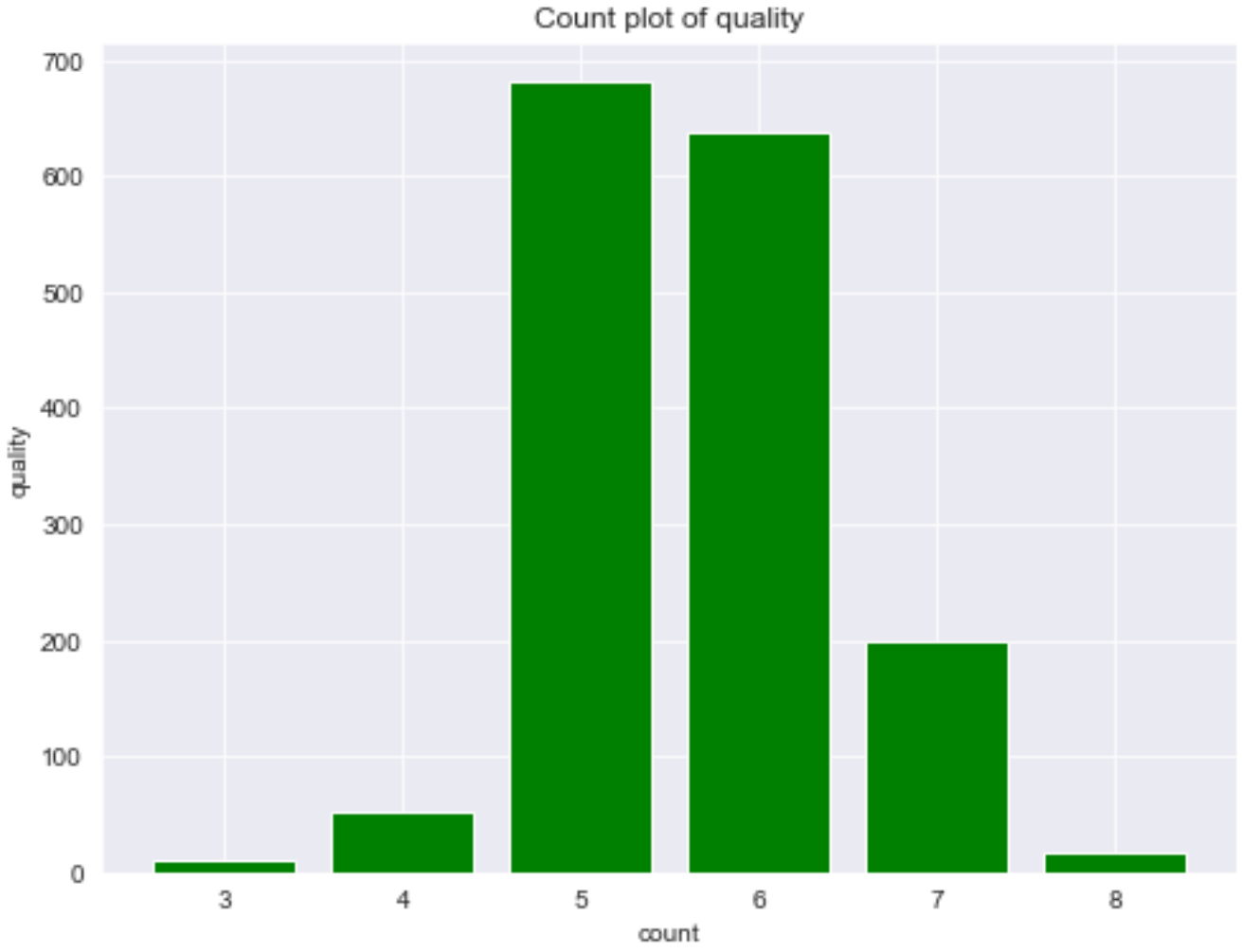
```
IPython 8.15.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/mvsla/OneDrive/Desktop/ml/eda1.py', wdir='C:/Users/mvsla/OneDrive/Desktop/ml')
   fixed acidity  volatile acidity  citric acid  ...  sulphates  alcohol  quality
0            7.4              0.70         0.00  ...       0.56      9.4        5
1            7.8              0.88         0.00  ...       0.68      9.8        5
2            7.8              0.76         0.04  ...       0.65      9.8        5
3           11.2              0.28         0.56  ...       0.58      9.8        6
4            7.4              0.70         0.00  ...       0.56      9.4        5

[5 rows x 12 columns]

(1599, 12)

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   int64
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
None

       fixed acidity  volatile acidity  ...      alcohol      quality
count    1599.000000       1599.000000  ...  1599.000000  1599.000000
mean        8.319637          0.527821  ...    10.422983     5.636023
std         1.741096          0.179060  ...     1.065668     0.807569
min         4.600000          0.120000  ...     8.400000     3.000000
25%         7.100000          0.390000  ...     9.500000     5.000000
50%         7.900000          0.520000  ...    10.200000     6.000000
75%         9.200000          0.640000  ...    11.100000     6.000000
max        15.900000          1.580000  ...    14.900000     8.000000

[8 rows x 12 columns]

fixed acidity           96
volatile acidity       143
citric acid             80
residual sugar          91
chlorides              153
free sulfur dioxide     60
total sulfur dioxide   144
density                436
pH                      89
sulphates               96
alcohol                 65
quality                  6
dtype: int64

fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
dtype: int64
```

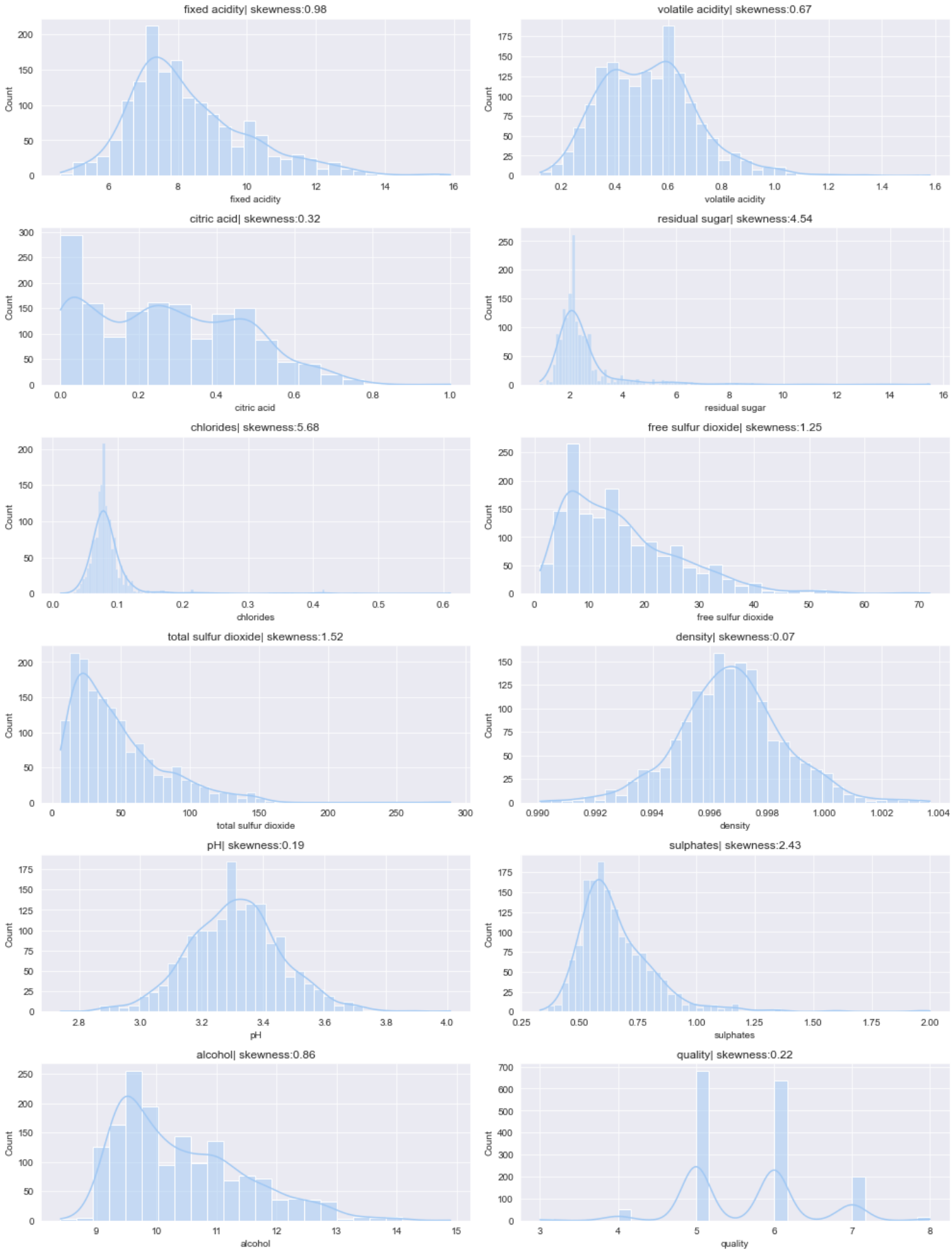
```
['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar', 'chlorides', 'free sulfur dioxide',
'total sulfur dioxide', 'density', 'pH', 'sulphates', 'alcohol', 'quality']
```

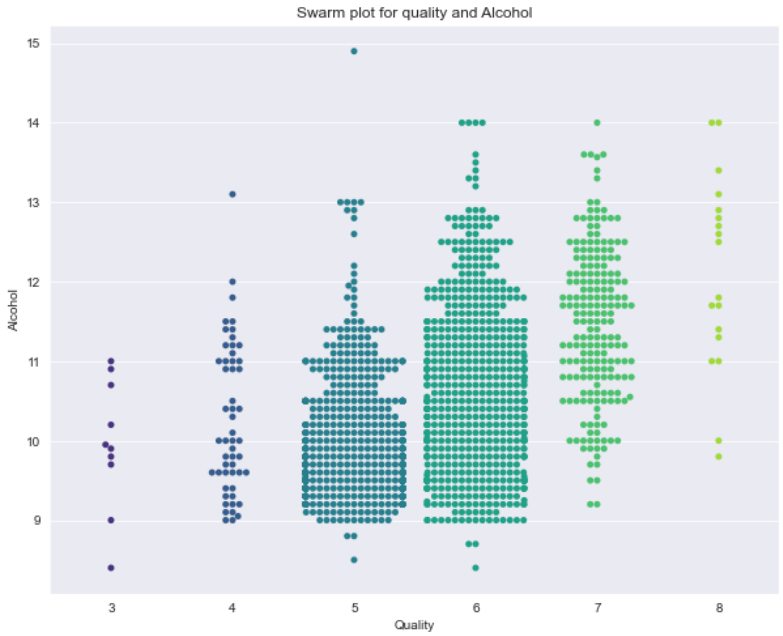
Important

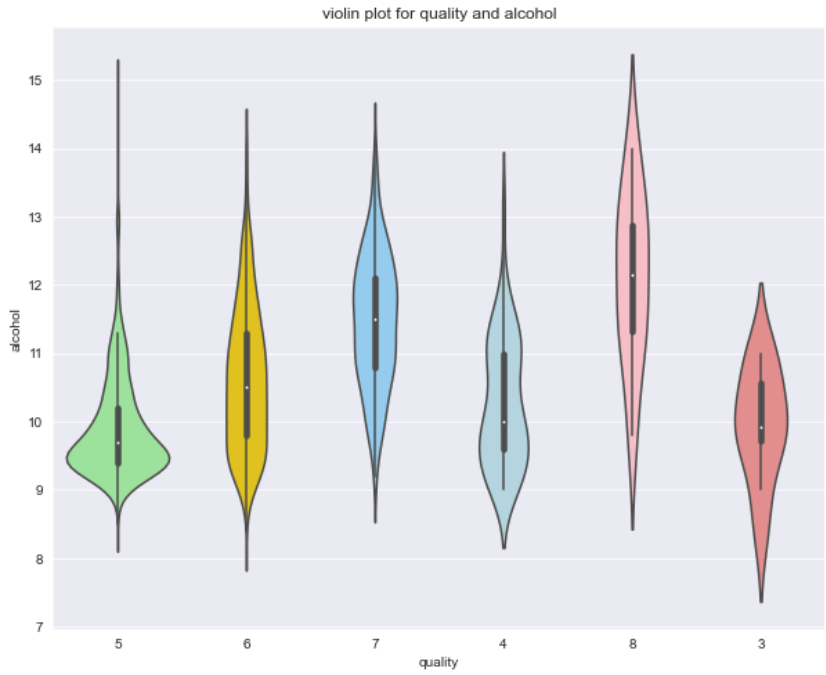
Figures are displayed in the Plots pane by default. To make them also appear inline in the console, you need to uncheck "Mute inline plotting" under the options menu of Plots.

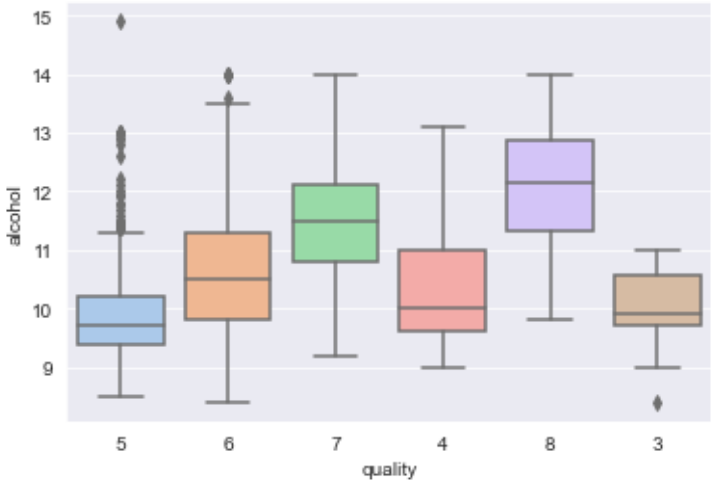
```
 <Figure size 720x432 with 0 Axes>
runfile('C:/Users/mvsla/OneDrive/Desktop/ml/eda1.py', wdir='C:/Users/mvsla/OneDrive/Desktop/ml')
```

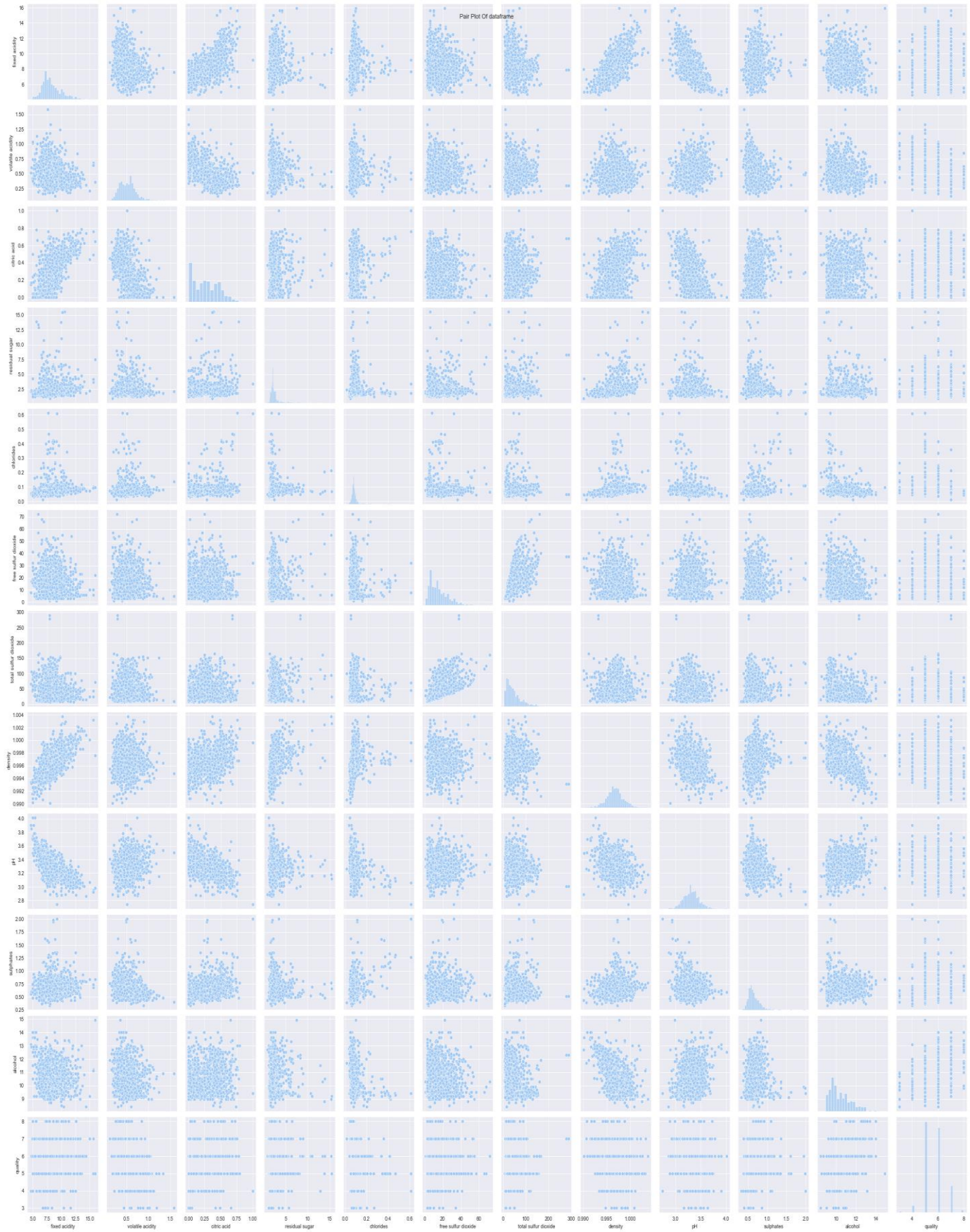


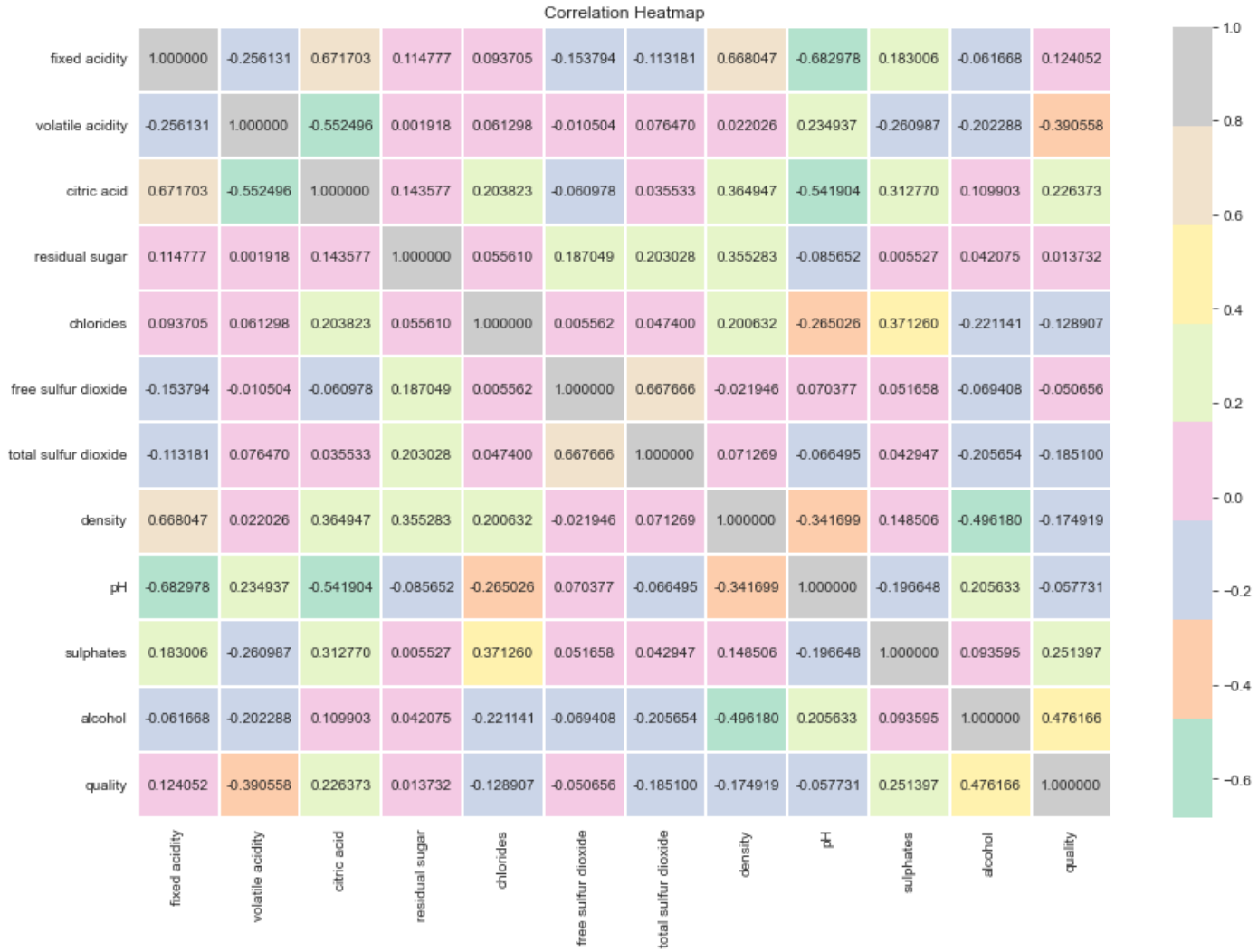

fixed acidity| skewness:0.98
Count
fixed acidity
volatile acidity| skewness:0.67
Count
volatile acidity
citric acid| skewness:0.32
Count
citric acid
residual sugar| skewness:4.54
Count
residual sugar
chlorides| skewness:5.68
Count
chlorides
free sulfur dioxide| skewness:1.25
Count
free sulfur dioxide
total sulfur dioxide| skewness:1.52
Count
total sulfur dioxide
density| skewness:0.07
Count
density
pH| skewness:0.19
Count
pH
sulphates| skewness:2.43
Count
sulphates
alcohol| skewness:0.86
Count
alcohol
quality| skewness:0.22
Count
quality


Swarm plot for quality and Alcohol
Alcohol
Quality
15
14
13
12
11
10
9
3
4
5
6
7
8


violin plot for quality and alcohol
alcohol
quality
15
14
13
12
11
10
9
8
7
5
6
7
4
8
3


alcohol
quality
15
14
13
12
11
10
9
5
6
7
4
8
3


Pair Plot Of dataframe
fixed acidity
volatile acidity
citric acid
residual sugar
chlorides
free sulfur dioxide
total sulfur dioxide
density
pH
sulphates
alcohol
quality



Observations:

- ✓ The count plot is showing that most of the wine is of quality mark 5 or 6.
- ➢ From the KDE plot we can infer that:
  - Fixed acidity attribute has the skewness as 0.98
  - volatile acidity has the skewness as 0.67
  - Citric acid has skewness as 0.32
  - Residual sugar has skewness as 4.54
  - Chlorides has the skewness as 5.68
  - Free Sulphur dioxide has the skewness as 1.25
  - Total sulphur dioxide has the skewness as 1.52
  - Density has the skewness as 0.07
  - pH level has the skewness as 0.19
  - Sulphates has the skewness as 2.43
  - Alcohol skewness is 0.86
  - Quality attribute skewness is 0.22

- ✓ Swarm plot is showing that the density of the data points is highest in the quality bars of 5 and then 6, later comes the bar of 7
- ✓ Violin plot is giving the observation that in the target feature quality has most of the data points in the quality value 5 and they are highly dense in between 9 and 10 percentage of alcohol
- ✓ The box plot is showing that there are multiple outliers in the 5 quality category with respective to alcohol percentage
- ✓ Pair plot is giving the correlation between each pair of features so it gives us a result of 144 plot diagrams
- ✓ The correlation heatmap is giving the correlation between each pair of the observations in the numeric format i.e. it is giving the results of the pairplots in the numeric format. The correlation heatmap is a visual representation of the covariance matrix of the entire dataset.

Conclusion:

Each of the data visualization technique has its own unique use and they can be used to understand the data better and assist in employing the appropriate machine learning algorithm to solve the problem.

TASK 8: K Nearest Neighbours Classification with elbow method

AIM: Creating a Classifier to for the given dataset using the K nearest neighbours algorithm and deciding the optimal k value using the elbow method and cross validation.

Methods and Description:

- K Nearest neighbours, or simply called as KNN is a simple, yet powerful classifier algorithm.
- It belongs to the class of Lazy learners, so the training phase time is almost negligible.
- When the test set arrives, then KNN takes the training data and then try to find the nearest 'k' neighbours to each test data element and assign the appropriate class to the test data item.
- The KNN algorithm is implemented on the diabetes datase, which is of dimensions 768 x 9 and has no missing values and categorical variables.
- The target feature is labelled as 'Outcome'.
- The training and testing data split is in the ratio of 70:30.
- Principal component Analysis is used to increase the accuracy of the model.
- The most important hyperparameter for this model is 'k' value and the most optimal 'k' value can be predicted using elbow method and cross validation.

Source Code:

```
#importing the necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings as wr
wr.filterwarnings('ignore')

#importing the dataset
df=pd.read_csv('diabetes.csv')
df.head()

#dataset information
print(df.dtypes=='object')
df.info()

#dataset dimensions
df.shape

#statistics of the numeric attributes in the dataset
df.describe()

#checking the missing values
df.isnull().sum()
```

```
#extracting the target feature
x=df.drop('Outcome',axis=1)
y=df['Outcome']

#feature scaling by standardization
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
ds=sc.fit_transform(x)
data=pd.DataFrame(ds,columns=x.columns)
print(data.head())

#pairplot for interaction among the features
sns.pairplot(df,hue='Outcome')

#splitting the dataset into training and testing
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(data,y,test_size=0.30,
                                               random_state=10)

#dimesions of features and the target feature
print(x.shape,y.shape)

#dimensions of the training and testing data
print('x_train:  ',x_train.shape)
print('y_train:  ',y_train.shape)
print('x_test:  ',x_test.shape)
print('y_test:  ',y_test.shape)
```

```
#importing K nearest Neighbours
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=9)

#training the model
knn.fit(x_train,y_train)

#performance evaluation
pred=knn.predict(x_test)
print(pred)

#accuracy
print(knn.score(x_test,y_test)*100)
```

```
#PCA for increasing the accuracy of the model
from sklearn.decomposition import PCA
pca=PCA(0.95)
x_pca=pca.fit_transform(x_train)
print(x_train)
print(pca.n_components_)

x_train_pca,x_test_pca,y_train,y_test=train_test_split(data,y,
                                        test_size=0.30,random_state=30)

from sklearn.neighbors import KNeighborsClassifier
knn_pca=KNeighborsClassifier(n_neighbors=9)
knn_pca.fit(x_train_pca,y_train)
knn_pca.score(x_test_pca,y_test)
pred_pca=knn_pca.predict(x_test_pca)
print(pred_pca)

#classification evaluation metrics
from sklearn.metrics import confusion_matrix,classification_report
cm=confusion_matrix(y_test,pred_pca)
print(cm)
```

```
#confusion matrix visualization
plt.figure(figsize=(10,8))
ax=sns.heatmap(cm,annot=True,cmap='Blues')
ax.set_title('Confusion Matrix with labels \n\n')
ax.set_xlabel('\nPredicted labels')
ax.set_ylabel('\nActual values')
plt.show()

#classification report of KNN
print(classification_report(y_test,pred))

#Classification report after pca
print(classification_report(y_test,pred_pca))
```

```
#elbow method for optimal k value
#cross validation
from sklearn.model_selection import cross_val_score

#accuracy rate calculation
accuracy_rate=[]
for i in range(1,40):
    knn=KNeighborsClassifier(n_neighbors=i)
    score=cross_val_score(knn,x,y,cv=10)
    accuracy_rate.append(score.mean())

#error rate calculation for plotting the graph
error_rate=[]
for i in range(1,40):
    knn=KNeighborsClassifier(n_neighbors=i)
    score=cross_val_score(knn,x,y,cv=10)
    error_rate.append(1-score.mean())

#plottimg the relation between the K value and error rate
plt.figure(figsize=(10,6))
plt.plot(range(1,40),error_rate,color='blue',marker='o',
         linestyle='dashed',markerfacecolor='red',markersize=10)
plt.title('error-rate va k-value')
plt.xlabel('k value')
plt.ylabel('error-rate')
plt.show()

#Checking for the accuracy at k=1
knn=KNeighborsClassifier(n_neighbors=1)
knn.fit(x_train,y_train)
y_pred1=knn.predict(x_test)
print('At K=1\n')
print(confusion_matrix(y_pred1,y_test),'\n')
print(classification_report(y_pred1,y_test))

#Checking the accuracy of the model at the most optimal k
#value from the elbow method i.e. K=18
knn=KNeighborsClassifier(n_neighbors=18)
knn.fit(x_train,y_train)
y_pred1=knn.predict(x_test)
print('At K=18\n')
print(confusion_matrix(y_pred1,y_test),'\n')
print(classification_report(y_pred1,y_test))
```

Output:

```
Pregnancies                 False
Glucose                     False
BloodPressure               False
SkinThickness               False
Insulin                     False
BMI                         False
DiabetesPedigreeFunction    False
Age                         False
Outcome                     False
dtype: bool
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

```
   Pregnancies   Glucose  ...  DiabetesPedigreeFunction       Age
0     0.639947  0.848324  ...                  0.468492  1.425995
1    -0.844885 -1.123396  ...                 -0.365061 -0.190672
2     1.233880  1.943724  ...                  0.604397 -0.105584
3    -0.844885 -0.998208  ...                 -0.920763 -1.041549
4    -1.141852  0.504055  ...                  5.484909 -0.020496

[5 rows x 8 columns]
```

```
(768, 8) (768,)
x_train:   (537, 8)
y_train:   (537,)
x_test:   (231, 8)
y_test:   (231,)
```

```
[1 0 1 0 0 0 1 0 1 0 0 1 0 0 1 1 0 0 1 0 1 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0
 0 1 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 1 0 0 0 1 1 1 1 0 0 0 0 0 0 0 1 0 0 0 1 1 0 0 0 0 0 1 0 1 0
 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 1 1 1 0 0 0 0 1 1 0 0
 1 0 0 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 1 1 0 0 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 1 0 0 1 0 0 0 1 1 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 1 0 0 0 1 0 0
 0 1 0 1 0 1 0 0 0]
70.12987012987013
```

```
     Pregnancies   Glucose  ...  DiabetesPedigreeFunction       Age
491    -0.547919 -0.998208  ...                 -0.543248  0.745293
69      0.046014  0.785730  ...                 -0.854320 -0.531023
667     1.827813 -0.309671  ...                 -0.999286  0.575118
566    -0.844885 -0.685236  ...                 -0.180834 -1.041549
97     -0.844885 -1.561556  ...                 -0.449624 -0.956462
..           ...       ...  ...                       ...       ...
369    -0.844885  0.378867  ...                 -0.718415  1.000557
320     0.046014  0.253678  ...                  0.166480 -0.190672
527    -0.250952 -0.153185  ...                 -1.101970 -0.786286
125    -0.844885 -1.029505  ...                  0.072856 -0.616111
265     0.342981 -0.779128  ...                  1.585936  0.830381

[537 rows x 8 columns]
```

```
8
[0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0
 0 1 0 0 1 0 0 1 1 1 1 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 1 1 1 0 0 1 0 0
 0 1 0 1 1 0 0 0 0 1 1 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 1 1 0 0 1 1 1 0 0 0 0 0
 0 0 0 0 0 1 1 0 0 1 1 0 0 1 0 0 0 0 0 0 0 1 0 1 1 0 1 0 0 0 1 0 0 0 0 0 1 0
 0 1 1 0 1 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 1 0 1 0 1 1 0 0 0 1 1 1 1 0 0 0 1
 0 0 0 0 0 0 1 1 1 0 1 0 1 0 1 0 0 1 0 0 0 0 0 0 0 1 0 1 0 0 0 1 0 1 0 0 0
 0 0 0 1 1 0 0 1 0]
[[129  30]
 [ 28  44]]
```

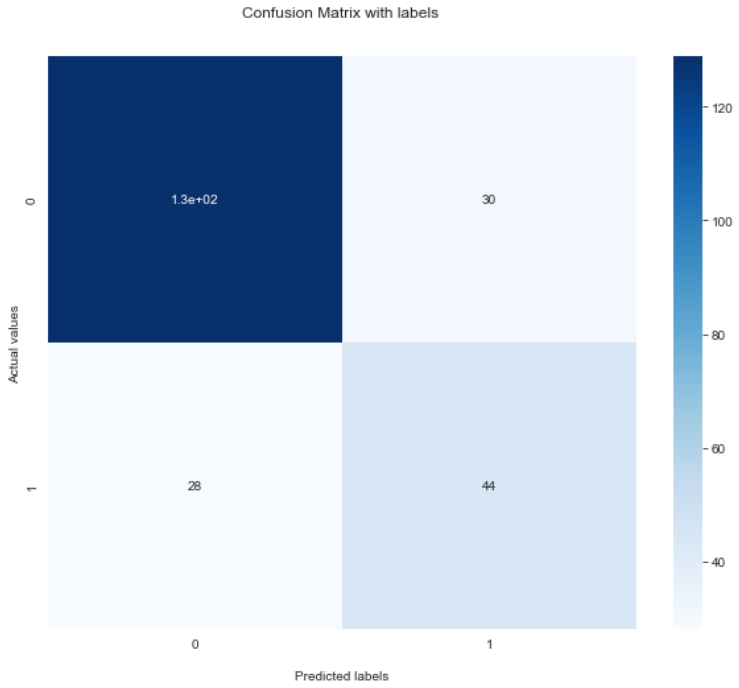
	precision	recall	f1-score	support
0	0.68	0.75	0.71	159
1	0.29	0.22	0.25	72
accuracy			0.58	231
macro avg	0.48	0.49	0.48	231
weighted avg	0.56	0.58	0.57	231

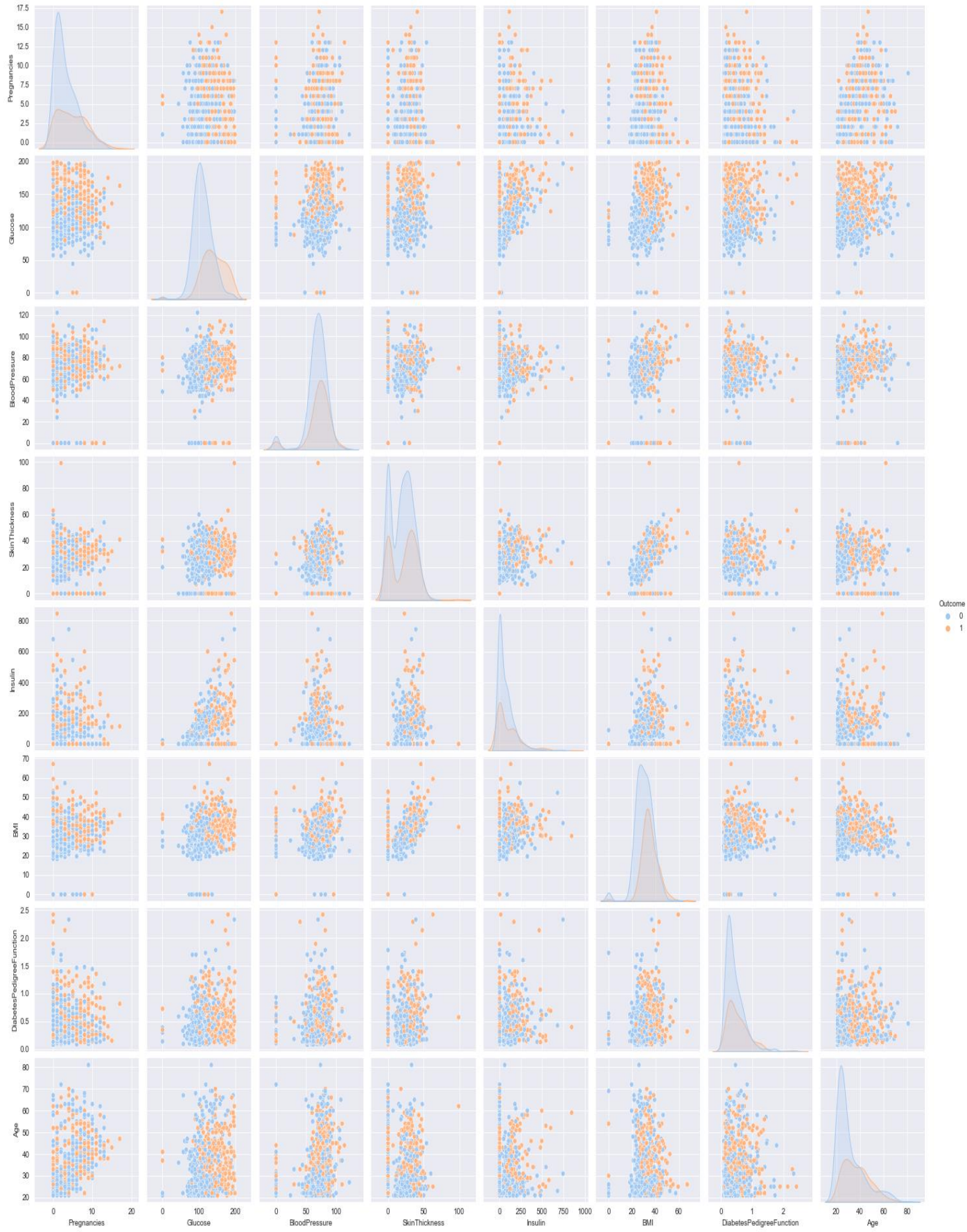
	precision	recall	f1-score	support
0	0.82	0.81	0.82	159
1	0.59	0.61	0.60	72
accuracy			0.75	231
macro avg	0.71	0.71	0.71	231
weighted avg	0.75	0.75	0.75	231

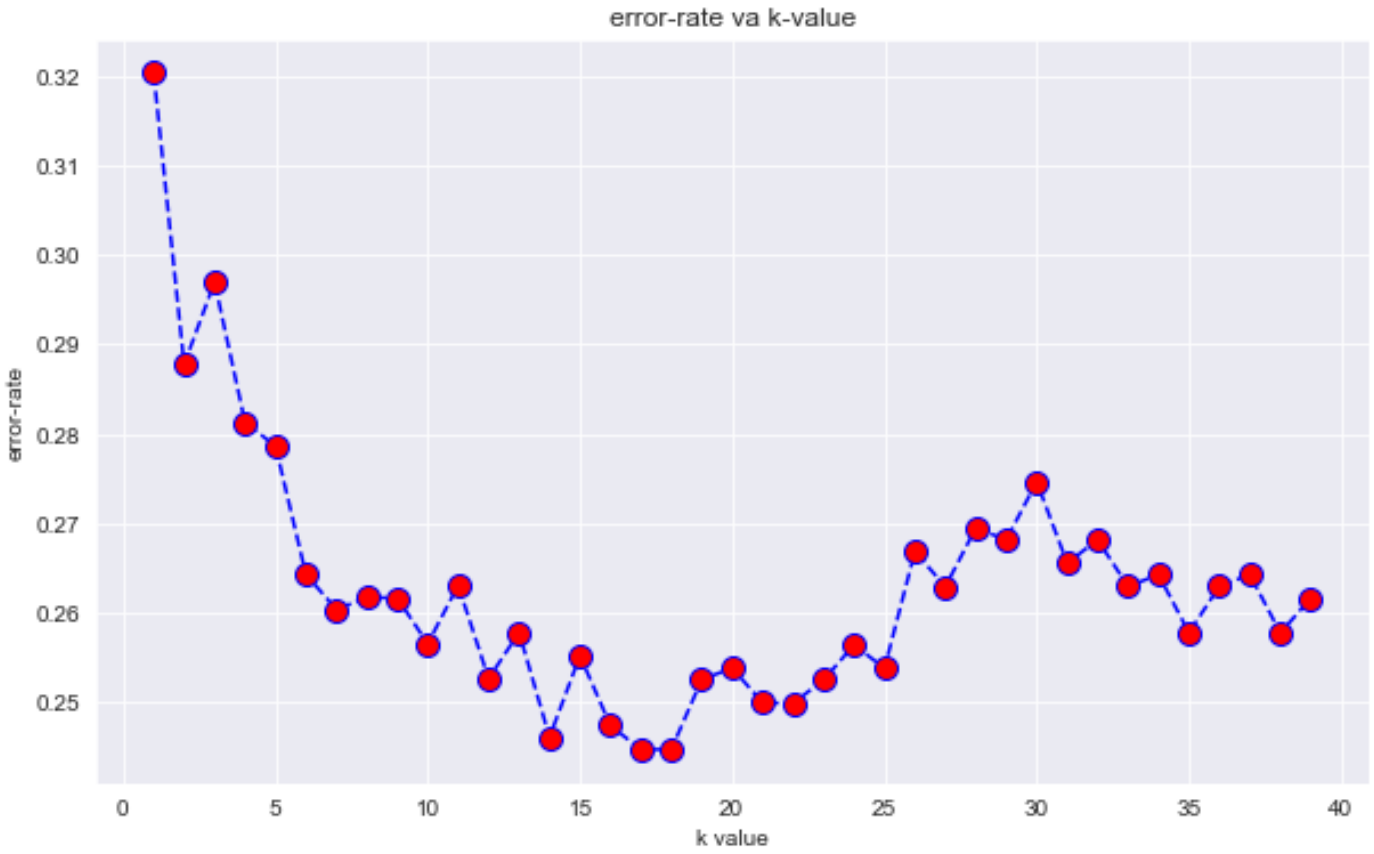
```
At K=1

[[95 39]
 [64 33]]
```

	precision	recall	f1-score	support
0	0.60	0.71	0.65	134
1	0.46	0.34	0.39	97
accuracy			0.55	231
macro avg	0.53	0.52	0.52	231
weighted avg	0.54	0.55	0.54	231

```
At K=18

[[148  64]
 [ 11   8]]

              precision    recall  f1-score   support

           0       0.93      0.70      0.80       212
           1       0.11      0.42      0.18        19

    accuracy                           0.68       231
   macro avg       0.52      0.56      0.49       231
weighted avg       0.86      0.68      0.75       231
```




Pregnancies
Glucose
BloodPressure
SkinThickness
Insulin
BMI
DiabetesPedigreeFunction
Age
Outcome
0
1



Observations:

- ✓ The KNN is showing an accuracy score as 70% at k=9
- ✓ After Principal Component Analysis, the score level has improved and reached to 74% at k=9
- ✓ F1 score at k=9 before PCA is 0.58, after PCA it is found to be 0.75.
- ✓ After using the elbow method to find the optimal K value, we found that at k=18 without PCA the F1 score value is found to be is 0.68.
- ✓ After PCA the F1 score has improved to 0.77 at k=18.

Conclusion:

The principal component analysis is very useful in improving the accuracy of the model by transforming the features into useful features, by reducing the noise and irrelevancy amongst them.

The elbow method is very useful in finding the most optimal 'k' value which has improved the performance of the model.

When both of these techniques are used together, they are resulting in the highest level of the accuracy and F1 score for the prediction.

TASK 9 : Fit Linear SVM to the non linear datasets

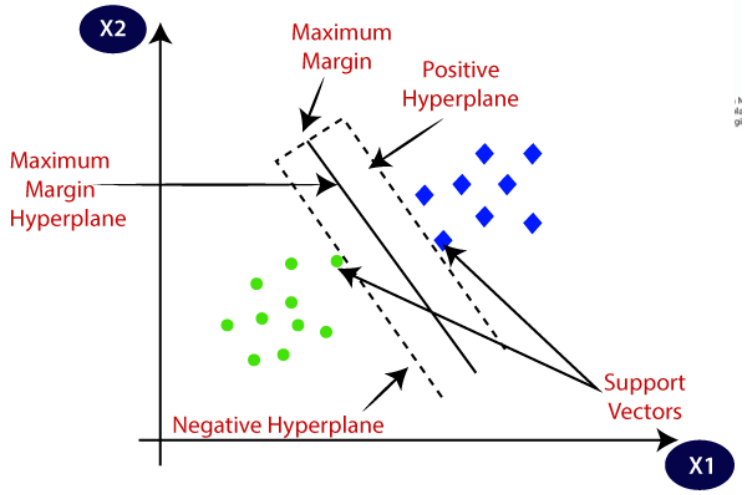
AIM : To fit linear Support Vector Machine on thre datasets : Breast Cancer,titanic dataset and Iris dataset.

Algorithm and Dataset Description :

- Breast Cancer Dataset is a famous dataset primarily used for prediction tasks.
- The target feature is diagnosis where we predict if the cancer is malignant or benign.
- It has 32 features but we selected only 2 features i.e. radius1 and texture 1
- It has no missing values and the features are numeric.

Support Vector Machine : Support Vector Machine is a powerful machine learning algorithm.

- It classifies the data based on finding the decision boundary with maximum margin.
- If the data is not linearly separable, SVM tries to map the data into higher dimensional feature space where the data can become linearly separable, called as kernel trick. Hence, that type of SVM is called kernel SVM.
- The distance parameter can be Euclidean distance.
- SVM has multiple hyperparameters involved in it like C, gamma.
- The distance metric considered and the kernel function used are also important hyperparameters.

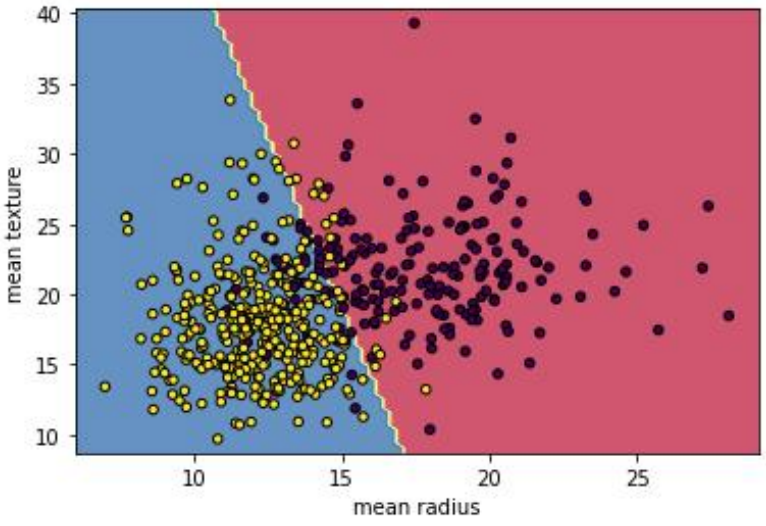


Source Code:

```
# Load the important packages
from sklearn.datasets import load_breast_cancer
import matplotlib.pyplot as plt
from sklearn.inspection import DecisionBoundaryDisplay
from sklearn.svm import SVC

# Load the datasets
cancer = load_breast_cancer()
X = cancer.data[:, :2]
y = cancer.target
print(X)
#Build the model
svm = SVC(kernel="linear", gamma=0.5, C=1.0)
# Trained the model
svm.fit(X, y)

# Plot Decision Boundary
DecisionBoundaryDisplay.from_estimator(
        svm,
        X,
        response_method="predict",
        cmap=plt.cm.Spectral,
        alpha=0.8,
        xlabel=cancer.feature_names[0],
        ylabel=cancer.feature_names[1],
    )

# Scatter plot
plt.scatter(X[:, 0], X[:, 1],
            c=y,
            s=20, edgecolors="k")
plt.show()
```

Output:

```
In [9]: runfile('C:/Use
Desktop/ml')
[[17.99 10.38]
 [20.57 17.77]
 [19.69 21.25]
 ...
 [16.6  28.08]
 [20.6  29.33]
 [ 7.76 24.54]]

In [10]:
```



Titanic Dataset:

- ❖ Titanic Dataset is mainly used for classification tasks.
- ❖ It has 11 features and the target class is Survived or not. Hence it is binary classification.
- ❖ There are missing values and they are handled using fillna() method.
- ❖ There are some unnecessary features in the dataset like the address, ticket type and many more. They were removed and only Pclass, Sex, Age, Fare, Embarked are selected as feature set.

Source Code:

```
#importing the libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings as wr
wr.filterwarnings('ignore')

#importing the dataset
df=pd.read_csv('C:/Users/mvsla/OneDrive/Desktop/ml/titanic_train.csv')
print(df.head())

#Exploratory data analysis
print(df.info(),'\n')
print(df.shape,'\n')
print(df.dtypes=='object','\n')
print(df.describe(),'\n')
print(df.isnull().sum())

print(df['Embarked'].value_counts())
embarked_mode=df['Embarked'].mode()
print(embarked_mode)
```

```
df['Embarked'].replace(str(embarked_mode),inplace=True)
print(df.isnull().sum())
df['Age'].fillna(df['Age'].mean(),inplace=True)

df['Cabin'].value_counts()
df['Cabin'].fillna(df['Cabin'].mode,inplace=True)

#seaborn plots
sns.set_style('darkgrid')
numerical_columns=df.select_dtypes(include=['int64','float64']).columns
plt.figure(figsize=(14,len(numerical_columns)*3))
for idx, feature in enumerate(numerical_columns,1):
    plt.subplot(len(numerical_columns),2,idx)
    sns.histplot(df[feature],kde=True)
    plt.title(f"{feature}| skewness:{round(df[feature].skew(),2)}")

plt.tight_layout()
plt.show()
```

```
sns.pairplot(df,hue='Survived')
plt.show()

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['Embarked'] = le.fit_transform(df['Embarked'])
df['Sex'] = le.fit_transform(df['Sex'])
df = df[df['Sex'] != 0]
df=df[df['Embarked']!=0]
df = df.reset_index(drop=True)

x=df.drop(['PassengerId','Ticket','SibSp','Parch','Name','Cabin','Survived'],axis=1)
y=df['Survived']
print(x.head())
```

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_scaled=sc.fit_transform(x)

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x_scaled,y,test_size=0.30,random_state=20)

#fitting an svm
from sklearn.svm import SVC
svm = SVC(kernel="linear", gamma=0.5, C=1.0)
svm.fit(x_train,y_train)

y_pred=svm.predict(x_test)

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

report=classification_report(y_test,y_pred)
print(report)
```

```
#confusion matrix visualization
plt.figure(figsize=(10,8))
ax=sns.heatmap(cm,annot=True,cmap='Blues')
ax.set_title('Confusion Matrix with labels \n\n')
ax.set_xlabel('\nPredicted labels')
ax.set_ylabel('\nActual values')
plt.show()
```

Outputs:

```
In [10]: runfile('C:/Users/mvsla/OneDrive/Desktop/ml/svm2.py', wdir='C:/Users/mvsl
Desktop/ml')
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name     Sex   Age  SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
2                             Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                           Allen, Mr. William Henry    male  35.0      0

   Parch            Ticket     Fare Cabin Embarked
```

```
   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
```

```
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
None

(891, 12)
```

]

```
PassengerId    False
Survived       False
Pclass         False
Name            True
Sex             True
Age            False
SibSp          False
Parch          False
Ticket          True
Fare           False
Cabin           True
Embarked        True
dtype: bool
```

```
             Parch        Fare
count  891.000000  891.000000
mean     0.381594   32.204208
std      0.806057   49.693429
min      0.000000    0.000000
25%      0.000000    7.910400
50%      0.000000   14.454200
75%      0.000000   31.000000
max      6.000000  512.329200
```

```
       PassengerId    Survived      Pclass         Age       SibSp  \
count   891.000000  891.000000  891.000000  714.000000  891.000000
mean    446.000000    0.383838    2.308642   29.699118    0.523008
std     257.353842    0.486592    0.836071   14.526497    1.102743
min       1.000000    0.000000    1.000000    0.420000    0.000000
25%     223.500000    0.000000    2.000000   20.125000    0.000000
50%     446.000000    0.000000    3.000000   28.000000    0.000000
75%     668.500000    1.000000    3.000000   38.000000    1.000000
max     891.000000    1.000000    3.000000   80.000000    8.000000
```

```
    Pclass  Sex        Age     Fare  Embarked
0       3    1  22.000000   7.2500         2
1       3    1  35.000000   8.0500         2
2       3    1  29.699118   8.4583         1
3       1    1  54.000000  51.8625         2
4       3    1   2.000000  21.0750         2
[[122  23]
 [  0   0]]
              precision    recall  f1-score   support

           0       0.84      1.00      0.91       122
           1       0.00      0.00      0.00        23

    accuracy                           0.84       145
   macro avg       0.42      0.50      0.46       145
weighted avg       0.71      0.84      0.77       145
```

Source Code for Iris Dataset:

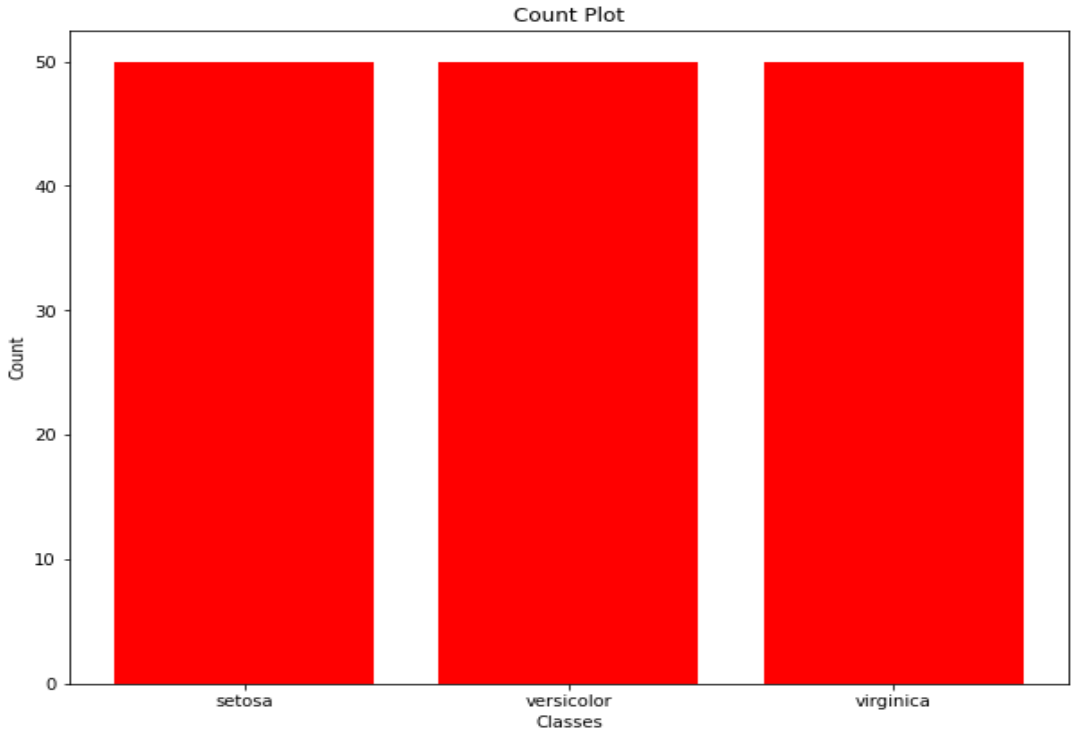
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings as wr
wr.filterwarnings('ignore')

df=pd.read_csv('iris.csv')
print(df.head())

print(df.info(),'\n')
print(df.shape,'\n')
print(df.dtypes=='object','\n')
print(df.describe(),'\n')
print(df.isnull().sum())

count_class=df['species'].value_counts()
plt.figure(figsize=(10,8))
plt.bar(count_class.index,count_class,color='red')
plt.xlabel('Classes')
plt.ylabel('Count')
plt.title('Count Plot')
plt.show()

sns.set_style('darkgrid')
numerical_columns=df.select_dtypes(include=['int64','float64']).columns
plt.figure(figsize=(14,len(numerical_columns)*3))
for idx, feature in enumerate(numerical_columns,1):
    plt.subplot(len(numerical_columns),2,idx)
    sns.histplot(df[feature],kde=True)
    plt.title(f"{feature}| skewness:{round(df[feature].skew(),2)}")
plt.tight_layout()
plt.show()
```

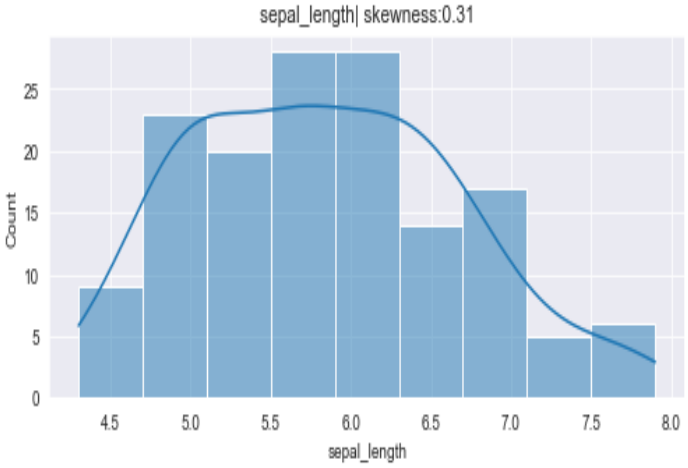
```
sns.pairplot(data=df,hue='species')
plt.show()

x=df.iloc[:,:-1].values
y=df.iloc[:,-1].values

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=10)

from sklearn.svm import SVC
svm=SVC(C=1.0,kernel='linear',gamma=0.5)
svm.fit(x_train,y_train)

y_pred=svm.predict(x_test)

from sklearn.metrics import confusion_matrix,classification_report
cm=confusion_matrix(y_test,y_pred)
print(cm)
rep=classification_report(y_test,y_pred)
print(rep)

x1=df.iloc[:,:2].values
y1=df.iloc[:,-1].values

from sklearn.svm import SVC
svm1=SVC(kernel='linear',C=1.0,gamma=0.5)
svm1.fit(x1,y1)

# Plot Decision Boundary
# Scatter plot
```

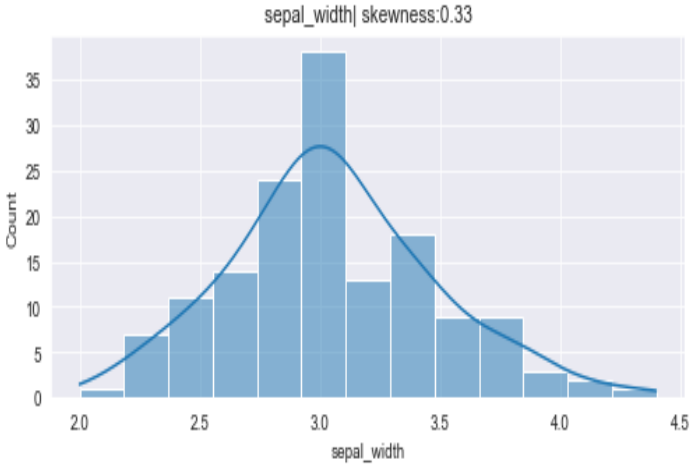
```
# Plot Decision Boundary
# Scatter plot

from sklearn.inspection import DecisionBoundaryDisplay
DecisionBoundaryDisplay.from_estimator(
        svm1,
        x1,
        response_method="predict",
        cmap=plt.cm.Spectral,
        alpha=0.8,
        xlabel=df.columns[0],
        ylabel=df.columns[1],
        )
plt.scatter(x[:,0],x[:,1],edgecolors='k')
plt.show()
```

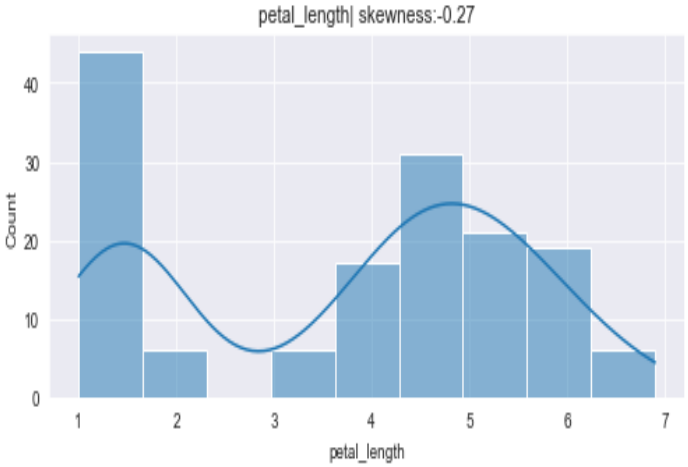
Outputs:

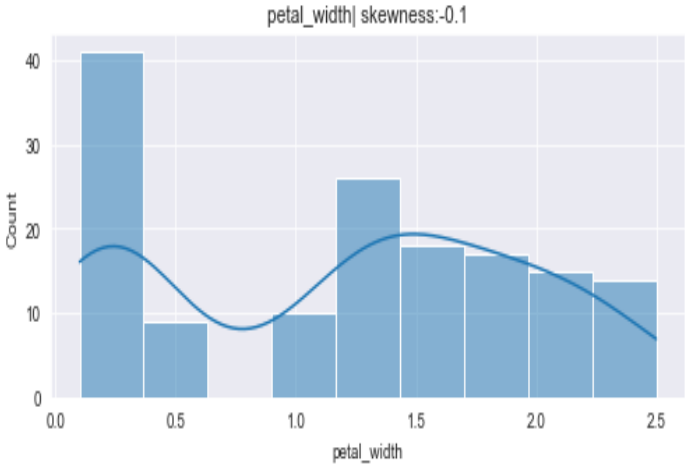
```
In [1]: runfile('C:/Users/MVSTU/OneDrive/Desktop/ml/svm3.py', wdir='C:/Users/M
Desktop/ml')
   sepal_length  sepal_width  petal_length  petal_width species
0           5.1          3.5           1.4          0.2  setosa
1           4.9          3.0           1.4          0.2  setosa
2           4.7          3.2           1.3          0.2  setosa
3           4.6          3.1           1.5          0.2  setosa
4           5.0          3.6           1.4          0.2  setosa
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
None

(150, 5)

sepal_length    False
sepal_width     False
petal_length    False
petal_width     False
species          True
dtype: bool

       sepal_length  sepal_width  petal_length  petal_width
count    150.000000   150.000000    150.000000   150.000000
mean       5.843333     3.054000      3.758667     1.198667
std        0.828066     0.433594      1.764420     0.763161
min        4.300000     2.000000      1.000000     0.100000
25%        5.100000     2.800000      1.600000     0.300000
50%        5.800000     3.000000      4.350000     1.300000
75%        6.400000     3.300000      5.100000     1.800000
max        7.900000     4.400000      6.900000     2.500000

sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64

[[14  0  0]
 [ 0 17  0]
 [ 0  0 14]]
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        14
  versicolor       1.00      1.00      1.00        17
   virginica       1.00      1.00      1.00        14

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

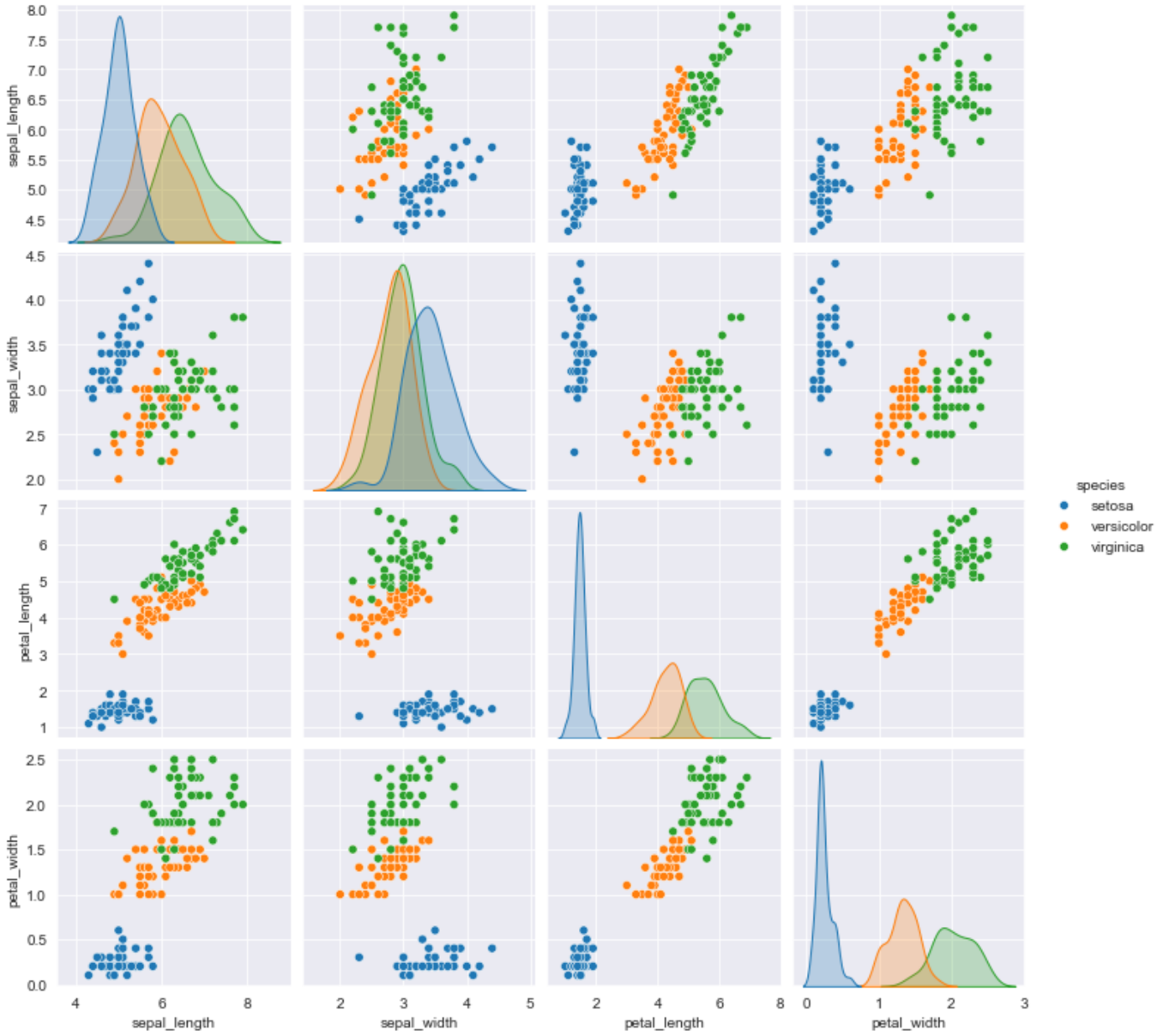

Count Plot
Count
Classes
setosa
versicolor
virginica

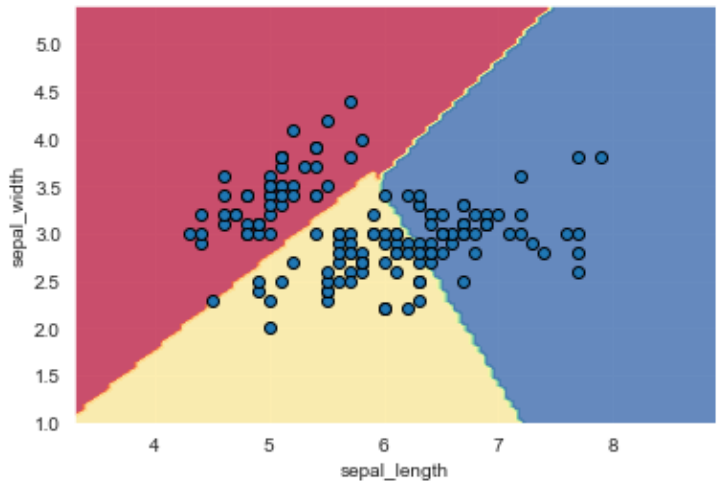

sepal_length| skewness:0.31
Count
sepal_length


sepal_width| skewness:0.33
Count
sepal_width


petal_length| skewness:-0.27
Count
petal_length


petal_width| skewness:-0.1
Count
petal_width


sepal_length
sepal_width
petal_length
petal_width
species
setosa
versicolor
virginica


sepal_width
sepal_length

Observations:

- ✓ SVM tries to separate the data in the linear feature space and the accuracy scores are calculated.
- ✓ The accuracy score for titanic dataset is calculated to be 84%.

Conclusion:

SVM results are a decent level of accuracy. The model performance can be improved using cross validation and going for different support vector kernels.

TASK 10: PERFORM DATA VISUALIZATION USING DIFFERENT LIBRARIES

AIM: Performing Data Visualization using different libraries

Dataset Description:

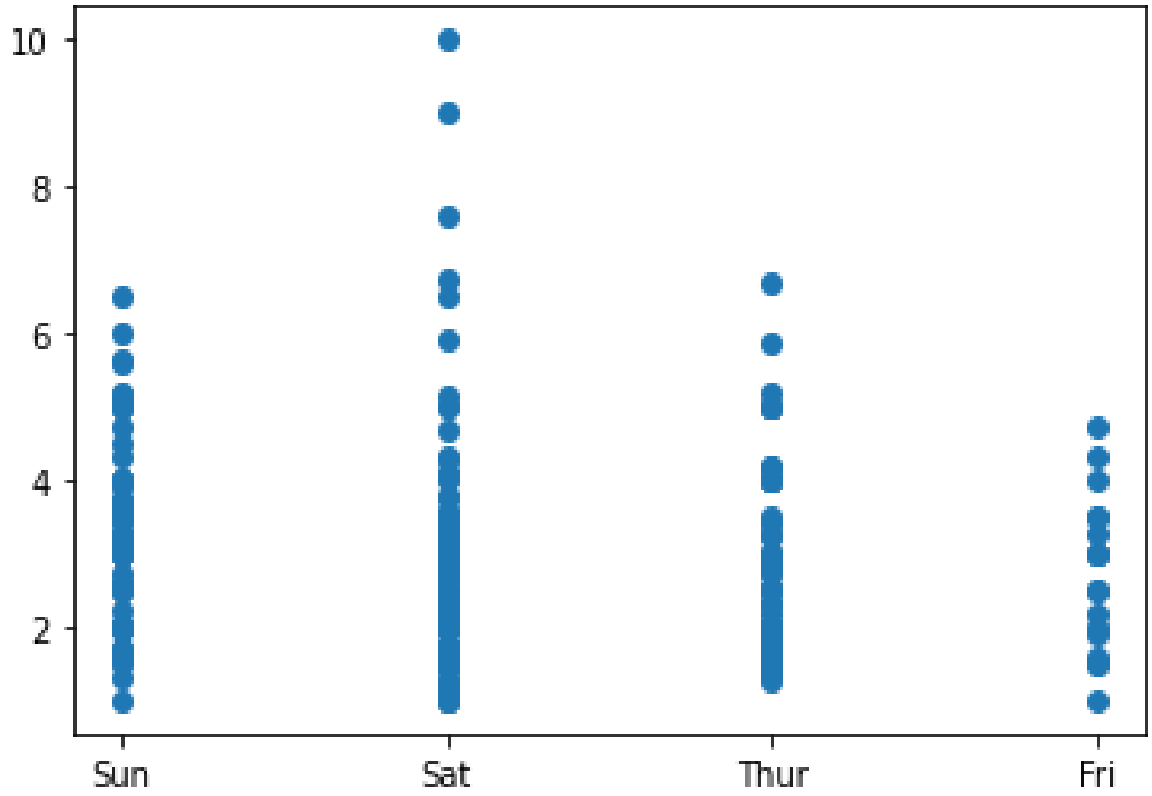
- ❖ The tips dataset is used for performing data visualizations/
- ❖ It is a dataset of dimensions 244 rows x 7 columns and has no missing values.

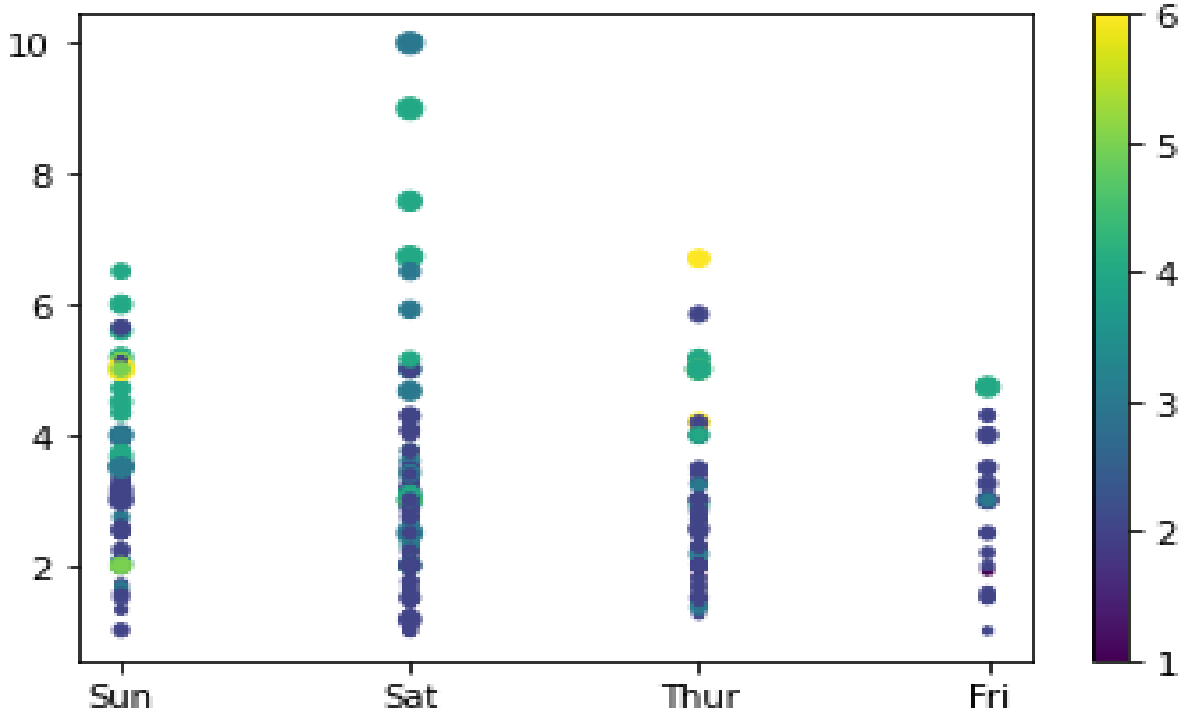
Source Code:

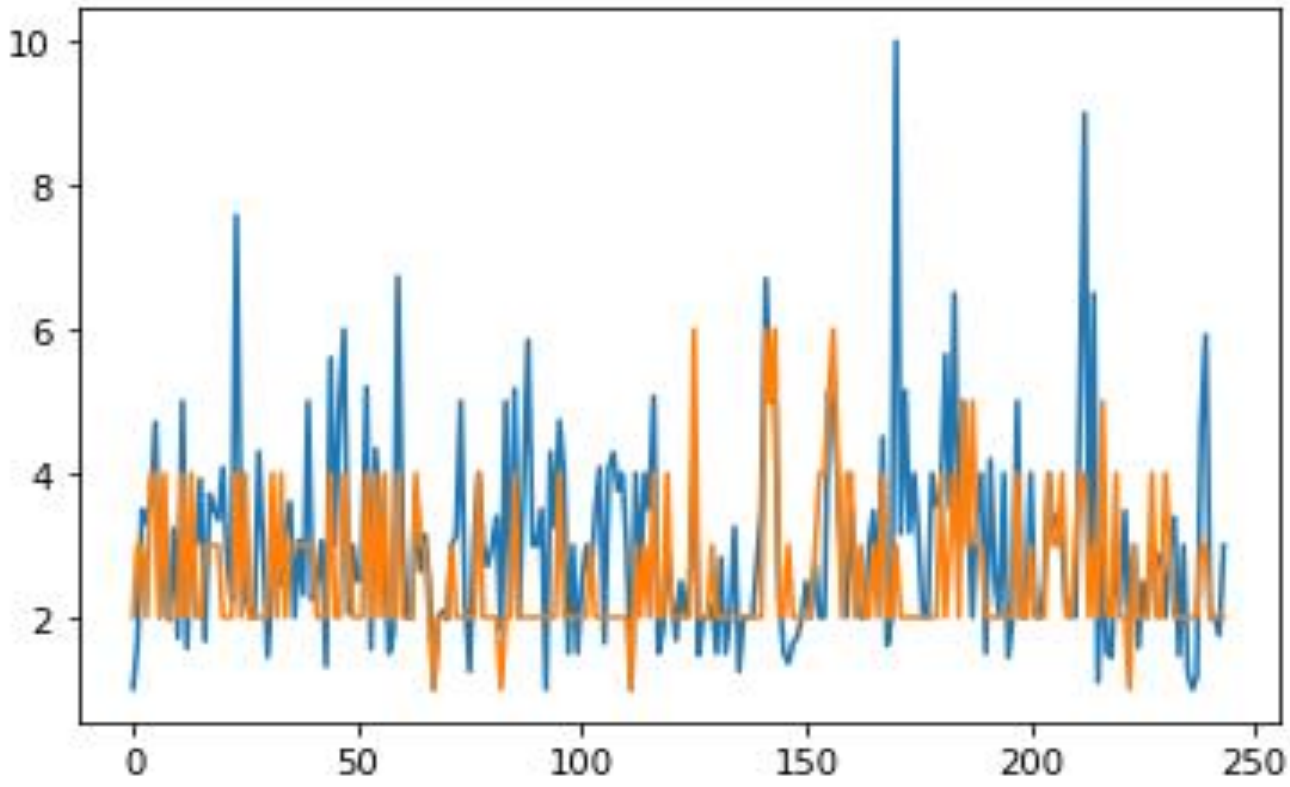
```
#data visualizations
import pandas as pd
data=pd.read_csv('tips.csv')
#USING MATPLOTLIB
import matplotlib.pyplot as plt
plt.scatter(data['day'],data['tip'])
plt.show()
#with colour bar
plt.scatter(data['day'],data['tip'],c=data['size'],s=data['total_bill'])
plt.colorbar()
plt.show()
#linechart
plt.plot(data['tip'])
plt.plot(data['size'])
plt.show()
#barplot
plt.bar(data['day'],data['tip'])
plt.show()
#histogram
plt.hist(data['total_bill'])
plt.show()
```

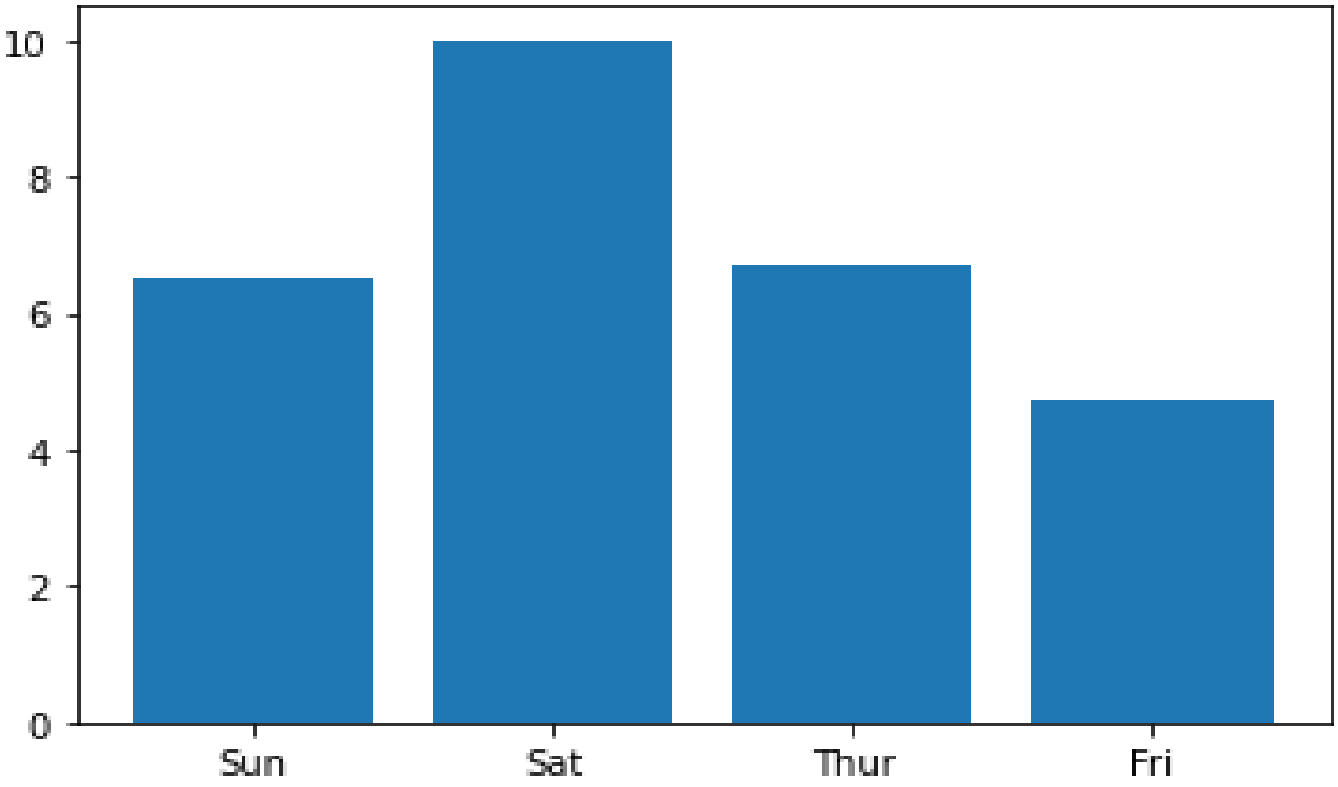
```
#USING SEABORN
import seaborn as sns
sns.lineplot(x='sex',y='total_bill',data=data)
plt.show()

sns.scatterplot(x='day',y='tip',data=data)
plt.show()
#using bokeh
from bokeh.plotting import figure,show
from bokeh.palettes import magma
graph=figure(title='Bokeh Scatterplot')
color=magma(256)
graph.scatter(data['total_bill'],data['tip'],color=color)
show(graph)

df=data['tip'].value_counts()
graph.line(df,data['tip'])
show(graph)

graph=figure(title='Bokeh Bar Chart')
graph.vbar(data['total_bill'],top=data['tip'])
show(graph)
```

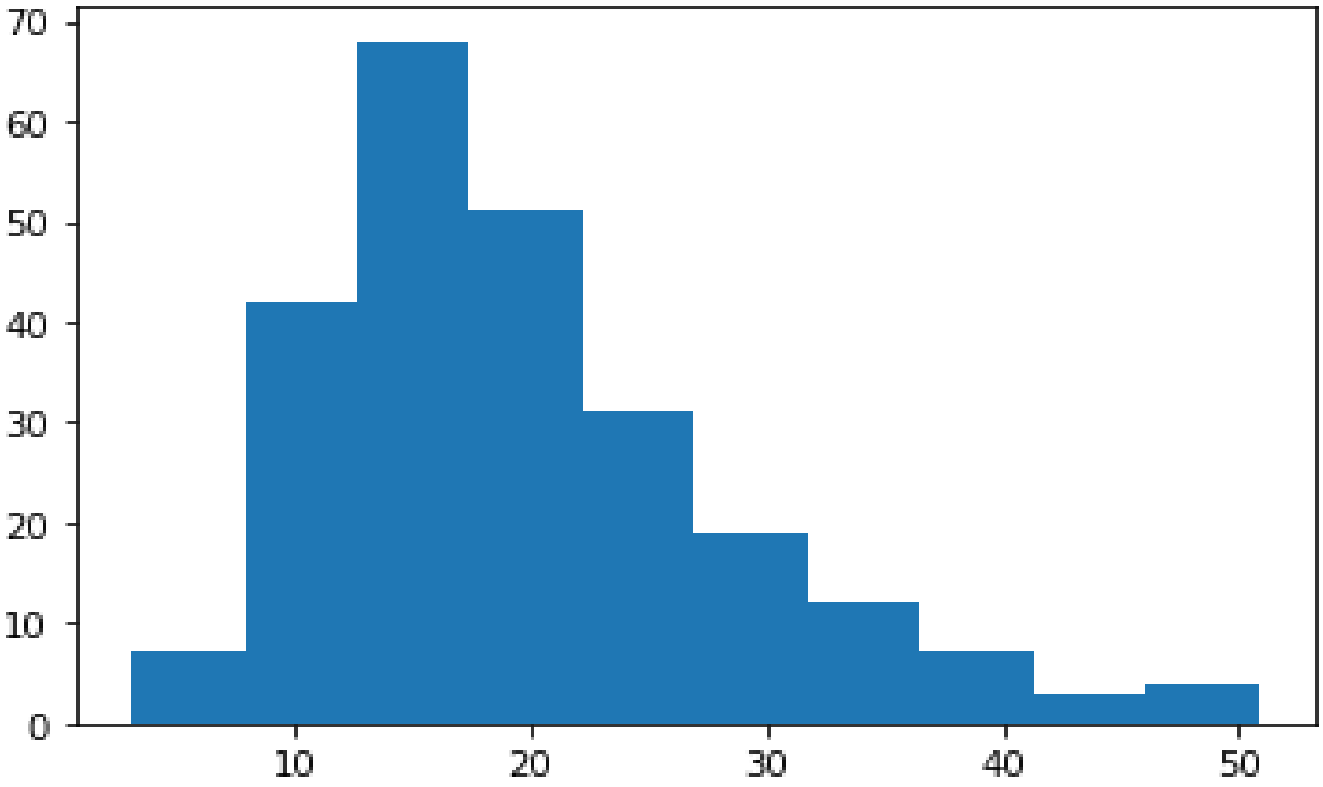
Outputs:

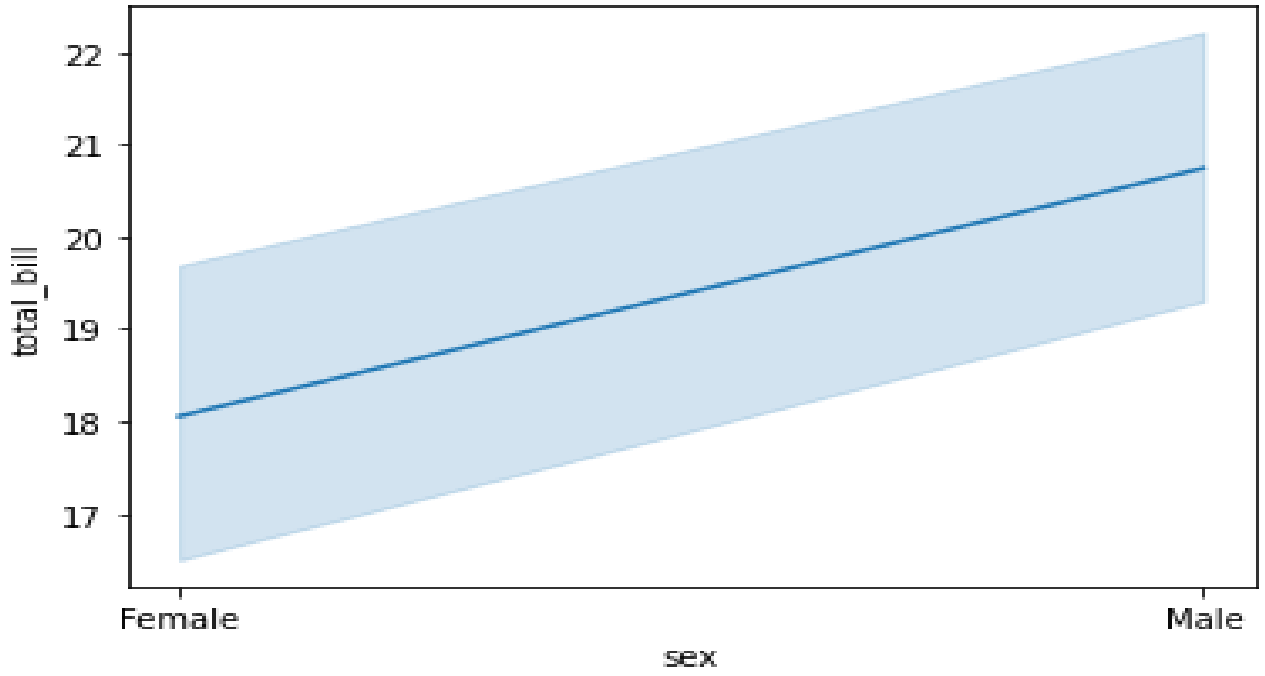


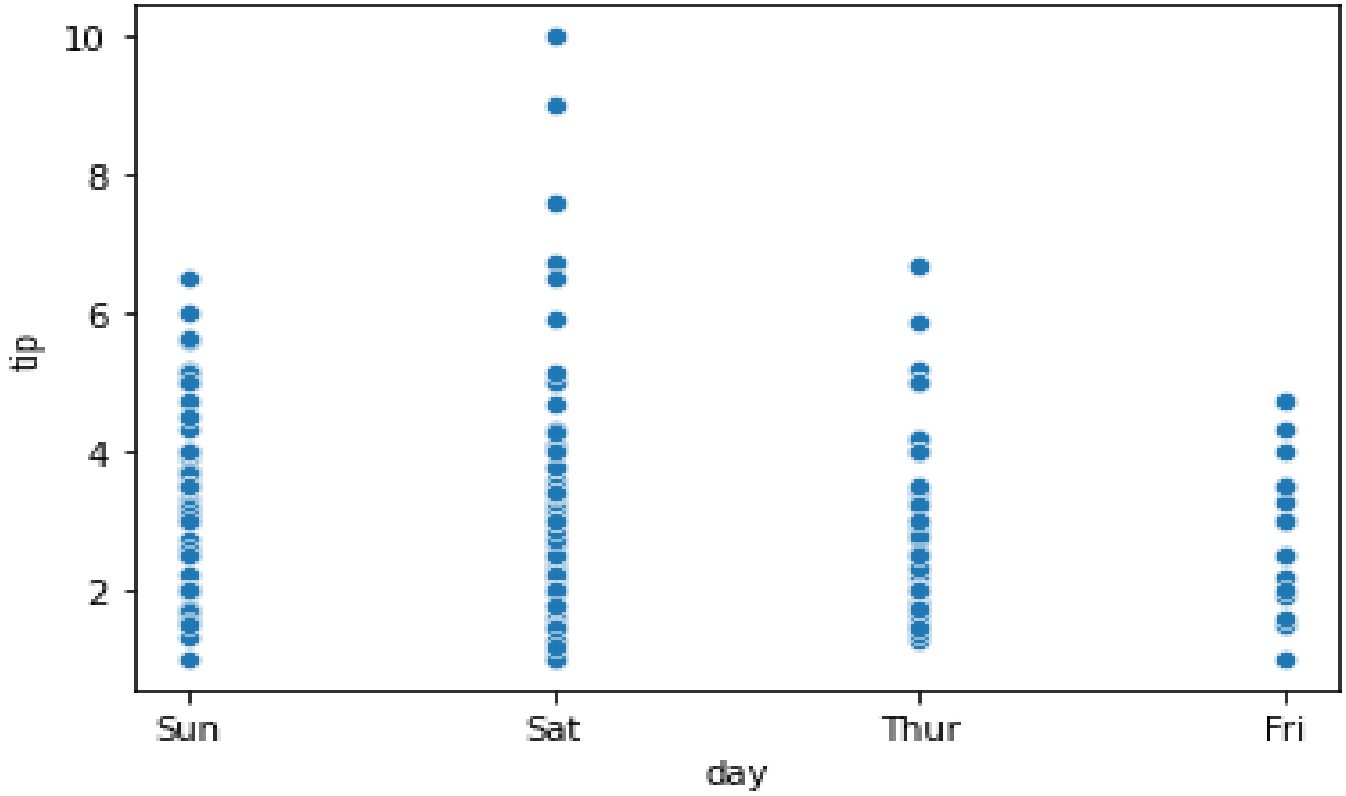

10
8
6
4
2
Sun
Sat
Thur
Fri
6
5
4
3
2
1

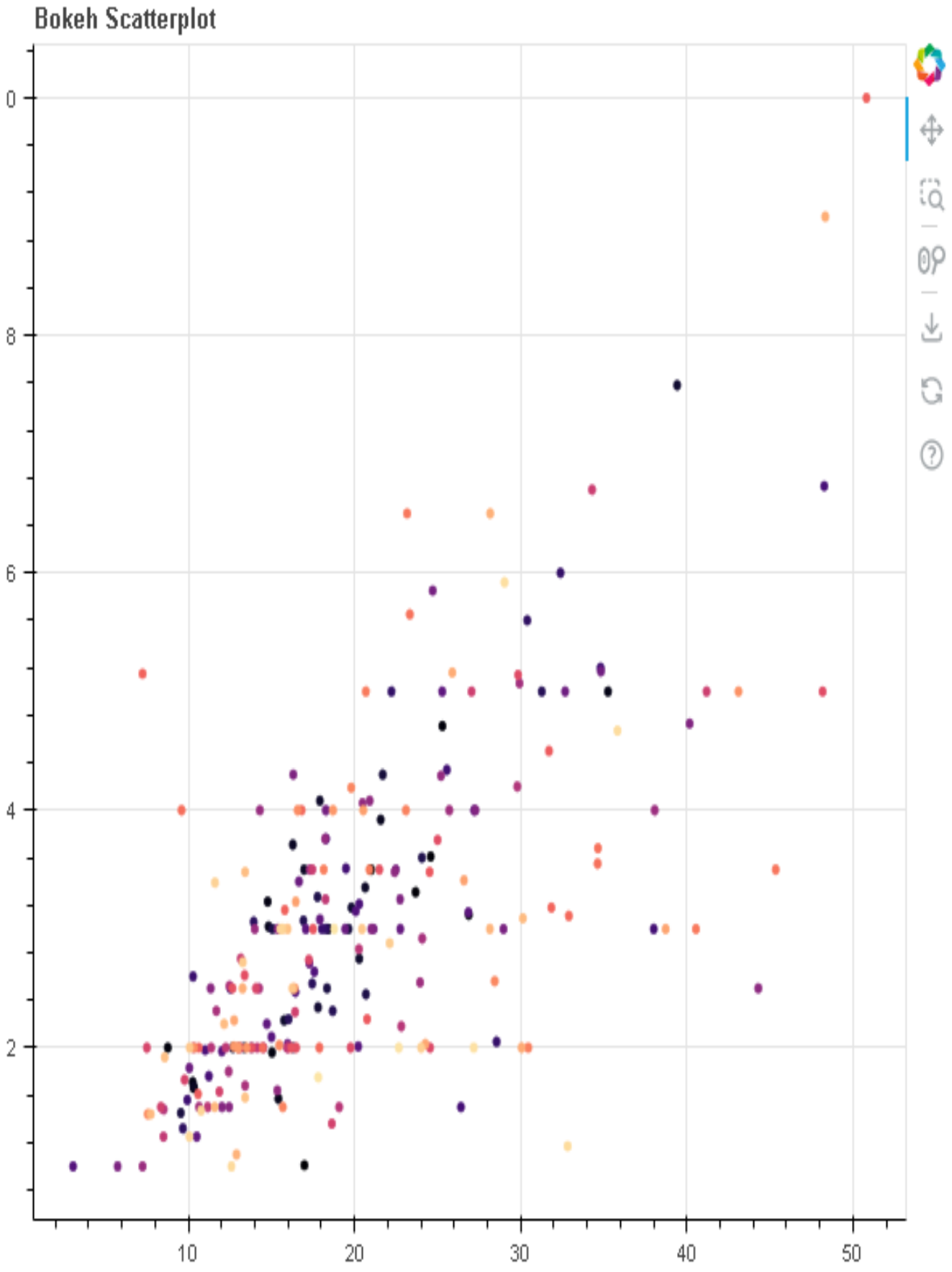

10
8
6
4
2
0
50
100
150
200
250

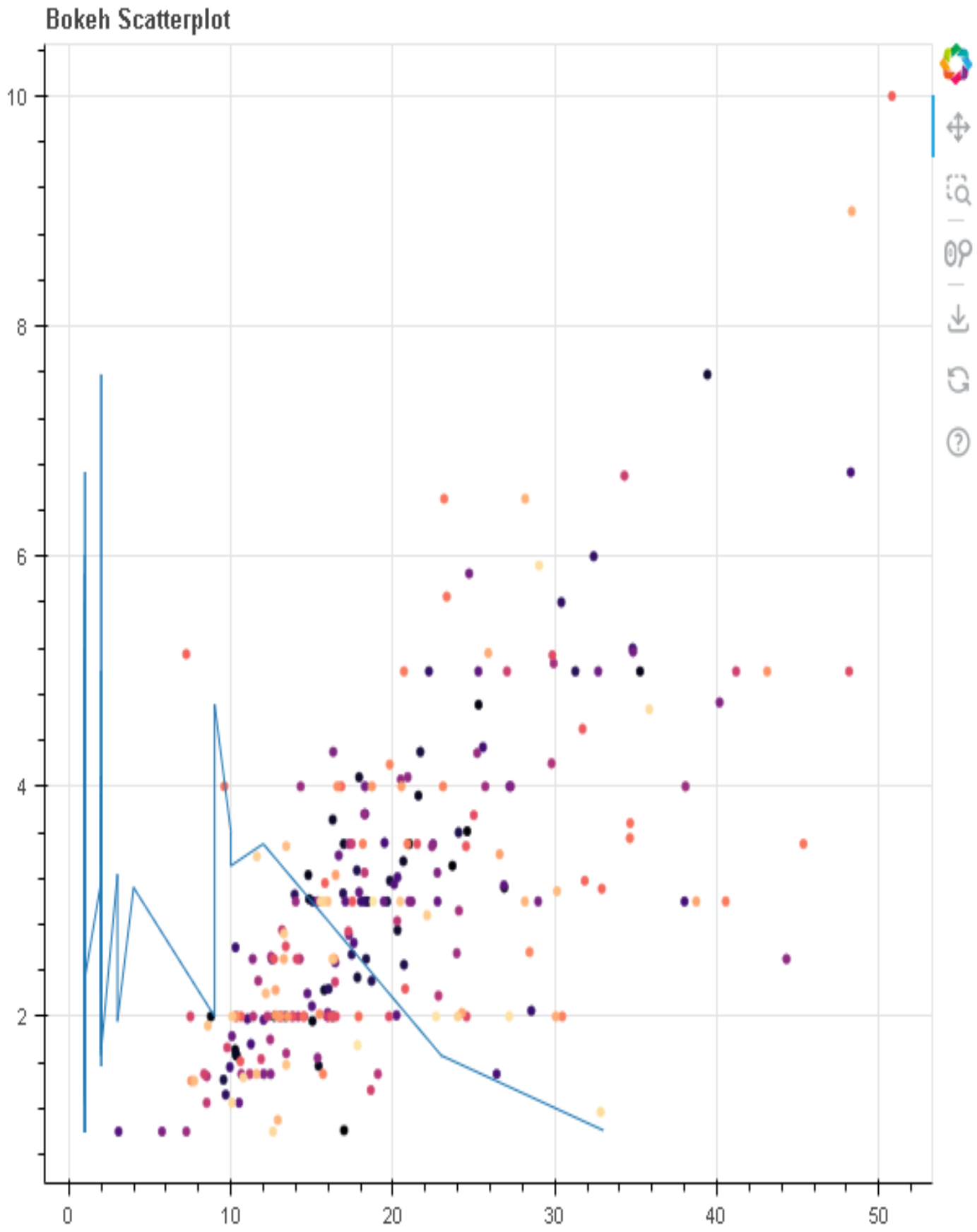

10
8
6
4
2
0
Sun
Sat
Thur
Fri

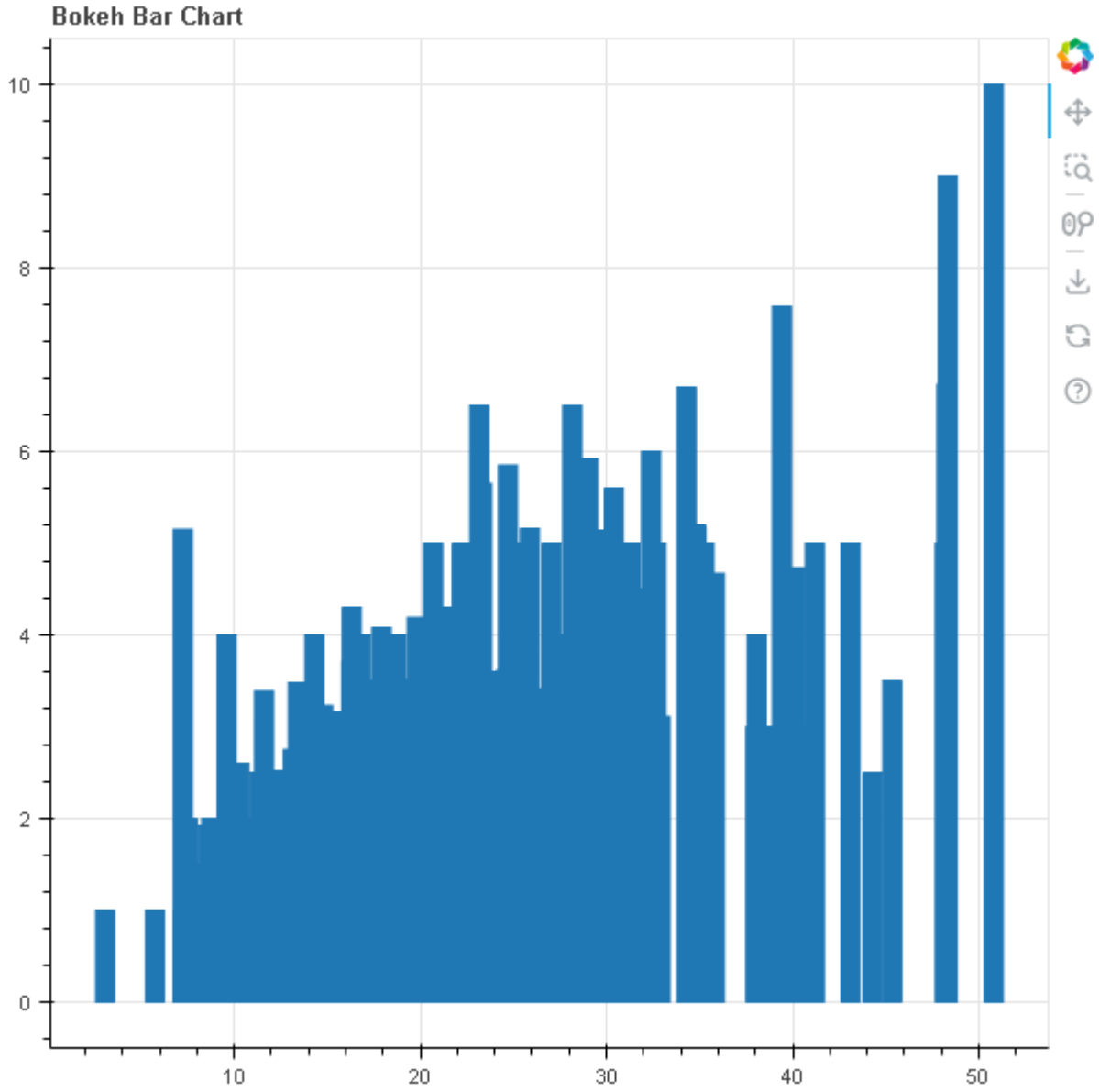

70
60
50
40
30
20
10
0
10
20
30
40
50


22
21
20
19
18
17
total_bill
Female
Male
sex


10
8
6
4
2
tip
Sun
Sat
Thur
Fri
day


Bokeh Scatterplot
0
8
6
4
2
10
20
30
40
50


Bokeh Scatterplot
10
8
6
4
2
0
10
20
30
40
50



Observations:

- ✓ Matplotlib is a powerful library where we can construct many plots. However, matplotlib is having many parameters to pass.
- ✓ Seaborn is another powerful library for visualization which can construct many plots and is easier to use
- ✓ Bokeh is a great data visualization library which can create some advanced visualizations and can generate the visualizations in web format.

Conclusion:

The type of library is to be chosen as per the requirement and as per the plot.

TASK 11: Fit a Classification model i.e. Decision Tree.

AIM : To fit a decision tree model the two datasets on balance scale and distance database and daily weather prediction.

Algorithm and Dataset Description:

- **Decision Tree:** Decision Tree is a supervised machine learning algorithm mainly used for classification tasks.
- The goal of Decision Tree is to predict the best possible decision from the past data.
- It is in tree like form where the decisions are represented by nodes and the edges contain the outcomes of the decision and the leaf nodes contain the final decision.

Dataset Description:

- The balance scale and distance database is present in UCI machine learning repository which is about psychological data.
- It is primarily used for classification tasks and the data set does not contain any missing values.
- The daily weather dataset is about the weather recorded on San Diego, California over a period of three years.
- The dataset has the dimensions as 1095 rows x 11 columns.
- The target feature is humidity level.

Source Code:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings as wr
wr.filterwarnings('ignore')
df=pd.read_csv(
        'https://archive.ics.uci.edu/ml/machine-learning-' +
        'databases/balance-scale/balance-scale.data',
        sep=',', header=None)
print(df.head())
print(df.shape)
print(df.info())
print(df.isnull().sum())
print(df.describe())

sns.set_palette('bright')
plots=plt.figure(figsize=(10,6))

sns.set_style('darkgrid')
num_col=df.select_dtypes(include=['int64','float64']).columns
plt.figure(figsize=(14,len(num_col)*3))
for idx, feature in enumerate(num_col,1):
    plt.subplot(len(num_col),2,idx)
    sns.histplot(df[feature],kde=True)
    plt.title(f"{feature}| skewness:{round(df[feature].skew(),2)}")

plt.tight_layout()
plt.show()
```

```
sns.pairplot(df,hue=0)
plt.show()

x=df.iloc[:,1:5].values
y=df.iloc[:,0]

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=20)


from sklearn.tree import DecisionTreeClassifier
gini=DecisionTreeClassifier(criterion='gini',max_depth=3,
                            random_state=100,min_samples_leaf=5)
gini.fit(x_train,y_train)
y_pred_gini=gini.predict(x_test)

entropy=DecisionTreeClassifier(criterion='entropy',random_state=100,
                               max_depth=3,min_samples_leaf=5)
entropy.fit(x_train,y_train)
y_pred_entropy=entropy.predict(x_test)

from sklearn.metrics import confusion_matrix,accuracy_score,classification_report

cm1=confusion_matrix(y_test,y_pred_gini)
acc1=accuracy_score(y_test,y_pred_gini)*100
rep1=classification_report(y_test,y_pred_gini)

cm2=confusion_matrix(y_test,y_pred_entropy)
acc2=accuracy_score(y_test,y_pred_entropy)*100
rep2=classification_report(y_test,y_pred_entropy)
```

```
print('The confusion matrix in case of using decision criteria as GINI INDEX is :\n',cm1)
print('The accuracy in case of using decision criteria as GINI INDEX is :\n',acc1)
print('The classification report in case of using decision criteria as GINI INDEX is :\n',rep1)

print('The confusion matrix in case of using decision criteria as ENTROPY is : \n',cm2)
print('The accuracy in case of using decision criteria as ENTROPY is : \n',acc2)
print('The classification report in case of using decision criteria as ENTROPY is : \n',rep2)
```

```
# Function to plot the decision tree
from sklearn import tree
def plot_decision_tree(clf_object, feature_names, class_names):
    plt.figure(figsize=(15, 10))
    plt.title('Decision Tree')
    tree.plot_tree(clf_object, filled=True, feature_names=feature_names, class_names=class_names, rounded=True)
    plt.show()

plot_decision_tree(gini,['X1','X2','X3','X4'],['B','L','R'])
plot_decision_tree(entropy,['X1','X2','X3','X4'],['B','L','R'])
```

Outputs:

```
In [1]: runfile('C:/Users/mvsl
Desktop/ml')
   0  1  2  3  4
0  B  1  1  1  1
1  R  1  1  1  2
2  R  1  1  1  3
3  R  1  1  1  4
4  R  1  1  1  5
(625, 5)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 625 entries, 0 to 624
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   0       625 non-null    object
 1   1       625 non-null    int64
 2   2       625 non-null    int64
 3   3       625 non-null    int64
 4   4       625 non-null    int64
dtypes: int64(4), object(1)
memory usage: 24.5+ KB
None
0    0
1    0
2    0
3    0
4    0
dtype: int64
```

	1	2	3	4
count	625.000000	625.000000	625.000000	625.000000
mean	3.000000	3.000000	3.000000	3.000000
std	1.415346	1.415346	1.415346	1.415346
min	1.000000	1.000000	1.000000	1.000000
25%	2.000000	2.000000	2.000000	2.000000
50%	3.000000	3.000000	3.000000	3.000000
75%	4.000000	4.000000	4.000000	4.000000
max	5.000000	5.000000	5.000000	5.000000

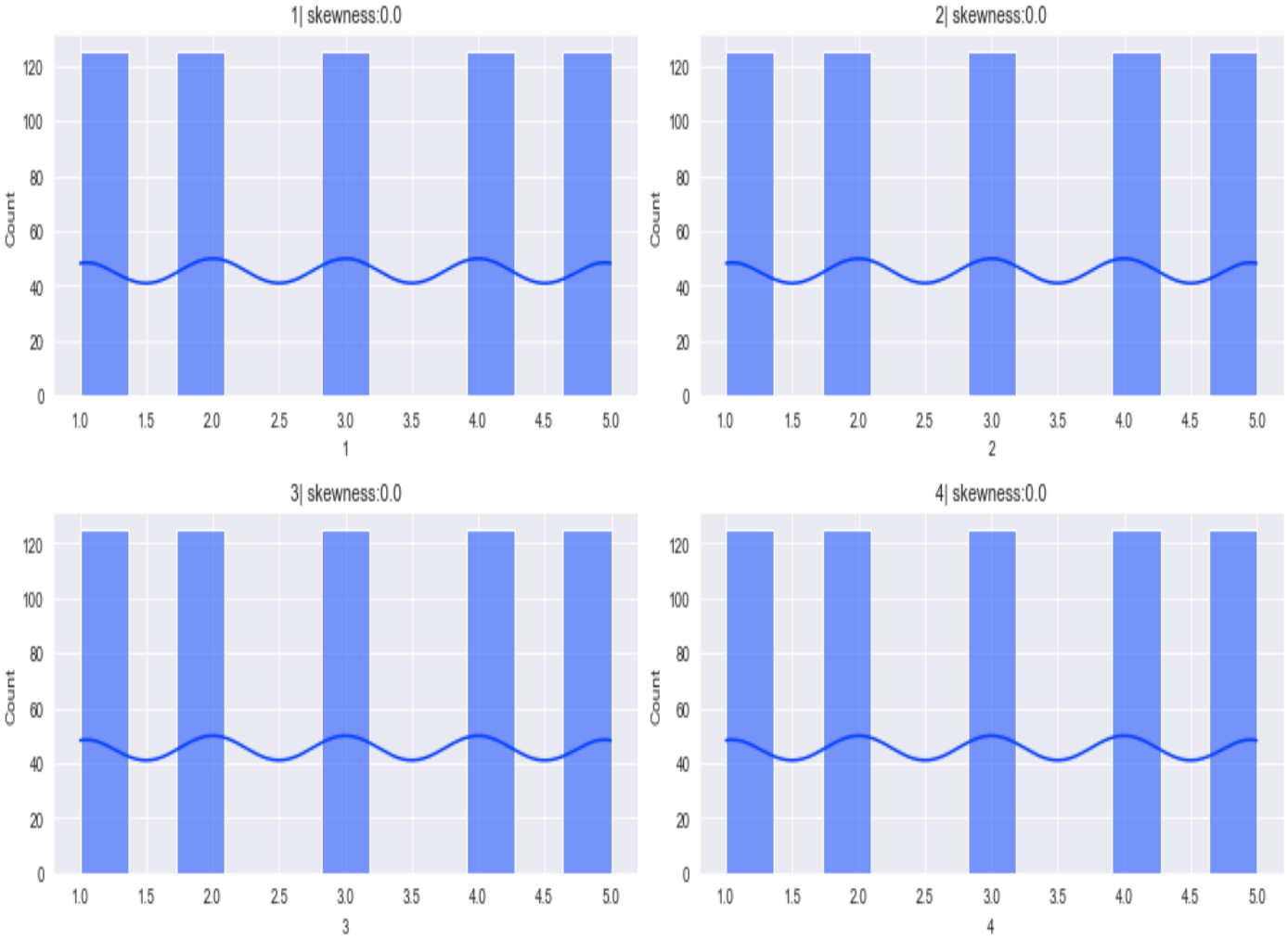
```
<Figure size 720x432 with 0 Axes>
```

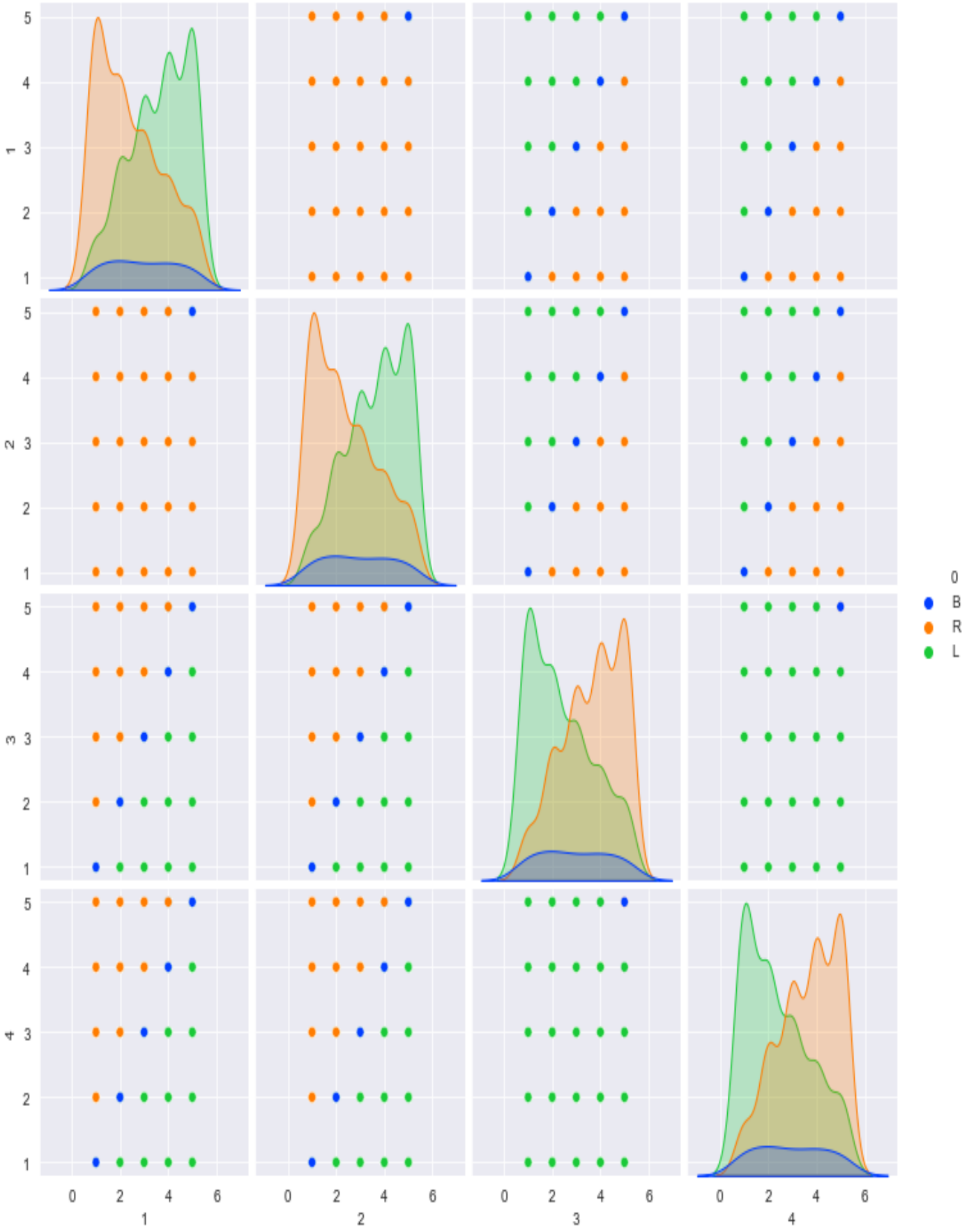
Important

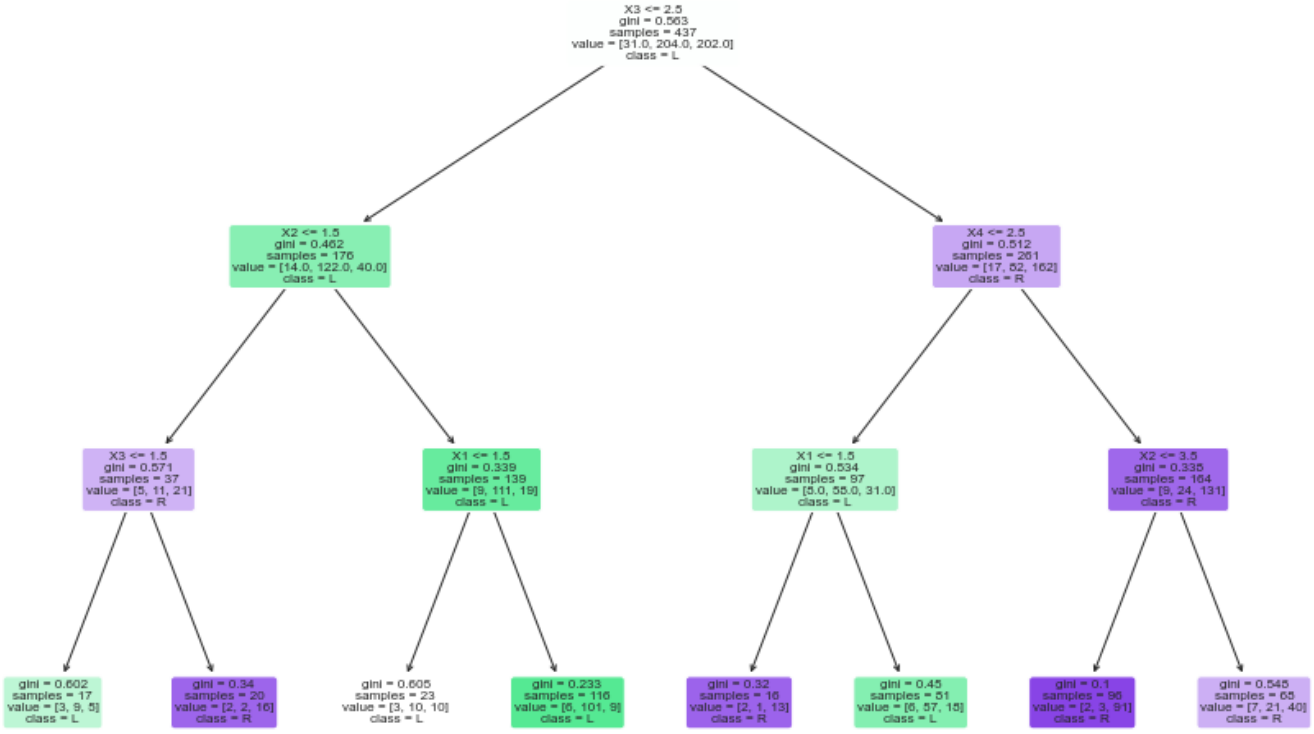
Figures are displayed in the Plots pane by default. To make them also appear inline in the console, you need to uncheck "Mute inline plotting" under the options menu of Plots.

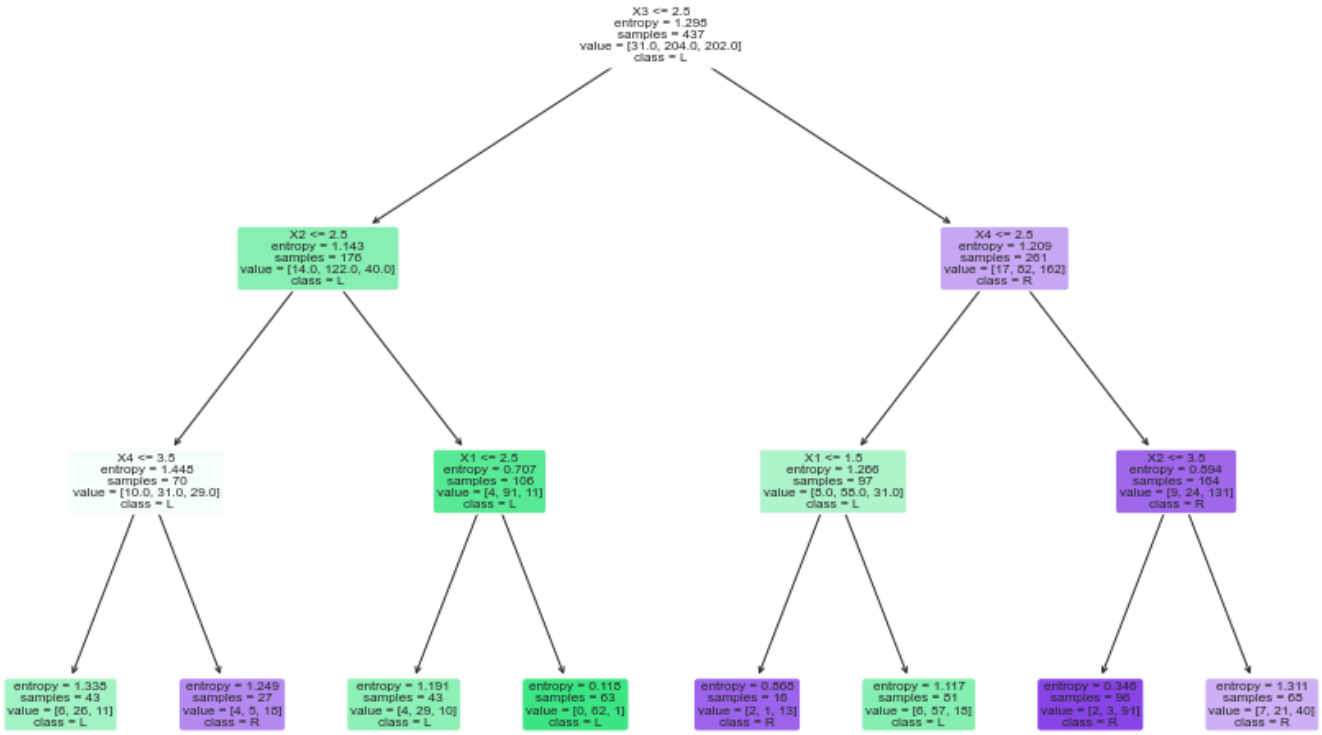
```
The confusion matrix in case of using decision criteria as GINI INDEX is :
[[ 0 11  7]
 [ 0 69 15]
 [ 0 28 58]]
The accuracy in case of using decision criteria as GINI INDEX is :
 67.5531914893617
The classification report in case of using decision criteria as GINI INDEX is :
              precision    recall  f1-score   support

           B       0.00      0.00      0.00        18
           L       0.64      0.82      0.72        84
           R       0.72      0.67      0.70        86

    accuracy                           0.68       188
   macro avg       0.45      0.50      0.47       188
weighted avg       0.62      0.68      0.64       188
```

```
The confusion matrix in case of using decision criteria as ENTROPY is :
 [[ 0 10  8]
 [ 0 68 16]
 [ 0 22 64]]
The accuracy in case of using decision criteria as ENTROPY is :
 70.2127659574468
The classification report in case of using decision criteria as ENTROPY is :
              precision    recall  f1-score   support

           B       0.00      0.00      0.00        18
           L       0.68      0.81      0.74        84
           R       0.73      0.74      0.74        86

    accuracy                           0.70       188
   macro avg       0.47      0.52      0.49       188
weighted avg       0.64      0.70      0.67       188


The system cannot find the path specified.
```




0
B
R
L
1
2
3
4


X3 <= 2.5
gini = 0.563
samples = 437
value = [31.0, 204.0, 202.0]
class = L
X2 <= 1.5
gini = 0.462
samples = 176
value = [14.0, 122.0, 40.0]
class = L
X4 <= 2.5
gini = 0.512
samples = 261
value = [17, 82, 162]
class = R
X3 <= 1.5
gini = 0.571
samples = 37
value = [5, 11, 21]
class = R
X1 <= 1.5
gini = 0.339
samples = 139
value = [9, 111, 19]
class = L
X1 <= 1.5
gini = 0.534
samples = 97
value = [5.0, 55.0, 31.0]
class = L
X2 <= 3.5
gini = 0.335
samples = 164
value = [9, 24, 131]
class = R
gini = 0.602
samples = 17
value = [3, 9, 5]
class = L
gini = 0.34
samples = 20
value = [2, 2, 16]
class = R
gini = 0.605
samples = 23
value = [3, 10, 10]
class = L
gini = 0.233
samples = 116
value = [6, 101, 9]
class = L
gini = 0.32
samples = 16
value = [2, 1, 13]
class = R
gini = 0.45
samples = 51
value = [5, 37, 15]
class = L
gini = 0.1
samples = 96
value = [2, 3, 91]
class = R
gini = 0.545
samples = 68
value = [7, 21, 40]
class = R


X3 <= 2.5
entropy = 1.295
samples = 437
value = [31.0, 204.0, 202.0]
class = L
X2 <= 2.5
entropy = 1.143
samples = 176
value = [14.0, 122.0, 40.0]
class = L
X4 <= 2.5
entropy = 1.209
samples = 261
value = [17, 82, 162]
class = R
X4 <= 3.5
entropy = 1.445
samples = 70
value = [10.0, 31.0, 29.0]
class = L
X1 <= 2.5
entropy = 0.707
samples = 106
value = [4, 91, 11]
class = L
X1 <= 1.5
entropy = 1.266
samples = 97
value = [5.0, 55.0, 31.0]
class = L
X2 <= 3.5
entropy = 0.594
samples = 164
value = [9, 24, 131]
class = R
entropy = 1.335
samples = 43
value = [6, 26, 11]
class = L
entropy = 1.249
samples = 27
value = [4, 5, 15]
class = R
entropy = 1.191
samples = 43
value = [4, 29, 10]
class = L
entropy = 0.115
samples = 63
value = [0, 62, 1]
class = L
entropy = 0.565
samples = 16
value = [2, 1, 13]
class = R
entropy = 1.117
samples = 51
value = [5, 37, 15]
class = L
entropy = 0.346
samples = 96
value = [2, 3, 91]
class = R
entropy = 1.311
samples = 68
value = [7, 21, 40]
class = R

Source Code for Daily Weather Data:

```
#importing the dataset
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings as wr
wr.filterwarnings('ignore')

#importing the dataset
df=pd.read_csv('daily_weather.csv')
print(df.head(),'\n')
print(df.shape,'\n')
print(df.isnull().sum())
```

```
#Data preprocessing
print(df.columns)
del df['number']
print(df.head(2))

clean_df=df.copy()
clean_df=clean_df.dropna()
clean_df.info()
clean_df.isnull().sum()
clean_df['high_humidity_label']=(clean_df['relative_humidity_3pm']>24.99)*1
print(clean_df['high_humidity_label'])
print(clean_df.head())

sns.pairplot(data=clean_df,hue='high_humidity_label')

sns.heatmap(clean_df.corr(),cmap='Pastel2',annot=True,fmt='.2f')
plt.totle('correlation heatmap')
```

```
del clean_df['relative_humidity_3pm']
clean_df.describe()

x=clean_df.iloc[:,:-1].values
y=clean_df.iloc[:,-1].values

from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_scaled=sc.fit_transform(x)

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x_scaled,y,test_size=0.30,random_state=100)

print(x_train.shape,'\n',x_test.shape,'\n',y_train.shape,'\n',y_test.shape)

from sklearn.tree import DecisionTreeClassifier
humidity_classifier=DecisionTreeClassifier(criterion='gini',max_depth=10)
humidity_classifier.fit(x_train,y_train)

print(type(humidity_classifier))

y_pred=humidity_classifier.predict(x_test)
```

```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
cm=confusion_matrix(y_test,y_pred)
print('The confusion matrix representation is: ',cm)

acc=accuracy_score(y_test,y_pred)*100
print('Accuracy is :',acc)

rep=classification_report(y_test,y_pred)
print('The classifictaion report is:',rep)

#tree plot
from sklearn import tree
plt.figure(figsize=(15,10))
tree.plot_tree(humidity_classifier,rounded=True,max_depth=5)
plt.show()
```

Output:

```
Python 3.10.13 | packaged by Anaconda, Inc. | (main, Sep 11 2023, 13:15:57) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython  -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/mvsla/OneDrive/Desktop/ml/decision_tree_weather.py', wdir='C:/Users/mvsla/OneDrive/Desktop/ml')
   number  air_pressure_9am  ...  relative_humidity_9am  relative_humidity_3pm
0       0        918.060000  ...              42.420000              36.160000
1       1        917.347688  ...              24.328697              19.426597
2       2        923.040000  ...               8.900000              14.460000
3       3        920.502751  ...              12.189102              12.742547
4       4        921.160000  ...              92.410000              76.740000

[5 rows x 11 columns]

(1095, 11)
number                    0
air_pressure_9am          3
air_temp_9am              5
avg_wind_direction_9am    4
avg_wind_speed_9am        3
max_wind_direction_9am    3
max_wind_speed_9am        4
rain_accumulation_9am     6
rain_duration_9am         3
relative_humidity_9am     0
relative_humidity_3pm     0
dtype: int64
Index(['number', 'air_pressure_9am', 'air_temp_9am', 'avg_wind_direction_9am',
       'avg_wind_speed_9am', 'max_wind_direction_9am', 'max_wind_speed_9am',
       'rain_accumulation_9am', 'rain_duration_9am', 'relative_humidity_9am',
       'relative_humidity_3pm'],
      dtype='object')
   air_pressure_9am  air_temp_9am  ...  relative_humidity_9am  relative_humidity_3pm
0        918.060000     74.822000  ...              42.420000              36.160000
1        917.347688     71.403843  ...              24.328697              19.426597
```

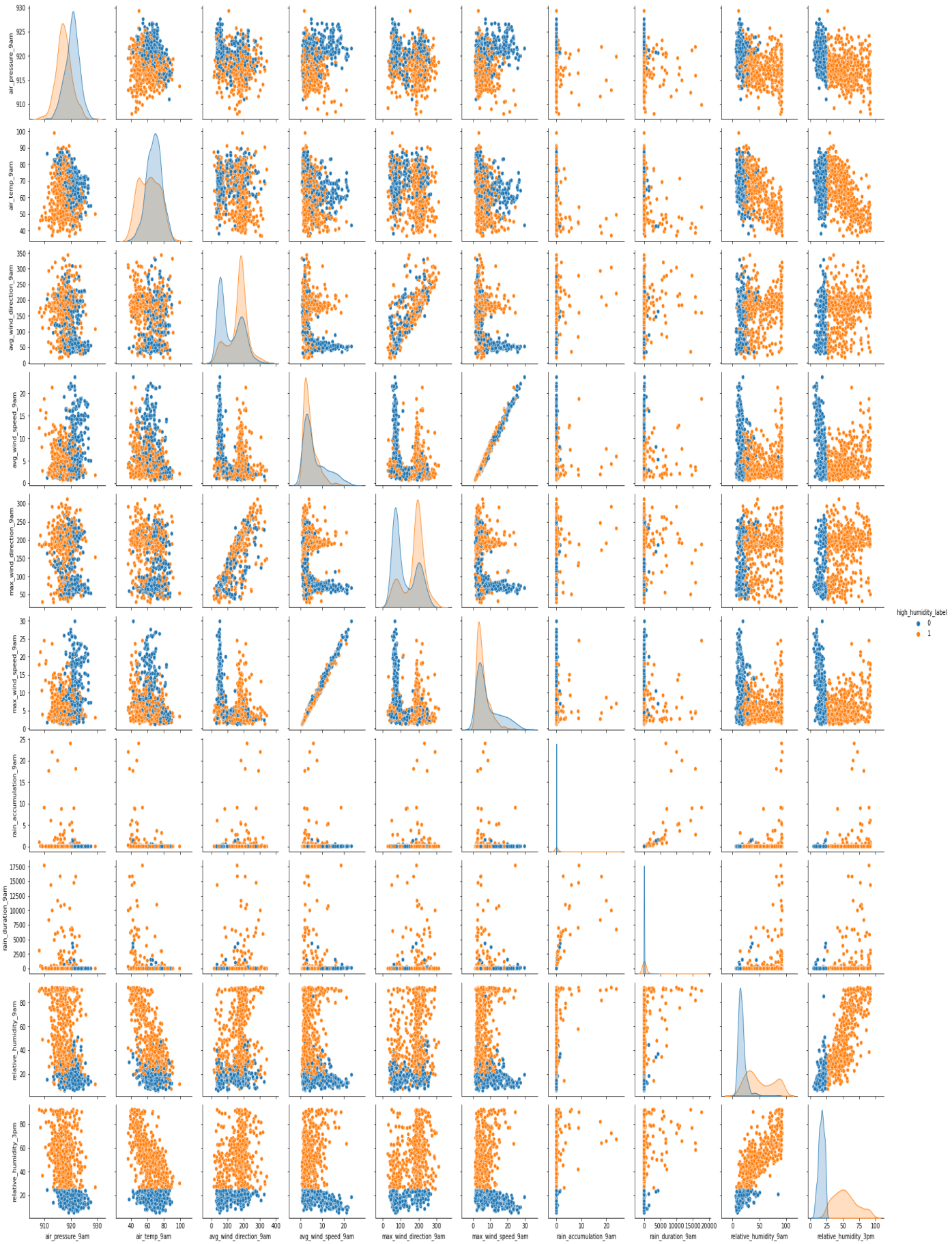
```
[2 rows x 10 columns]
<class 'pandas.core.frame.DataFrame'>
Index: 1064 entries, 0 to 1094
Data columns (total 10 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   air_pressure_9am        1064 non-null   float64
 1   air_temp_9am            1064 non-null   float64
 2   avg_wind_direction_9am  1064 non-null   float64
 3   avg_wind_speed_9am      1064 non-null   float64
 4   max_wind_direction_9am  1064 non-null   float64
 5   max_wind_speed_9am      1064 non-null   float64
 6   rain_accumulation_9am   1064 non-null   float64
 7   rain_duration_9am       1064 non-null   float64
 8   relative_humidity_9am   1064 non-null   float64
 9   relative_humidity_3pm   1064 non-null   float64
dtypes: float64(10)
memory usage: 91.4 KB
```

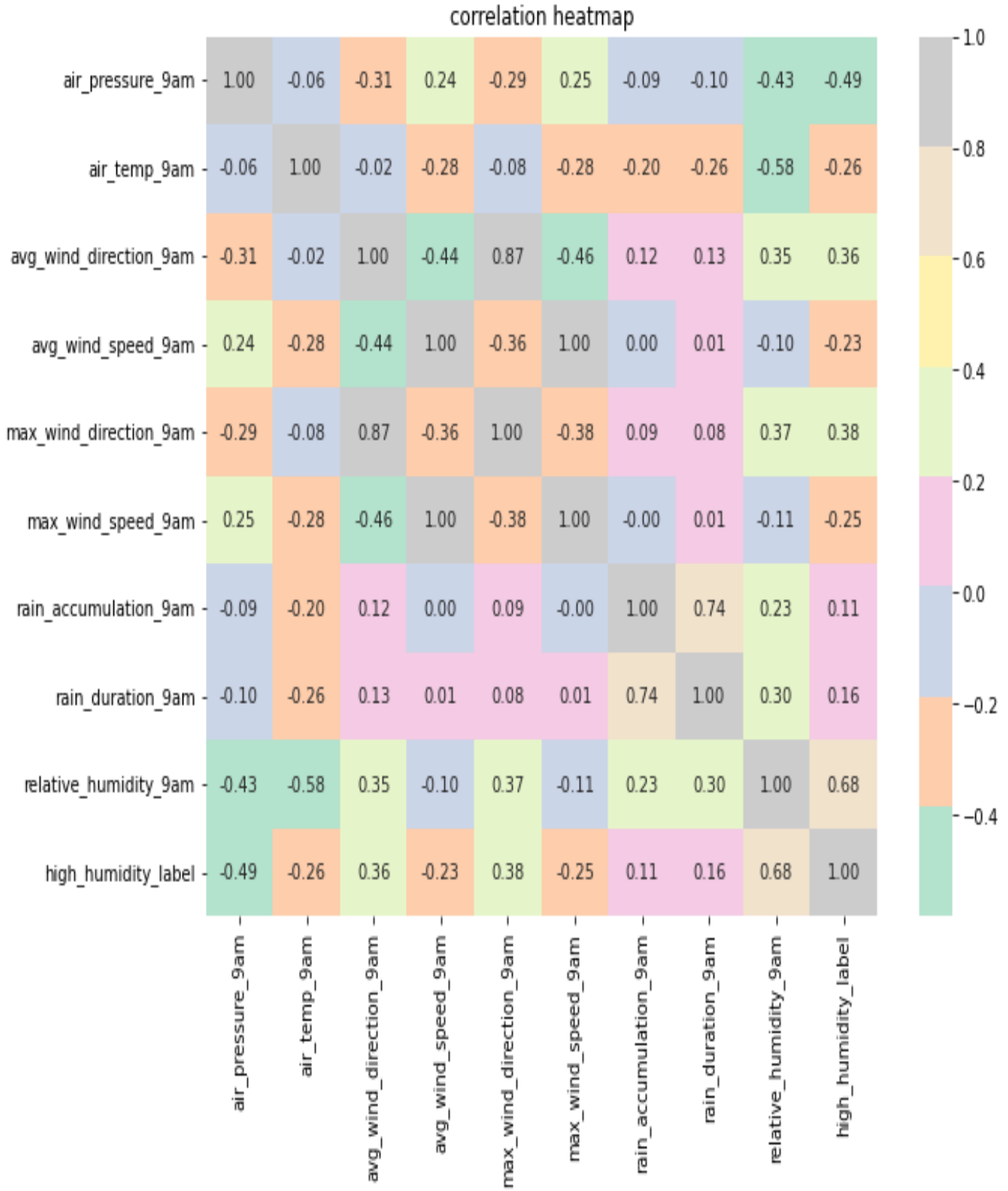
```
0       1
1       0
2       0
3       0
4       1
       ..
1090    1
1091    1
1092    1
1093    1
1094    0
Name: high_humidity_label, Length: 1064, dtype: int32
   air_pressure_9am  air_temp_9am  ...  relative_humidity_3pm  high_humidity_label
0        918.060000     74.822000  ...              36.160000                    1
1        917.347688     71.403843  ...              19.426597                    0
2        923.040000     60.638000  ...              14.460000                    0
3        920.502751     70.138895  ...              12.742547                    0
4        921.160000     44.294000  ...              76.740000                    1

[5 rows x 11 columns]
(744, 9)
 (320, 9)
 (744,)
 (320,)
<class 'sklearn.tree._classes.DecisionTreeClassifier'>
The confusion matrix representation is:  [[139  21]
 [ 22 138]]
Accuracy is : 86.5625
The classifictaion report is:              precision    recall  f1-score   support

           0       0.86      0.87      0.87       160
           1       0.87      0.86      0.87       160

    accuracy                           0.87       320
   macro avg       0.87      0.87      0.87       320
weighted avg       0.87      0.87      0.87       320
```

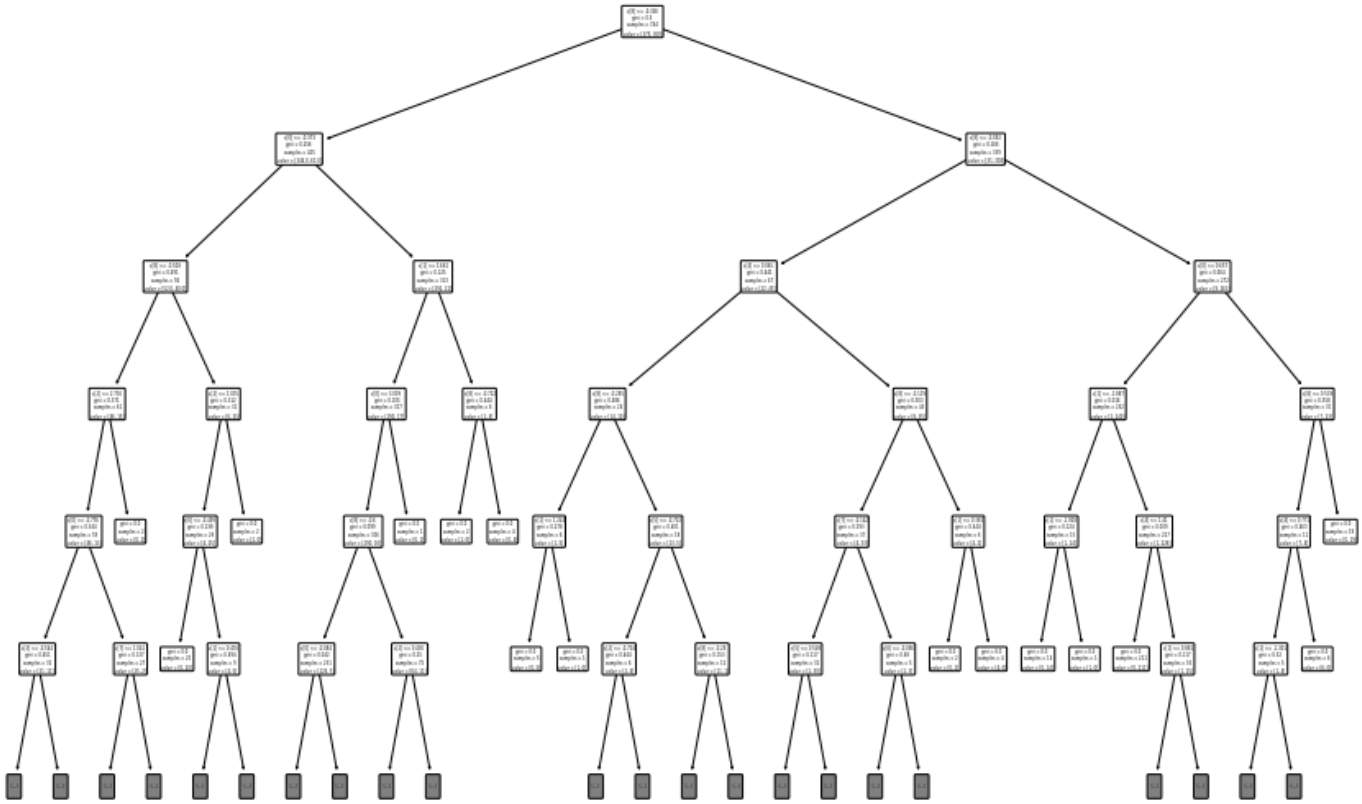
Important

Figures are displayed in the Plots pane by default. To make them also appear inline in the console, you need to uncheck "Mute inline plotting" under the options menu of Plots.

```
The system cannot find the path specified.
```


air_pressure_9am
air_temp_9am
avg_wind_direction_9am
avg_wind_speed_9am
max_wind_direction_9am
max_wind_speed_9am
rain_accumulation_9am
rain_duration_9am
relative_humidity_9am
relative_humidity_3pm
high_humidity_label
0
1


correlation heatmap
air_pressure_9am 1.00 -0.06 -0.31 0.24 -0.29 0.25 -0.09 -0.10 -0.43 -0.49
air_temp_9am -0.06 1.00 -0.02 -0.28 -0.08 -0.28 -0.20 -0.26 -0.58 -0.26
avg_wind_direction_9am -0.31 -0.02 1.00 -0.44 0.87 -0.46 0.12 0.13 0.35 0.36
avg_wind_speed_9am 0.24 -0.28 -0.44 1.00 -0.36 1.00 0.00 0.01 -0.10 -0.23
max_wind_direction_9am -0.29 -0.08 0.87 -0.36 1.00 -0.38 0.09 0.08 0.37 0.38
max_wind_speed_9am 0.25 -0.28 -0.46 1.00 -0.38 1.00 -0.00 0.01 -0.11 -0.25
rain_accumulation_9am -0.09 -0.20 0.12 0.00 0.09 -0.00 1.00 0.74 0.23 0.11
rain_duration_9am -0.10 -0.26 0.13 0.01 0.08 0.01 0.74 1.00 0.30 0.16
relative_humidity_9am -0.43 -0.58 0.35 -0.10 0.37 -0.11 0.23 0.30 1.00 0.68
high_humidity_label -0.49 -0.26 0.36 -0.23 0.38 -0.25 0.11 0.16 0.68 1.00
air_pressure_9am
air_temp_9am
avg_wind_direction_9am
avg_wind_speed_9am
max_wind_direction_9am
max_wind_speed_9am
rain_accumulation_9am
rain_duration_9am
relative_humidity_9am
high_humidity_label
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4



Observations:

- ✓ The accuracy of the decision tree in case of using gini index for Balance weights and distance database is 68%
- ✓ The accuracy of the decision tree model in case of entropy for the Balance dataset is 70%
- ✓ The accuarcy of the decision tree model in case of using gini index on the daily weather dataset is 87%
- ✓ Decision tree is highly prone to overfitting as the model tries to learn from the training data in the most perfect form.
- ✓ So to avoid the problem of overfitting, techniques like prepruning and postpruning are used.
- ✓ Random Forest model is an ensembling model machine learning where multiple decision trees are used and the overfitting problem is resolved.

Conclusion:

The Decision tree model has been fitted to the given datasets.

TASK 12: Perform One Way ANOVA

AIM: To perform one way ANOVA

Concept and Dataset Description:

- ❖ Concept of ANOVA: ANOVA or Analysis Of Variance is a statistical technique where we divide the variations into variations due to known factors and variations due to unknown factors.
- ❖ ANOVA is based on agricultural experiments and is a powerful technique used in statistics to test various factors as an alternative to independence of two means t-test in statistics.
- ❖ Dataset used is CVD Dataset.

Source Code :

```
#basic scipy packages
import pandas as pd

#stats packages
from scipy import stats

#graphing packages
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('whitegrid')
cvd_dataset=pd.read_csv('CVD_Dataset_EDA_ANOVA.csv')

sns.boxplot(x='Power',y='GrowthRate',data=cvd_dataset)
plt.xlabel(r'Power(w)')
plt.ylabel(r'Growth Rate ($\AA$/min')

cvd_pivot=cvd_dataset.pivot(index='RunID',
                            columns='Power',
                            values='GrowthRate')
print(cvd_pivot)
```

```
def degrees_of_freedom(df_pivot):
    #between the samples is across the row
    #within the server is down the row
    n=df_pivot.count(axis=1)
    a=df_pivot.count()
    n=n.iloc[0]
    a=a.iloc[0]
    TotalN=n*a
    dft=a-1
    dfe=TotalN-a
    dftotal=TotalN-1
    return dft,dfe,dftotal,n,a
dft,dfe,dftotal,n,a=degrees_of_freedom(cvd_pivot)
print('degrees of freedom between treatments: {:d}'.format(dft))
print("Degrees of freedom within treatments: {:d}".format(dfe))
print("Total degrees of freedom: {:d}".format(dftotal))
```

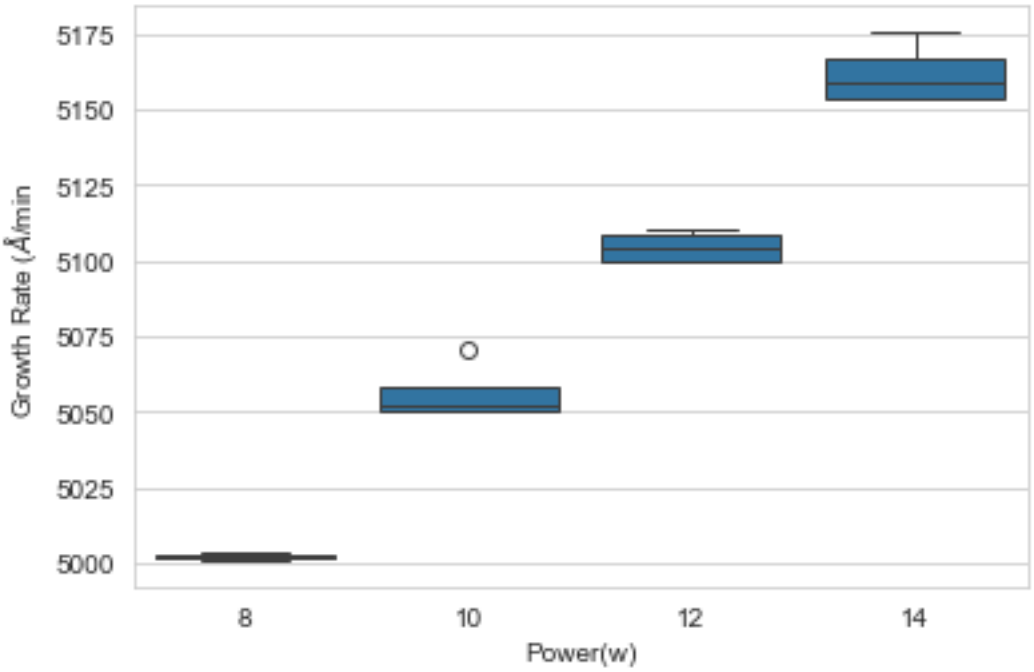
```
def compute_sums(df_pivot):
    yi_sum=df_pivot.sum()
    yi_avg=df_pivot.mean()
    y_sum=yi_sum.sum()
    y_avg=yi_avg.mean()
    return yi_sum,yi_avg,y_sum,y_avg
yi_sum,yi_avg,y_sum,y_avg=compute_sums(cvd_pivot)
print(r"The sum under each Power (W) treatment is: {}".format(yi_sum))
print(r"The average under each Power (W) treatment is: {}".format(yi_avg))
print(r"The grand sum is: {:.2f}".format(y_sum))
print(r"The grand average is: {:.2f}".format(y_avg))

def sum_of_squares(yi_avg,y_avg,df_pivot):
    sstr=(yi_avg-y_avg)**2
    sstr=sstr.sum()*n
    sse=df_pivot.sub(yi_avg.values)**2
    sse=sse.sum().sum()
    sst=sstr+sse
    return sstr,sse,sst
sstr,sse,sst=sum_of_squares(yi_avg,y_avg,cvd_pivot)
print("Sum of Squares between treatments is: {:.2f}".format(sstr))
print("Sum of Squares within treatments is: {:.2f}".format(sse))
print("Total Sum of Squares is: {:.2f}".format(sse))
```

```
def mean_squares(sstr,sse,dft,dfe):
    mstr=sstr/dft
    mse=sse/dfe
    return mstr,mse
mstr,mse=mean_squares(sstr,sse,dft,dfe)
print("Mean squares between treatments is: {:.2f}".format(mstr))
print("Mean squares within treatments is: {:.2f}".format(mse))

f=mstr/mse
p=stats.f.sf(f,dft,dfe)
print("F-value is: {:.2f}".format(f))
print("p-value is: {:.3f}".format(p))

#built in method for one way anova
f_val, p_val = stats.f_oneway(cvd_pivot[8],
               cvd_pivot[10], cvd_pivot[12], cvd_pivot[14])
print("F-value is: {:.2f}".format(f_val))
print("p-value is: {:.3f}".format(p_val))
```

Outputs:

```
Power          8           10          12          14
RunID
1      5002.320019  5070.505941  5110.123564  5163.853637
2      5003.601457  5053.803062  5100.022294  5153.563207
3      5001.024675  5050.102792  5108.673395  5175.444188
4      5001.576370  5050.176954  5100.200780  5153.171650
degrees of freedom between treatments: 3
Degrees of freedom within treatments: 12
Total degrees of freedom: 15
The sum under each Power (W) treatment is: Power
8     20008.522520
10    20224.588749
12    20419.020034
14    20646.032682
dtype: float64
The average under each Power (W) treatment is: Power
8     5002.130630
10    5056.147187
12    5104.755008
14    5161.508171
dtype: float64
The grand sum is: 81298.16
The grand average is: 5081.14
Sum of Squares between treatments is: 55535.33
Sum of Squares within treatments is: 707.23
Total Sum of Squares is: 707.23
Mean squares between treatments is: 18511.78
```

```
Mean squares between treatments is: 18511.78
Mean squares within treatments is: 58.94
F-value is: 314.10
p-value is: 0.000
F-value is: 314.10
p-value is: 0.000
```



Observations:

- ✓ The method of least squares is used to estimate different types of least square values in ANOVA.
- ✓ The calculated F- value is found to be 314.10
- ✓ The p-value is found to be 0.00
- ✓ Since the p-value is found to be very small, we may reject the null hypothesis that the power does not have an influence on the growth rate.

Conclusion:

From the Analysis Of Variance, we can conclude that the power has influence on the growth rate from the given CVD dataset.

TASK 13: Work with JSON data

AIM : To work with JSON data i.e. JavaScript Object Notation Data format.

Concept:

- ❖ Most of the data in the real world is in the form of JSON format.
- ❖ JSON is a semi structured format of the data where the values are found in the key-value pairs.
- ❖ Most of the online data is found in HTML/DHTML/JSON format.
- ❖ So, it is imperative to work with JSON data when working in real-time projects.
- ❖ Here, iris data is loaded in the JSON format.

Source Code :

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings as wr
wr.filterwarnings('ignore')

df=pd.read_json('iris.json')
print(df.head(5))

print(df.shape)
print(df.info())
print(df.isnull().sum())
print(df.describe())

#count plot
count_plots=df['species'].value_counts()
plt.figure(figsize=(10,8))
plt.bar(count_plots.index,count_plots,color='blue')
plt.title('count plot for class balance')
plt.xlabel('count of classes')
plt.ylabel('classes')
plt.show()
```

```
#bivariate analysis
#creating the kernel density plots
sns.set_style('darkgrid')
numerical_columns=df.select_dtypes(include=['int64','float64']).columns
plt.figure(figsize=(14,len(numerical_columns)*3))
for idx, feature in enumerate(numerical_columns,1):
    plt.subplot(len(numerical_columns),2,idx)
    sns.histplot(df[feature],kde=True)
    plt.title(f"{feature}| skewness:{round(df[feature].skew(),2)}")

plt.tight_layout()
plt.show()

sns.pairplot(data=df,palette='pastel',hue='species')
plt.show()
```

```
x=df.iloc[:,:-1].values
y=df.iloc[:,-1].values

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=10

from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
lr.fit(x_train,y_train)

y_pred=lr.predict(x_test)

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

rep=classification_report(y_test,y_pred)
print(rep)
```

Outputs:

```
In [1]: runfile('C:/Users/mvsla/OneDrive/Desktop/ml/json_iris.py', wdir='C:/Users/m
OneDrive/Desktop/ml')
   sepalLength  sepalWidth  petalLength  petalWidth species
0          5.1         3.5          1.4         0.2  setosa
1          4.9         3.0          1.4         0.2  setosa
2          4.7         3.2          1.3         0.2  setosa
3          4.6         3.1          1.5         0.2  setosa
4          5.0         3.6          1.4         0.2  setosa
(150, 5)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   sepalLength  150 non-null    float64
 1   sepalWidth   150 non-null    float64
 2   petalLength  150 non-null    float64
 3   petalWidth   150 non-null    float64
 4   species      150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
None
```

```
sepalLength    0
sepalWidth     0
petalLength    0
petalWidth     0
species        0
dtype: int64
       sepalLength  sepalWidth  petalLength  petalWidth
count   150.000000  150.000000   150.000000  150.000000
mean      5.843333    3.057333     3.758000    1.199333
std       0.828066    0.435866     1.765298    0.762238
min       4.300000    2.000000     1.000000    0.100000
25%       5.100000    2.800000     1.600000    0.300000
50%       5.800000    3.000000     4.350000    1.300000
75%       6.400000    3.300000     5.100000    1.800000
max       7.900000    4.400000     6.900000    2.500000
```

```
[[14  0  0]
 [ 0 17  0]
 [ 0  0 14]]
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        14
  versicolor       1.00      1.00      1.00        17
   virginica       1.00      1.00      1.00        14

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

Observations:

- ✓ The JSON data is easier to work with when pandas method pd.read_json() is used.

Conclusion:

JSON data is easy and very important to understand and work with if we are in the data engineering field.

TASK 14: Apply Simple and Multiple Linear Regression

AIM : Application of simple linear regression on Salary Dataset and Multiple Linear Regression on USA Housing Dataset.

Algorithm and Dataset Description:

Algorithm: Linear Regression

- Regression is another predictive analytics task where we try to predict a real number instead of a category.
- Linear regression uses a straight line to model the data and then uses that model to predict the value.
- Simple linear regression equation is **y=a+bx** implying that the target feature has only one predictor variable.
- Multiple linear regression equation is $\mathbf{y= b_0+b_1x_1+b_2x_2+.....b_nx_n}$ implying that the target feature is predicted using multiple predictor variables.

Datasets Description:

- Salary dataset is a simple dataset of size 30 rows x 2 columns.
- USA housing dataset is a dataset of size 5000 rows x 7 columns where price is the target feature.

Source Code For Salary Dataset:

```
#importing the libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings as wr
wr.filterwarnings('ignore')

#importing the dataset
df=pd.read_csv("C:/Users/mvsla/OneDrive/Documents/Linear_regression_on_datasets[1]/Week-2- Packages, EDA, Simple Linear regression/Salary_dataset.csv
print(df.head())

df=df.drop(df.columns[[0]],axis=1)

#EDA and plots
print(df.isnull().sum(),'\n')
print(df.describe,'\n')
print(df.shape,'\n')
print(df.info(),'\n')
print(df.duplicated().sum(),'\n')

#plot to visualize the relationship between the x and y
plt.figure(figsize=(10,8))
sns.scatterplot(data=df,x='YearsExperience',y='Salary')
plt.show()
```

```
#x=df['YearsExperience']
#y=df['Salary']

x=df.iloc[:,:-1].values
y=df.iloc[:,-1].values

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=10)
print('Training data X shape:',x_train.shape)
print('Training data Y shape:',y_train.shape)
print('Testing data X shape:',x_test.shape)
print('Testing data Y shape:',y_test.shape)

#x_train=x_train[:,np.newaxis]
#x_test=x_test[:,np.newaxis]
```

```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()

lr.fit(x_train,y_train)
y_pred=lr.predict(x_test)

from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
print('Mean squared error is: ',mean_squared_error(y_test,y_pred))
print('Mean absolute error is: ',mean_absolute_error(y_test,y_pred))
print('R sqaured error is: ',r2_score(y_test,y_pred))

# Intecept and coeff of the line
print('Intercept of the model:',lr.intercept_)
print('Coefficient of the line:',lr.coef_)
x1= [[1.3]]
y_pred1=lr.predict(x1)
print(y_pred1)

y1 = lr.coef_ * 1.3 + lr.intercept_
print(y1)

#visualizing the test set results
plt.scatter(x_test,y_test,color='red')
plt.plot(x_train,lr.predict(x_train),color='blue')
plt.title('Salary vs Experience test set results')
plt.xlabel('years of experience')
plt.ylabel('Salary')
plt.show()
```

Output:

```
mvsla/OneDrive/Desktop/ml')
   Unnamed: 0  YearsExperience   Salary
0           0              1.2  39344.0
1           1              1.4  46206.0
2           2              1.6  37732.0
3           3              2.1  43526.0
4           4              2.3  39892.0
YearsExperience    0
Salary             0
dtype: int64
```

```
<bound method NDFrame.describe of     YearsExperience    Salary
0                1.2   39344.0
1                1.4   46206.0
2                1.6   37732.0
3                2.1   43526.0
4                2.3   39892.0
5                3.0   56643.0
6                3.1   60151.0
7                3.3   54446.0
8                3.3   64446.0
9                3.8   57190.0
10               4.0   63219.0
11               4.1   55795.0
12               4.1   56958.0
13               4.2   57082.0
14               4.6   61112.0
15               5.0   67939.0
16               5.2   66030.0
17               5.4   83089.0
```

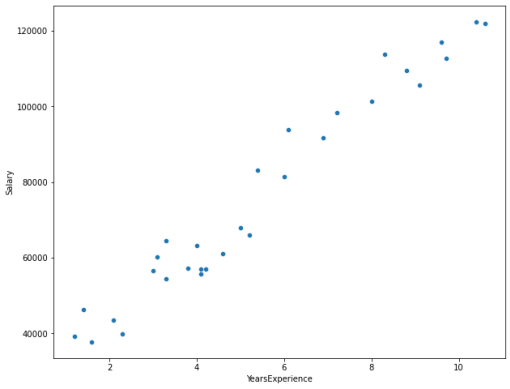
```
18              6.0   81364.0
19              6.1   93941.0
20              6.9   91739.0
21              7.2   98274.0
22              8.0  101303.0
23              8.3  113813.0
24              8.8  109432.0
25              9.1  105583.0
26              9.6  116970.0
27              9.7  112636.0
28             10.4  122392.0
29             10.6  121873.0>

(30, 2)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   YearsExperience  30 non-null     float64
 1   Salary           30 non-null     float64
dtypes: float64(2)
memory usage: 608.0 bytes
None

0
```

Important

Figures are displayed in the Plots pane by default. To make them also appear inline in the console, you need to uncheck "Mute inline plotting" under the options menu of Plots.

```
 Training data X shape: (21, 1)
Training data Y shape: (21,)
Testing data X shape: (9, 1)
Testing data Y shape: (9,)
Mean squared error is:  21713548.637118697
Mean absolute error is:  4067.597514637223
R sqaured error is:  0.9647278344670827
Intercept of the model: 26277.03296973191
Coefficient of the line: [9303.95933197]
[38372.18010129]
[38372.18010129]
```



Source Code for USA Housing:

```
#importing the libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings as wr
wr.filterwarnings('ignore')

#importing the dataset
df=pd.read_csv('USA_housing.csv')
print(df.head())

print(df.isnull().sum(),'\n')
print(df.shape)
print(df.info(),'\n')
print(df.describe())

df=df.drop('Address',axis=1)

#EDA using plots
sns.pairplot(data=df,hue='Price')
plt.show()
#Correlation heatmap
sns.heatmap(data=df.corr(),cmap='Pastel2',annot=True,fmt='.2f')
plt.show()
```

```
#Feature scaling
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
df_scaled=sc.fit_transform(df)

#Extracting the target and features from the data frame
x=df.drop('Price',axis=1)
y=df['Price']

#training and testing data splitting
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=10)
print('x_train shape is: ',x_train.shape,'\n y_train shape is:',y_train.shape,'\n

# Model selection
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)

y_pred=lr.predict(x_test)

from sklearn.metrics import r2_score,mean_squared_error
print('The R squared value is :', r2_score(y_test,y_pred))
print('The mean squared value is: ',mean_squared_error(y_test,y_pred))
```

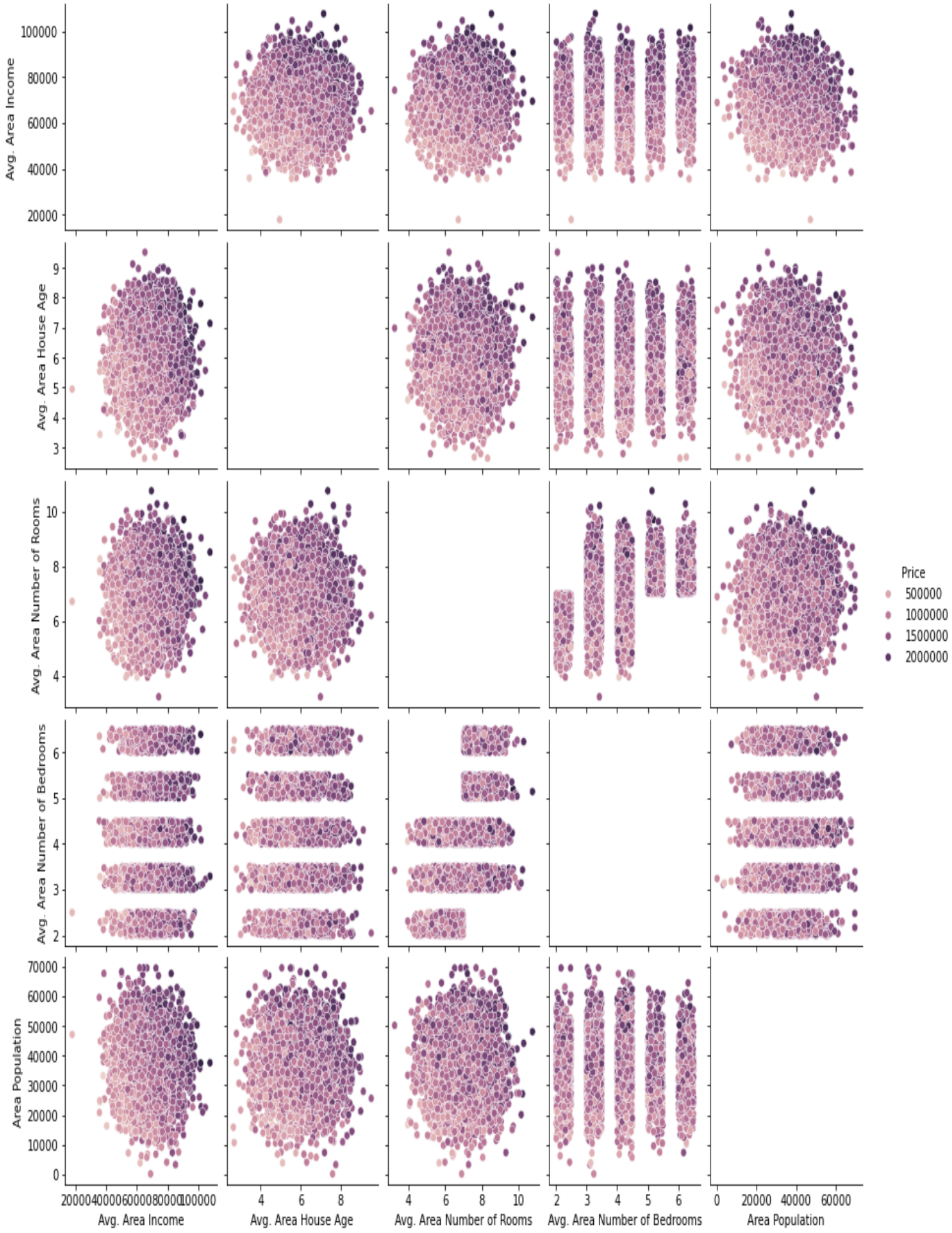
Outputs:

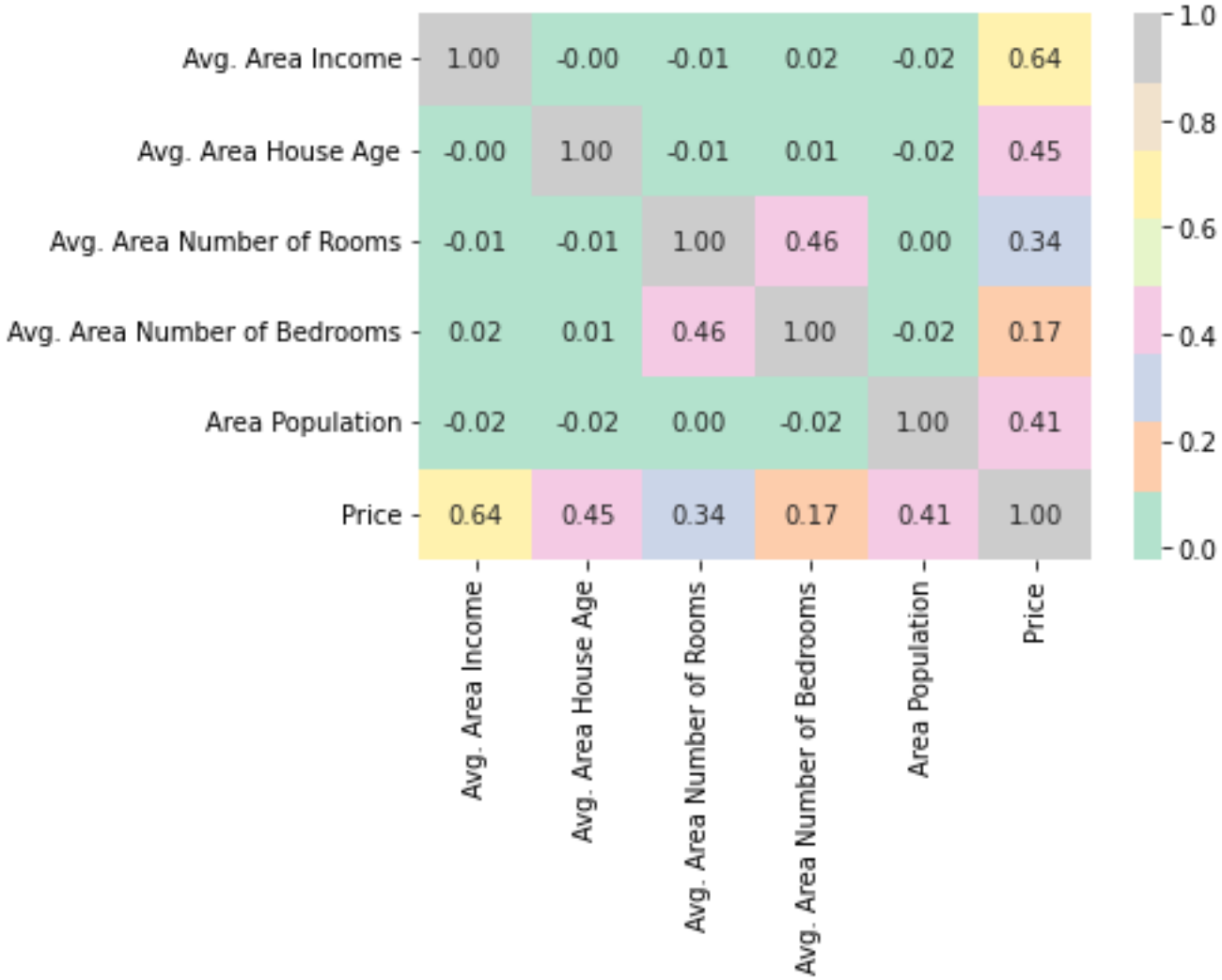
```
In [1]: runfile('C:/Users/mvsla/OneDrive/Desktop/ml/linear_regression_2.py', wdir='C:/Users/
mvsla/OneDrive/Desktop/ml')
   Avg. Area Income  ...                                            Address
0      79545.458574  ...  208 Michael Ferry Apt. 674\nLaurabury, NE 3701...
1      79248.642455  ...  188 Johnson Views Suite 079\nLake Kathleen, CA...
2      61287.067179  ...  9127 Elizabeth Stravenue\nDanieltown, WI 06482...
3      63345.240046  ...                          USS Barnett\nFPO AP 44820
4      59982.197226  ...                         USNS Raymond\nFPO AE 09386

[5 rows x 7 columns]
Avg. Area Income                0
Avg. Area House Age             0
Avg. Area Number of Rooms       0
Avg. Area Number of Bedrooms    0
Area Population                 0
Price                           0
Address                         0
dtype: int64
```

```
(5000, 7)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 7 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   Avg. Area Income              5000 non-null   float64
 1   Avg. Area House Age           5000 non-null   float64
 2   Avg. Area Number of Rooms     5000 non-null   float64
 3   Avg. Area Number of Bedrooms  5000 non-null   float64
 4   Area Population               5000 non-null   float64
 5   Price                         5000 non-null   float64
 6   Address                       5000 non-null   object
dtypes: float64(6), object(1)
memory usage: 273.6+ KB
None
```

```
       Avg. Area Income  Avg. Area House Age  ...  Area Population         Price
count       5000.000000          5000.000000  ...      5000.000000  5.000000e+03
mean       68583.108984             5.977222  ...     36163.516039  1.232073e+06
std        10657.991214             0.991456  ...      9925.650114  3.531176e+05
min        17796.631190             2.644304  ...       172.610686  1.593866e+04
25%        61480.562388             5.322283  ...     29403.928702  9.975771e+05
50%        68804.286404             5.970429  ...     36199.406689  1.232669e+06
75%        75783.338666             6.650808  ...     42861.290769  1.471210e+06
max       107701.748378             9.519088  ...     69621.713378  2.469066e+06

[8 rows x 6 columns]
```

```
 x_train shape is:  (3500, 5)
 y_train shape is: (3500,)
 x_test shape is:  (1500, 5)
 y_test shape is:  (1500,)
The R squared value is : 0.9159275718398349
The mean squared value is:  10408992254.124008
```


Avg. Area Income
Avg. Area House Age
Avg. Area Number of Rooms
Avg. Area Number of Bedrooms
Area Population
Price
500000
1000000
1500000
2000000



Observations:

- ✓ The linear regression is used to predict the future value using the given data using the regression equations.
- ✓ The R squared value is found to be 0.96 for salary dataset and 0.91 for USA Housing dataset.
- ✓ The mean squared error in salary dataset is : 21713548.637118697
- ✓ The mean squared error in USA Housing dataset is : 10408992254.124008

Conclusion:

The mean squared error value must be minimised and the R squared value must be maximized in regression problems. Simple linear regression is picked when the dataset has only one feature and multiple linear regression is picked when the dataset has more than one feature.

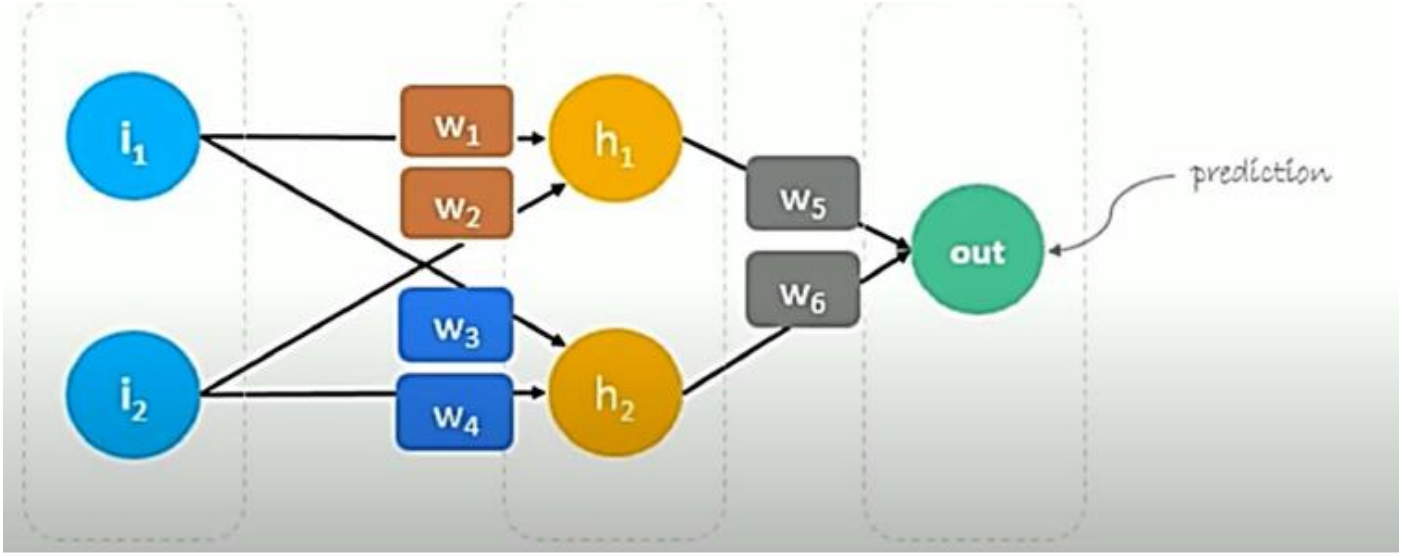
TASK 15: Apply an artificial neural network

AIM : Applying of an artificial neural network using keras on churn modelling dataset.

Concept and dataset description:

Concept of ANN:

- Artificial Neural Networks are inspired from the working of human brain.
- Neural Networks consist of small units called neurons or perceptrons, which can have multiple inputs $x_1, x_2 \ldots x_n$ and outputs y_1.
- The inputs of the neurons with corresponding number of weights and one bias term is added.
- Multiple neurons are comined together to form layers.
- These layers are connected to form the network, by feeding the outputs of one layer as the inputs of the next layers, hence forming the multi layer architecture.
- A simple diagram of the ANN is as follows:



Dataset Description:

- The dataset is called as 'Churn Modelling'.
- It is a very important dataset where the data is about the customers' data of them staying with the bank or leaving the bank.
- It has 10,000 rows and 14 columns, with no missing values.
- Since it is a bigger dataset, it is important to move to deep learning models i.e. ANN to solve the predictive problem
- The ANN was implemented using keras library, with tensorflow framework at the backend.

Source Code:

```
#Neural Network
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
# Importing label encoder for encoding, category data
from sklearn.preprocessing import LabelEncoder
#read the dataset
df = pd.read_csv("Churn_Modelling.csv")
print(df.head())
print(df.dtypes)
print(df.columns)
print(df.shape)
#drop un-necessary columns
df=df.drop(['RowNumber','CustomerId','Surname'], axis=1)
print(df.shape)
print(df.dtypes)
# object dtype means categorical data, see classes of objects
print(df.Gender.value_counts())
print(df.Geography.value_counts())
#object is categorical data, so convert to numeric by encoding
obj_list= ['Geography','Gender']
```

```
for col in obj_list:
    encoder = LabelEncoder()
    encoder.fit(df[col])
    df[col] = encoder.transform(df[col])
#see all dtypes are numeric
print(df.dtypes)
print(df.head())
#separate X and target
X = df.iloc[:,:9].values
y = df.iloc[:,-1].values
# apply train test split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 1)
# Feature Scaling, observe som efeatures of X are with too large values, so scale down to small values
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
#build your keras ANN model
from keras.models import Sequential
from keras.layers import Dense
# Defining the ANN model
model = Sequential()
# Adding the first hidden layer, by passing X with 8 features
model.add(Dense(units = 4, kernel_initializer = 'uniform', activation = 'relu', input_dim = 9))

# Adding the second hidden layer
model.add(Dense(units = 4, kernel_initializer = 'uniform', activation = 'relu'))

# Adding the output layer
model.add(Dense(units = 1, kernel_initializer = 'uniform', activation = 'sigmoid'))

# Compiling the ANN
model.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
model.summary()
# Fitting the ANN to the Training set
model.fit(X_train, y_train, batch_size = 2, epochs = 5)
y_pred = model.predict(X_test)
print(y_pred[10:20])
print(y_test[10:20])
```

Outputs:

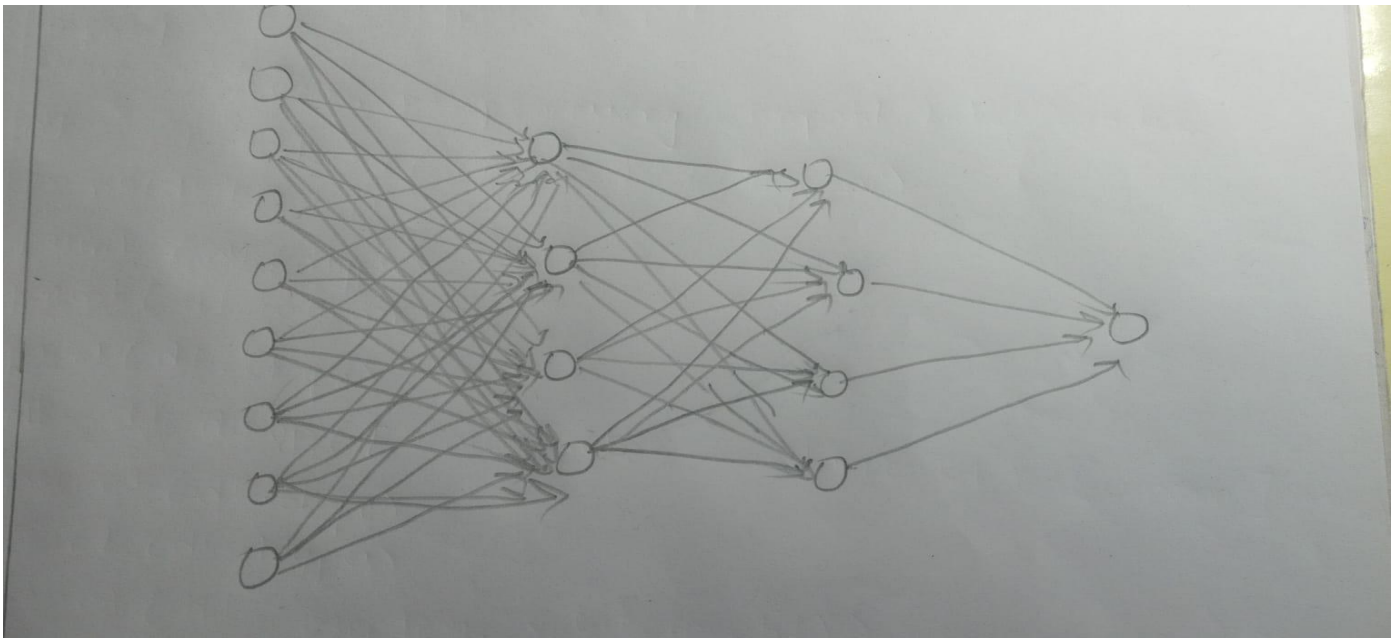
```
In [1]: runfile('C:/Users/mvsla/OneDrive/Desktop/ml/neural_network.py', wdir='C:/Users/mvsla/
OneDrive/Desktop/ml')
   RowNumber  CustomerId   Surname  ...  IsActiveMember EstimatedSalary Exited
0          1    15634602  Hargrave  ...               1       101348.88      1
1          2    15647311      Hill  ...               1       112542.58      0
2          3    15619304      Onio  ...               0       113931.57      1
3          4    15701354      Boni  ...               0        93826.63      0
4          5    15737888  Mitchell  ...               1        79084.10      0
```

```
[5 rows x 14 columns]
RowNumber            int64
CustomerId           int64
Surname             object
CreditScore          int64
Geography           object
Gender              object
Age                  int64
Tenure               int64
Balance            float64
NumOfProducts        int64
HasCrCard            int64
IsActiveMember       int64
EstimatedSalary    float64
Exited               int64
dtype: object
Index(['RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Geography',
       'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'EstimatedSalary', 'Exited'],
      dtype='object')
```

```
Name: count, dtype: int64
CreditScore          int64
Geography            int32
Gender               int32
Age                  int64
Tenure               int64
Balance            float64
NumOfProducts        int64
HasCrCard            int64
IsActiveMember       int64
EstimatedSalary    float64
Exited               int64
dtype: object
   CreditScore  Geography  Gender  ...  IsActiveMember  EstimatedSalary  Exited
0          619          0       0  ...               1        101348.88       1
1          608          2       0  ...               1        112542.58       0
2          502          0       0  ...               0        113931.57       1
3          699          0       0  ...               0         93826.63       0
4          850          2       0  ...               1         79084.10       0

[5 rows x 11 columns]
```

```
(10000, 14)
(10000, 11)
CreditScore          int64
Geography           object
Gender              object
Age                  int64
Tenure               int64
Balance            float64
NumOfProducts        int64
HasCrCard            int64
IsActiveMember       int64
EstimatedSalary    float64
Exited               int64
dtype: object
Gender
Male      5457
Female    4543
Name: count, dtype: int64
Geography
France     5014
Germany    2509
Spain      2477
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 4)                 40

 dense_1 (Dense)             (None, 4)                 20

 dense_2 (Dense)             (None, 1)                 5

=================================================================
Total params: 65
Trainable params: 65
Non-trainable params: 0
```



```
_________________________________________________________________
Epoch 1/5
4000/4000 [==============================] - 18s 4ms/step - loss: 0.4539 - accuracy: 0.8109
Epoch 2/5
4000/4000 [==============================] - 11s 3ms/step - loss: 0.4176 - accuracy: 0.8286
Epoch 3/5
4000/4000 [==============================] - 11s 3ms/step - loss: 0.4103 - accuracy: 0.8294
Epoch 4/5
4000/4000 [==============================] - 11s 3ms/step - loss: 0.4070 - accuracy: 0.8315
Epoch 5/5
4000/4000 [==============================] - 13s 3ms/step - loss: 0.4047 - accuracy: 0.8321
63/63 [==============================] - 0s 2ms/step
[[0.26987427]
 [0.08518809]
 [0.19799112]
 [0.0469233 ]
 [0.36926296]
 [0.05397568]
 [0.02361764]
 [0.12670958]
 [0.0902103 ]
 [0.3102065 ]]
[0 1 0 0 0 0 0 0 0 0]
```

Observations:

- ✓ Each epoch consists of one forward pa(ss and one backward pass over the entire data.
- ✓ The gradient descent type is set to stochastic gradient descent.
- ✓ The weights and bias i.e. the total number of trainable parameters are : 65
- ✓ The weights and bias are updated using the concept of gradient descent with backpropagation to optimize the cost function.
- ✓ The accuracy of the network has increased from 81% to 83% in 5 epochs or 5 iterations.

- ✓ We can see that the predicted values are closer to the actual values and are lying between 0 and 1 but not exactly 0 and 1 as the activation function used at the output layer is sigmoid activation function which constitutes the range (0,1).
- ✓ The activation functions are used to introduce the non-linearity into the linear computation in the neural network.
- ✓ The hyperparameters such as learning rate, no.of neurons, no.of hidden layers, gradient descent type, epochs and many more control the model's learning of the weights and bias term.
- ✓ The implementation of NN using keras is quite easy compared to that in tensorflow.

Conclusion:

The neural network is applied to the dataset and 83% accuracy is achieved.

TASK 16: PERFORM K-MEANS CLUSTERING ON THE DATA

AIM : To perform k-means clustering on the given dataset.

Concept and Dataset Description :

- ❖ K-means clustering is an unsupervised machine learning where we try to group the data elements based on similarity.
- ❖ K-means works on the concept that the similar items are closer to each other and dissimilar items are farther from each other.
- ❖ So, it is a distance based approach in the concept of clustering.
- ❖ The centroids of the k clusters are randomly initialized from the data elements and the distances from each data points are calculated.
- ❖ The datapoint with lower distance from the centroid are assigned to that cluster, and centroids are recalculated by comupting the average of the values in the cluster.
- ❖ The distances are computed again and the reassignment takes place. This process is repeated until no re-assignment takes place.
- ❖ The entire clustering process is applied on iris dataset for this particular task.

Source Code:

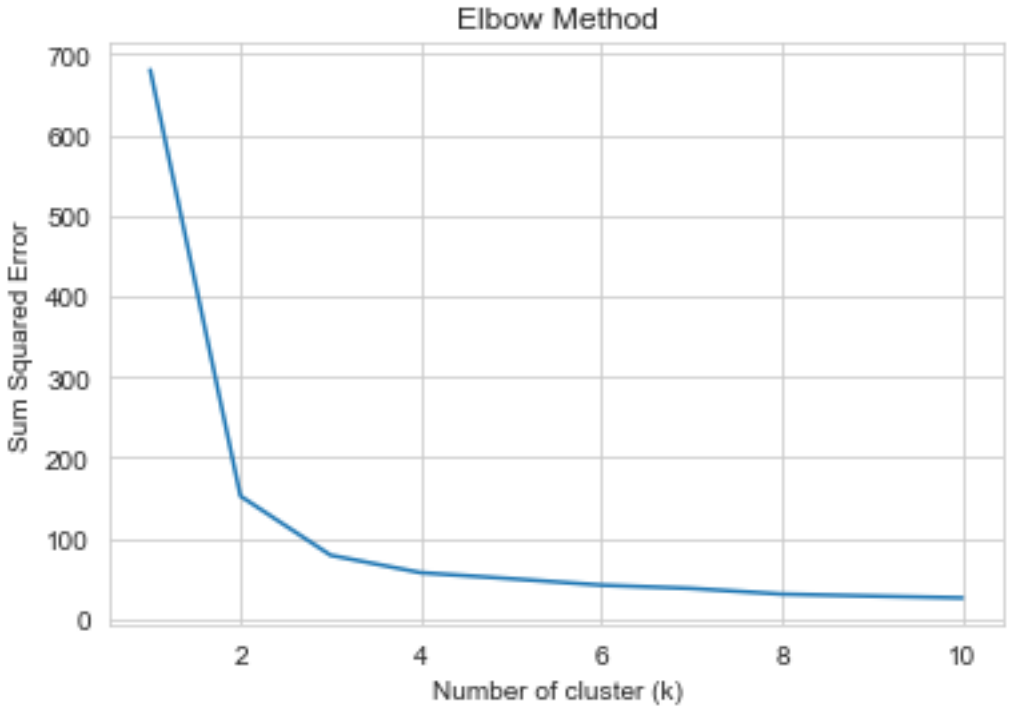
```
#importing the necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.cm as cm
from sklearn.cluster import KMeans

#importing the dataset
df=pd.read_csv('iris.csv')
print(df.head())

#extracting the features
x=df.iloc[:,:-1].values

sse=[]
for k in  range(1,11):
    km=KMeans(n_clusters=k,random_state=2)
    km.fit(x)
    sse.append(km.inertia_)
```

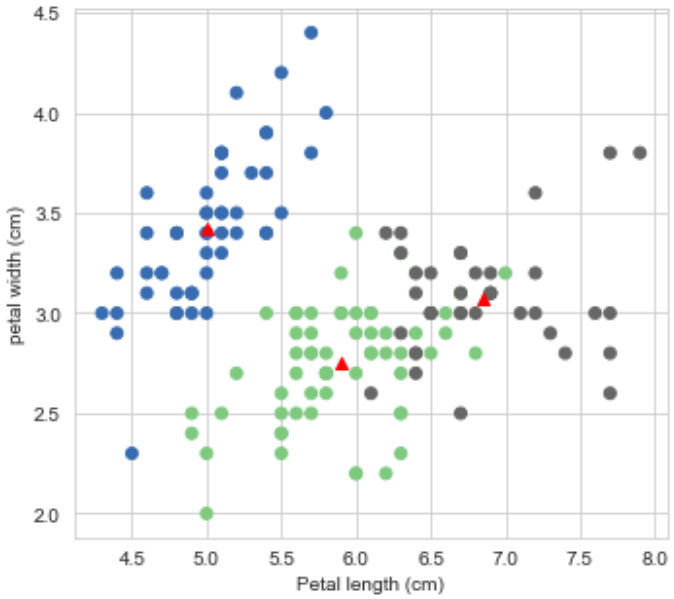
```
sns.set_style('whitegrid')
g=sns.lineplot(x=range(1,11),y=sse)

g.set(xlabel="Number of cluster (k)",
      ylabel="Sum Squared Error",
      title='Elbow Method')
plt.show()

kmeans=KMeans(n_clusters=3,random_state=2)
kmeans.fit(x)

print(kmeans.cluster_centers_)

pred=kmeans.fit_predict(x)
print(pred)
```

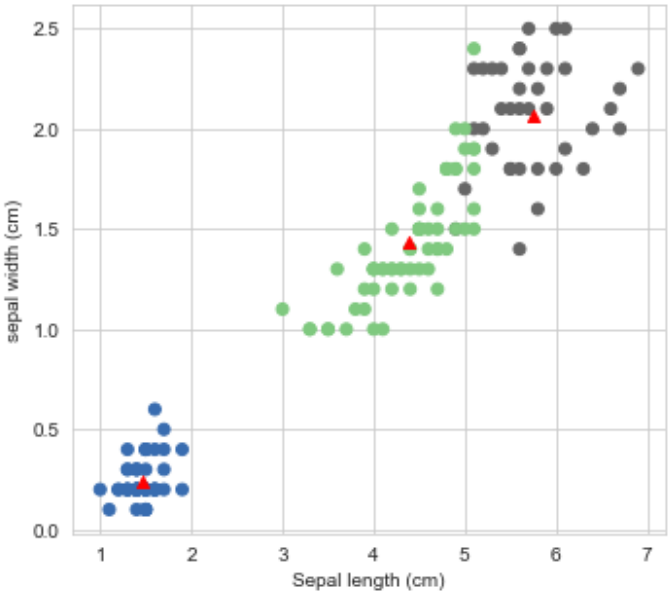
```
#plotting the cluster centers with datapoints
plt.figure(figsize=(12,5))
plt.subplot(1,2,1)
plt.scatter(x[:,0],x[:,1],c=pred,cmap=cm.Accent)
plt.grid(True)
for center in kmeans.cluster_centers_:
    center=center[:2]
    plt.scatter(center[0],center[1],marker='^',c='red')
plt.xlabel('Petal length (cm)')
plt.ylabel('petal width (cm)')

plt.subplot(1,2,2)
plt.scatter(x[:,2],x[:,3],c=pred,cmap=cm.Accent)
plt.grid(True)
for center in kmeans.cluster_centers_:
    center=center[2:4]
    plt.scatter(center[0],center[1],marker='^',c='red')
plt.xlabel('Sepal length (cm)')
plt.ylabel('sepal width (cm)')
plt.show()
```

Outputs:

```
   sepal_length  sepal_width  petal_length  petal_width species
0           5.1          3.5           1.4          0.2  setosa
1           4.9          3.0           1.4          0.2  setosa
2           4.7          3.2           1.3          0.2  setosa
3           4.6          3.1           1.5          0.2  setosa
4           5.0          3.6           1.4          0.2  setosa
[[5.9016129  2.7483871  4.39354839 1.43387097]
 [5.006      3.418      1.464      0.244     ]
 [6.85       3.07368421 5.74210526 2.07105263]]
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 2 2 2 2 0 2 2 2 2
 2 2 0 0 2 2 2 2 0 2 0 2 0 2 2 0 0 2 2 2 2 2 0 2 2 2 2 0 2 2 2 0 2 2 2 0 2
 2 0]
```







Observations:

- ✓ The elbow method was chosen to choose the optimal k value which gives us the optimal number of clusters and it is found to be 3 clusters.
- ✓ The clusters are visualized using the scatter plots and we can identify three different clusters with their centroids marked with red.

Conclusion:

Elbow method is chosen to pick up the optimal number of clusters because the choice of K value is very important as K means clustering is computationally intensive.