ex1

April 18, 2022

0.1 Exercise sheet 1

Cüneyt Erem 3277992 s6cuerem@uni-bonn.de

Nkeh Victor Ndiwago 3504121 s0vinkeh@uni-bonn.de

Paula Romero Jiménez 3320220 s0parome@uni-bonn.de

---

ex1)

---

1. Load the Iris dataset into your notebook from Scikit-Learn

[ ]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

#load iris dataset
from sklearn import datasets
iris = datasets.load_iris()
```

2. Report the descriptive statistics of the features of the iris dataset

```
a. Mean, Median,  Mode
b. Variance, MAD, Standard deviation
c. Quantiles, IQR
```

[ ]:
```python
# Dataset preview
print (iris.feature_names)     # Names of features or columns in iris dataset
print (iris.target_names)      # Names of targets in iris dataset
print (iris.data.shape)
```

```
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width
(cm)']
['setosa' 'versicolor' 'virginica']
(150, 4)
```

[ ]:
```python
# Convert the data into Dataframe
```

```
#iris_df = pd.DataFrame(data= np.c_[iris['data'], iris['target']],columns=␣
 ↪iris['feature_names'] + ['Species'])
iris_df = pd.DataFrame(iris.data, columns=iris.feature_names)
#print(iris_df)
print(iris_df.head())
```

```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
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
```

[ ]:
```
# This function gives us statistical information about the dataset
iris_df.describe()
```

[ ]:
```
       sepal length (cm)  sepal width (cm)  petal length (cm)  \
count         150.000000        150.000000         150.000000
mean            5.843333          3.057333           3.758000
std             0.828066          0.435866           1.765298
min             4.300000          2.000000           1.000000
25%             5.100000          2.800000           1.600000
50%             5.800000          3.000000           4.350000
75%             6.400000          3.300000           5.100000
max             7.900000          4.400000           6.900000

       petal width (cm)
count        150.000000
mean           1.199333
std            0.762238
min            0.100000
25%            0.300000
50%            1.300000
75%            1.800000
max            2.500000
```

a. Mean, Median, Mode

[ ]:
```
iris_df.mean()
```

[ ]:
```
sepal length (cm)    5.843333
sepal width (cm)     3.057333
petal length (cm)    3.758000
petal width (cm)     1.199333
dtype: float64
```

[ ]:
```
iris_df.median()
```

[ ]:
```
sepal length (cm)    5.80
sepal width (cm)     3.00
petal length (cm)    4.35
petal width (cm)     1.30
dtype: float64
```

[ ]:
```
iris_df.mode()
```

[ ]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.0	3.0	1.4	0.2
1	NaN	NaN	1.5	NaN

b. Variance, MAD, Standard deviation

[ ]:
```
iris_df.var()
```

[ ]:
```
sepal length (cm)    0.685694
sepal width (cm)     0.189979
petal length (cm)    3.116278
petal width (cm)     0.581006
dtype: float64
```

[ ]:
```
iris_df.mad()
```

[ ]:
```
sepal length (cm)    0.687556
sepal width (cm)     0.336782
petal length (cm)    1.562747
petal width (cm)     0.658133
dtype: float64
```

[ ]:
```
iris_df.std()
```

[ ]:
```
sepal length (cm)    0.828066
sepal width (cm)     0.435866
petal length (cm)    1.765298
petal width (cm)     0.762238
dtype: float64
```

c. Quantiles, IQR

[ ]:
```
iris_df.quantile(.75)
```

[ ]:
```
sepal length (cm)    6.4
sepal width (cm)     3.3
petal length (cm)    5.1
petal width (cm)     1.8
Name: 0.75, dtype: float64
```

[ ]:
```
iris_df.quantile(.5)
```

[ ]:
```
sepal length (cm)    5.80
sepal width (cm)     3.00
petal length (cm)    4.35
petal width (cm)     1.30
Name: 0.5, dtype: float64
```

[ ]:
```
iris_df.quantile(.25)
```

[ ]:
```
sepal length (cm)    5.1
sepal width (cm)     2.8
petal length (cm)    1.6
petal width (cm)     0.3
Name: 0.25, dtype: float64
```

[ ]:
```
IQR=iris_df.quantile(.75) - iris_df.quantile(.25)
print(IQR)
```

```
sepal length (cm)    1.3
sepal width (cm)     0.5
petal length (cm)    3.5
petal width (cm)     1.5
dtype: float64
```

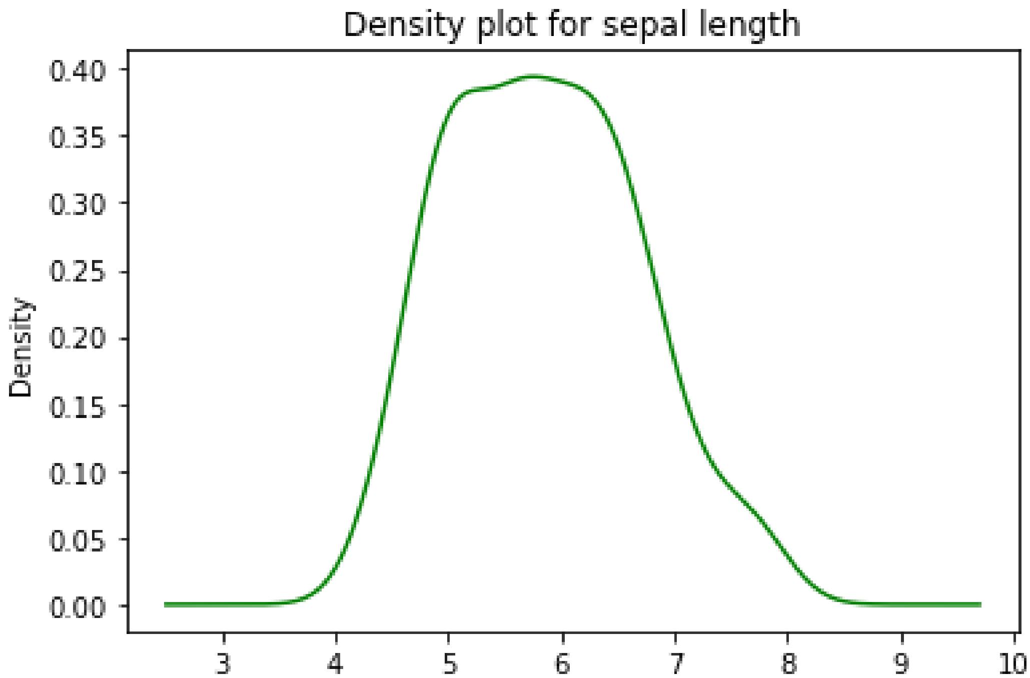
3. Plot a density plot for each of the variables. Interpret the plots.

[ ]:
```
iris_df.rename(columns = {'sepal length (cm)':'sepal_length', 'sepal width␣
↪(cm)':'sepal_width', 'petal length (cm)':'petal_length', 'petal width (cm)':
↪'petal_width' }, inplace = True)
```

[ ]:
```
list(iris_df)
```

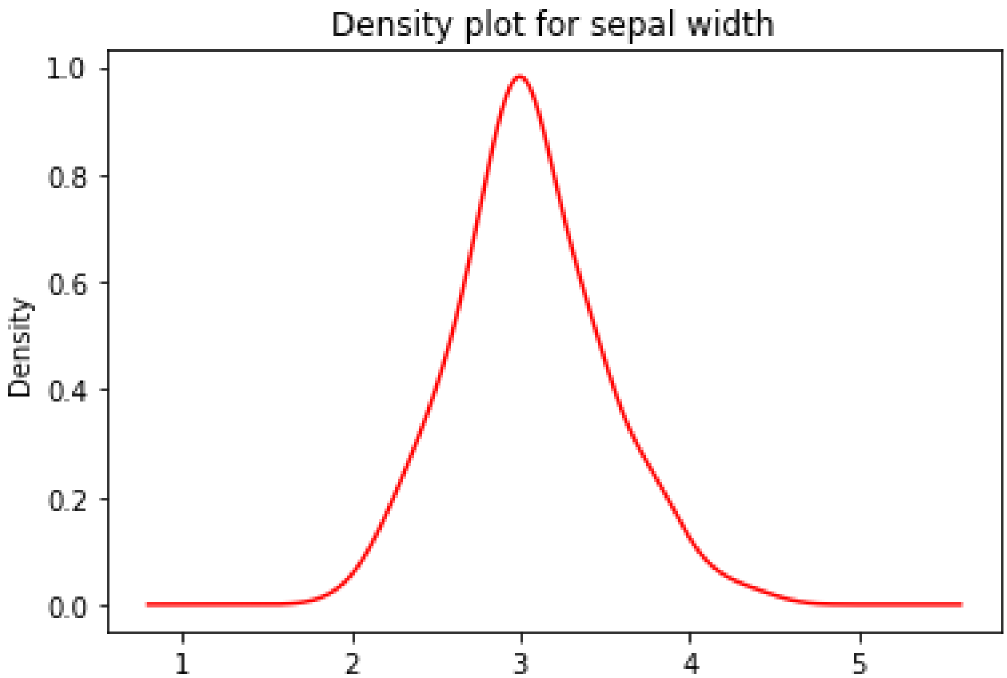
[ ]:
```
['sepal_length', 'sepal_width', 'petal_length', 'petal_width']
```

[ ]:
```
# Plot for the sepal length
iris_df.sepal_length.plot.density(color='green')
plt.title('Density plot for sepal length')
plt.show()
```



According to the plot, most of the sepals measure around 6 cm long, and if we compare that value with the ones obtained in our mean, median and mode it is correct.

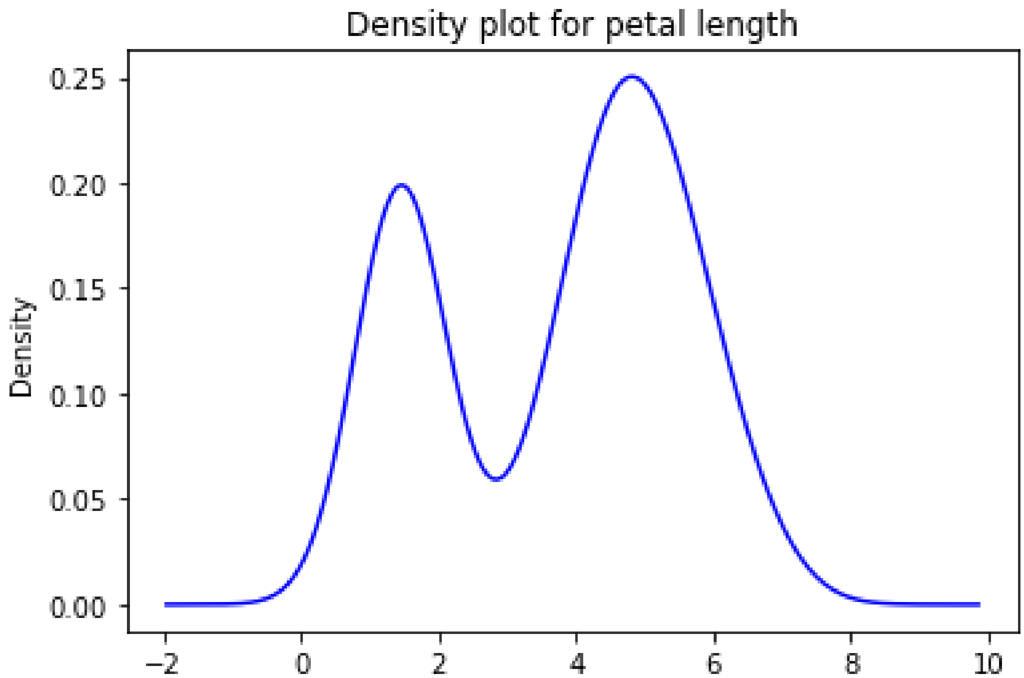
[ ]:
```
# Plot for the sepal width
iris_df.sepal_width.plot.density(color='red')
plt.title('Density plot for sepal width')
plt.show()
```



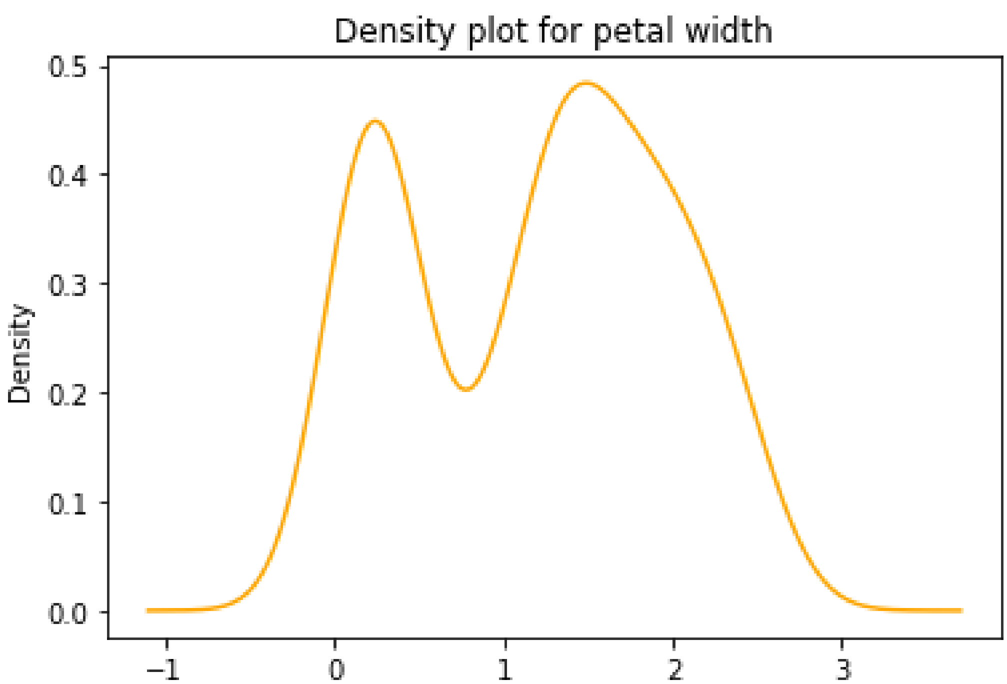
We can observe a clear predominance of 3 cm width, and our values of mean, median and mode are 3, 3.05 and 3.

So we could say in sepals the common points are in one same region, allowing then such a low variance, MAD and standard deviation.

[ ]:
```
# Plot for the petal length
iris_df.petal_length.plot.density(color='blue')
plt.title('Density plot for petal length')
plt.show()
```




```
[ ]: # Plot for the petal width
iris_df.petal_width.plot.density(color='orange')
plt.title('Density plot for petal width')
plt.show()
```



In the case of the petals, we have two peaks of density. Therefore, our values of mean, median and mode are not going to coincide with the ones in the plot. Moreover, in petals we observe a higher variance and standard deviation, which is normal taking into account the difference between the common data points.

4. Create a violin plot for the sepal width feature for each class. What can be seen from the plots?

[ ]:
```
iris_df2 = pd.DataFrame(data= np.c_[iris['data'], iris['target']],columns=␣
↪iris['feature_names'] + ['Species'])

iris_df2['Species'].value_counts()
```

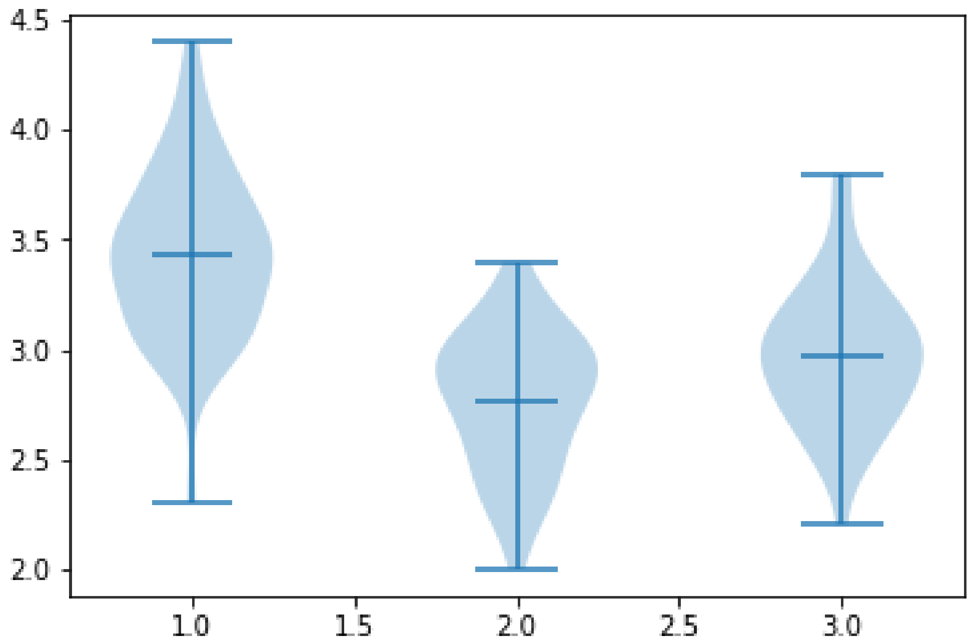
[ ]:
```
0.0    50
1.0    50
2.0    50
Name: Species, dtype: int64
```

[ ]:
```
setosa = iris_df.iloc[0:49,1]
versicolor = iris_df.iloc[50:99,1]
virginica = iris_df.iloc[100:149,1]
sepal_w = [setosa, versicolor, virginica]
```

[ ]:
```
fig, ax = plt.subplots()

ax.violinplot(sepal_w, positions=None, vert=True, widths=0.5, showmeans=True,␣
↪showextrema=True, showmedians=False, points=100)
```

[ ]:
```
{'bodies': [<matplotlib.collections.PolyCollection at 0x7fb10f9cbfd0>,
  <matplotlib.collections.PolyCollection at 0x7fb10f9de2e0>,
  <matplotlib.collections.PolyCollection at 0x7fb10f9de5b0>],
 'cmeans': <matplotlib.collections.LineCollection at 0x7fb1706c3ca0>,
 'cmaxes': <matplotlib.collections.LineCollection at 0x7fb10f9dea00>,
 'cmins': <matplotlib.collections.LineCollection at 0x7fb10f9ded00>,
 'cbars': <matplotlib.collections.LineCollection at 0x7fb113f0a100>}
```



We observe the difference between the sepal's width of the three species. Setosa is going to have wider sepals than virginica, being versicolor's ones the smallest of the three.

---

ex2)

---

1) Load the banknote authentication dataset from the given data_banknote_authentication.csv file. How many rows and columns does the dataset contain?

[ ]:
```
import pandas as pd
#import os
path="data_banknote_authentication.csv"
dataset=pd.read_csv(path)
#print(dataset)
print(dataset.head())
```

```
   Variance  Skewness  Curtosis  Entropy  Class
0       NaN    8.6661   -2.8073 -0.44699    0.0
1   4.54590    8.1674   -2.4586 -1.46210    0.0
2   3.86600   -2.6383    1.9242      NaN    0.0
3   3.45660    9.5228   -4.0112 -3.59440    0.0
4   0.32924   -4.4552    4.5718      NaN    0.0
```

```
[ ]: print(f"there are {len(dataset.index)} rows and {len(dataset.columns)} columns␣
     ↪in the dataset")
```

```
there are 1382 rows and 5 columns in the dataset
```

2) Mention the different types of varaibles.Which types does your dataset contain There are two main types of variables :Categorical varaible and Numeric variable Categorical also known as quantitative is subdivided into Nominal(no ordering) and ordinal variable(involves order) while Numerical also known as quantitative variable is subdivided into discrete(takes particular values) and continuous variable(measured on a continous scale).The dataset contains continous variables.

3) Count the number of duplicate rows in the dataset.How can you remove the duplicate rows

---

```
[ ]: print(dataset.loc[dataset.duplicated(keep="first"),:])
     dataset.duplicated()
     print(f"There are {dataset.duplicated().sum()} duplicated rows")
```

```
      Variance  Skewness  Curtosis  Entropy  Class
190    0.92970   -3.7971   4.64290 -0.29570    0.0
195   -1.85840    7.8860  -1.66430 -1.83840    0.0
268    0.92970   -3.7971   4.64290 -0.29570    0.0
284   -1.30000   10.2678  -2.95300 -5.86380    0.0
300    0.32920   -4.4552   4.57180 -0.98880    0.0
315    0.32920   -4.4552   4.57180 -0.98880    0.0
345   -1.85840    7.8860  -1.66430 -1.83840    0.0
427   -1.30000   10.2678  -2.95300 -5.86380    0.0
436    0.37980    0.7098   0.75720 -0.44440    0.0
476    0.37980    0.7098   0.75720 -0.44440    0.0
615   -0.20620    9.2207  -3.70440 -6.81030    0.0
691    0.57060   -0.0248   1.24210 -0.56210    0.0
727   -2.64790   10.1374  -1.33100 -5.47070    0.0
1372       NaN    8.6661  -2.80730 -0.44699    0.0
1373   4.54590    8.1674  -2.45860 -1.46210    0.0
1374   3.86600   -2.6383   1.92420      NaN    0.0
1375   3.45660    9.5228  -4.01120 -3.59440    0.0
1376   0.32924   -4.4552   4.57180      NaN    0.0
1377   4.36840    9.6718  -3.96060 -3.16250    0.0
1378   3.59120    3.0129   0.72888  0.56421    0.0
1379   2.09220   -6.8100       NaN -0.60216    0.0
1380   3.20320    5.7588  -0.75345 -0.61251    0.0
1381   1.53560    9.1772  -2.27180 -0.73535    0.0
There are 23 duplicated rows
```

[ ]:
```python
drop= dataset.drop_duplicates()
#print(drop)
print(drop.head(), '\n')
print(drop.info())
```

```
   Variance  Skewness  Curtosis  Entropy  Class
0       NaN    8.6661   -2.8073 -0.44699    0.0
1   4.54590    8.1674   -2.4586 -1.46210    0.0
2   3.86600   -2.6383    1.9242      NaN    0.0
3   3.45660    9.5228   -4.0112 -3.59440    0.0
4   0.32924   -4.4552    4.5718      NaN    0.0

<class 'pandas.core.frame.DataFrame'>
Int64Index: 1359 entries, 0 to 1371
Data columns (total 5 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Variance  1217 non-null   float64
 1   Skewness  1224 non-null   float64
 2   Curtosis  1229 non-null   float64
 3   Entropy   1215 non-null   float64
 4   Class     1224 non-null   float64
dtypes: float64(5)
memory usage: 63.7 KB
None
```

4) Count the number of missing values in the dataset

[ ]:
```python
missing_values= dataset.isna().sum().sum()
print(f"There are {missing_values} missing values in the dataset.")
```

```
There are 690 missing values in the dataset.
```

5) How an you deal with missing values in a dataset? implement one of the possible methods deleting the rows with a null value for a particular feature replacing the null value with mean meadian or mode assinging a unique category predicting the missing value using algorithms which support missing values

[ ]:
```python
print('before: ', dataset.isna().sum(), '\n')
dataset.fillna(dataset.mean().round(1), inplace=True)
print('after: ', dataset.isna().sum())
```

```
before:  Variance    143
Skewness    135
Curtosis    131
Entropy     146
Class       135
dtype: int64
```

```
after:  Variance    0
Skewness    0
Curtosis    0
Entropy     0
Class       0
dtype: int64
```

---

ex3)

---

1) Load the dataset from the given dataset.csv file

```
[ ]: import pandas as pd
     import seaborn as sns
```

```
[ ]: dataset2 = pd.read_csv("dataset.csv")
     print(dataset2.head())
```

```
   Unnamed: 0  feature_1  feature_2  feature_3  feature_4
0           0   0.286672   0.006711   0.178739  -4.997212
1           1   0.230586   0.013423   0.351505  -4.297285
2           2   0.074979   0.020134   0.879812  -3.884994
3           3   0.187541   0.026846   0.226149  -3.590439
4           4   0.422490   0.033557   0.424136  -3.360375
```

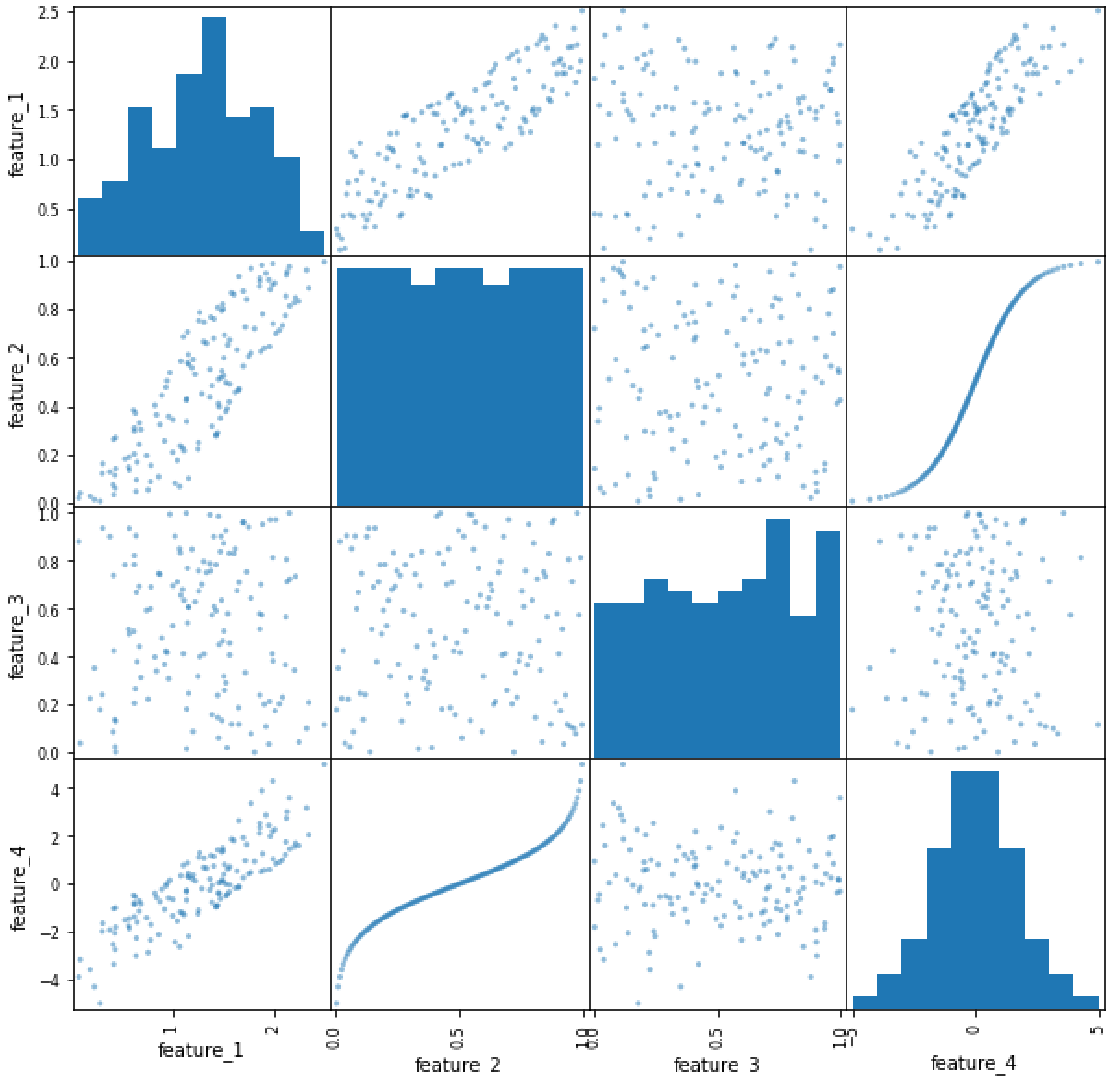
---

2) Plot the scatterplot matrix for the given dataset. What can be seen in the scatterplot matrix?

```
[ ]: pd.plotting.scatter_matrix(dataset2.iloc[:,1:], figsize=(10,10))
```

```
[ ]: array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7fb10f187e20>,
             <matplotlib.axes._subplots.AxesSubplot object at 0x7fb10f12d3d0>,
             <matplotlib.axes._subplots.AxesSubplot object at 0x7fb10fa14790>,
             <matplotlib.axes._subplots.AxesSubplot object at 0x7fb10fa5dd30>],
            [<matplotlib.axes._subplots.AxesSubplot object at 0x7fb10fabbe80>,
             <matplotlib.axes._subplots.AxesSubplot object at 0x7fb1706cb5b0>,
             <matplotlib.axes._subplots.AxesSubplot object at 0x7fb10fae0df0>,
             <matplotlib.axes._subplots.AxesSubplot object at 0x7fb111b7d970>],
            [<matplotlib.axes._subplots.AxesSubplot object at 0x7fb10fa58400>,
             <matplotlib.axes._subplots.AxesSubplot object at 0x7fb10fa970a0>,
             <matplotlib.axes._subplots.AxesSubplot object at 0x7fb10f0f1c40>,
             <matplotlib.axes._subplots.AxesSubplot object at 0x7fb10f0a64f0>],
            [<matplotlib.axes._subplots.AxesSubplot object at 0x7fb10f0d0d30>,
             <matplotlib.axes._subplots.AxesSubplot object at 0x7fb10f0855b0>,
             <matplotlib.axes._subplots.AxesSubplot object at 0x7fb10f031df0>,
             <matplotlib.axes._subplots.AxesSubplot object at 0x7fb10efe3670>]],
```

```
dtype=object)
```



scatter_matrix shows correlation between different features. feature 1-2 and 1-4 have positive weak correlation, 1-3 has no correlation, 2-4 has positive strong correlation between them.

---

3) Which correlation would suit the comparison of feature_1 and feature_3? Calculate the relevant correlation coefficient for the 2 features

[ ]:
```
dataset_corr = dataset2[['feature_1','feature_3']]

print('pearson: \n', dataset_corr.corr(method='pearson'), '\n')
print('kendall: \n', dataset_corr.corr(method='kendall'), '\n')
print('spearman: \n', dataset_corr.corr(method='spearman'), '\n')
```

```
pearson:
           feature_1  feature_3
feature_1   1.000000  -0.004628
feature_3  -0.004628   1.000000

kendall:
           feature_1  feature_3
feature_1   1.000000  -0.006251
feature_3  -0.006251   1.000000

spearman:
           feature_1  feature_3
feature_1    1.00000   -0.01755
feature_3   -0.01755    1.00000
```

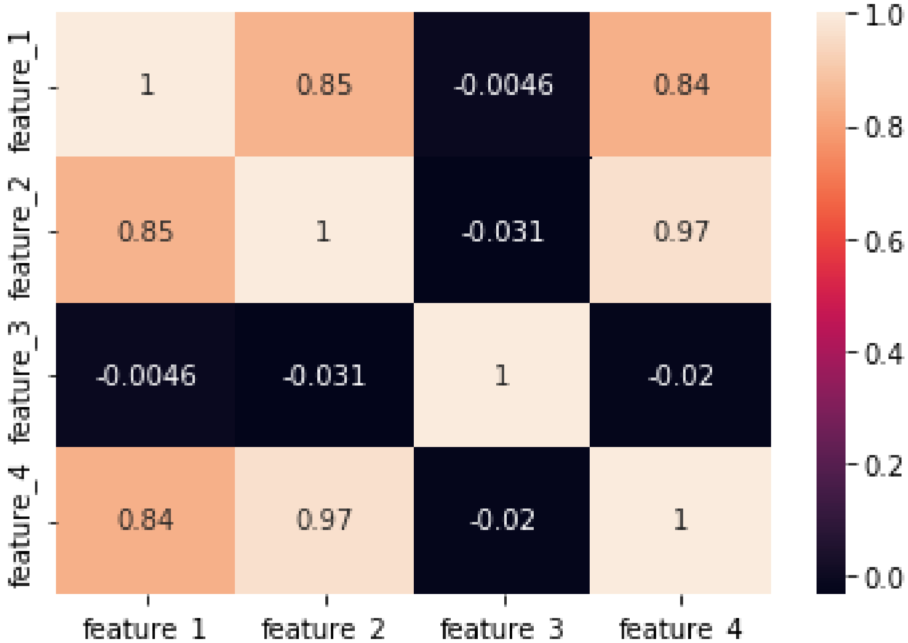
there is no obvious correlation between them, and lowest negative correlation is pearson

---

4) Plot the correlation heatmap of the entire dataset

[ ]:
```
sns.heatmap(dataset2.iloc[:,1:].corr(method='pearson'), xticklabels=dataset2.
↪iloc[:,1:].columns, yticklabels=dataset2.iloc[:,1:].columns, annot=True)
```

[ ]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb10efacb50>



```
[ ]:
```