2024

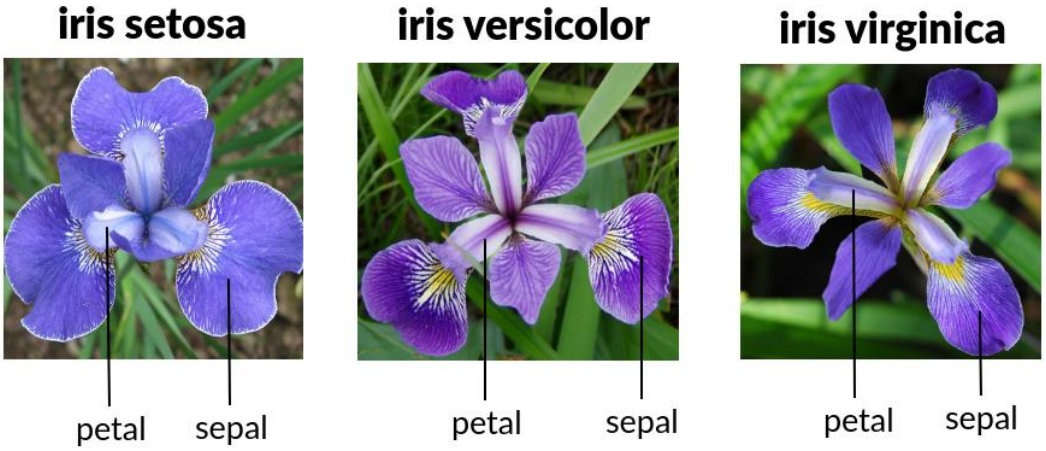
ANALYSIS OF IRIS DATASET

EXPLORATORY DATA ANALYSIS (EDA) AND VISUALIZATION

Submitted by :

AARTHY SM

Abstract



This project involves the analysis of the Iris dataset, a well-known dataset in the field of machine learning and statistics. The objective is to perform a comprehensive exploratory data analysis (EDA) and visualization to understand the dataset's structure and uncover key patterns.

The project begins with data import and preliminary exploration, including reviewing the dataset's head and tail. A sanity check ensures that the data is correctly formatted and cleansed by removing unnecessary columns and verifying its structure.

Subsequently, the project addresses missing values by identifying and calculating their percentages, ensuring data integrity. Exploratory data analysis follows, including descriptive statistics for both categorical and numeric variables, and an examination of unique and count values.

Visualization techniques, such as scatter plots, bar plots, and pair plots, are used to illustrate relationships between features and provide a deeper insight into the dataset. A heatmap of correlations is also generated to reveal the strength of relationships between different variables.

In conclusion, the project provides a detailed understanding of the Iris dataset, highlighting key patterns and relationships. This analysis forms the foundation for any subsequent modelling and predictive tasks.

Source : kaggle.com

Table of Contents

1.Introduction

1.1 Objective

The document provides a detailed analysis of the Iris dataset through exploratory data analysis (EDA) and visualization. The aim is to understand the dataset's structure, identify key patterns, and gain insights through various analytical techniques.

1.2 Dataset Overview

1.2.1 About the Dataset

The Iris dataset was used in R.A. Fisher's classic 1936 paper, The Use of Multiple Measurements in Taxonomic Problems, and can also be found on the UCI Machine Learning Repository.

It includes three iris species with 50 samples each as well as some properties about each flower. One flower species is linearly separable from the other two, but the other two are not linearly separable from each other.

1.2.2 Input Variables

1. Id
2. SepalLengthCm
3. SepalWidthCm
4. PetalLengthCm
5. PetalWidthCm
6. Species
    - Setosa
    - Versicolor
    - Virginica

7. Dataset characteristics - Multivariate
8. Number of Observations - 150(50 in each of 3 classes)
9. Number of varibles - 4 numeric values
10. Missing values - N/A.

2.Data Import and Exploration

2.1 Import Libraries

1. ***Pandas*** - Provides data structures and functions for efficient data manipulation and analysis.
2. ***NumPy*** - Supports numerical operations and efficient array handling.
3. ***Matplotlib.pyplot*** - Enables creation of static, interactive, and animated visualizations in Python
4. ***Seaborn*** - Simplifies the creation of informative and attractive statistical graphics based on Matplotlib.

```
# import neccessary libraries

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

2.2 Read Dataset

```
# Read Dataset

data = pd.read_csv("Iris_model.csv")
```

2.3 Head and Tail

```
# Head

data.head()
```

```
# Tail

data.tail()
```

Head - Shows the first few rows of the DataFrame.

```
# Head

data.head()
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	setosa
1	2	4.9	3.0	1.4	0.2	setosa
2	3	4.7	3.2	1.3	0.2	setosa
3	4	4.6	3.1	1.5	0.2	setosa
4	5	5.0	3.6	1.4	0.2	setosa

Tail - Displays the last few rows of the DataFrame.

```
# Tail

data.tail()
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
145	146	6.7	3.0	5.2	2.3	virginica
146	147	6.3	2.5	5.0	1.9	virginica
147	148	6.5	3.0	5.2	2.0	virginica
148	149	6.2	3.4	5.4	2.3	virginica
149	150	5.9	3.0	5.1	1.8	virginica

3.Sanity Check of Data

- To ensure the dataset is accurate and reliable by identifying and addressing errors, inconsistencies, and anomalies before further analysis.

3.1 Naming the Columns as given in dataset

```
data.columns
```

```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

The data.columns output reveals that the dataset has 6 columns:

1. ***Id*** --> Unique identifier for each entry.
2. ***SepalLengthCm*** --> Sepal length in centimetres.
3. ***SepalWidthCm*** --> Sepal width in centimetres.
4. ***PetalLengthCm*** --> Petal length in centimetres.
5. ***PetalWidthCm*** --> Petal width in centimetres.
6. ***Species*** --> Species of the Iris flower.

3.2 Removing Id Column

- ID columns are non-informative for analysis and can be excluded to simplify the dataset.

```
# drop Id (unneccessary column in dataset)

data=data.drop(columns=['Id'])
```

```
data.columns
```

```
Index(['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

3.3 Shape

- To determine the dimensions of the dataset, including the number of rows and columns, which helps in understanding its structure and ensuring it aligns with expectations.

```
# Shape

data.shape
```

```
(150, 5)
```

The data.shape output indicates that the dataset has 150 rows and 5 columns.

3.4 Info

- To provide a summary of the dataset's structure, including data types, non-null counts, and column names, which helps in assessing data quality and preparing for analysis.

```
# info

data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   SepalLengthCm  150 non-null    float64
 1   SepalWidthCm   150 non-null    float64
 2   PetalLengthCm  150 non-null    float64
 3   PetalWidthCm   150 non-null    float64
 4   Species        150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

- The data.info() output reveals that the dataset is a Pandas Data Frame with 150 entries, indexed from 0 to 149.
- It contains 5 columns: SepalLengthCm , SepalWidthCm, PetalLengthCm , PetalWidthCm and Species
- All columns have 150 non-null entries.
- The first four columns are of type float64, representing the measurements in centimetres

- The Species column is of type object , indicating the species of the Iris flower.
- The dataset's memory usage is approximately 6.0 KB.

4.Handling Missing Values

4.1 Finding Missing values

- To identify and assess missing data in the dataset, which is crucial for ensuring data completeness and accuracy before proceeding with analysis.

```
# Finding missing values

data.isnull().sum()
```

```
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

```
# Percentage of missing values

data.isnull().sum()/data.shape[0]
```

```
SepalLengthCm    0.0
SepalWidthCm     0.0
PetalLengthCm    0.0
PetalWidthCm     0.0
Species          0.0
dtype: float64
```

This dataset has no missing values, indicating completeness and ready-to-use for analysis.

5.Exploratory Data Analysis (EDA)

- To perform a preliminary examination of the dataset, identifying key patterns and basic statistics to understand its structure and content.

Investigating the data

5.1 Descriptive Statistics for numeric values

- Provides a summary of key statistical measures such as mean, median, standard deviation, and range for numeric data, helping to understand the distribution and variability of the dataset.

```
# Descriptive Statistics for numeric values

data.describe()
```

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
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

The data.describe() output provides statistical summary of the numerical columns in the dataset:

- **SepalLengthCm** : The sepal length ranges from 4.30 cm to 7.90 cm, with a mean of 5.84 cm and a standard deviation of 0.83 cm.

- **SepalWidthCm:** The sepal width varies from 2.00 cm to 4.40 cm, with a mean of 3.05 cm and a standard deviation of 0.43 cm.
- **PetalLengthCm:** The petal length ranges from 1.00 cm to 6.90 cm, with an average of 3.76 cm and a standard deviation of 1.76 cm.
- **PetalWidthCm:** The petal width varies from 0.10 cm to 2.50 cm, with a mean of 1.20 cm and a standard deviation of 0.76 cm.

These statistics provide a comprehensive overview of the distribution and variability of the sepal and petal measurements in the dataset.

The data.describe() analysis helps us understand the central tendency and variability of the measurements.

It shows average values and the range of sepal and petal dimensions, indicating their distribution and spread.

This information is crucial for identifying patterns, assessing data quality, and informing further analysis or modelling.

5. 1.1 Minimum Value

- To identify the smallest value in a numeric dataset, which helps in understanding the range and distribution of the data.

```
data.min()

SepalLengthCm       4.3
SepalWidthCm        2.0
PetalLengthCm       1.0
PetalWidthCm        0.1
Species          setosa
dtype: object
```

The data.min() output reveals the minimum values for each column in the dataset:

SepalLengthCm --> 4.3 cm

SepalWidthCm --> 2.0 cm

PetalLengthCm --> 1.0 cm

PetalWidthCm --> 0.1 cm

Species --> setosa

Additionally, the species with the minimum recorded value is **setosa.** This indicates that the smallest measurements in the dataset are associated with the **setosa** species.

5.1.2 Maximum Value

- To determine the largest value in a numeric dataset, which helps in understanding the range and distribution of the data.

```
data.max()

SepalLengthCm          7.9
SepalWidthCm           4.4
PetalLengthCm          6.9
PetalWidthCm           2.5
Species          virginica
dtype: object
```

The data.max() output shows the maximum values for each column in the dataset:

SepalLengthCm --> 7.9 cm

SepalWidthCm --> 4.4 cm

PetalLengthCm --> 6.9 cm

PetalWidthCm --> 2.5 cm

Species --> virginica

These maximum values indicate the highest recorded measurements and species category, with **virginica** being the species associated with the largest values.

5.2 Descriptive Statistics for object values

- Provides insights into the frequency and distribution of categorical data, including counts and unique values, helping to understand the composition and diversity of the dataset.

```
# Descriptive Statistics for object values

data.describe(include="object")
```

	Species
count	150
unique	3
top	setosa
freq	50

The data.describe(include="object") output provides a summary of the categorical column Species:

Count --> 150 entries

Unique --> 3 distinct species

Top --> setosa (most frequently occurring species)

Frequency --> setosa appears 50 times

This summary highlights that there are three unique species in the dataset, with **setosa** being the most common.

5.2.1 Unique Values

- To identify all distinct values in a categorical dataset, helps in understanding the diversity and variability of object features.

```
# Species unique values

data.Species.unique()
```

```
array(['setosa', 'versicolor', 'virginica'], dtype=object)
```

The unique values in the Species column are:

- setosa
- versicolor
- virginica

These represent the three distinct species of Iris flowers present in the dataset.

5.2.2 Count of each Species values

- To determine the frequency of each category in the dataset, which helps in understanding the distribution and balance of categorical data.

```
# Count of each Species values

data.Species.value_counts()
```

```
Species
setosa        50
versicolor    50
virginica     50
Name: count, dtype: int64
```

The count of each species in the dataset:

- Setosa --> 50
- versicolor --> 50
- virginica --> 50

Each species is equally represented in the dataset, with 50 entries for each.

6. Visualization

- To graphically represent data patterns and relationships, making it easier to interpret and communicate insights from the dataset.

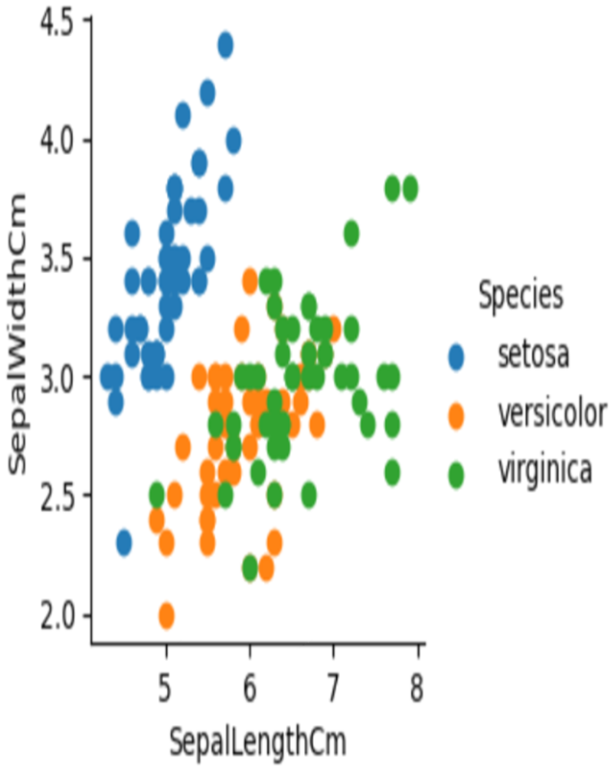
Distribution of features

6.1 Scatterplot

- Displays the relationship between two numeric variables, illustrating correlations and trends within the data.

Scatterplot [Sepal Length & Width]

```
# Scatter plot to visualize unique features in color
sns.FacetGrid(data,hue="Species").map(plt.scatter,"SepalLengthCm","SepalWidthCm").add_legend()
plt.show()
```



The scatter plot visualizes the relationship between sepal length (SepalLengthCm) and sepal width (SepalWidthCm) for different Iris species.

setosa , versicolor, and virginica

- is represented by a distinct color, making it easy to distinguish between them.

The plot reveals:

setosa is clustered in a lower sepal length and width range.

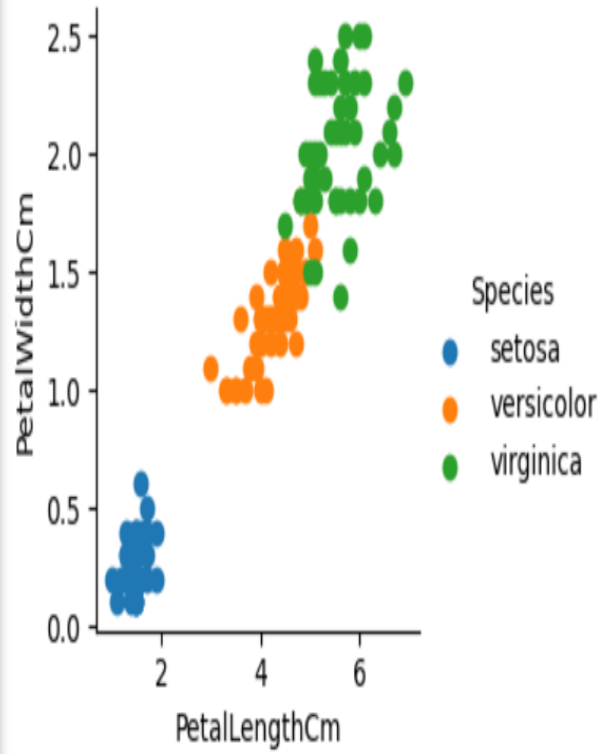
versicolor shows a more varied distribution across both dimensions.

virginica generally has a larger sepal length and width compared to the other species.

This visualization effectively highlights the differences and overlaps in sepal dimensions among the species.

Scatterplot [Petal Length & Width]

```
sns.FacetGrid(data,hue="Species").map(plt.scatter,"PetalLengthCm","PetalWidthCm").add_legend()
plt.show()
```



The scatter plot visualizes the relationship between petal length (PetalLengthCm) and petal width (PetalWidthCm) for different Iris species.

setosa , versicolor, and virginica

- is represented by a distinct color, making them easily distinguishable.

The plot shows:

setosa Generally clustered in the lower range of petal length and width.

versicolor Exhibits a moderate range of petal dimensions.

virginica Typically has larger petal lengths and widths compared to the other species.

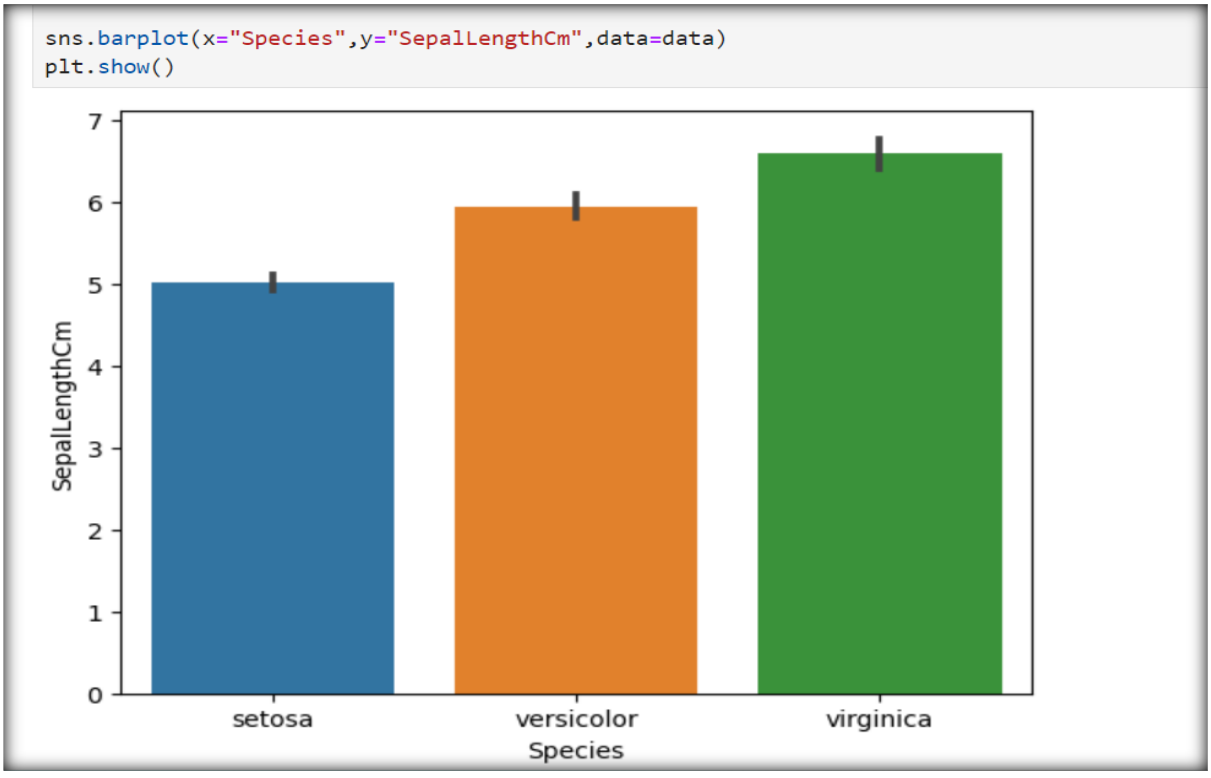
This visualization highlights the distinct separation in petal dimensions among the species, with virginica showing the most pronounced petal characteristics.

6.2 Boxplot

- To compare categorical data by displaying the frequency or count of each category, making it easier to analyze and interpret distribution pattern.

Boxplot of Sepal Length

```
sns.barplot(x="Species",y="SepalLengthCm",data=data)
plt.show()
```



The bar plot visualizes the average sepal length (SepalLengthCm) for each Iris species.

The plot shows:

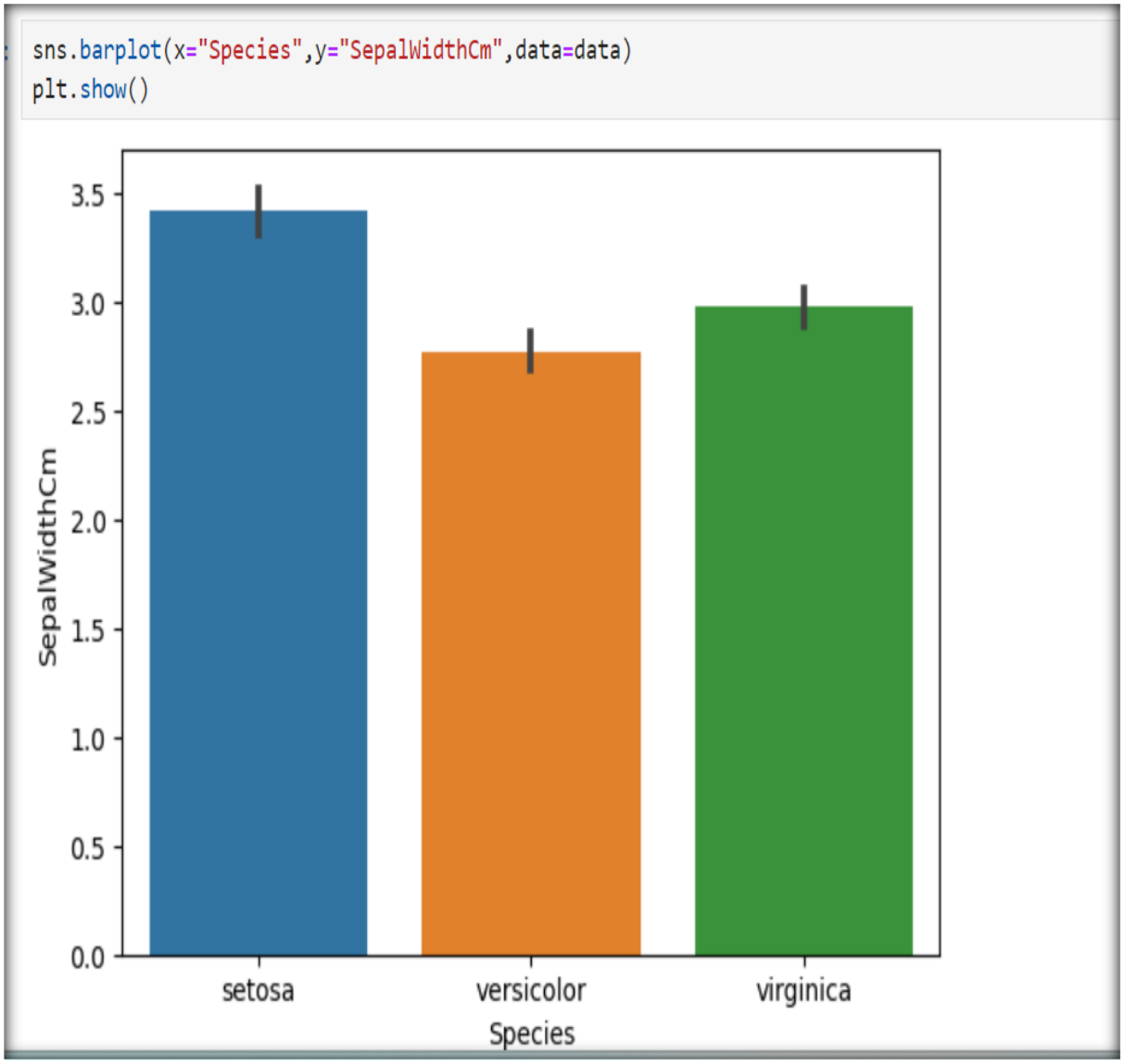
setosa --> Has the shortest average sepal length.

Versicolor--> Displays a slightly longer average sepal length than setosa.

Virginica --> Has the longest average sepal length among the three species.

This bar plot provides a clear comparison of the average sepal length across different species, illustrating how virginica typically has larger sepals compared to the other species.

Boxplot of Sepal Width



The bar plot visualizes the average sepal width (SepalWidthCm) for each Iris species.

The plot shows:

setosa --> Exhibits the widest average sepal width.

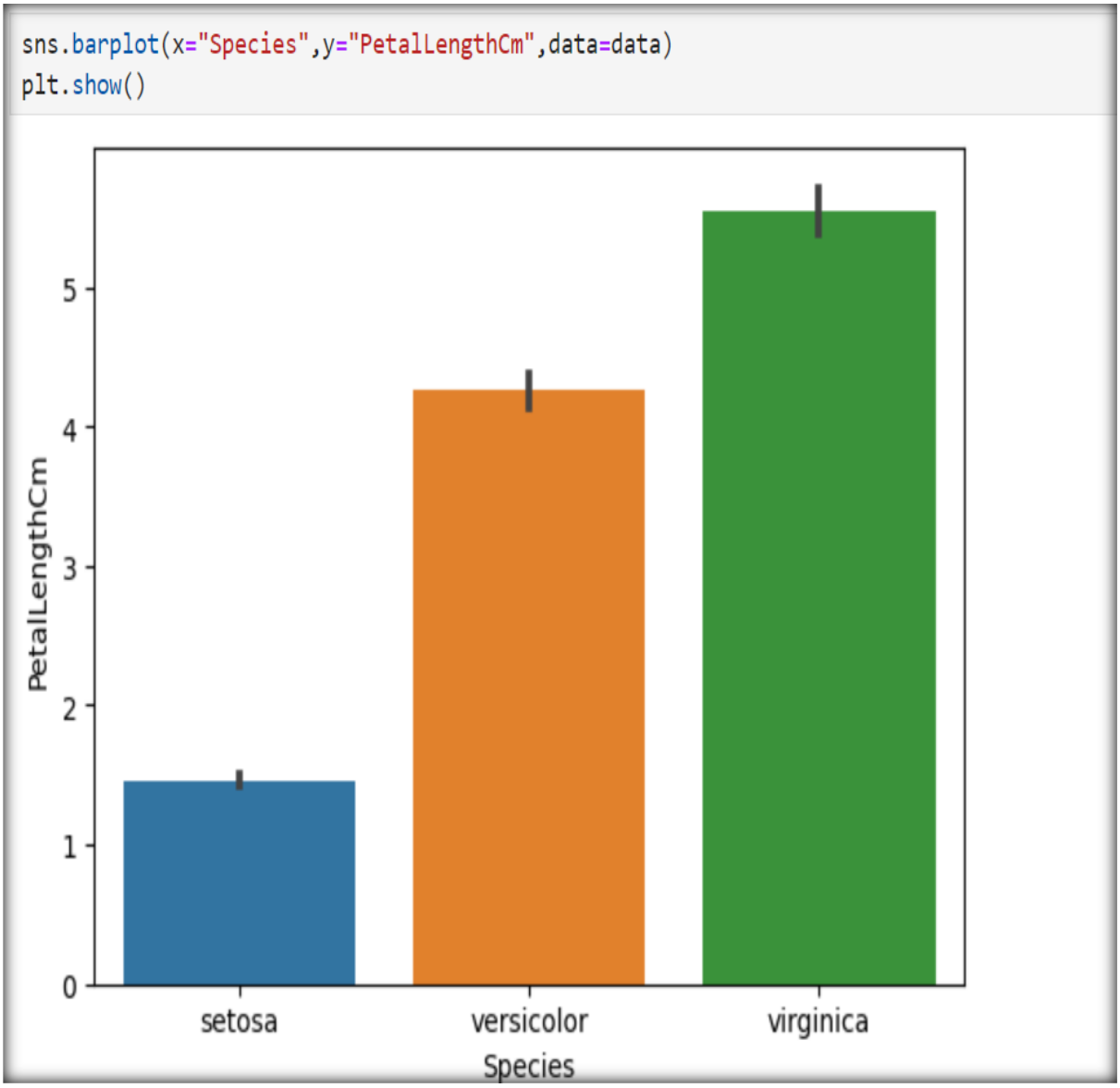
Versicolor --> Shows a moderate average sepal width, narrower than setosa.

Virginica --> Has the narrowest average sepal width among the three species.

This bar plot helps to compare the average sepal width across species, highlighting that setosa typically has wider sepals compared to the other species.

Boxplot of Petal Length

```
sns.barplot(x="Species",y="PetalLengthCm",data=data)
plt.show()
```



The bar plot visualizes the average petal length (PetalLengthCm) for each Iris species.

The plot shows:

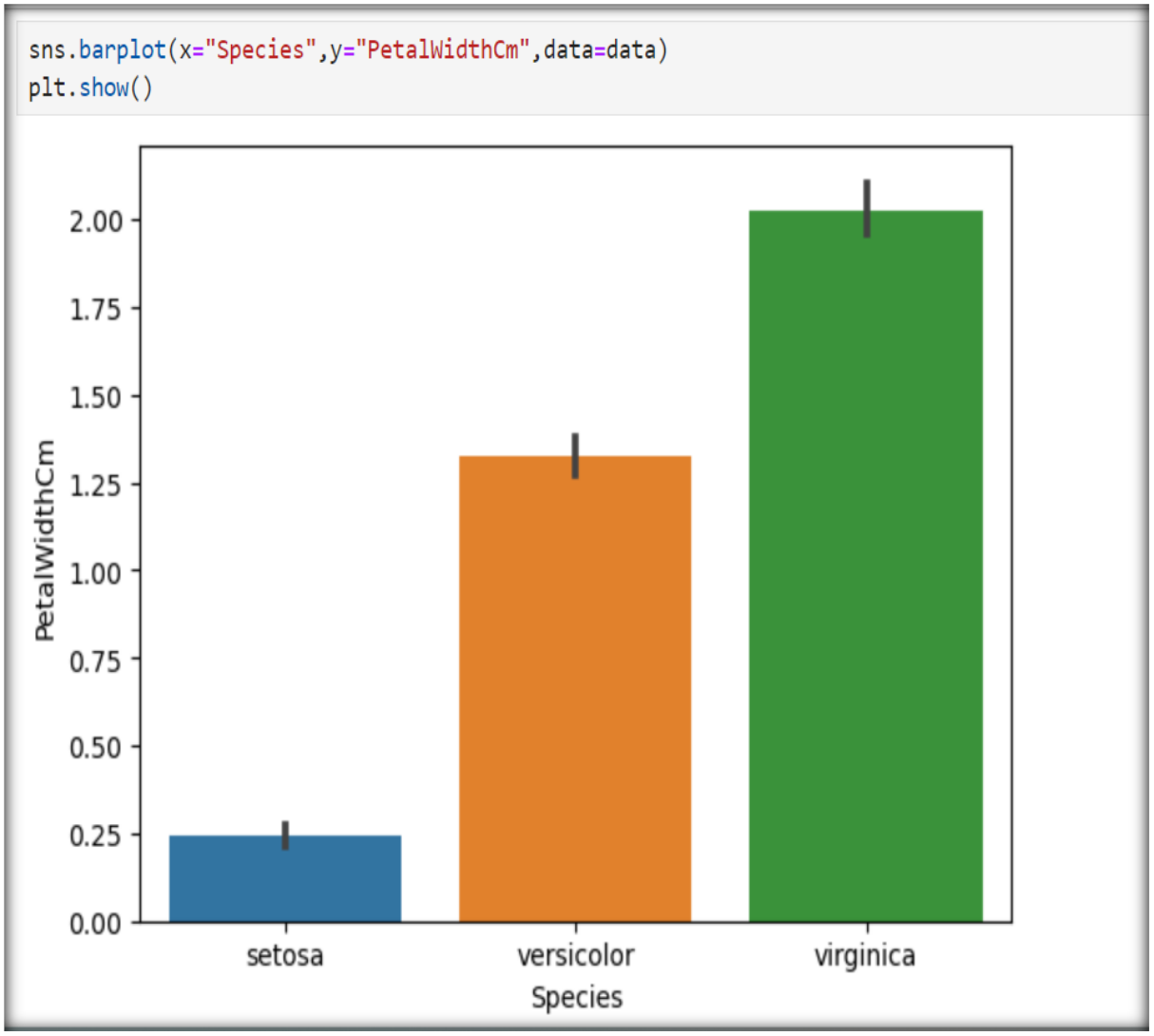
setosa --> Has the shortest average petal length.

versicolor --> Shows a moderate average petal length, longer than setosa.

Virginica --> Has the longest average petal length among the three species.

This bar plot effectively compares the average petal length across species, illustrating that virginica typically has longer petals compared to setosa and versicolor.

Boxplot of Petal Width



The bar plot visualizes the average petal width (PetalWidthCm) for each Iris species.

The plot shows:

Setosa --> Has the smallest average petal width.

Versicolor --> Exhibits a moderate average petal width, wider than setosa.

Virginica --> Displays the widest average petal width among the three species.

This bar plot highlights the differences in average petal width across species, with virginica having the widest petals compared to setosa and versicolor.

6.3 Pairplot

- To visualize pairwise relationships and distributions among multiple numeric variables, helping to identify correlations and patterns across the dataset.

The sns.pairplot() visualizes pairwise relationships between the numerical features in the dataset, with different colours representing each Iris species. The plot shows **scatter plots for each pair of features**, along with **distplots on the diagonal**.

Key observations include:

Distinct Clusters -----> **setosa**, **versicolor**, and **virginica** show distinct clustering patterns in the feature space, particularly in petal dimensions.

Feature Correlations-----> The pairwise plots reveal how features like petal length and petal width are correlated, with virginica often showing larger values compared to the other species.

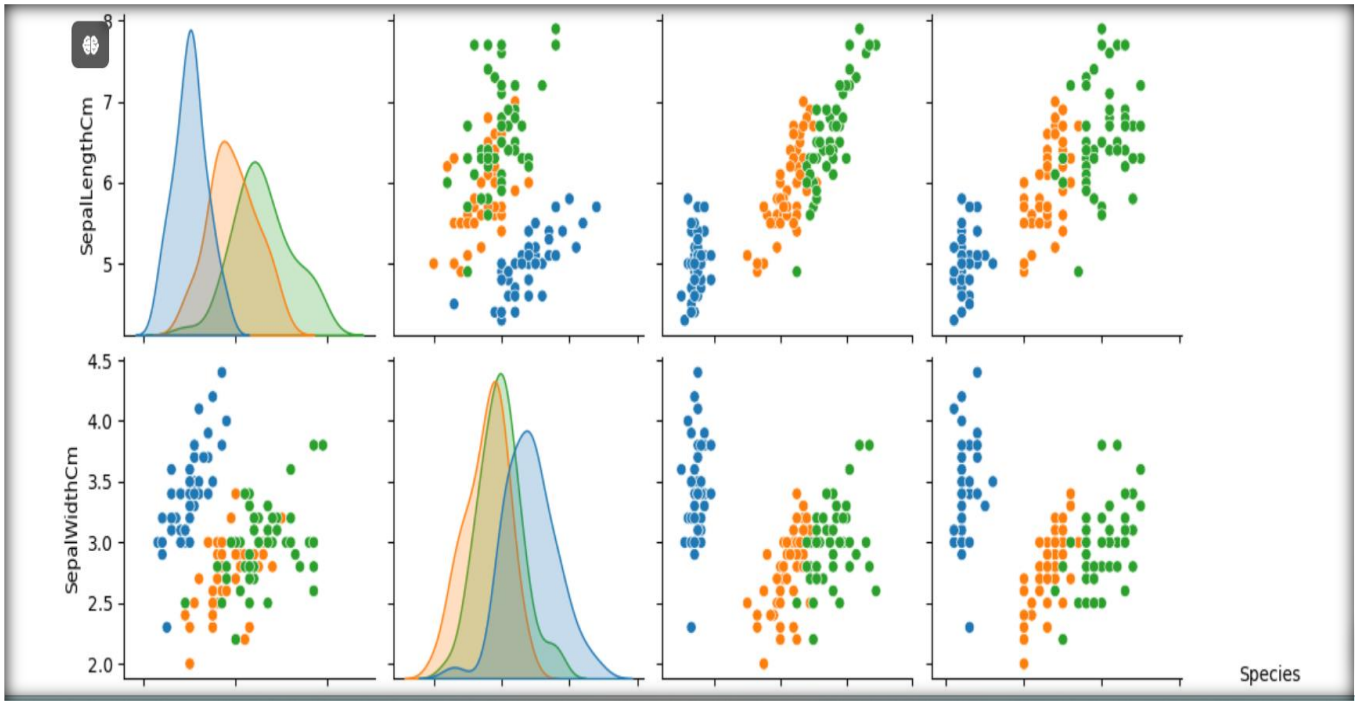
Overlap -----> There is some overlap between **versicolor** and **virginica** in their feature ranges, while **setosa** is more distinctly separated.

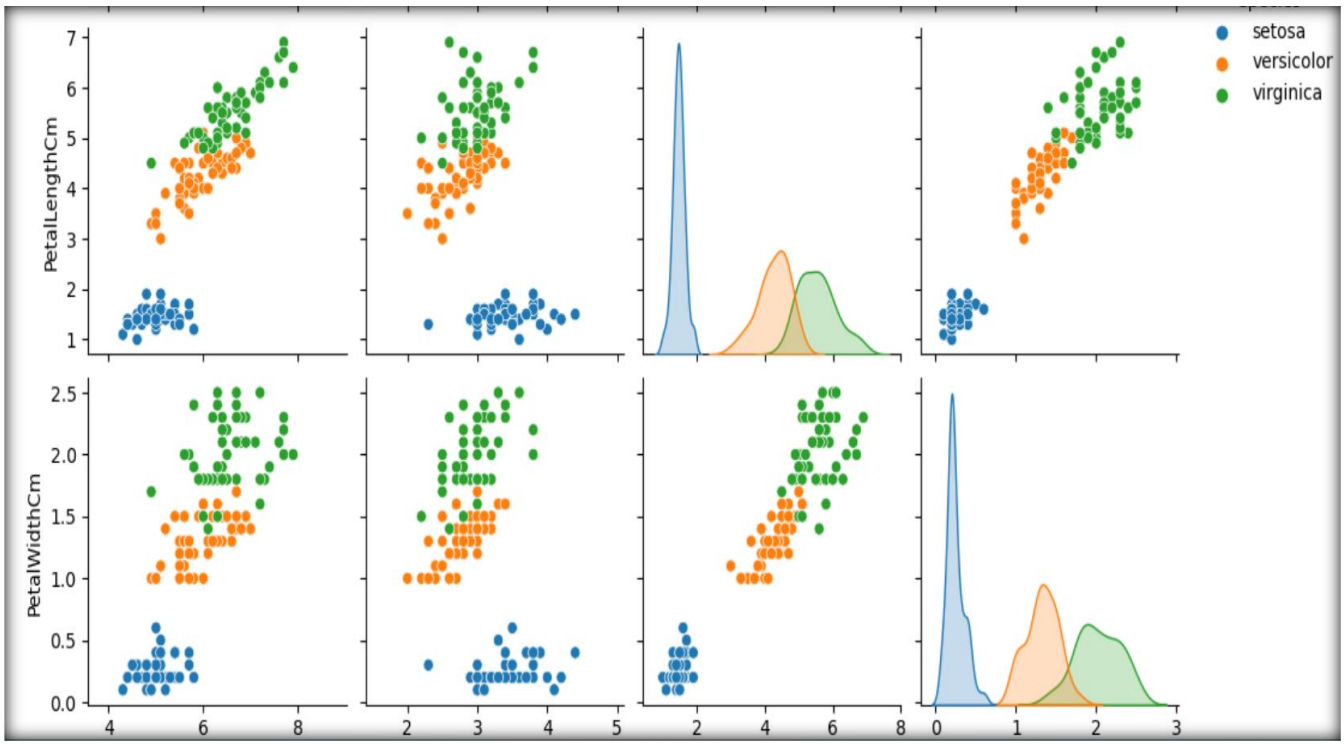
This visualization provides a comprehensive view of the relationships between features and helps to differentiate between the species based on their measurements.

```
# Pairplot to visualize relationships between features and species

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

Pairplot of Species





7.Correlation

- To measure and visualize the strength and direction of relationships between numeric variables, aiding in understanding how variables interact with one another.

```
# Correlation

data.select_dtypes(include="number").corr()
```

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
SepalLengthCm	1.000000	-0.109369	0.871754	0.817954
SepalWidthCm	-0.109369	1.000000	-0.420516	-0.356544
PetalLengthCm	0.871754	-0.420516	1.000000	0.962757
PetalWidthCm	0.817954	-0.356544	0.962757	1.000000

```
s=data.select_dtypes(include="number").corr()
print(s)
```

```
               SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
SepalLengthCm       1.000000     -0.109369       0.871754      0.817954
SepalWidthCm       -0.109369      1.000000      -0.420516     -0.356544
PetalLengthCm       0.871754     -0.420516       1.000000      0.962757
PetalWidthCm        0.817954     -0.356544       0.962757      1.000000
```

The data.select_dtypes(include="number").corr() function computes the correlation matrix for the numerical columns in the dataset. This matrix shows the **pairwise correlation coefficients between each feature,** indicating the **strength and direction of their linear relationships.**

Key observations from the correlation matrix include:

Petal Length and Petal Width -----> These features have a high **positive correlation**, indicating that as petal length increases, petal width tends to increase as well.

Sepal Length and Sepal Width-----> These features have **a negative** correlation, suggesting that larger sepals tend to be narrower.

Petal Dimensions vs. Sepal Dimensions----> Petal length and width have low to moderate correlations with sepal dimensions.

This correlation matrix helps in understanding the relationships between features and can guide feature selection and engineering for further analysis or modelling.

7.1 Heatmap

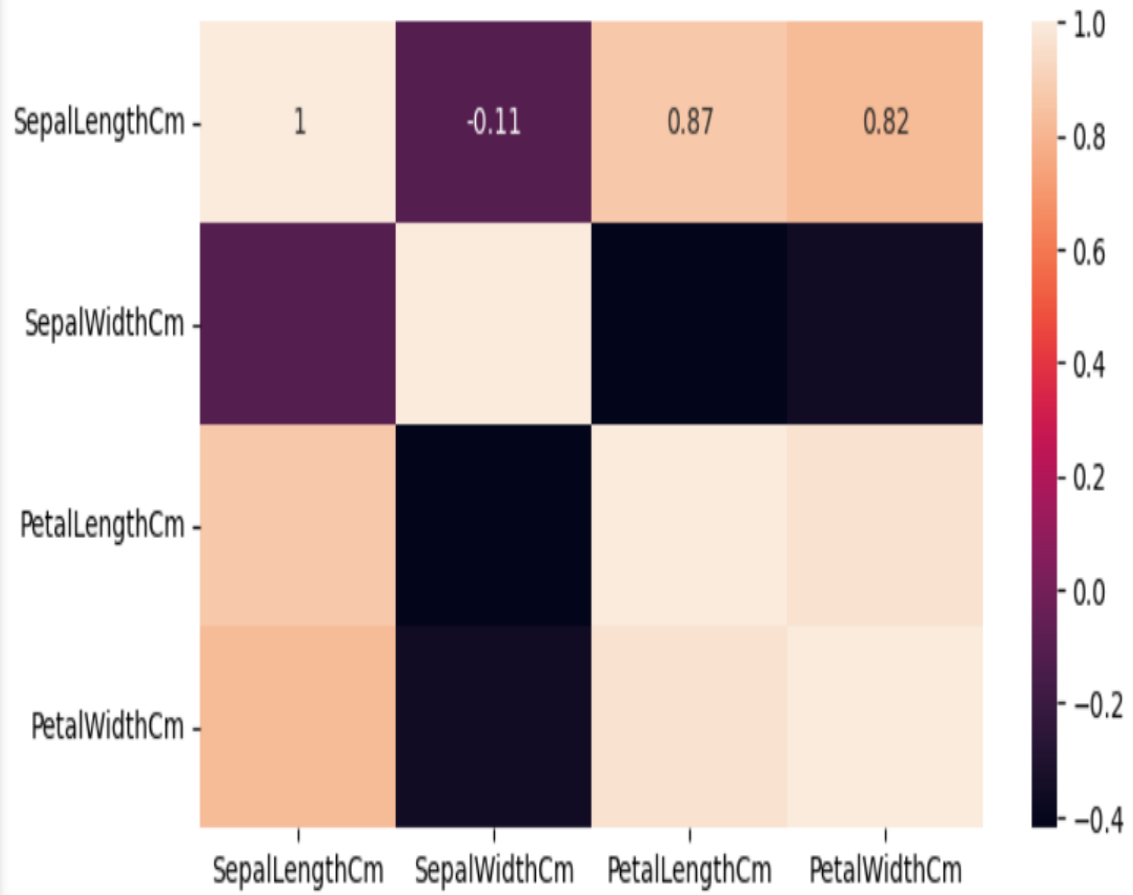
- To visually represent the correlation matrix between numeric variables, making it easier to identify patterns and relationships through color-coded values.

Heatmap

```
# Correlation analysis to identify relationship between parameters

plt.figure(figsize=(8,4))
sns.heatmap(s,annot=True)
```

```
<Axes: >
```



The sns.heatmap() function with the annot=True parameter creates a heatmap of the correlation matrix s, where the correlation coefficients are displayed as annotations within the heatmap cells.

Color Intensity-----> The color intensity represents the strength of the correlation, with **darker colors indicating stronger** correlations.

Annotations -----> The numeric values inside the heatmap cells show the exact correlation coefficients between pairs of features.

Correlation Insights -----> This visualization provides a clear, visual representation of the correlations between numerical features, making it easier to identify strong or weak relationships.

This heatmap is useful for quickly assessing feature relationships and understanding how different variables are interrelated in the dataset.

Interpretation

- A high **positive** correlation between **Petal Width** and **Petal Length**, (0.96)
- A high **positive** correlation between **Petal Length** and **Sepal Length**, (0.87)
- A high **positive** correlation between **Petal Width** and **Sepal Length** (0.81)

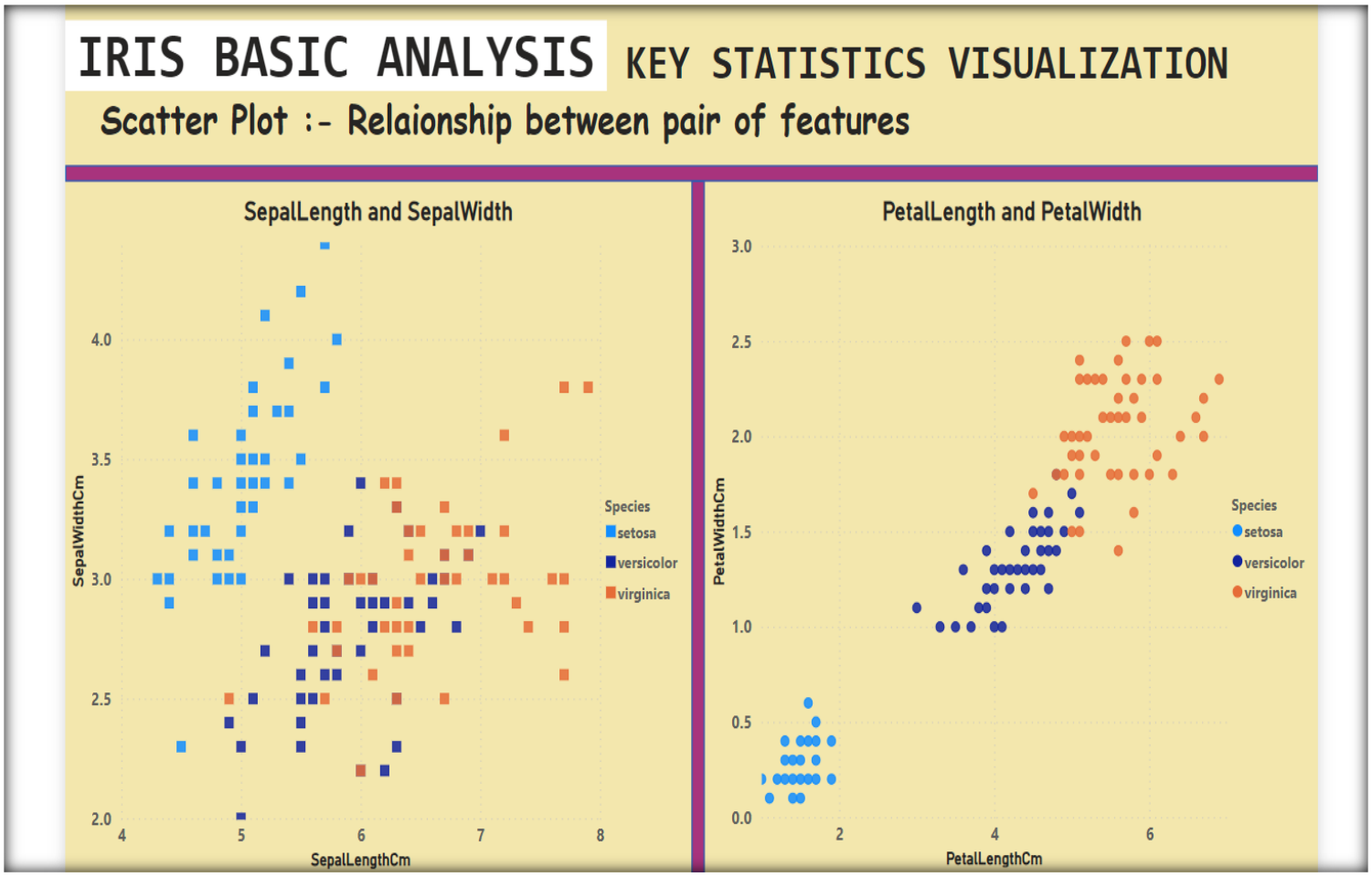
As such, observed correlations between these three attributes:

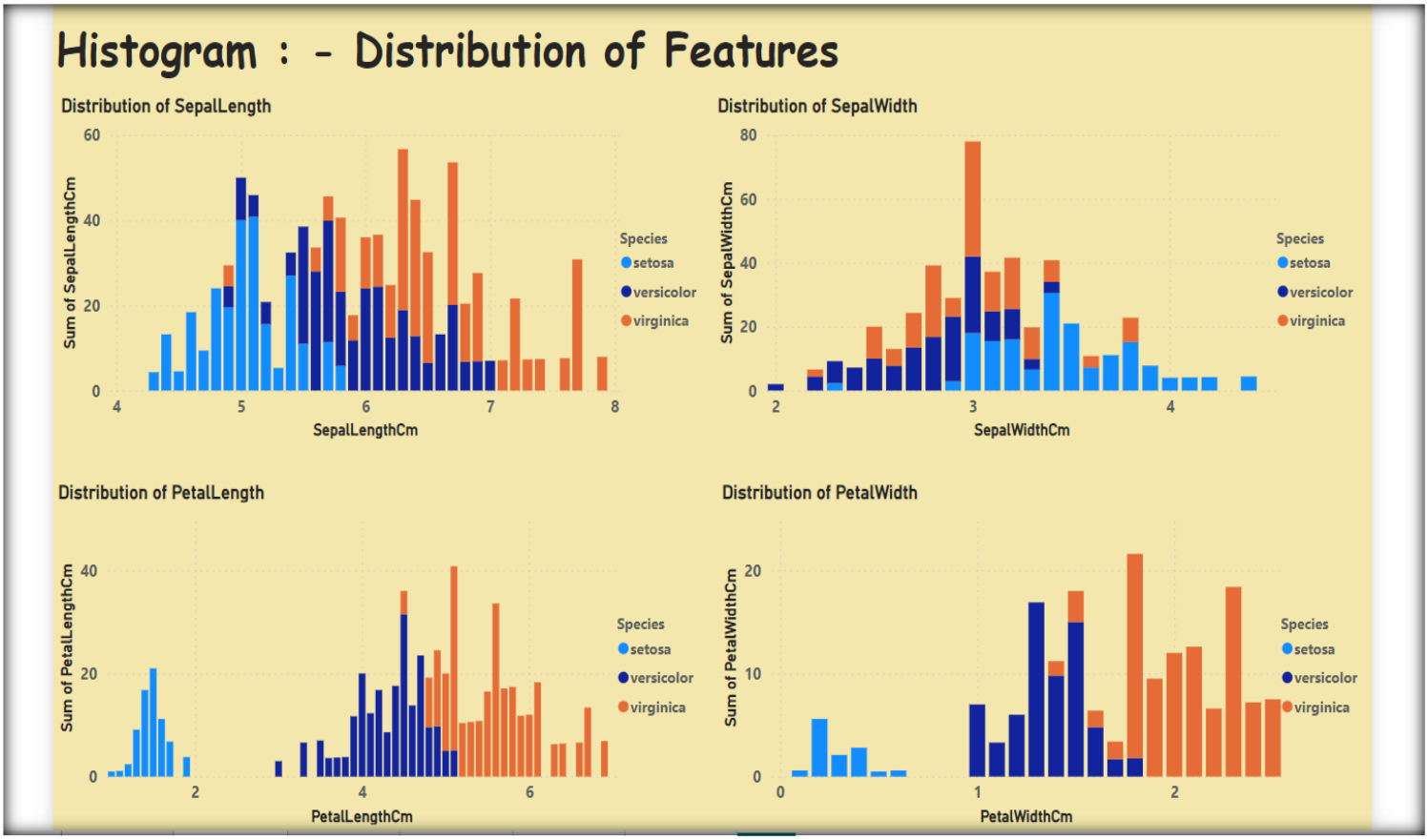
1. Petal Width
2. Petal Length
3. Sepal Length

- +1 variables are correlated,
- -1 inversely correlated.

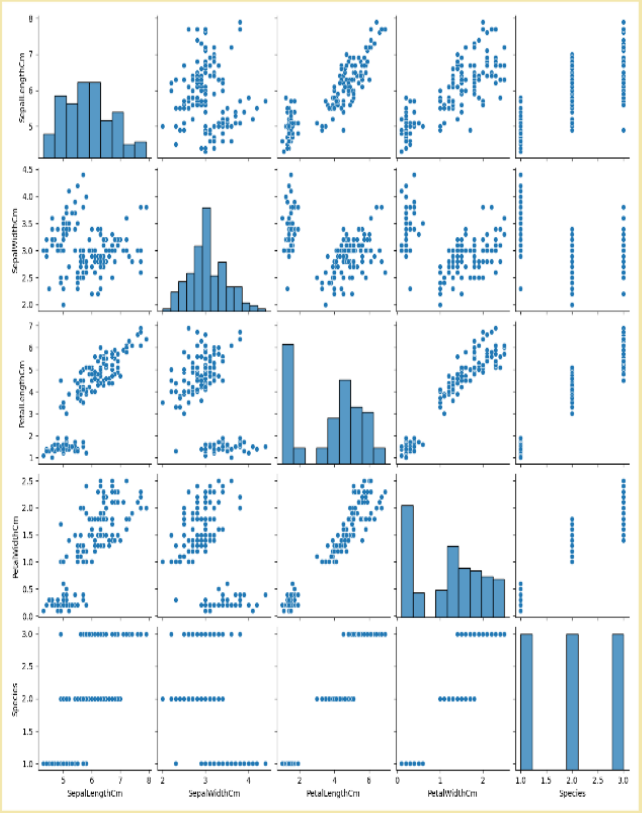
8. *Visualization of the Iris Dataset*

8.1 Using Power BI



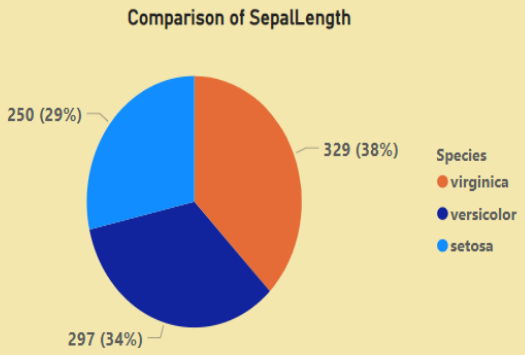


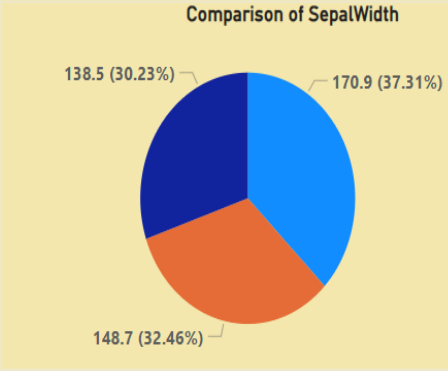
Pair Plot : - Pairwise Relationship and Feature



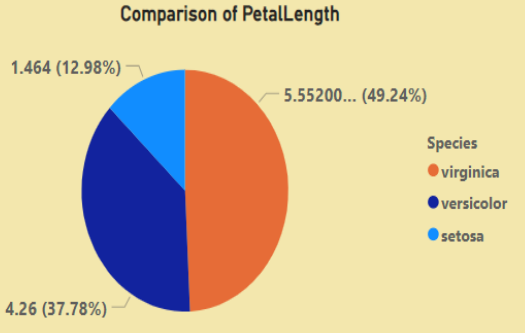
ANALYZE TRENDS & PATTERNS

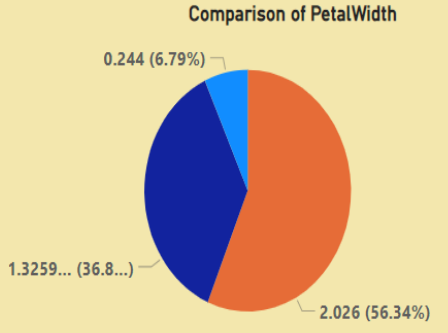
Pie Chart : - Proportion of Categories





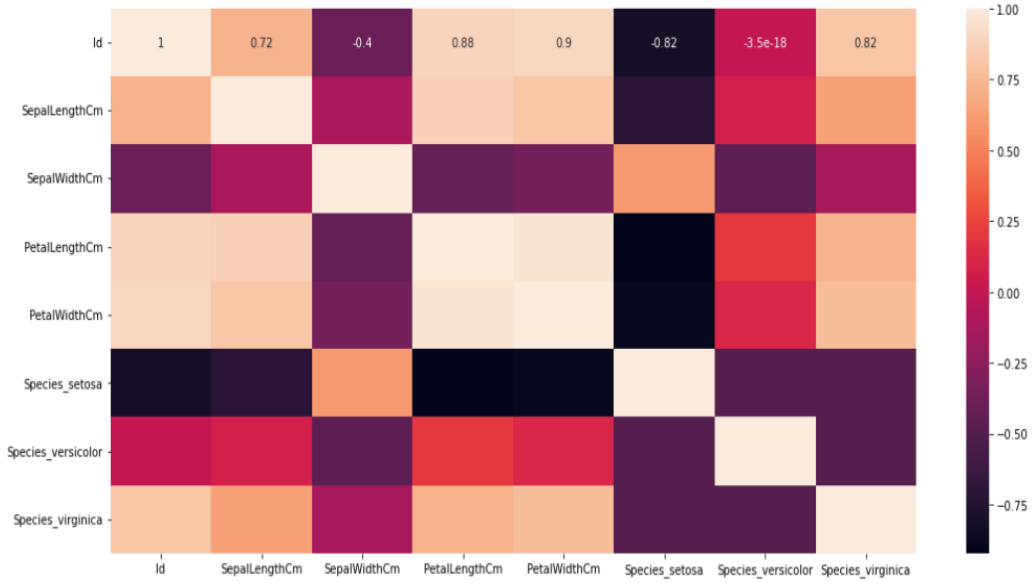
Species
- setosa
- versicolor
- virginica



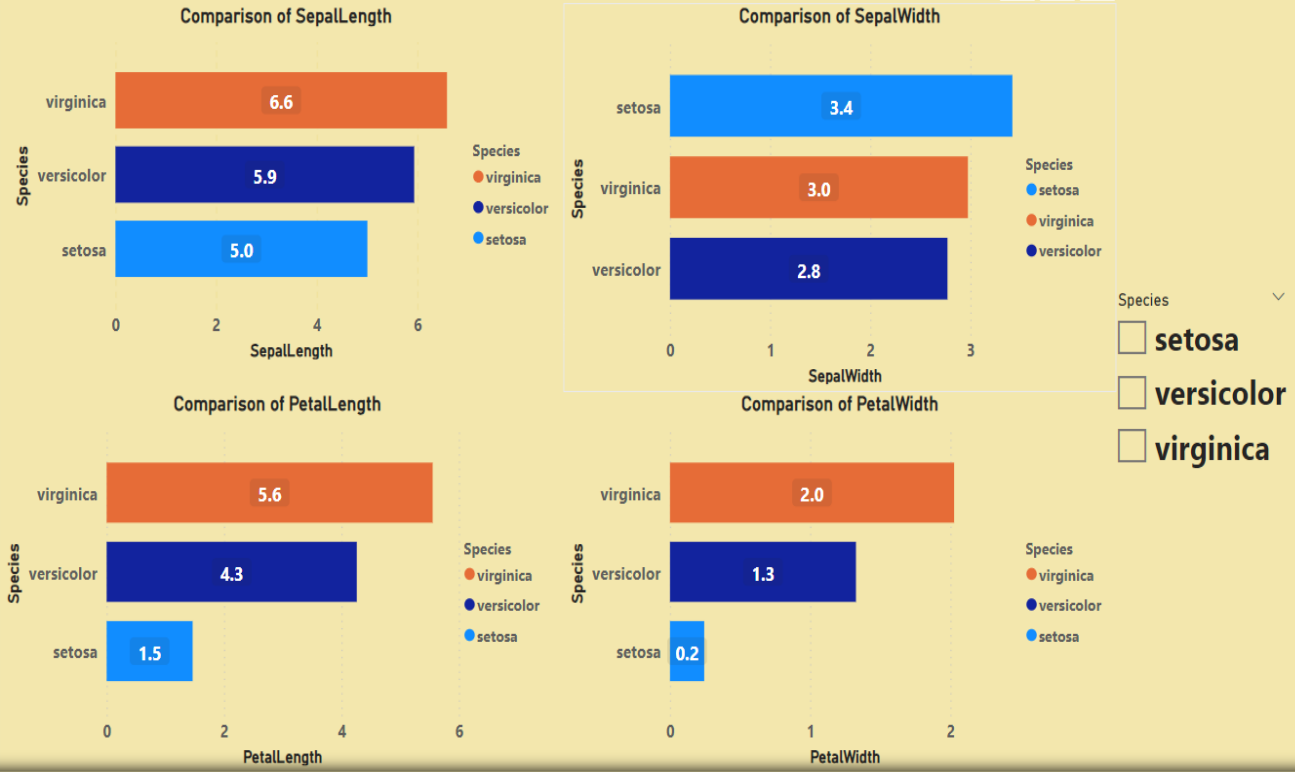


CORRELATION MATRIX

Interrelationships between Features



Bar Chart : - Comparison of Feature Averages



9.Conclution

This analysis provided a comprehensive overview of the Iris dataset. By examining the data structure, handling missing values, performing exploratory data analysis, and visualizing key relationships, gained valuable insights into the dataset. This process aids in understanding the dataset's features and the relationships between them.