Maryam Mounad - Muhammad Humza Arain - Mehak Nadeem

Exercise 1- Descriptive Statistics & Data Visualization

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

In [5]:
```
from sklearn.datasets import load_iris
import pandas as pd
from scipy.stats import median_abs_deviation
import matplotlib.pyplot as plt
```

In [6]:
```
#1
# Loading the dataset
iris = load_iris()
df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
df.head()
```

Out[6]:

	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

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

In [ ]:
```
#a.
mean = df.mean()
print(f"mean of data:\n{mean}\n")

median = df.median()
print(f"median of data:\n{median}\n")

mode = df.mode()
print(f"mode of data:\n{mode}\n")
```

```
mean of data:
sepal length (cm)    5.843333
sepal width (cm)     3.057333
petal length (cm)    3.758000
petal width (cm)     1.199333
dtype: float64

median of data:
sepal length (cm)    5.80
sepal width (cm)     3.00
petal length (cm)    4.35
petal width (cm)     1.30
dtype: float64

mode of data:
   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
```

In [ ]:
```
#b.
variance= df.var()
print(f"variance of data:\n{variance}")

mad = median_abs_deviation(df)
print(f"Mean Absolute Deviation of data:\n{mad}\n")

sd= df.std()
print(f"standard deviation of data:\n{sd}")
```

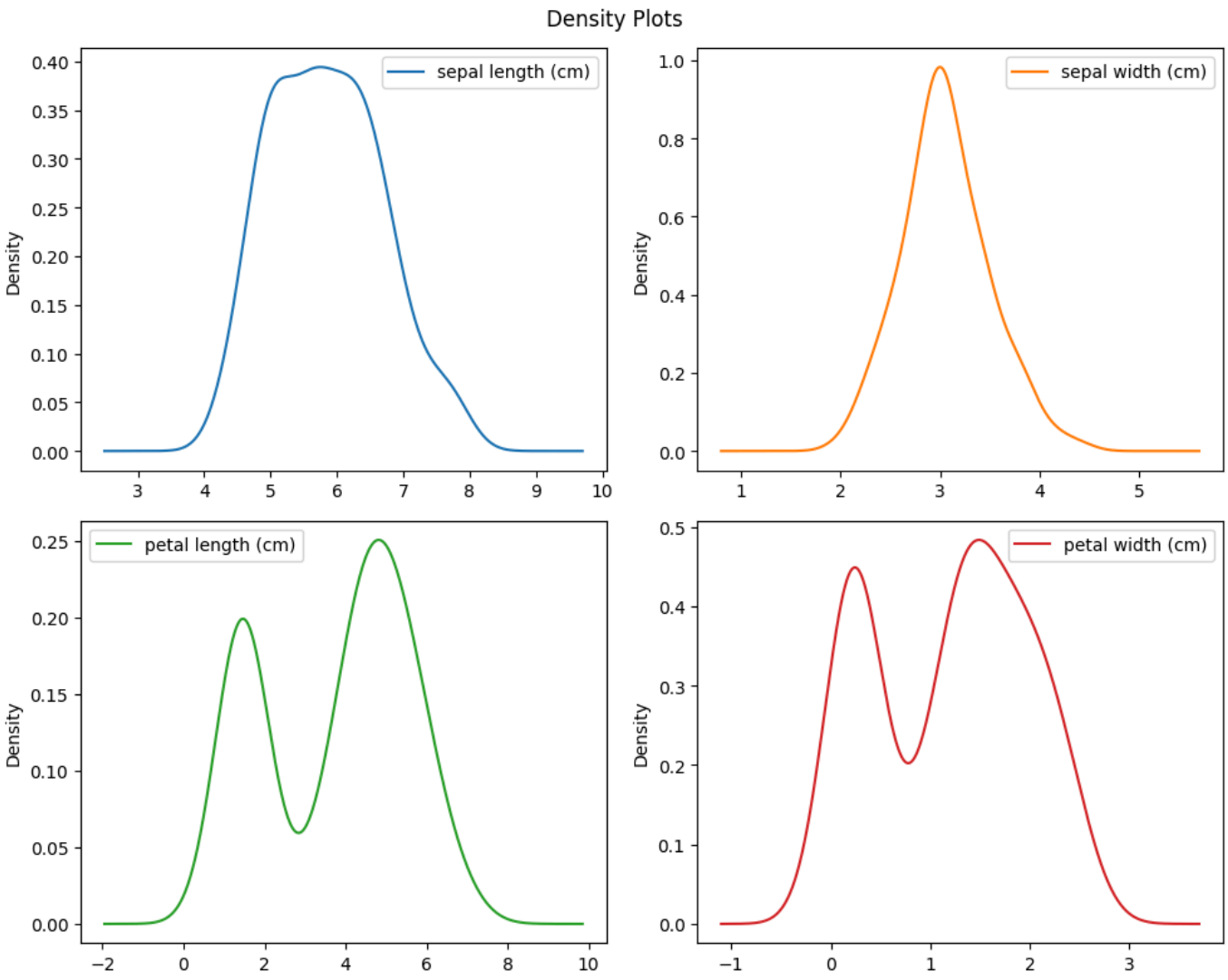
```
variance of data:
sepal length (cm)    0.685694
sepal width (cm)     0.189979
petal length (cm)    3.116278
petal width (cm)     0.581006
dtype: float64
Mean Absolute Deviation of data:
[0.7  0.3  1.25 0.7 ]

standard deviation of data:
sepal length (cm)    0.828066
sepal width (cm)     0.435866
petal length (cm)    1.765298
petal width (cm)     0.762238
dtype: float64
```

In [ ]:
```
#c.
quantiles = df.quantile([0.25, 0.5, 0.75])
print(f"Quantiles of data:\n{quantiles}\n")

q1 = df.quantile(0.25)
q3 = df.quantile(0.75)
iqr = q3 - q1
print(f"IQR\n{iqr}")
```

```
Quantiles of data:
      sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0.25                5.1               2.8               1.60               0.3
0.50                5.8               3.0               4.35               1.3
0.75                6.4               3.3               5.10               1.8

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.

In [10]:
```
df.plot(kind='kde', subplots=True, layout=(2, 2), figsize=(10, 8), sharex=False, sharey=False
plt.tight_layout()
plt.show()
```

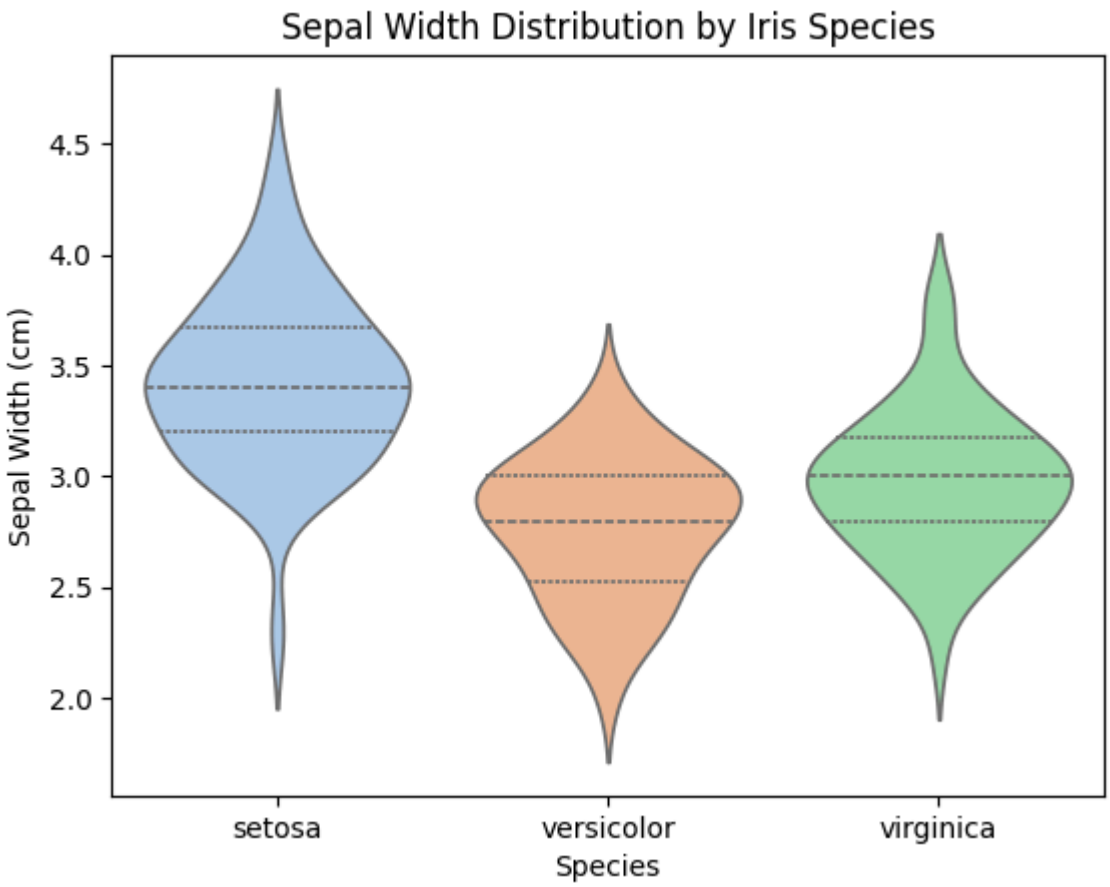


- interpretation:

1. Sepal Length: the plot is Bell-like and slightly flat at the top.it shows that most sepal lengths are concentrated between 5 and 7 cm. Few flowers have very short or very long sepals.
2. Sepal Width: the plot is very tall and narrow. it shows that sepal width is quite consistent, and clusters tightly around 3 cm.
3. Petal Length: the plot is Bimodal so it shows two subgroups:One group with short petals, Another with longer petals. This reflects how petal length probably differs between 2 classes.
4. Petal Width: this plot is also bimodal which indicates two common petal width ranges, corresponding to distinct groups in the data.

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

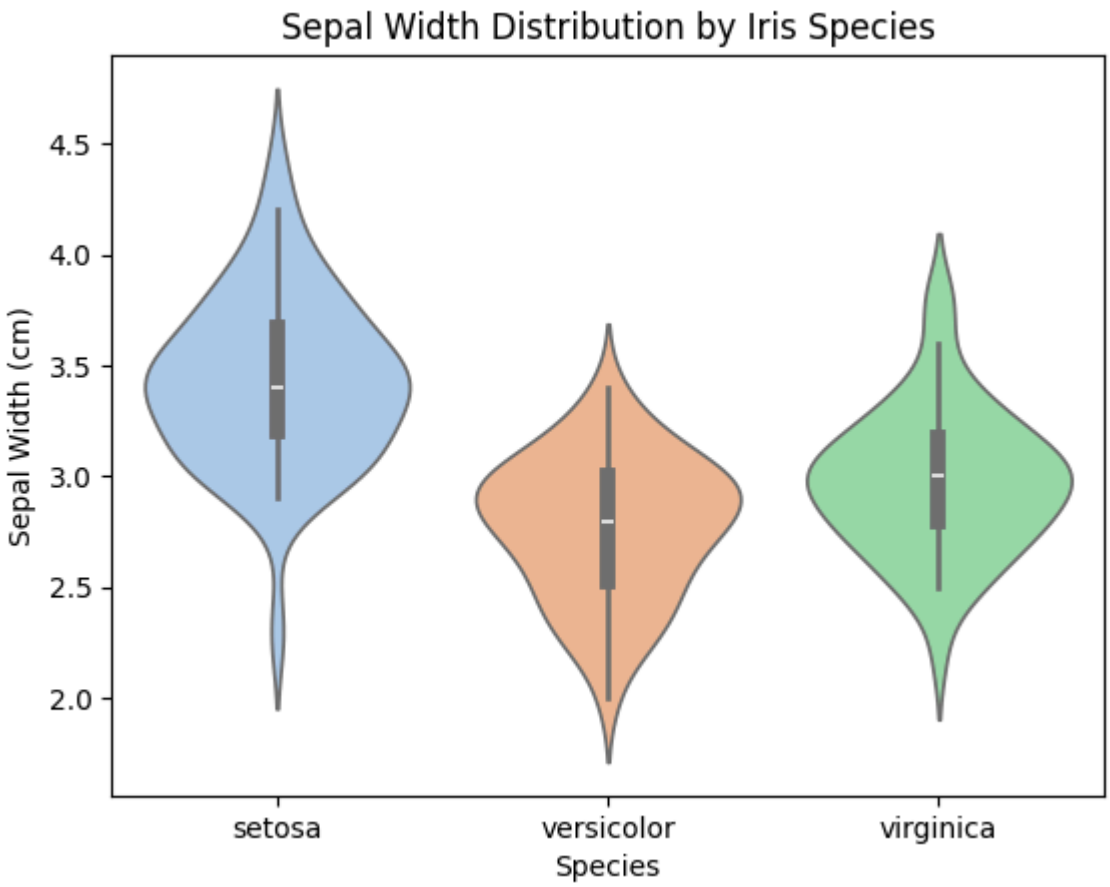
In [11]:
```
import seaborn as sns
df['species'] = pd.Categorical.from_codes(codes=iris.target, categories=iris.target_names)

# Create violin plot for sepal width
sns.violinplot(data=df, x='species', y='sepal width (cm)', inner= 'quartile' , hue= 'species'

plt.title(label="Sepal Width Distribution by Iris Species")
plt.xlabel(xlabel="Species")
plt.ylabel(ylabel="Sepal Width (cm)")
plt.show()
```



- Setosa has wider sepals, this feature might help in classification because it make it distinct from the other species
- Versicolor and Virginica have an overlapping sepal width distribution, this feature alone wouldn't be sufficient to distinguish them.

5. Combine your violin plot with boxplots as shown in the lecture.

In [12]:
```
df['species'] = pd.Categorical.from_codes(codes=iris.target, categories=iris.target_names)

sns.violinplot(data=df, x='species', y='sepal width (cm)', inner= 'box' , hue= 'species',pale

plt.title(label="Sepal Width Distribution by Iris Species")
plt.xlabel(xlabel="Species")
plt.ylabel(ylabel="Sepal Width (cm)")
plt.show()
```



Exercise 02 Data Pre-processing

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

In [13]:
```
import pandas as pd
#This script reads a CSV file containing banknote authentication data and prints the number
# of rows and columns in the dataset.
banknote: pd.DataFrame =pd.read_csv(filepath_or_buffer="data_banknote_authentication.csv")

rows, columns = banknote.shape
print("Number of rows: ", rows)
print("Number of columns: ", columns)
```

```
Number of rows:  1382
Number of columns:  5
```

2. Mention the different types of variables. Which types does your dataset contain? (2 point)

There are mainly two types of variables: Quantitative: It includes Continuous and Discrete Variable. Qualitative: It includes Nominal and Ordinal variables.

For our Banknote Authentication Dataset, we have Variance, Skewness, Curtosis and Entropy. These are continuous (Quantitative) variables. Whereas Class is qualitative variable.

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

We can use the method drop_duplicates() method to remove the duplicates

In [14]:
```
#This script checks for duplicates in the dataset and prints the number of duplicate rows.
#It uses the duplicated() function to identify duplicate rows and sums them to get the total c
duplicate=banknote.duplicated()
print("Number of duplicate rows: ", duplicate.sum())
```

```
Number of duplicate rows:  23
```

In [15]:
```
#This script removes duplicate rows from the dataset using the drop_duplicates() function.
remove_duplicate=banknote.drop_duplicates()
print("Number of duplicate rows: ", remove_duplicate.duplicated().sum())
```

```
Number of duplicate rows:  0
```

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

In [16]:
```
from pandas import DataFrame
#This script checks for missing values in the dataset and prints the total number of missing v
#It uses the isnull() function to identify missing values and sums them to get the total count

missing: DataFrame = banknote.isnull()
print("Number of missing values: ", missing.sum().sum())
```

```
Number of missing values:  690
```

5. How can you deal with missing values in your dataset? Implement one of the possible methods (2 points)

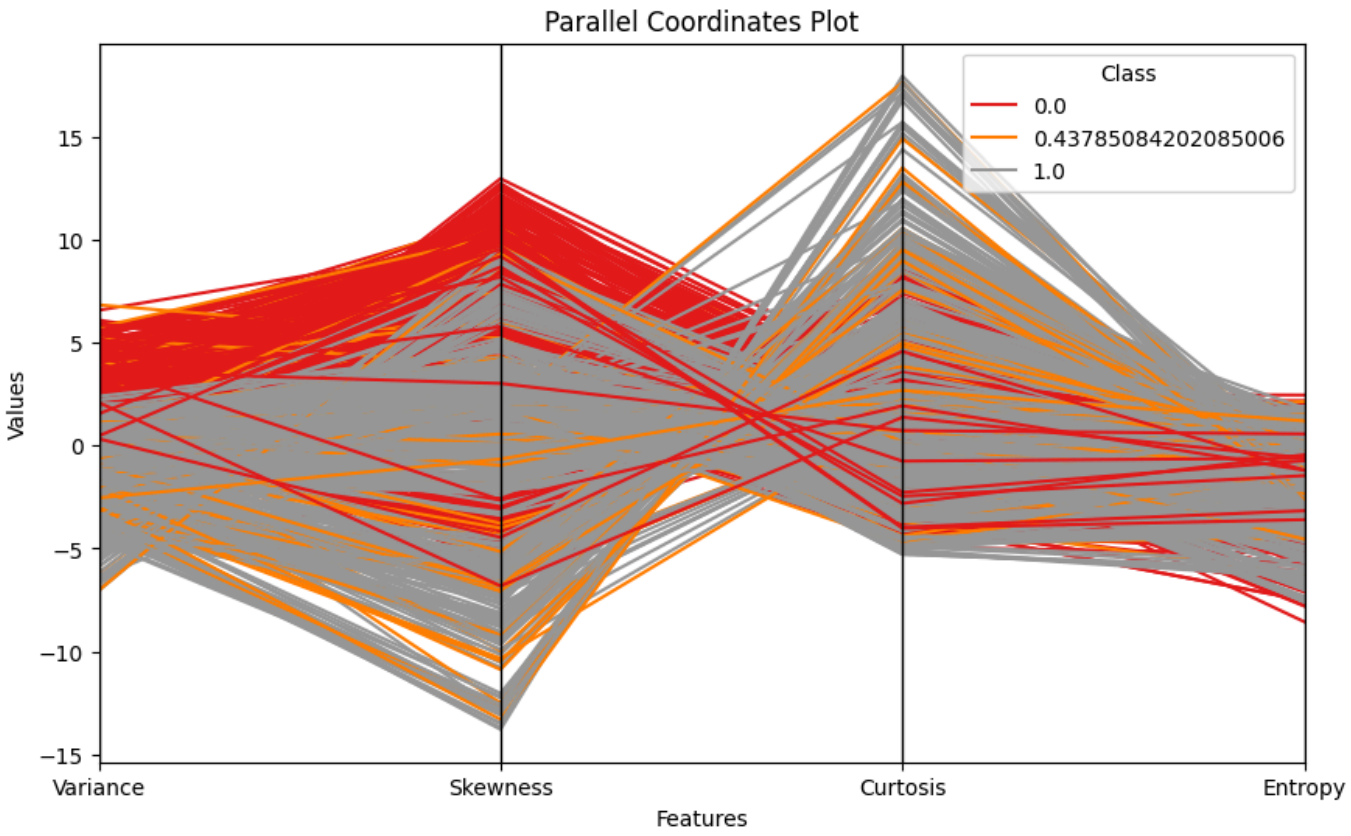
We can replace the missing value by mean to handle the missing value

In [17]:
```
from pandas import DataFrame
#This script handles missing values in the dataset by replacing them with the mean value of th
#It uses the fillna() function to replace missing values with the mean of each column.

handling_missing_values: DataFrame=banknote.fillna(banknote.mean())
print("Missing values handled by replacing with mean value of the column." )
```

```
Missing values handled by replacing with mean value of the column.
```

6. Based on the dataset you get after dealing with missing values, create a parallel coordinates plot. Color the lines based on class assignment.

In [18]:
```
#This script creates a parallel coordinates plot to visualize the features of the dataset.
import matplotlib.pyplot as plt
from pandas.plotting import parallel_coordinates
plt.figure(figsize=(10, 6))
#This script uses the parallel_coordinates() function to create a parallel coordinates plot.
parallel_coordinates(frame=handling_missing_values, class_column= 'Class', colormap=plt.get_cm
plt.title(label="Parallel Coordinates Plot")
plt.xlabel(xlabel="Features")
plt.ylabel(ylabel="Values")
plt.legend(title="Class", loc="upper right")
plt.grid()
plt.show()
```



Exercise 03 Scatterplot

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

2. Plotting

In [19]:
```
from pandas.plotting import scatter_matrix
import matplotlib.pyplot as plt
#This script creates a scatter matrix plot to visualize the relationships between features in
df=pd.read_csv("dataset.csv")
if 'Unnamed: 0' in df.columns:
    # Remove the 'Unnamed: 0' column if it exists
    df = df.drop(columns=['Unnamed: 0']) #This script removes the 'Unnamed: 0' column from th
scatter_matrix(df, alpha=0.2, figsize=(10, 10), diagonal='kde')
plt.suptitle("Scatter Matrix Plot")
plt.show()
```

Scatter Matrix Plot