Biomedical Data Science & AI

Exercise sheet 1 - Introduction

Submitted to:

Mohamed Aborageh : s0moabor@uni-bonn.de

Vinay Srinivas Bharadhwaj: s0vibhar@uni-bonn.de

Yasamin Salimi: yasisali@uni-bonn.de

Exercise 1 - Descriptive Statistics & Data Visualization (total: 9 points)

1. Load the Iris dataset into your notebook from Scikit-Learn. (2 points)

In [ ]:
```
#Importing Scikit Library

import sklearn
from sklearn import datasets
```

In [ ]:
```
#calling all datasets
dir(datasets)
```

Out[ ]:
```
['__all__',
 '__builtins__',
 '__cached__',
 '__doc__',
 '__file__',
 '__loader__',
 '__name__',
 '__package__',
 '__path__',
 '__spec__',
 '_base',
 '_california_housing',
 '_covtype',
 '_kddcup99',
 '_lfw',
 '_olivetti_faces',
 '_openml',
 '_rcv1',
 '_samples_generator',
 '_species_distributions',
 '_svmlight_format_fast',
 '_svmlight_format_io',
 '_twenty_newsgroups',
 'clear_data_home',
 'dump_svmlight_file',
 'fetch_20newsgroups',
 'fetch_20newsgroups_vectorized',
 'fetch_california_housing',
 'fetch_covtype',
 'fetch_kddcup99',
 'fetch_lfw_pairs',
 'fetch_lfw_people',
 'fetch_olivetti_faces',
```

```
'fetch_openml',
'fetch_rcv1',
'fetch_species_distributions',
'get_data_home',
'load_boston',
'load_breast_cancer',
'load_diabetes',
'load_digits',
'load_files',
'load_iris',
'load_linnerud',
'load_sample_image',
'load_sample_images',
'load_svmlight_file',
'load_svmlight_files',
'load_wine',
'make_biclusters',
'make_blobs',
'make_checkerboard',
'make_circles',
'make_classification',
'make_friedman1',
'make_friedman2',
'make_friedman3',
'make_gaussian_quantiles',
'make_hastie_10_2',
'make_low_rank_matrix',
'make_moons',
'make_multilabel_classification',
'make_regression',
'make_s_curve',
'make_sparse_coded_signal',
'make_sparse_spd_matrix',
'make_sparse_uncorrelated',
'make_spd_matrix',
'make_swiss_roll']
```

In [ ]:

```
#Loading Iris dataset and Iris-data

iris=datasets.load_iris()
print("Feature Names:")
print("\n")
print(iris.feature_names)
print("\n")
print("Iris Data:")
print("\n")
print(iris.data)
print("\n")
print("Iris Target:")
print("\n")
print(iris.target)
print("\n")
print("Iris Target Names:")
print("\n")
print(iris.target_names)
print("\n")
print("Iris Dataset Description:")
print("\n")
print(iris.DESCR)
```

```
Feature Names:


['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']


Iris Data:
```

```
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]
 [5.4 3.9 1.7 0.4]
 [4.6 3.4 1.4 0.3]
 [5.  3.4 1.5 0.2]
 [4.4 2.9 1.4 0.2]
 [4.9 3.1 1.5 0.1]
 [5.4 3.7 1.5 0.2]
 [4.8 3.4 1.6 0.2]
 [4.8 3.  1.4 0.1]
 [4.3 3.  1.1 0.1]
 [5.8 4.  1.2 0.2]
 [5.7 4.4 1.5 0.4]
 [5.4 3.9 1.3 0.4]
 [5.1 3.5 1.4 0.3]
 [5.7 3.8 1.7 0.3]
 [5.1 3.8 1.5 0.3]
 [5.4 3.4 1.7 0.2]
 [5.1 3.7 1.5 0.4]
 [4.6 3.6 1.  0.2]
 [5.1 3.3 1.7 0.5]
 [4.8 3.4 1.9 0.2]
 [5.  3.  1.6 0.2]
 [5.  3.4 1.6 0.4]
 [5.2 3.5 1.5 0.2]
 [5.2 3.4 1.4 0.2]
 [4.7 3.2 1.6 0.2]
 [4.8 3.1 1.6 0.2]
 [5.4 3.4 1.5 0.4]
 [5.2 4.1 1.5 0.1]
 [5.5 4.2 1.4 0.2]
 [4.9 3.1 1.5 0.2]
 [5.  3.2 1.2 0.2]
 [5.5 3.5 1.3 0.2]
 [4.9 3.6 1.4 0.1]
 [4.4 3.  1.3 0.2]
 [5.1 3.4 1.5 0.2]
 [5.  3.5 1.3 0.3]
 [4.5 2.3 1.3 0.3]
 [4.4 3.2 1.3 0.2]
 [5.  3.5 1.6 0.6]
 [5.1 3.8 1.9 0.4]
 [4.8 3.  1.4 0.3]
 [5.1 3.8 1.6 0.2]
 [4.6 3.2 1.4 0.2]
 [5.3 3.7 1.5 0.2]
 [5.  3.3 1.4 0.2]
 [7.  3.2 4.7 1.4]
 [6.4 3.2 4.5 1.5]
 [6.9 3.1 4.9 1.5]
 [5.5 2.3 4.  1.3]
 [6.5 2.8 4.6 1.5]
 [5.7 2.8 4.5 1.3]
 [6.3 3.3 4.7 1.6]
 [4.9 2.4 3.3 1. ]
 [6.6 2.9 4.6 1.3]
 [5.2 2.7 3.9 1.4]
 [5.  2.  3.5 1. ]
 [5.9 3.  4.2 1.5]
 [6.  2.2 4.  1. ]
 [6.1 2.9 4.7 1.4]
 [5.6 2.9 3.6 1.3]
 [6.7 3.1 4.4 1.4]
 [5.6 3.  4.5 1.5]
 [5.8 2.7 4.1 1. ]
```

```
[6.2 2.2 4.5 1.5]
[5.6 2.5 3.9 1.1]
[5.9 3.2 4.8 1.8]
[6.1 2.8 4.  1.3]
[6.3 2.5 4.9 1.5]
[6.1 2.8 4.7 1.2]
[6.4 2.9 4.3 1.3]
[6.6 3.  4.4 1.4]
[6.8 2.8 4.8 1.4]
[6.7 3.  5.  1.7]
[6.  2.9 4.5 1.5]
[5.7 2.6 3.5 1. ]
[5.5 2.4 3.8 1.1]
[5.5 2.4 3.7 1. ]
[5.8 2.7 3.9 1.2]
[6.  2.7 5.1 1.6]
[5.4 3.  4.5 1.5]
[6.  3.4 4.5 1.6]
[6.7 3.1 4.7 1.5]
[6.3 2.3 4.4 1.3]
[5.6 3.  4.1 1.3]
[5.5 2.5 4.  1.3]
[5.5 2.6 4.4 1.2]
[6.1 3.  4.6 1.4]
[5.8 2.6 4.  1.2]
[5.  2.3 3.3 1. ]
[5.6 2.7 4.2 1.3]
[5.7 3.  4.2 1.2]
[5.7 2.9 4.2 1.3]
[6.2 2.9 4.3 1.3]
[5.1 2.5 3.  1.1]
[5.7 2.8 4.1 1.3]
[6.3 3.3 6.  2.5]
[5.8 2.7 5.1 1.9]
[7.1 3.  5.9 2.1]
[6.3 2.9 5.6 1.8]
[6.5 3.  5.8 2.2]
[7.6 3.  6.6 2.1]
[4.9 2.5 4.5 1.7]
[7.3 2.9 6.3 1.8]
[6.7 2.5 5.8 1.8]
[7.2 3.6 6.1 2.5]
[6.5 3.2 5.1 2. ]
[6.4 2.7 5.3 1.9]
[6.8 3.  5.5 2.1]
[5.7 2.5 5.  2. ]
[5.8 2.8 5.1 2.4]
[6.4 3.2 5.3 2.3]
[6.5 3.  5.5 1.8]
[7.7 3.8 6.7 2.2]
[7.7 2.6 6.9 2.3]
[6.  2.2 5.  1.5]
[6.9 3.2 5.7 2.3]
[5.6 2.8 4.9 2. ]
[7.7 2.8 6.7 2. ]
[6.3 2.7 4.9 1.8]
[6.7 3.3 5.7 2.1]
[7.2 3.2 6.  1.8]
[6.2 2.8 4.8 1.8]
[6.1 3.  4.9 1.8]
[6.4 2.8 5.6 2.1]
[7.2 3.  5.8 1.6]
[7.4 2.8 6.1 1.9]
[7.9 3.8 6.4 2. ]
[6.4 2.8 5.6 2.2]
[6.3 2.8 5.1 1.5]
[6.1 2.6 5.6 1.4]
[7.7 3.  6.1 2.3]
[6.3 3.4 5.6 2.4]
```

```
 [6.4 3.1 5.5 1.8]
 [6.  3.  4.8 1.8]
 [6.9 3.1 5.4 2.1]
 [6.7 3.1 5.6 2.4]
 [6.9 3.1 5.1 2.3]
 [5.8 2.7 5.1 1.9]
 [6.8 3.2 5.9 2.3]
 [6.7 3.3 5.7 2.5]
 [6.7 3.  5.2 2.3]
 [6.3 2.5 5.  1.9]
 [6.5 3.  5.2 2. ]
 [6.2 3.4 5.4 2.3]
 [5.9 3.  5.1 1.8]]


Iris Target:


[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 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 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]


Iris Target Names:


['setosa' 'versicolor' 'virginica']


Iris Dataset Description:


.. _iris_dataset:

Iris plants dataset
--------------------

**Data Set Characteristics:**

    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, predictive attributes and the class
    :Attribute Information:
        - sepal length in cm
        - sepal width in cm
        - petal length in cm
        - petal width in cm
        - class:
                - Iris-Setosa
                - Iris-Versicolour
                - Iris-Virginica

    :Summary Statistics:

    ============== ==== ==== ======= ===== ====================
                    Min  Max   Mean    SD   Class Correlation
    ============== ==== ==== ======= ===== ====================
    sepal length:   4.3  7.9   5.84   0.83    0.7826
    sepal width:    2.0  4.4   3.05   0.43   -0.4194
    petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)
    petal width:    0.1  2.5   1.20   0.76    0.9565  (high!)
    ============== ==== ==== ======= ===== ====================

    :Missing Attribute Values: None
    :Class Distribution: 33.3% for each of 3 classes.
    :Creator: R.A. Fisher
    :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
    :Date: July, 1988
```

```
The famous Iris database, first used by Sir R.A. Fisher. The dataset is taken
from Fisher's paper. Note that it's the same as in R, but not as in the UCI
Machine Learning Repository, which has two wrong data points.

This is perhaps the best known database to be found in the
pattern recognition literature.  Fisher's paper is a classic in the field and
is referenced frequently to this day.  (See Duda & Hart, for example.)  The
data set contains 3 classes of 50 instances each, where each class refers to a
type of iris plant.  One class is linearly separable from the other 2; the
latter are NOT linearly separable from each other.

.. topic:: References

   - Fisher, R.A. "The use of multiple measurements in taxonomic problems"
     Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to
     Mathematical Statistics" (John Wiley, NY, 1950).
   - Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Scene Analysis.
     (Q327.D83) John Wiley & Sons.  ISBN 0-471-22361-1.  See page 218.
   - Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New System
     Structure and Classification Rule for Recognition in Partially Exposed
     Environments".  IEEE Transactions on Pattern Analysis and Machine
     Intelligence, Vol. PAMI-2, No. 1, 67-71.
   - Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule".  IEEE Transactions
     on Information Theory, May 1972, 431-433.
   - See also: 1988 MLC Proceedings, 54-64.  Cheeseman et al"s AUTOCLASS II
     conceptual clustering system finds 3 classes in the data.
   - Many, many more ...
```

1. Report the descriptive statistics of the features of the iris dataset. (3 points)

In [ ]:
```
#Importing libraries

import pandas as pd
import numpy as np
import seaborn as sns
import scipy.stats as stats
from scipy import stats
from scipy.stats import iqr
from collections import Counter
```

a. Mean, Median, Mode

In [ ]:
```
iris=sns.load_dataset("iris")
print("The column names are:",iris.columns)
```

```
The column names are: Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_w
idth',
       'species'],
      dtype='object')
```

In [ ]:
```
#Define Variable to calculate Total-Count, Mean, Median

#For Sepal Length:
Total_count_sl = iris["sepal_length"].sum()
mean_sl = iris["sepal_length"].mean()
median_sl = iris["sepal_length"].median()

#For Sepal Width:
Total_count_sw = iris["sepal_width"].sum()
mean_sw = iris["sepal_width"].mean()
median_sw = iris["sepal_width"].median()

#For Petal Length:
Total_count_pl = iris["petal_length"].sum()
mean_pl = iris["petal_length"].mean()
```

```
median_pl = iris["petal_length"].median()

#For Petal Width:
Total_count_pw = iris["petal_width"].sum()
mean_pw = iris["petal_width"].mean()
median_pw = iris["petal_width"].median()

#define Mode Function:
def Mode(column):
    M = Counter(column)
    return [i for i, k in c.items() if k == c.most_common(1)[0][1]]

#Defining Variable for Mode:
SL = np.array(iris["sepal_length"])
SW = np.array(iris["sepal_width"])
PL = np.array(iris["petal_length"])
PW = np.array(iris["petal_width"])
```

In [ ]:

```
print("Sepal Length:")
print("Total count of Sepal Length:",round(Total_count_sl,4), "\n", "Mean of Sepal L
print("\n")
print("Sepal Width:")
print("Total count of Sepal Width:",round(Total_count_sw,4), "\n", "Mean of Sepal Wi
print("\n")
print("Petal Length:")
print("Total count of Petal Length:",round(Total_count_pl,4), "\n", "Mean of Petal L
print("\n")
print("Petal Width:")
print("Total count of Petal Width:",round(Total_count_pw,4), "\n", "Mean of Petal Wi
print("\n")
print("Mode - Iris Dataset Columns:")
print("Mode of Sepal Length:", stats.mode(SL))
print("Mode of Sepal Width:", stats.mode(SW))
print("Mode of Petal Length:", stats.mode(PL))
print("Mode of Petal Width:", stats.mode(PW))
```

```
Sepal Length:
Total count of Sepal Length: 876.5
 Mean of Sepal Length: 5.8433
 Median of Sepal Length: 5.8


Sepal Width:
Total count of Sepal Width: 458.6
 Mean of Sepal Width: 3.0573
 Median of Sepal Width: 3.0


Petal Length:
Total count of Petal Length: 563.7
 Mean of Petal Length: 3.758
 Median of Petal Length: 4.35


Petal Width:
Total count of Petal Width: 179.9
 Mean of Petal Width: 1.1993
 Median of Petal Width: 1.3


Mode - Iris Dataset Columns:
Mode of Sepal Length: ModeResult(mode=array([5.]), count=array([10]))
Mode of Sepal Width: ModeResult(mode=array([3.]), count=array([26]))
Mode of Petal Length: ModeResult(mode=array([1.4]), count=array([13]))
Mode of Petal Width: ModeResult(mode=array([0.2]), count=array([29]))
```

b. Variance, MAD, Standard deviation

In [ ]:
```
print("Variance - Iris Dataset Columns:")
print("Sepal Length:", round(np.var(SL),4))
print("Sepal Width:", round(np.var(SW),4))
print("Petal Length:", round(np.var(PL),4))
print("Petal Width:", round(np.var(PW),4), "\n")
print("Mean Absolute Deviation(MAD) - Iris Dataset Columns:")
print("Sepal Length:", round(stats.median_abs_deviation(SL),4), "\n", "Sepal Width:"
print("Standard Deviation - Iris Dataset Columns:")
print("Sepal Length:", round(SL.std(axis=0),4), "\n", "Sepal Width:", round(SW.std(a
```

```
Variance - Iris Dataset Columns:
Sepal Length: 0.6811
Sepal Width: 0.1887
Petal Length: 3.0955
Petal Width: 0.5771

Mean Absolute Deviation(MAD) - Iris Dataset Columns:
Sepal Length: 0.7
 Sepal Width: 0.3
 Petal Length: 1.25
 Petal Width: 0.7

Standard Deviation - Iris Dataset Columns:
Sepal Length: 0.8253
 Sepal Width: 0.4344
 Petal Length: 1.7594
 Petal Width: 0.7597
```

c. Quantiles, IQR

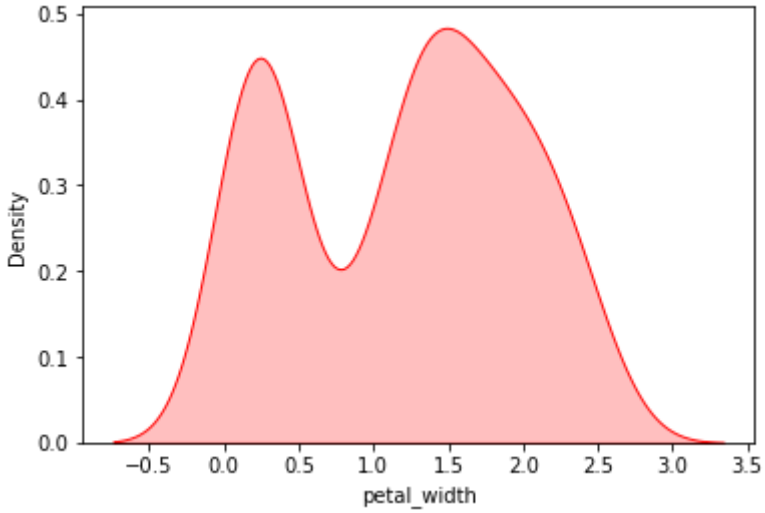
In [ ]:
```
print("Quantile - Iris Dataset Columns:", "\n")
print("Sepal Legth:")
print("Q1:", np.quantile(SL, .25))
print("Q2:", np.quantile(SL, .50))
print("Q3:", np.quantile(SL, .75))
print("100th Quantile:", np.quantile(SL, .1), "\n")
print("Sepal Width:")
print("Q1:", np.quantile(SW, .25))
print("Q2:", np.quantile(SW, .50))
print("Q3:", np.quantile(SW, .75))
print("100th Quantile:", np.quantile(SW, .1), "\n")
print("Petal Legth:")
print("Q1:", np.quantile(PL, .25))
print("Q2:", np.quantile(PL, .50))
print("Q3:", np.quantile(PL, .75))
print("100th Quantile:", np.quantile(PL, .1), "\n")
print("Petal Width:")
print("Q1:", np.quantile(PW, .25))
print("Q2:", np.quantile(PW, .50))
print("Q3:", np.quantile(PW, .75))
print("100th Quantile:", np.quantile(PW, .1), "\n\n")
print("Inter Quantile Range - Iris Dataset Columns:")
print("Sepal Legth:", iqr(SL))
print("Sepal Width:", iqr(SW))
print("Petal Legth:", iqr(PL))
print("Petal Width:", iqr(PW))
```

```
Quantile - Iris Dataset Columns:

Sepal Legth:
Q1: 5.1
Q2: 5.8
Q3: 6.4
100th Quantile: 4.8

Sepal Width:
```

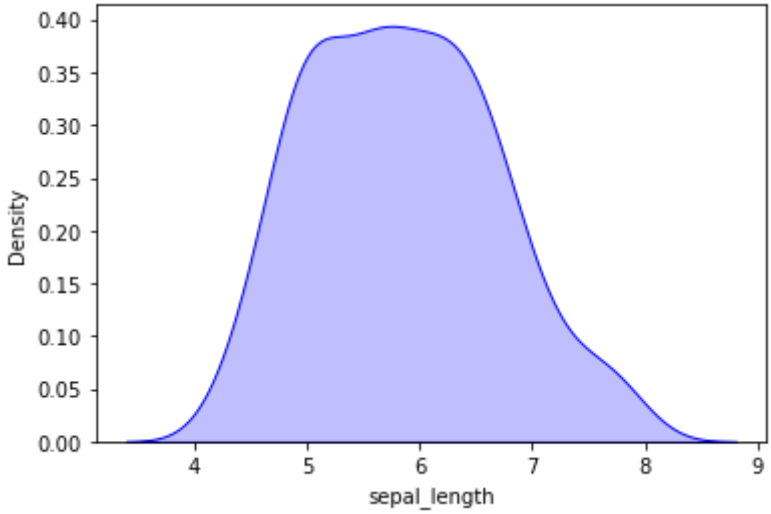
```
Q1: 2.8
Q2: 3.0
Q3: 3.3
100th Quantile: 2.5

Petal Legth:
Q1: 1.6
Q2: 4.35
Q3: 5.1
100th Quantile: 1.4

Petal Width:
Q1: 0.3
Q2: 1.3
Q3: 1.8
100th Quantile: 0.2


Inter Quantile Range - Iris Dataset Columns:
Sepal Legth: 1.3000000000000007
Sepal Width: 0.5
Petal Legth: 3.4999999999999996
Petal Width: 1.5
```

1. Plot a density plot for each of the variables. Interpret the plots. (2 points)

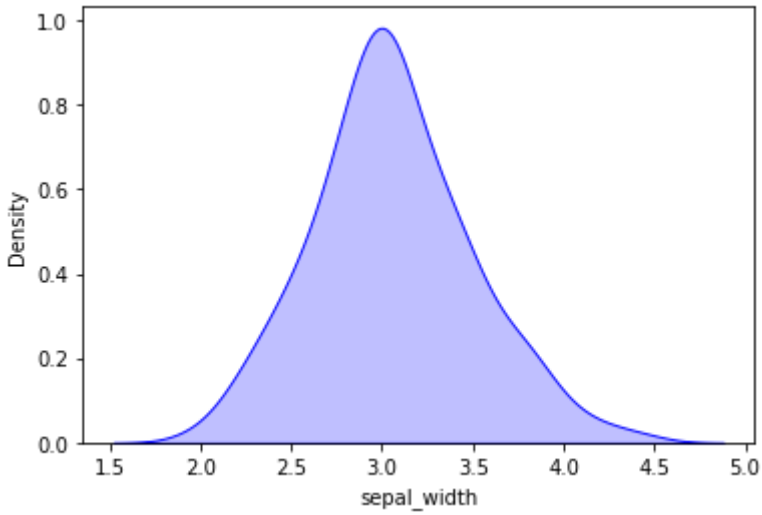
In [ ]:
```
kg=sns.kdeplot(iris['petal_width'], shade=True, color="r")
```



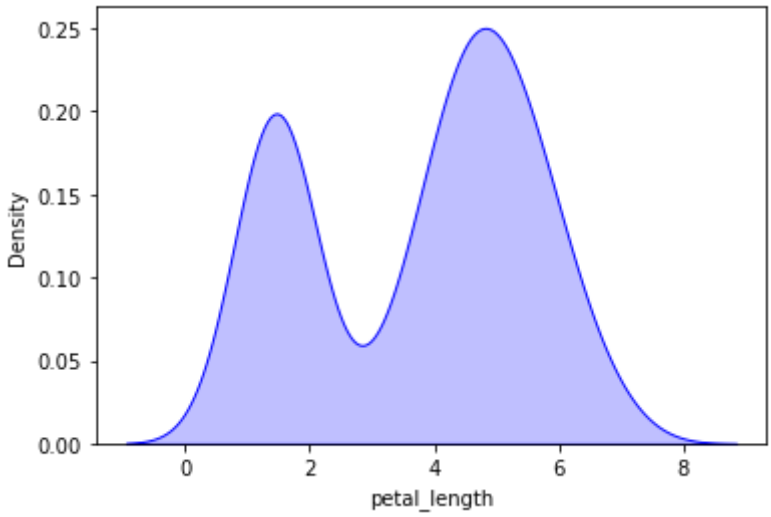
In [ ]:
```
kg=sns.kdeplot(iris['sepal_length'], shade=True,color='b')
```



In [ ]:
```
kg=sns.kdeplot(iris['sepal_width'], shade=True,color='b')
```



In [ ]:
```
kg=sns.kdeplot(iris['petal_length'], shade=True,color='b')
```



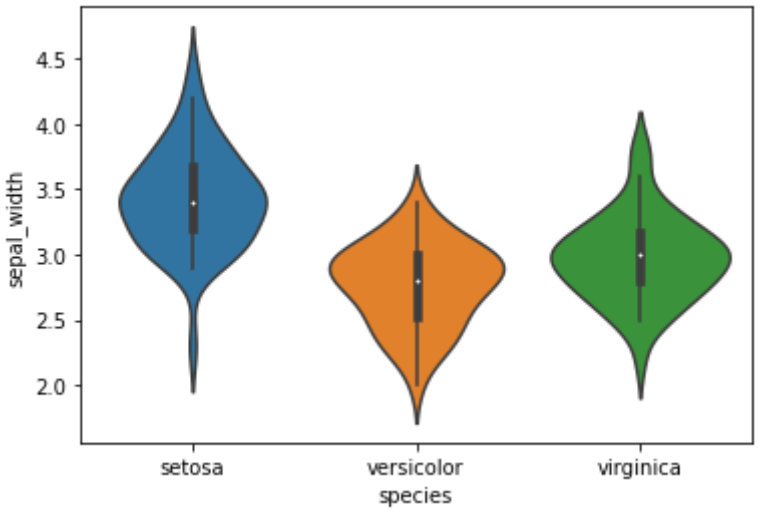
The Density Plot shows the smoothed distribution of the points along the numeric axis. Area under the curve in a range of values indicates the proportion of values in that range. The peaks of a Density Plot help display where values are concentrated over the interval

Several distribution shapes can be possible. In our data petal_lenght and petal_weidth plot is bimodal density plots but sepal_length,sepal_width shows normal plots

1. Create a violin plot for the sepal width feature for each class. What can be seen from the plots? (2 points)

In [ ]:
```
sns.violinplot(x="species", y="sepal_width", data=iris, size=6)
```

Out[ ]:
```
<AxesSubplot:xlabel='species', ylabel='sepal_width'>
```



1-the white dot represents the median 2-the thick gray bar in the center represents the interquartile range 3-the thin gray line represents the rest of the distribution, except for points that are determined to be “outliers” using a method that is a function of the interquartile range.

This violin plot shows the relationship of classes to sepal width. Wider sections of the violin plot represent a higher probability the skinnier sections represent a lower probability.

Exercise 2 - Data Pre-processing (total: 9 points)

1. Load the heart dataset from the given heart.csv file. How many rows and columns does the dataset contain? (2 points)

In [ ]:
```
#Loading data
import pandas as pd
def load_dataset():
    with open("heart.csv") as file:
        dataset = pd.read_csv(file)
        return dataset

data = load_dataset()
#print(data)

#Dataset information
print("Number of rows:", len(data.index))
print("Number of columns:", len(data.columns))
```

```
Number of rows: 312
Number of columns: 14
```

1. How many unique values does each column contain? (1 point)

In [ ]:
```
col_names = list(data.columns)
for col in col_names:
    print("Number of unique values in {} are".format(col),len(data[col].unique()))
```

```
Number of unique values in ï»¿age are 41
Number of unique values in sex are 3
Number of unique values in cp are 5
Number of unique values in trestbps are 50
Number of unique values in chol are 153
Number of unique values in fbs are 3
Number of unique values in restecg are 4
```

```
Number of unique values in thalach are 92
Number of unique values in exang are 2
Number of unique values in oldpeak are 41
Number of unique values in slope are 4
Number of unique values in ca are 6
Number of unique values in thal are 5
Number of unique values in target are 2
```

1. Count the number of duplicate rows in the dataset. How can you remove the duplicate rows? (2 points)

In [ ]:
```
d = data[data.duplicated()]
print("Number of duplicate rows are:", len(d))
```

```
Number of duplicate rows are: 9
```

Answer: We can remove duplicates by using pandas function: drop_duplicates()

In [ ]:
```
data.drop_duplicates()
```

Out[ ]:

	ï»¿age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	tar
0	63	1.0	3.0	145.0	233.0	1.0	0.0	150.0	0	2.3	0.0	0.0	1.0	
2	37	1.0	2.0	130.0	250.0	0.0	1.0	187.0	0	3.5	0.0	0.0	2.0	
3	41	0.0	1.0	130.0	204.0	0.0	0.0	172.0	0	1.4	2.0	0.0	2.0	
4	56	1.0	1.0	120.0	236.0	0.0	1.0	178.0	0	0.8	2.0	0.0	2.0	
5	57	0.0	0.0	120.0	354.0	0.0	1.0	163.0	1	0.6	2.0	0.0	2.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
307	57	0.0	0.0	140.0	241.0	0.0	1.0	123.0	1	0.2	1.0	0.0	3.0	
308	45	1.0	3.0	110.0	264.0	0.0	1.0	132.0	0	1.2	1.0	0.0	3.0	
309	68	1.0	0.0	144.0	193.0	1.0	1.0	141.0	0	3.4	1.0	2.0	3.0	
310	57	1.0	0.0	130.0	131.0	0.0	1.0	115.0	1	1.2	1.0	1.0	3.0	
311	57	0.0	1.0	130.0	236.0	0.0	0.0	174.0	0	0.0	1.0	1.0	2.0	

303 rows × 14 columns

1. Count the number of missing values in the dataset. (1 points)

In [ ]:
```
print("Total number of missing values in dataset:", data.isnull().sum().sum(), "\n")

print("Number of missing values per columns are: \n",data.isnull().sum())
```

```
Total number of missing values in dataset: 17

Number of missing values per columns are:
 ï»¿age      0
sex         3
cp          1
trestbps    1
chol        1
fbs         5
restecg     1
thalach     1
```

```
exang       0
oldpeak     1
slope       1
ca          1
thal        1
target      0
dtype: int64
```

1. How can you deal with missing values in your dataset? Implement one of the possible methods

Answer: We can deal with missing values by replacing them with:

- Mean
- Median
- Most frequenct value
- Zero

In [ ]:
```
#Replacing missing value with median

for col in col_names:
    median = data[col].median()
    data[col] = data[col].fillna(median)

data.isnull().sum() #Re-check of missing values
```

Out[ ]:
```
ï»¿age      0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

Exercise 3 - Correlation (total: 7 points)

1. Load the dataset from the given dataset.tsv file. (1 points)

In [ ]:
```
import pandas as pd
import csv

path = r'C:\Users\Shubhi Ambast\Desktop\dataset.tsv'
tsv_read = pd.read_csv(path, sep='\t')
print(tsv_read)
tsv_read.shape
```

```
     Unnamed: 0  feature_1  feature_2  feature_3  feature_4
0             0   0.006711   0.286672  -4.997212   0.178739
1             1   0.013423   0.230586  -4.297285   0.351505
2             2   0.020134   0.074979  -3.884994   0.879812
3             3   0.026846   0.187541  -3.590439   0.226149
4             4   0.033557   0.422490  -3.360375   0.424136
..          ...        ...        ...        ...        ...
143         143   0.966443   1.777039   3.360375   0.078540
```

```
144         144   0.973154   2.158306   3.590439   0.998950
145         145   0.979866   1.885833   3.884994   0.574135
146         146   0.986577   1.993843   4.297285   0.812999
147         147   0.993289   2.503264   4.997212   0.116215

[148 rows x 5 columns]
```

Out[ ]:
```
(148, 5)
```

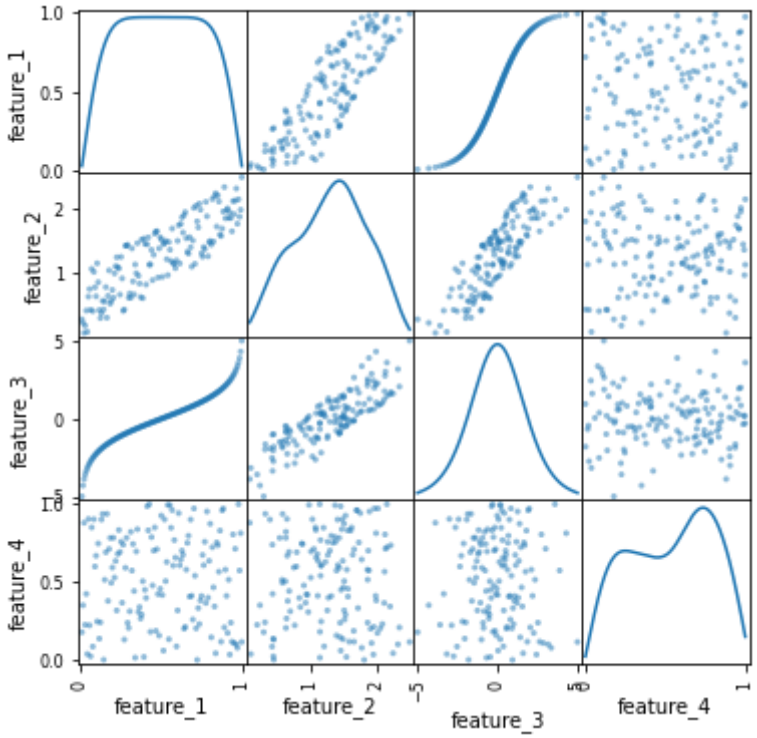
1. Plot the scatterplot matrix for the given dataset. What can be seen in the scatterplot matrix? (2 points)

In [ ]:
```
tsv = tsv_read.iloc[:,1:]
tsv.isna().any() #To check any NaN or NA is given dataset
```

Out[ ]:
```
feature_1    False
feature_2    False
feature_3    False
feature_4    False
dtype: bool
```

In [ ]:
```
pd.plotting.scatter_matrix(tsv, figsize = (6,6),diagonal="kde")
```

Out[ ]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x0000021F9A8504F0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000021F9AF43970>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000021F9AF71DC0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000021F9AFAA280>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000021F9AFD66D0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000021F9B001A60>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000021F9B001B50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000021F9B03A040>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000021F9B093850>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000021F9B0BDCA0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000021F9B0F9160>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000021F9B1235B0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000021F9B150A00>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000021F9B17BE50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000021F9B1B52E0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000021F9B1E0760>]],
      dtype=object)
```



1. Which correlation would suit the comparison of feature_1 and feature_3? Calculate the relevant correlation coefficient for the 2 features. (2 points)

For comaprison of feature_1 and feature_3 , suitable correlation would be Spearman rank correlation test since it does not carry any assumptions about the distribution of the data.

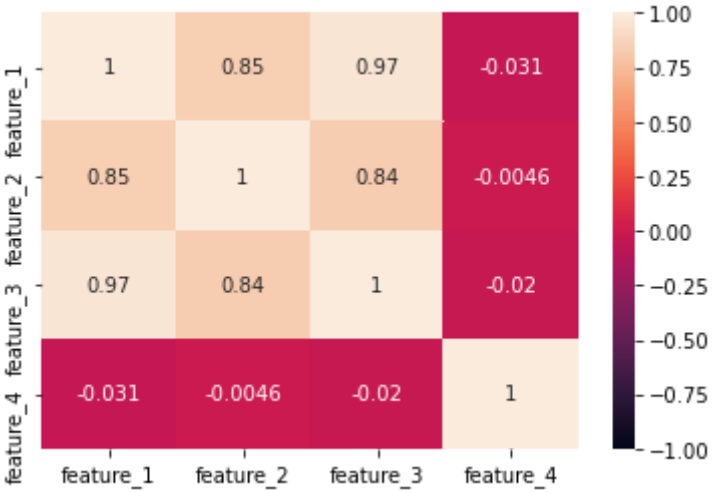
In [ ]:
```
#Three ways to calculate
data = tsv[['feature_1','feature_3']]

correlation_pearson = data.corr(method='pearson')
print('Pearson Correlation:')
print(correlation_pearson)

print('\n')

correlation_spearman= data.corr(method='spearman')
print('Spearman Correlation:')
print(correlation_spearman)

print('\n')

correlation_kendall= data.corr(method='kendall')
print('Kendall Correlation:')
print(correlation_kendall)
```

```
Pearson Correlation:
           feature_1  feature_3
feature_1   1.000000   0.968507
feature_3   0.968507   1.000000


Spearman Correlation:
           feature_1  feature_3
feature_1        1.0        1.0
feature_3        1.0        1.0


Kendall Correlation:
           feature_1  feature_3
feature_1        1.0        1.0
feature_3        1.0        1.0
```

1. Plot the correlation heatmap of the entire dataset. (2 points)

In [ ]:
```
import seaborn as sns

Var_Corr = tsv.corr()
# plot the heatmap and annotation on it
sns.heatmap(Var_Corr, xticklabels=Var_Corr.columns, yticklabels=Var_Corr.columns, an
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x21f9d67daf0>
```


feature_1
feature_2
feature_3
feature_4
1
0.85
0.97
-0.031
0.85
1
0.84
-0.0046
0.97
0.84
1
-0.02
-0.031
-0.0046
-0.02
1
feature_1
feature_2
feature_3
feature_4
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00