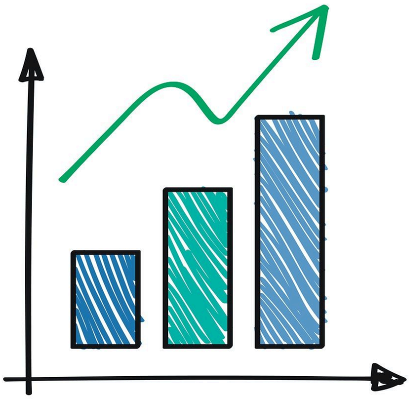
Exploratory Data Analysis



Name : Sai Purvi S

Date : 30th December, 2024

Topic : Exploratory Data Analysis on Iris Dataset

Tools Used : Python (Pandas, Seaborn, Matplotlib)

Dataset taken from “Greek For Greek”

Problem Statement –

The goal of this project is to perform Exploratory Data Analysis (EDA) on the Iris dataset to understand the underlying patterns, relationships, and distributions among the features of different Iris flower species — Iris-setosa, Iris-versicolor, and Iris-virginica.

By analyzing key parameters such as sepal length, sepal width, petal length, and petal width, this study aims to identify distinguishing characteristics between species, detect potential correlations, and visualize the data effectively for better interpretation.

The insights from this analysis can serve as a foundation for building accurate machine learning models for flower classification and contribute to a deeper understanding of biological patterns in plant taxonomy.

Data Set –

The Dataset is taken from “Greek For Greek”.

Methodology –

The project followed a systematic approach to perform Exploratory Data Analysis (EDA) on the Iris dataset. The key steps are described below:

1. Data Collection
- The Iris dataset was imported using the pandas library.
- The dataset contains 150 records with 5 features: Sepal Length, Sepal Width, Petal Length, Petal Width, and Species.

2. Data Inspection
- Used df.head(), df.shape(), and df.info() to understand the structure, number of records, and data types.
- Performed df.describe() to compute summary statistics such as mean, median, and standard deviation.

3. Data Cleaning and Preparation

- Checked for missing values using df.isnull().sum() — none were found.
- Verified duplicate records using drop_duplicates() and removed any if present.
- Checked class balance using df.value_counts("Species") to ensure equal representation of all three flower types.

4. Data Visualization

- Created scatter plots to observe relationships between sepal and petal dimensions using Seaborn.
- Used histograms to study the distribution of each numerical feature.
- Applied FacetGrid plots for species-wise comparison of features, making patterns and distinctions more visible.

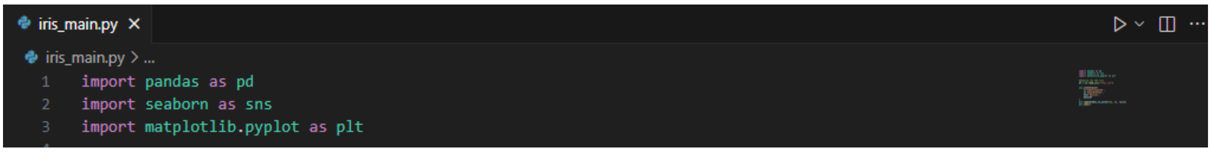
5. Exploration and Insights

- Identified clear separation patterns between species based on petal dimensions.
- Observed overlap in sepal measurements, suggesting they are less effective for species classification.

6. Tools and Libraries Used

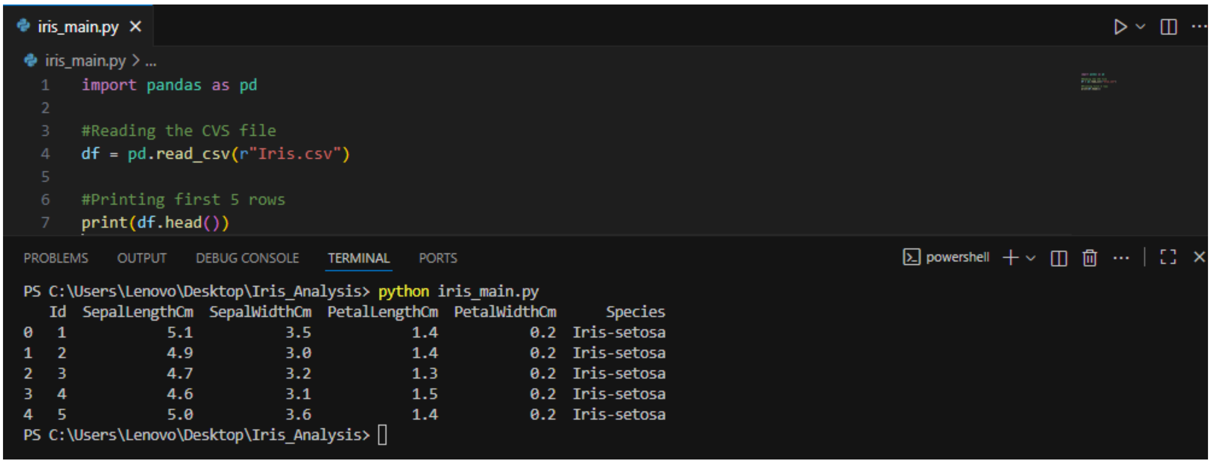
- pandas for data handling and cleaning.
- matplotlib and seaborn for data visualisation

INPUTS and OUTPUTS –



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

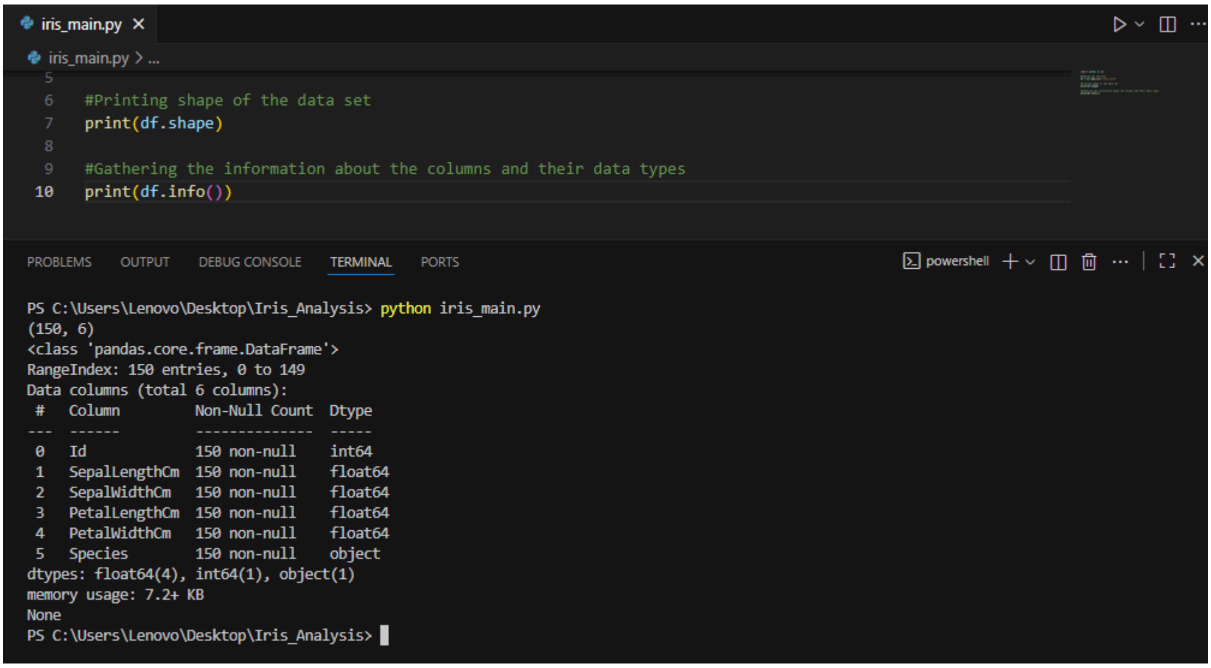
Importing all the necessary packages for this analysis. Pandas are used to read the file. Seaborn and matplotlib are used for data visualisation.



```python
import pandas as pd

#Reading the CVS file
df = pd.read_csv(r"Iris.csv")

#Printing first 5 rows
print(df.head())
```

```
PS C:\Users\Lenovo\Desktop\Iris_Analysis> python iris_main.py
   Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm      Species
0   1            5.1           3.5            1.4           0.2  Iris-setosa
1   2            4.9           3.0            1.4           0.2  Iris-setosa
2   3            4.7           3.2            1.3           0.2  Iris-setosa
3   4            4.6           3.1            1.5           0.2  Iris-setosa
4   5            5.0           3.6            1.4           0.2  Iris-setosa
PS C:\Users\Lenovo\Desktop\Iris_Analysis> 
```

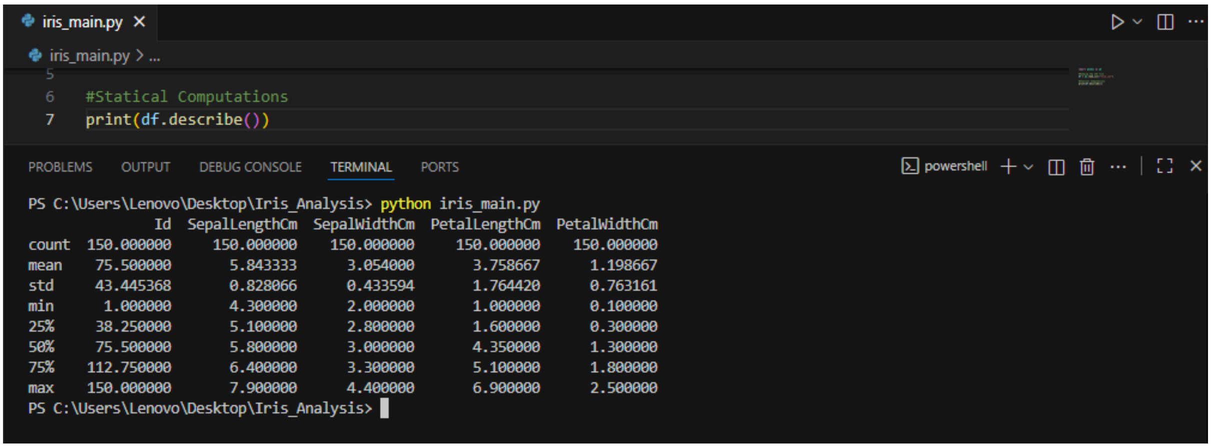
The df.head() function displays the first five rows of the dataset, providing a quick preview of the data. It helps to verify that the dataset has been loaded correctly and to understand the structure and column names.



```python

#Printing shape of the data set
print(df.shape)

#Gathering the information about the columns and their data types
print(df.info())
```

```
PS C:\Users\Lenovo\Desktop\Iris_Analysis> python iris_main.py
(150, 6)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             150 non-null    int64  
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object 
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
None
PS C:\Users\Lenovo\Desktop\Iris_Analysis> 
```

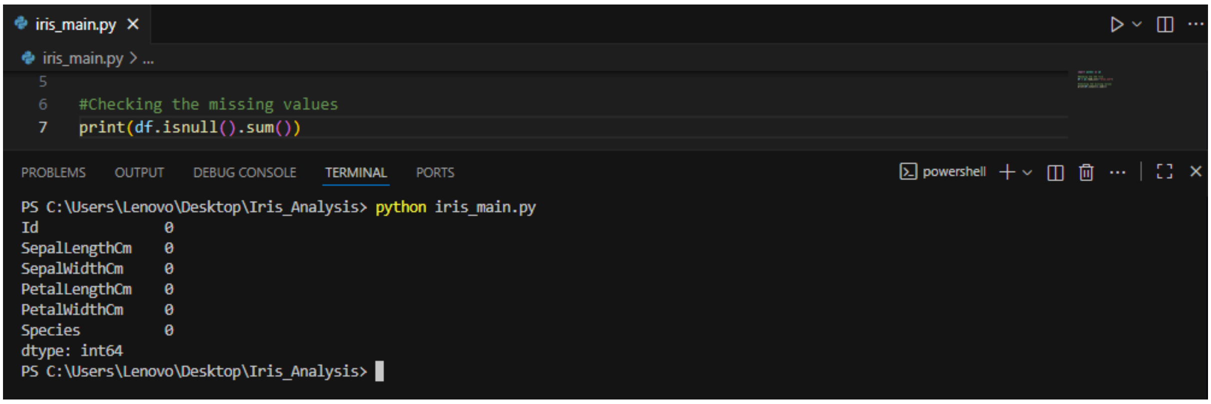
df.shape gives the shape of the loaded dataset. We have 150 rows and 6 columns. df.info() is used to gather the information about the columns and their data types. We can see that only one column has categorical data and all the other columns are of the numeric type with non-Null entries.



```python
#Statical Computations
print(df.describe())
```

```
PS C:\Users\Lenovo\Desktop\Iris_Analysis> python iris_main.py
               Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
count  150.000000     150.000000    150.000000     150.000000    150.000000
mean    75.500000       5.843333      3.054000       3.758667      1.198667
std     43.445368       0.828066      0.433594       1.764420      0.763161
min      1.000000       4.300000      2.000000       1.000000      0.100000
25%     38.250000       5.100000      2.800000       1.600000      0.300000
50%     75.500000       5.800000      3.000000       4.350000      1.300000
75%    112.750000       6.400000      3.300000       5.100000      1.800000
max    150.000000       7.900000      4.400000       6.900000      2.500000
PS C:\Users\Lenovo\Desktop\Iris_Analysis> 
```

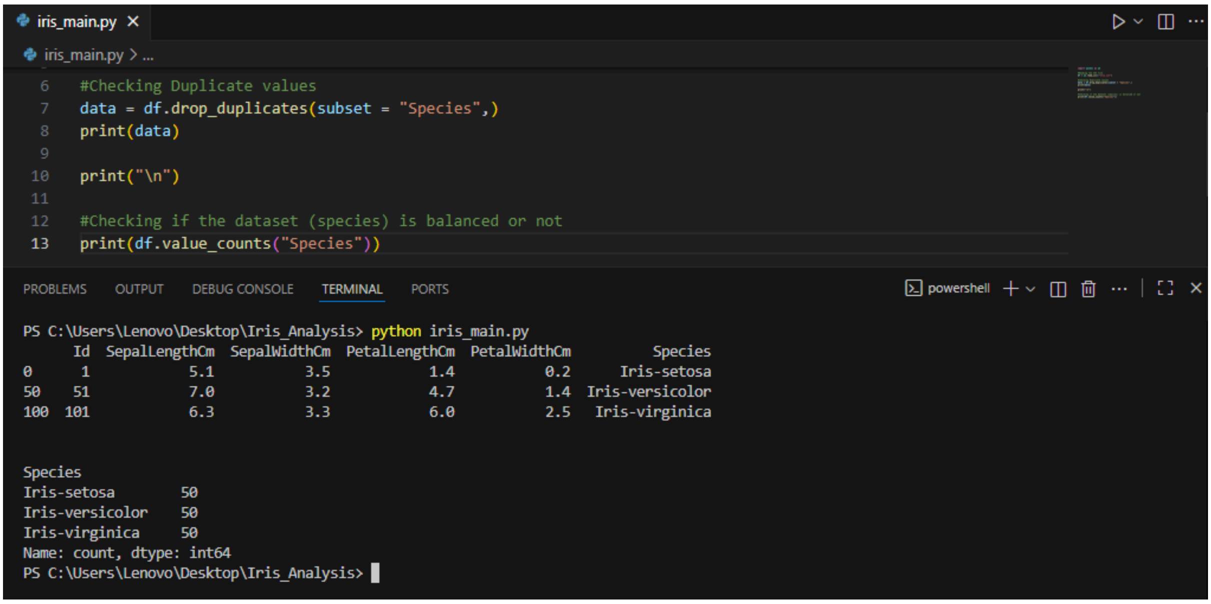
df.describe() is used to perform all the statistical computations on the numeric columns.



```python
#Checking the missing values
print(df.isnull().sum())
```

```
PS C:\Users\Lenovo\Desktop\Iris_Analysis> python iris_main.py
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
PS C:\Users\Lenovo\Desktop\Iris_Analysis> 
```

df.isnull().sum() is used to check if there are any missing values. We can see that there are no missing values.

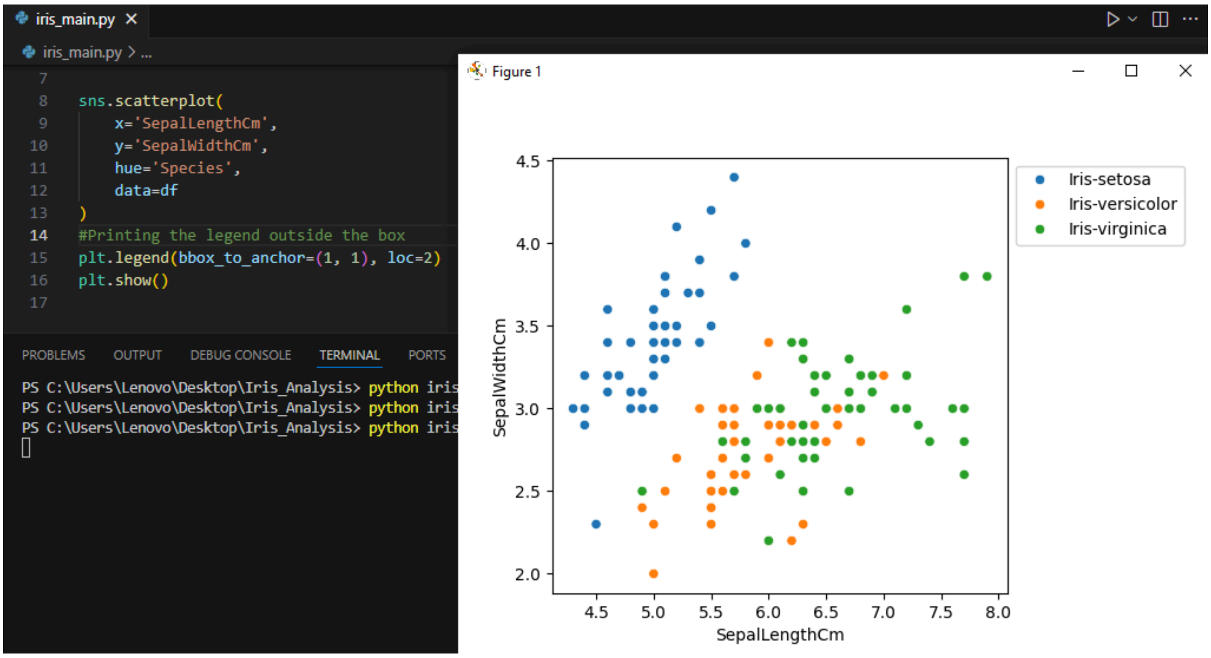


```python
#Checking Duplicate values
data = df.drop_duplicates(subset = "Species",)
print(data)

print("\n")

#Checking if the dataset (species) is balanced or not
print(df.value_counts("Species"))
```

```
PS C:\Users\Lenovo\Desktop\Iris_Analysis> python iris_main.py
      Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm          Species
0      1            5.1           3.5            1.4           0.2      Iris-setosa
50    51            7.0           3.2            4.7           1.4  Iris-versicolor
100  101            6.3           3.3            6.0           2.5   Iris-virginica


Species
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: count, dtype: int64
PS C:\Users\Lenovo\Desktop\Iris_Analysis> 
```

First we are checking if there are any duplicate values using df.drop_duplicates(). We notice that there are 3 types of Iris species – Iris-setosa, Iris-versicolor and Iris-virginica.

Next we are checking if the dataset (i.e, species) are balanced or not using df.value_counts(). And we notice that there are equal number of rows (i.e, 50 rows each)

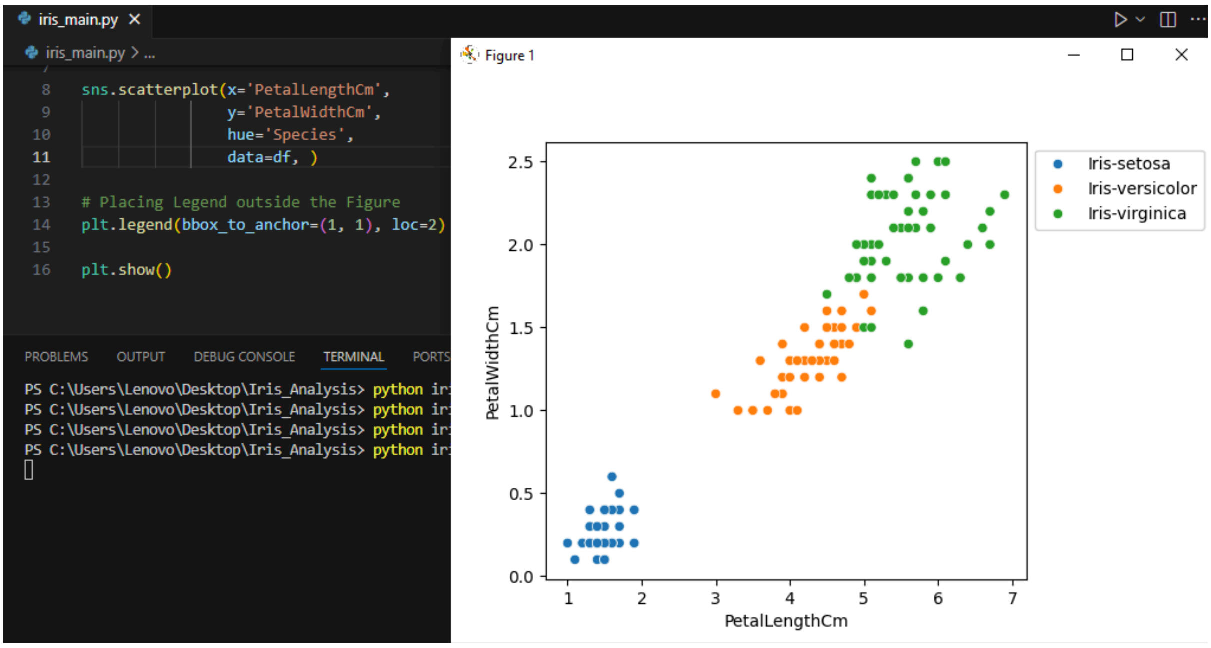


```
sns.scatterplot(
    x='SepalLengthCm',
    y='SepalWidthCm',
    hue='Species',
    data=df
)
#Printing the legend outside the box
plt.legend(bbox_to_anchor=(1, 1), loc=2)
plt.show()
```

```
PS C:\Users\Lenovo\Desktop\Iris_Analysis> python iris
PS C:\Users\Lenovo\Desktop\Iris_Analysis> python iris
PS C:\Users\Lenovo\Desktop\Iris_Analysis> python iris
```

Here we are comparing Sepal Length and Sepal Width.

We notice that Setosa has smaller Sepal Length but larger Sepal Width. Virginica has larger Sepal Length but smaller Sepal Width. Species Versicolor lies in between these two extremes.

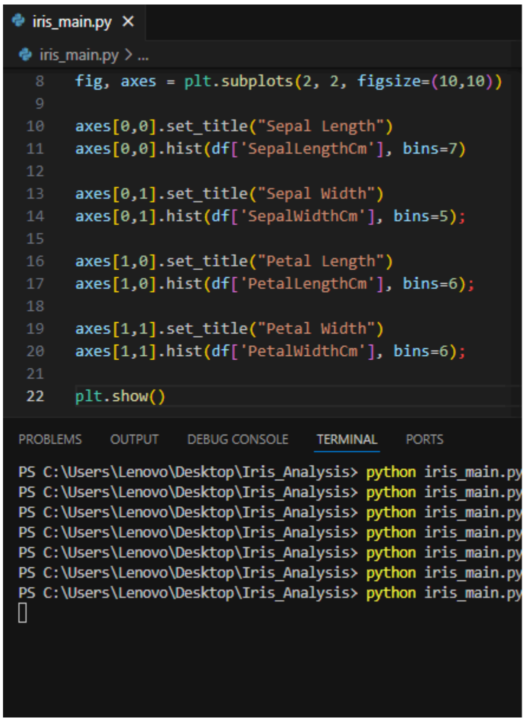


```
sns.scatterplot(x='PetalLengthCm',
                y='PetalWidthCm',
                hue='Species',
                data=df, )

# Placing Legend outside the Figure
plt.legend(bbox_to_anchor=(1, 1), loc=2)

plt.show()
```

```
PS C:\Users\Lenovo\Desktop\Iris_Analysis> python ir
PS C:\Users\Lenovo\Desktop\Iris_Analysis> python ir
PS C:\Users\Lenovo\Desktop\Iris_Analysis> python ir
PS C:\Users\Lenovo\Desktop\Iris_Analysis> python ir
```

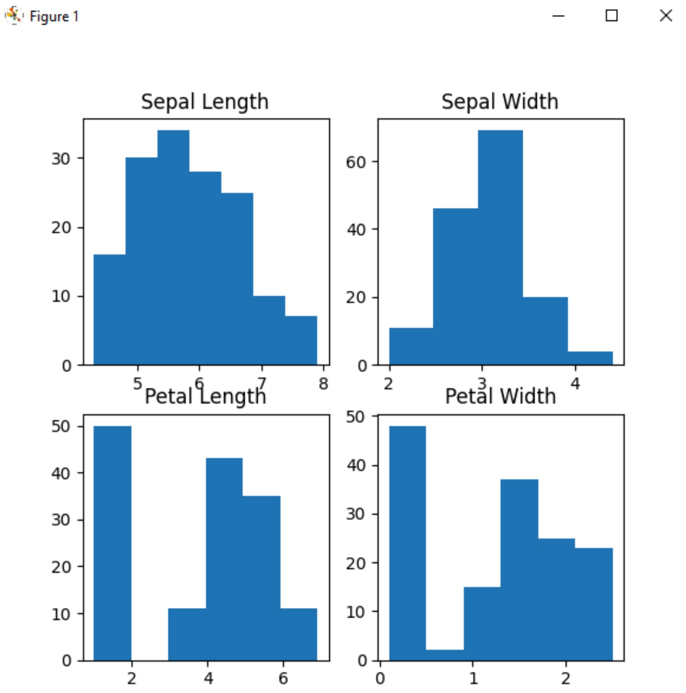
Here we are comparing Petal length and Petal Width

We notice that Setosa has smaller Petal Length and Width. Virginie has the largest Petal Length and Width. But Species Vericolor lies between these two extremes.

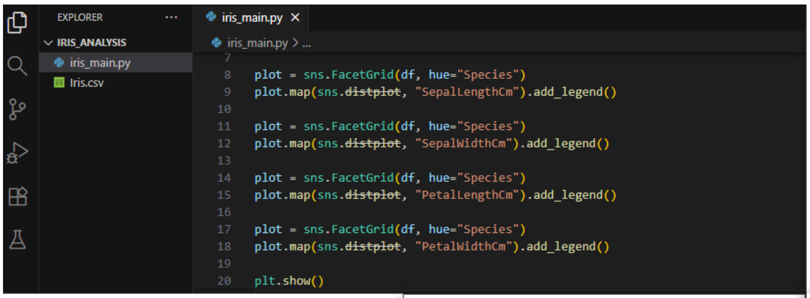


```
fig, axes = plt.subplots(2, 2, figsize=(10,10))

axes[0,0].set_title("Sepal Length")
axes[0,0].hist(df['SepalLengthCm'], bins=7)

axes[0,1].set_title("Sepal Width")
axes[0,1].hist(df['SepalWidthCm'], bins=5);

axes[1,0].set_title("Petal Length")
axes[1,0].hist(df['PetalLengthCm'], bins=6);

axes[1,1].set_title("Petal Width")
axes[1,1].hist(df['PetalWidthCm'], bins=6);

plt.show()
```

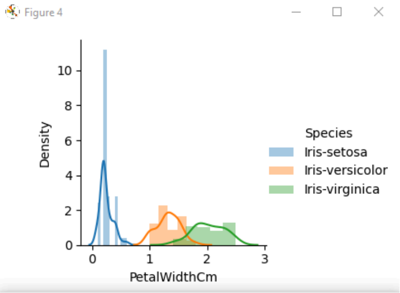
```
PS C:\Users\Lenovo\Desktop\Iris_Analysis> python iris_main.py
PS C:\Users\Lenovo\Desktop\Iris_Analysis> python iris_main.py
PS C:\Users\Lenovo\Desktop\Iris_Analysis> python iris_main.py
PS C:\Users\Lenovo\Desktop\Iris_Analysis> python iris_main.py
PS C:\Users\Lenovo\Desktop\Iris_Analysis> python iris_main.py
PS C:\Users\Lenovo\Desktop\Iris_Analysis> python iris_main.py
PS C:\Users\Lenovo\Desktop\Iris_Analysis> python iris_main.py
```

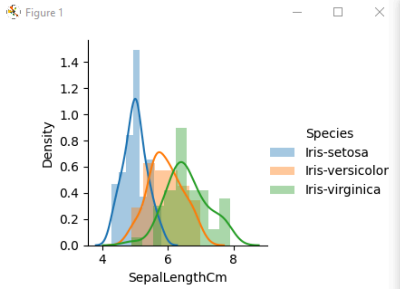


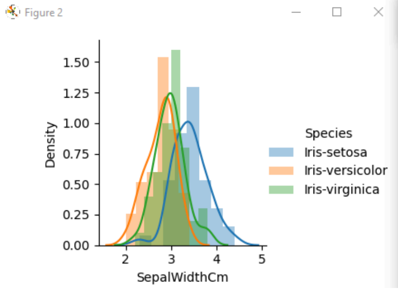
Histogram allows us to see the distribution of data for various columns. Here we observe - The highest frequency of the sepal length is between 30 and 35 which is between 5.5 and 6. The highest frequency of the sepal Width is around 70 which is between 3.0 and 3.5. The highest frequency of the petal length is around 50 which is between 1 and 2. The highest frequency of the petal width is between 40 and 50 which is between 0.0 and 0.5

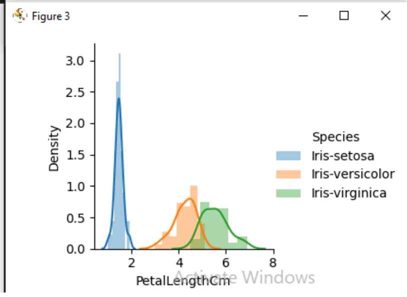


```
plot = sns.FacetGrid(df, hue="Species")
plot.map(sns.distplot, "SepalLengthCm").add_legend()

plot = sns.FacetGrid(df, hue="Species")
plot.map(sns.distplot, "SepalWidthCm").add_legend()

plot = sns.FacetGrid(df, hue="Species")
plot.map(sns.distplot, "PetalLengthCm").add_legend()

plot = sns.FacetGrid(df, hue="Species")
plot.map(sns.distplot, "PetalWidthCm").add_legend()

plt.show()
```









Histogram with Displot plot is used basically for the univariant set of observations and visualizes it through a histogram i.e. only one observation and hence we choose one particular column of the dataset. We observe - In the case of Sepal Length, there is a huge amount of overlapping. In the case of Sepal Width also, there is a huge amount of overlapping. In the case of Petal Length, there is a very little amount of overlapping. In the case of Petal Width also, there is a very little amount of overlapping.

Conclusion –

The Exploratory Data Analysis (EDA) of the Iris dataset provided valuable insights into the characteristics of different Iris flower species. The analysis showed that petal length and petal width are the most significant features for distinguishing between species, while sepal measurements show some overlap. The dataset was clean, well-balanced, and free from missing or duplicate values, making it ideal for further machine learning applications. Overall, this project successfully explored the data, identified key patterns, and built a strong foundation for predictive modeling and species classification.