IRIS DATASET: EXPLORATORY DATA ANALYSIS

```
# importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

#Setting up the Kaggle API credentials
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/

!kaggle datasets download -d arshid/iris-flower-dataset
```

```
Warning: Your Kaggle API key is readable by other users on this
system! To fix this, you can run 'chmod 600 /root/.kaggle/kaggle.json'
Downloading iris-flower-dataset.zip to /content
  0% 0.00/0.99k [00:00<?, ?B/s]
100% 0.99k/0.99k [00:00<00:00, 2.01MB/s]
```

```
# Unzipping the zipped dataset file
import zipfile
zip_ref = zipfile.ZipFile('/content/iris-flower-dataset.zip')
zip_ref.extractall('/content')
zip_ref.close()

# Loading the dataset
data = pd.read_csv('IRIS.csv')

data.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Gaining information from data

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal_length  150 non-null    float64
```

```
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

Data Insights:

1. All columns are not having any Null Entries
2. Four columns are numerical type
3. Only Single column categorical type

Statistical Insight

```
data.describe()
```

```
       sepal_length  sepal_width  petal_length  petal_width
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
```

Data Insights:

1. Mean values
2. Standard Deviation
3. Minimum Values
4. Maximum Values

Checking For Duplicate Entries

```
data[data.duplicated()]
```

```
     sepal_length  sepal_width  petal_length  petal_width
species
34            4.9          3.1           1.5          0.1     Iris-
setosa
37            4.9          3.1           1.5          0.1     Iris-
setosa
142           5.8          2.7           5.1          1.9  Iris-
virginica
```

There are 3 duplicates, therefore we must check whether each species data set is balanced in no's or no.

Checking the balance

```
data['species'].value_counts()
```

```
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: species, dtype: int64
```

Each species has 50 records each. Therefore we shouldn't delete the entries as it might imbalance the data sets and hence will prove to be less useful for valuable insights.

Checking for Missing Values

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

```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

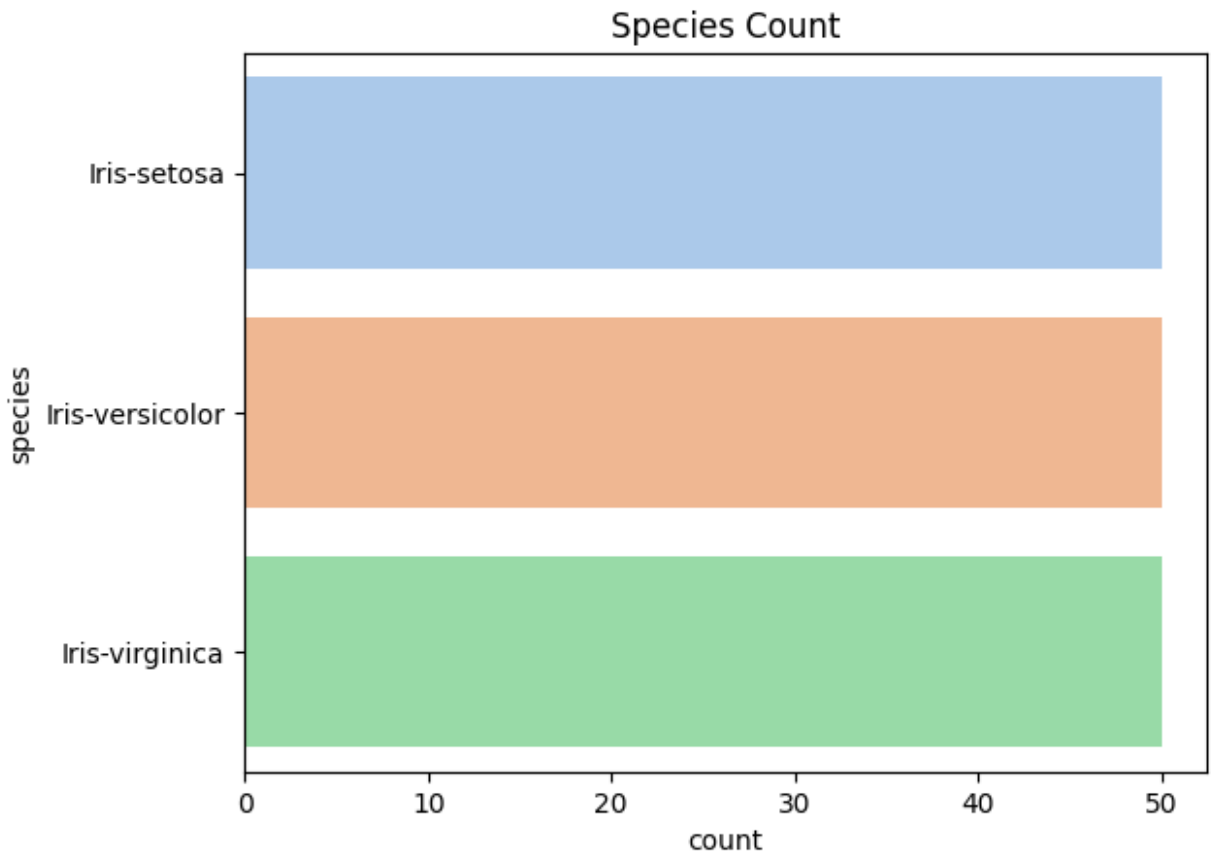
```
##### There are no missing values in our dataset. Hence we proceed
with Data Visualization.
```

Data Visualization

Getting the Species Count using Count Plot

```
plt.title("Species Count")
sns.countplot(data['species'], palette = "pastel")
```

```
<ipython-input-17-0a8eb65e1972>:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `y` variable to `hue` and set
`legend=False` for the same effect.

  sns.countplot(data['species'], palette = "pastel")

<Axes: title={'center': 'Species Count'}, xlabel='count',
ylabel='species'>
```



Data Insight:

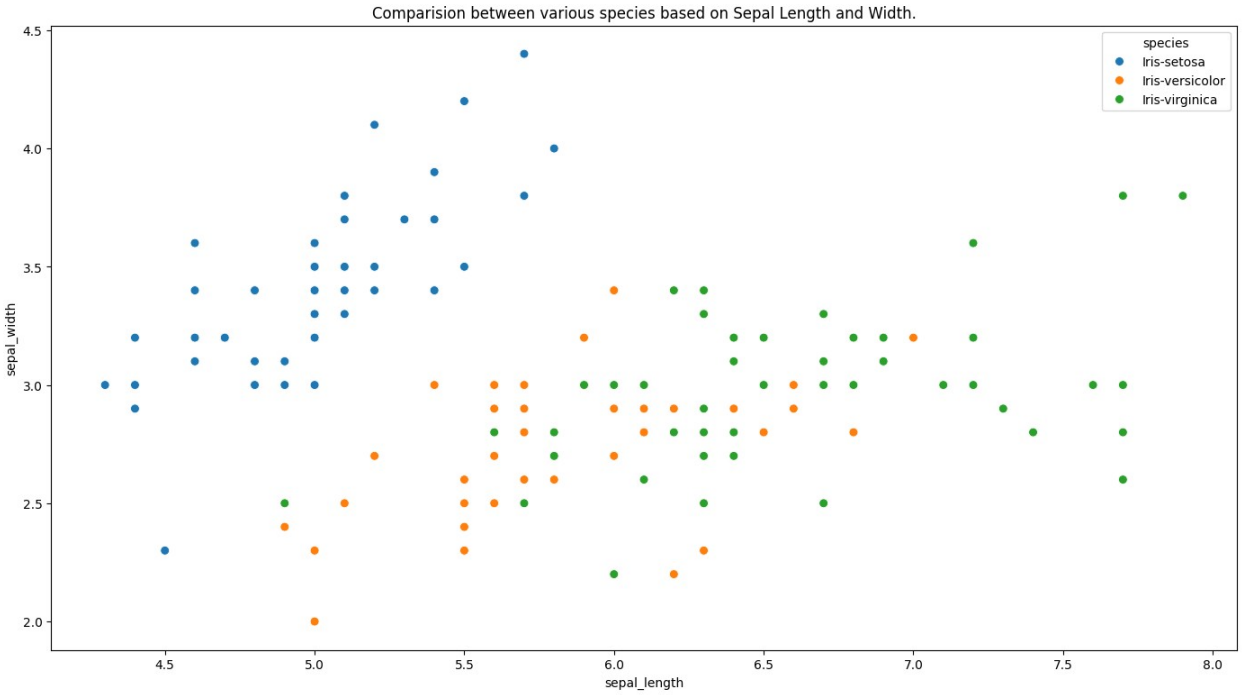
- This further visualizes that species are well balanced
- Each species (Iris virginica, setosa, versicolor) has 50 as it's count.

UNIVARIATE ANALYSIS BY SCATTER PLOT

Comparison between various species based on sepal length and width

```
plt.figure(figsize = (17,9))
plt.title("Comparision between various species based on Sepal Length
and Width.")
sns.scatterplot(data=data, x='sepal_length', y='sepal_width',
hue='species', s=50)
```

```
<Axes: title={'center': 'Comparision between various species based on
Sepal Length and Width.'}, xlabel='sepal_length',
ylabel='sepal_width'>
```



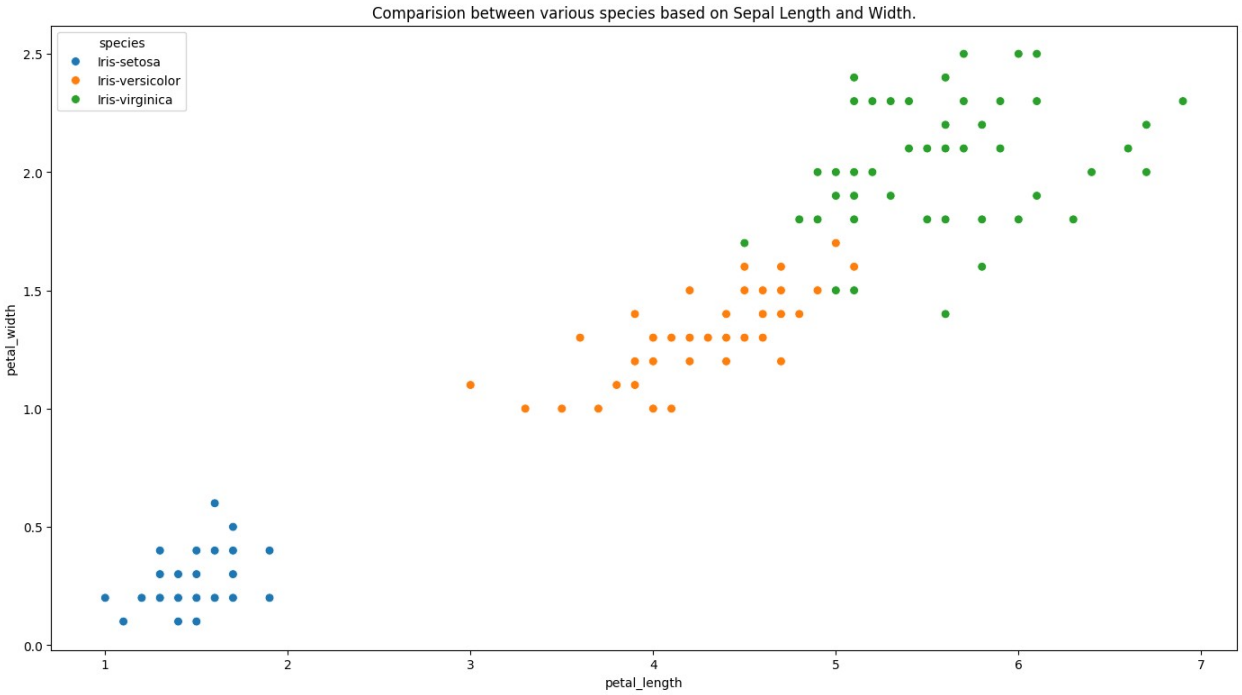
Data Insights:

- Iris Setosa species has smaller sepal length but higher width.
- Versicolor lies in almost middle for length as well as width.
- Virginica has larger sepal lengths and smaller sepal widths.

Comparison between various species based on petal length and width

```
plt.figure(figsize = (17,9))
plt.title("Comparision between various species based on Sepal Length
and Width.")
sns.scatterplot(data=data, x='petal_length', y='petal_width',
hue='species', s=50)
```

```
<Axes: title={'center': 'Comparision between various species based on
Sepal Length and Width.'}, xlabel='petal_length',
ylabel='petal_width'>
```



Data Insights

- Setosa species have the smallest petal length as well as petal width
- Versicolor species have average petal length and petal width
- Virginica species have the highest petal length as well as petal width

Checking the Mean, Median values of each Species and applying Box plot.

```
data.groupby('species').agg(["mean","median"])
```

```
                 sepal_length        sepal_width        petal_length
\
                         mean median        mean median          mean
median
species

Iris-setosa             5.006    5.0       3.418    3.4         1.464
1.50
Iris-versicolor         5.936    5.9       2.770    2.8         4.260
4.35
Iris-virginica          6.588    6.5       2.974    3.0         5.552
5.55

                 petal_width
                        mean median
species
Iris-setosa            0.244    0.2
Iris-versicolor        1.326    1.3
Iris-virginica         2.026    2.0
```

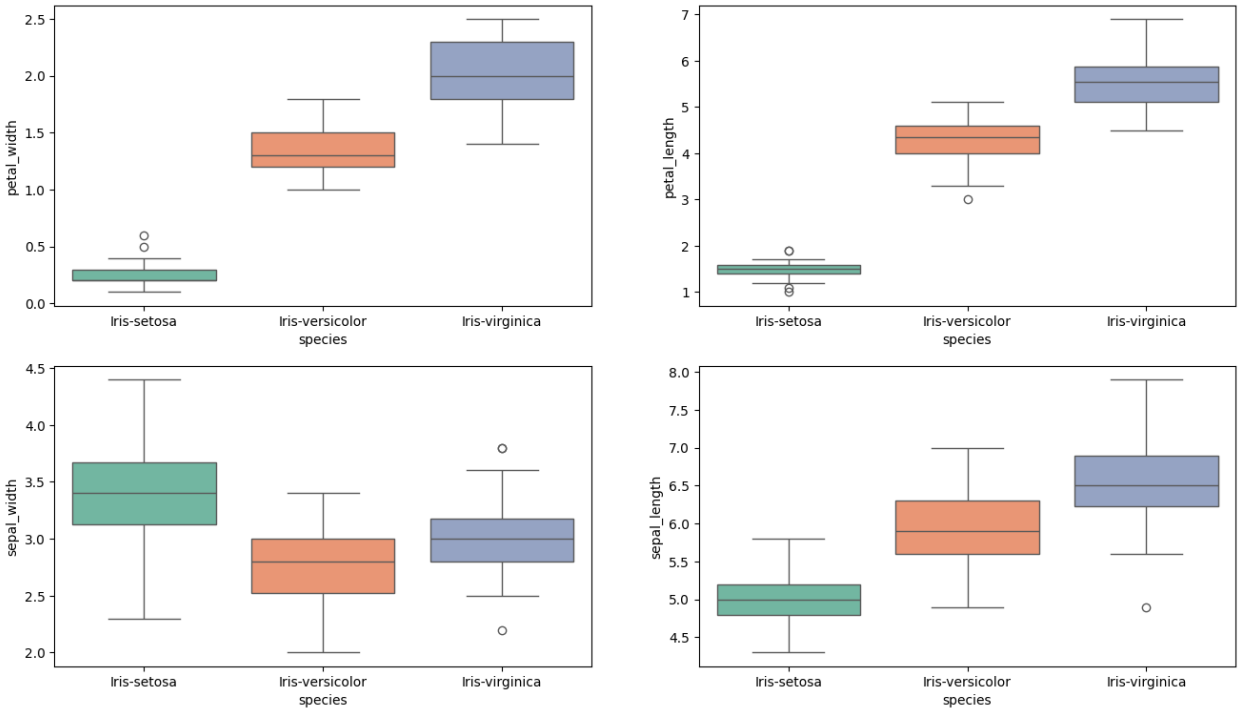
Data insights:

- The mean length of the sepal is highest for Iris-virginica followed by Iris-versicolor and then Iris-setosa.
- The mean width of the sepal is highest for Iris-setosa followed by Iris-virginica and then Iris-versicolor.
- The mean length of the petal is highest highest for Iris-virginica followed by Iris-versicolor and then Iris-setosa.
- The mean width of the petal is highest highest for Iris-virginica followed by Iris-versicolor and then Iris-setosa.

Box plot to know about the distribution

```
fig, axes = plt.subplots(2,2,figsize =(16,9))
sns.boxplot(data= data, x = 'species', y = 'petal_width', orient= 'v',
ax= axes[0,0], palette='Set2')

sns.boxplot(data= data, x = 'species', y = 'petal_length', orient=
'v', ax= axes[0,1], palette='Set2')

sns.boxplot(data= data, x = 'species', y = 'sepal_width', orient= 'v',
ax= axes[1,0],palette='Set2')

sns.boxplot(data= data, x = 'species', y = 'sepal_length', orient=
'v', ax= axes[1,1],palette='Set2')
plt.show()


<ipython-input-36-2ef2d4214bd7>:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.boxplot(data= data, x = 'species', y = 'petal_width', orient=
'v', ax= axes[0,0], palette='Set2')
<ipython-input-36-2ef2d4214bd7>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.boxplot(data= data, x = 'species', y = 'petal_length', orient=
'v', ax= axes[0,1], palette='Set2')
<ipython-input-36-2ef2d4214bd7>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.
```

```
  sns.boxplot(data= data, x = 'species', y = 'sepal_width', orient=
'v', ax= axes[1,0],palette='Set2')
<ipython-input-36-2ef2d4214bd7>:8: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.boxplot(data= data, x = 'species', y = 'sepal_length', orient=
'v', ax= axes[1,1],palette='Set2')
```



Data insights:

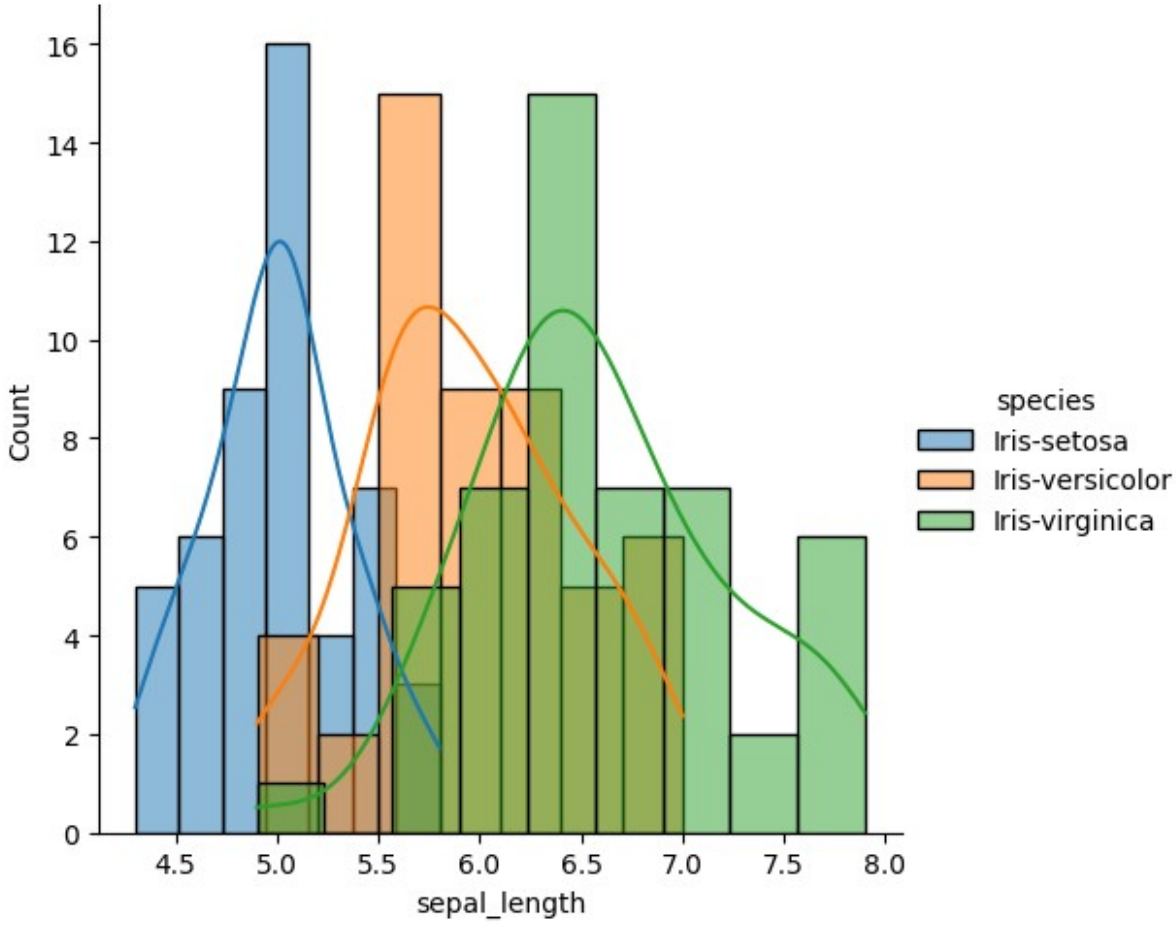
- Setosa is having smaller feature and less distributed
- Versicolor is distributed in a average manner and average features
- Virginica is highly distributed with large no .of values and features
- Clearly the mean/ median values are being shown by each plots for various features(sepal length & width, petal length & width)

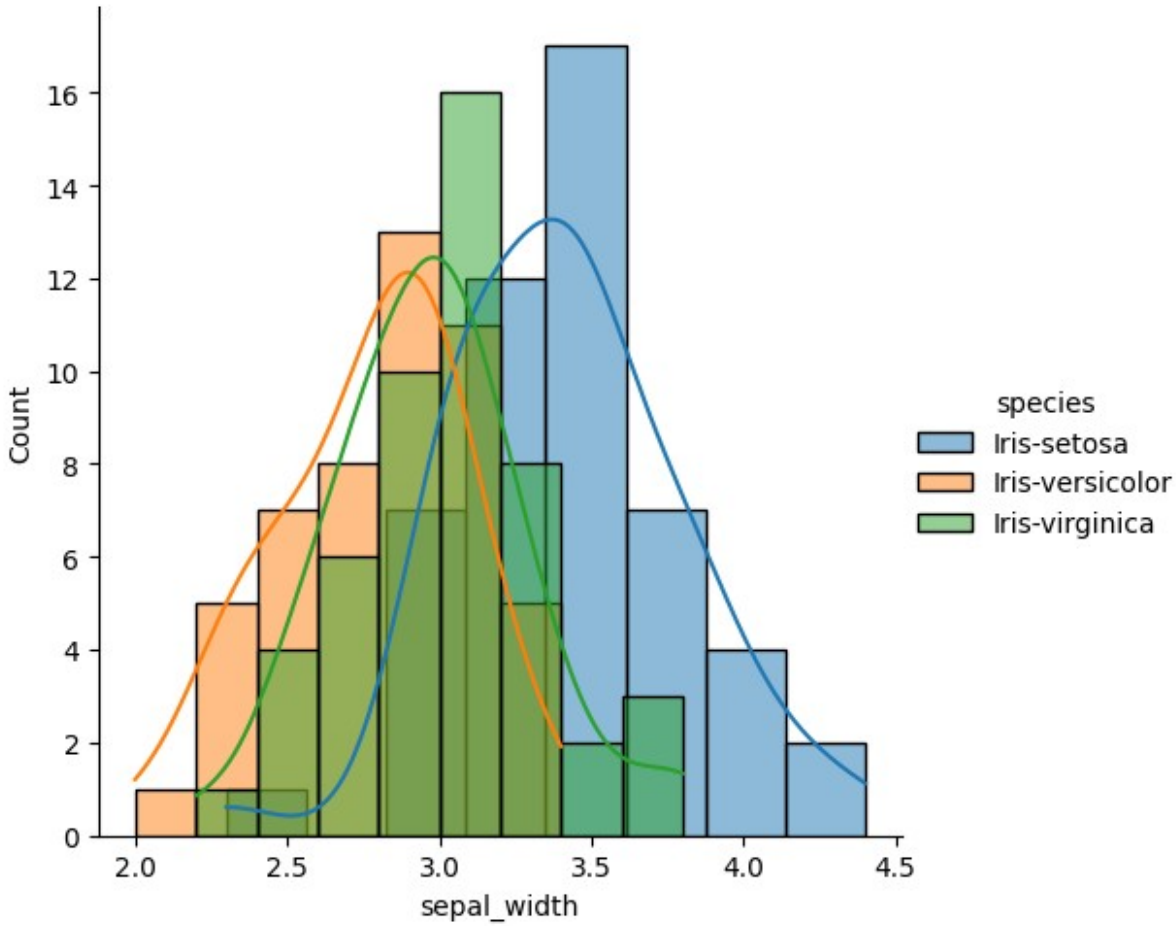
Plotting the Histogram & Probability Density Function (PDF)

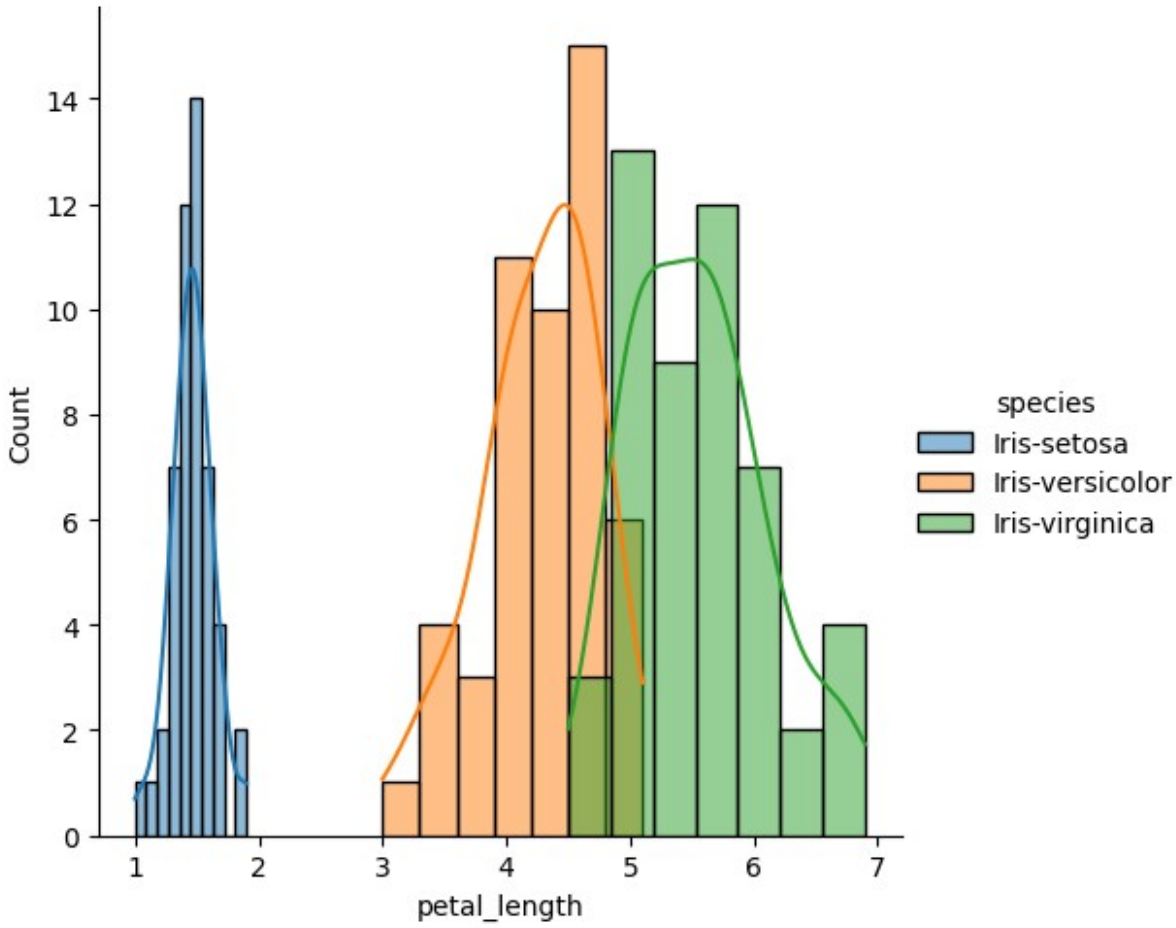
Here we are plotting the probability density function(PDF) with each feature as a variable on X-axis and it's histogram and corresponding kernel density plot on Y-axis.

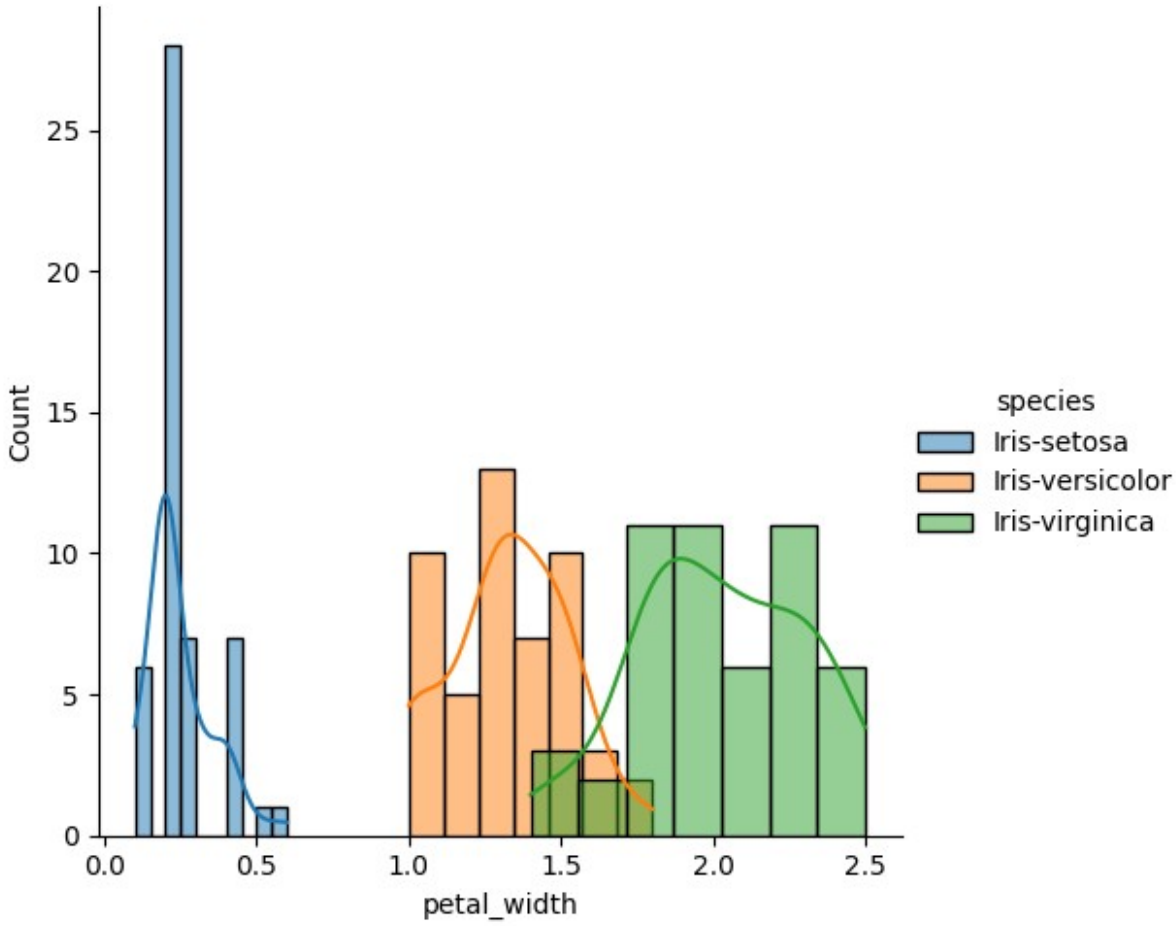
```
import seaborn as sns
import matplotlib.pyplot as plt
```

```
# Features to plot and their corresponding titles
features = ["sepal_length", "sepal_width", "petal_length",
"petal_width"]
titles = ["Sepal Length Distribution", "Sepal Width Distribution",
          "Petal Length Distribution", "Petal Width Distribution"]

# Create a FacetGridA for each feature
for feature, title in zip(features, titles):
    g = sns.FacetGrid(data, hue="species", height=5)
    g.map(sns.histplot, feature, kde=True)
    g.add_legend()
    g.set_titles(title)

# Show the plot
plt.show()
```




Count
sepal_width
species
Iris-setosa
Iris-versicolor
Iris-virginica


Count
petal_length
species
Iris-setosa
Iris-versicolor
Iris-virginica



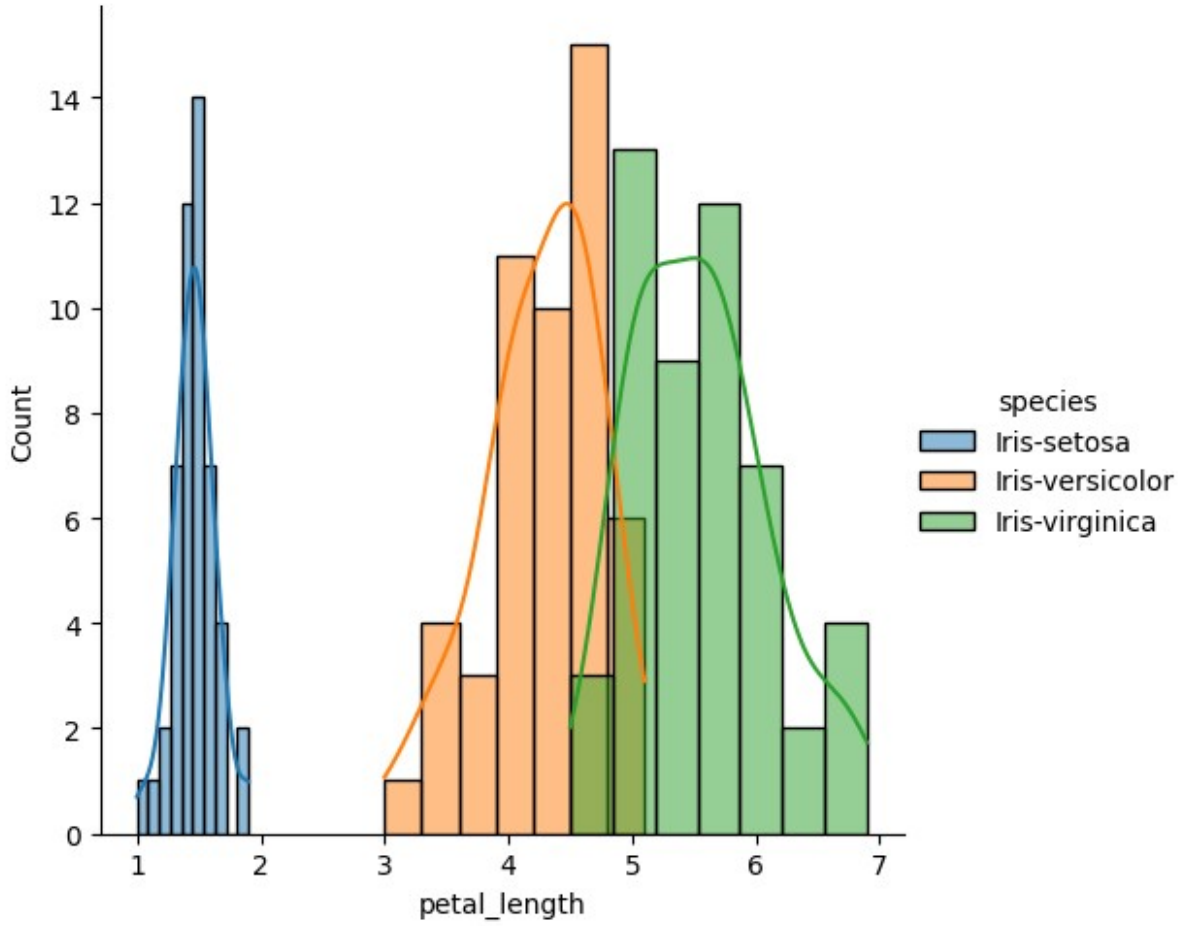
Data Insights:

- Plot 1 shows that there is a significant amount of overlap between the species on sepal length, so it is not an effective Classification feature.
- Plot 2 shows that there is even higher overlap between the species on sepal width, so it is not an effective Classification feature
- Plot 3 shows that petal length is a good Classification feature as it clearly separates the species . The overlap is extremely less (between Versicolor and Virginica) , Setosa is well separated from the rest two.
- Just like Plot 3, Plot 4 also shows that petal width is a good Classification feature . The overlap is significantly less (between Versicolor and Virginica) , Setosa is well separated from the rest two

Choosing Plot 3 (Classification feature as Petal Length)to distinguish among the species

```
# Create a FacetGrid with the Iris dataset
g = sns.FacetGrid(data, hue="species", height=5)
```

```
# Map a distribution plot onto the FacetGrid for the "sepal_length"
column
g.map(sns.histplot, "petal_length", kde = True)

# Add a legend
g.add_legend()

# Show the plot
plt.show()
```



The pdf curve of Iris Setosa ends roughly at 2.1

Data Insights:

- The pdf curve of Iris Setosa ends roughly at 2.1
- If petal length < 2.1, then species is Iris Setosa.
- The point of intersection between pdf curves of Versicolor and Virginica is roughly at 4.8.
- If petal length > 2.1 and petal length < 4.8 then species is Iris Versicolor.

- If petal length > 4.8 then species is Iris Virginica.

CONCLUSION:

Through the utilization of diverse graphical techniques like Count Plot, Scatter Plot, Box Plot, and Histogram, we conducted a comprehensive exploration of the Iris dataset, uncovering valuable insights about its features and distributions. Count Plot provided an overview of species distribution, while Scatter Plot facilitated the examination of relationships between features. Box Plot enabled us to analyze feature distributions across species, identifying potential outliers and variations. Lastly, Histogram offered detailed insights into individual feature distributions, notably revealing distinct clusters in petal length distribution across species. From this analysis, it became evident that petal length could serve as a significant discriminative feature for classifying Iris species. This thorough exploration not only deepened our understanding of the dataset but also provided valuable insights for potential feature selection and modeling tasks.