Exploratory Data Analysis (EDA) on Iris Dataset

Project ID - #CC69855

Name - Jyoti Ghaytadak

Project Level - Entry Level

Assigned By- CodeClause Internship

Project Details-

Aim -

Conduct exploratory data analysis on the famous Iris dataset to understand its characteristics and relationships between features.

Description -

Use libraries like Pandas, Matplotlib, and Seaborn to visualize patterns, distributions, and relationships in the Iris dataset

Technologies -

Python, Pandas, Matplotlib, Seaborn

Objective -

To classify the species of iris flower on the basis of data.

In [26]:

```
# Importing relevant libraries
```

In [27]:

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

In [28]:

```
# Loading data
```

In [29]:

```
iris_data = pd.read_csv(r"C:\Users\C ZONE\Downloads\iris.csv")
iris_data
```

Out[29]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

In [30]:

```
## Getting the head of data
iris_data.head()
```

Out[30]:

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

In [31]:

```
## Getting the columns of data set
iris_data.columns
```

Out[31]:

```
Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width',
       'species'],
      dtype='object')
```

In [32]:

```
## Renaming my columns name
iris_data.rename(columns={'sepal_length':'sepallength','sepal_width':'sepalwidt
                          'petal_width':'petalwidth'},inplace=True)
```

In [33]:
```
# Statistics about dataset
```

In [34]:
```
iris_data.describe()
```

Out[34]:

	sepallength	sepalwidth	petallength	petalwidth
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

In [35]:
```
# Gaining information from data
```

In [36]:
```
iris_data.info()
```

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

In [37]:
```
iris_data.shape
```

Out[37]:
```
(150, 5)
```

In [38]:
```
# checking for null values
```

In [39]:
```
iris_data.isnull().sum()
```

Out[39]:
```
sepallength    0
sepalwidth     0
petallength    0
petalwidth     0
species        0
dtype: int64
```

In [40]:

```
# Checking For Duplicate Entries
```

In [41]:

```
iris_data[iris_data.duplicated()]
```

Out[41]:

	sepallength	sepalwidth	petallength	petalwidth	species
34	4.9	3.1	1.5	0.1	setosa
37	4.9	3.1	1.5	0.1	setosa
142	5.8	2.7	5.1	1.9	virginica

In [42]:

```
# Checking the balance
```

In [43]:

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

Out[43]:

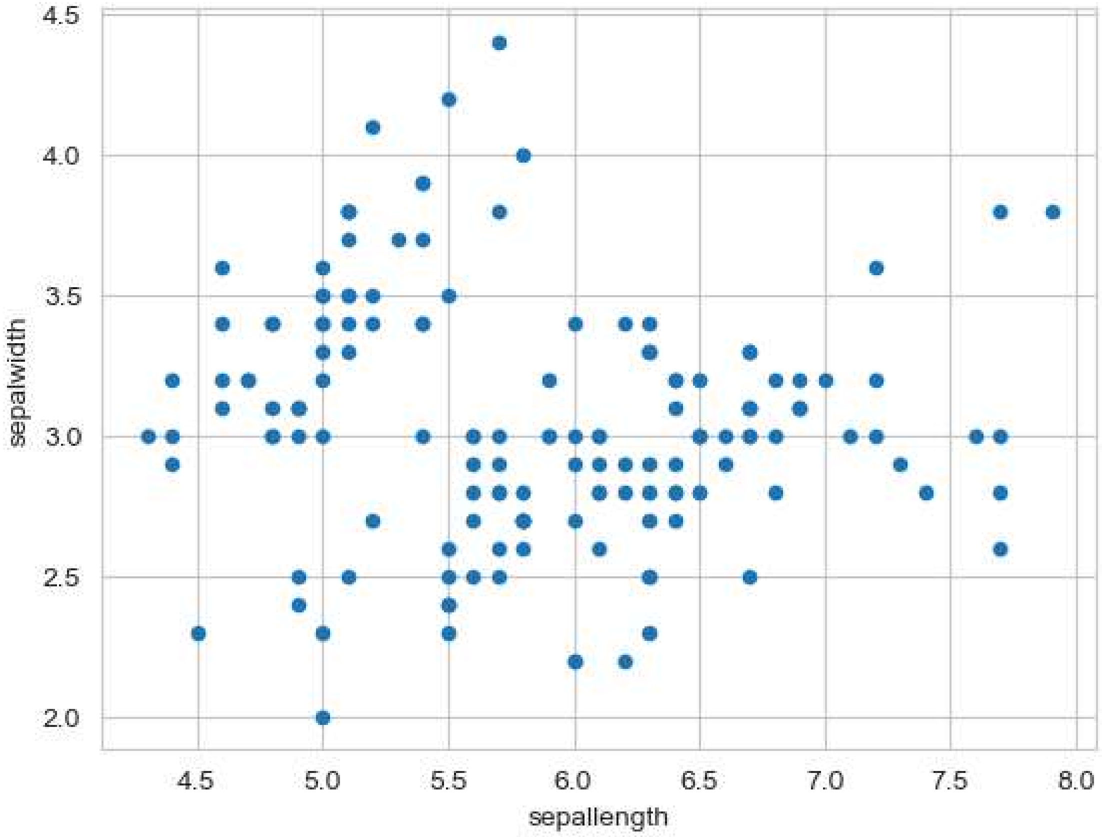
```
setosa        50
versicolor    50
virginica     50
Name: species, dtype: int64
```

Data Visualization

2-D Scatter plot

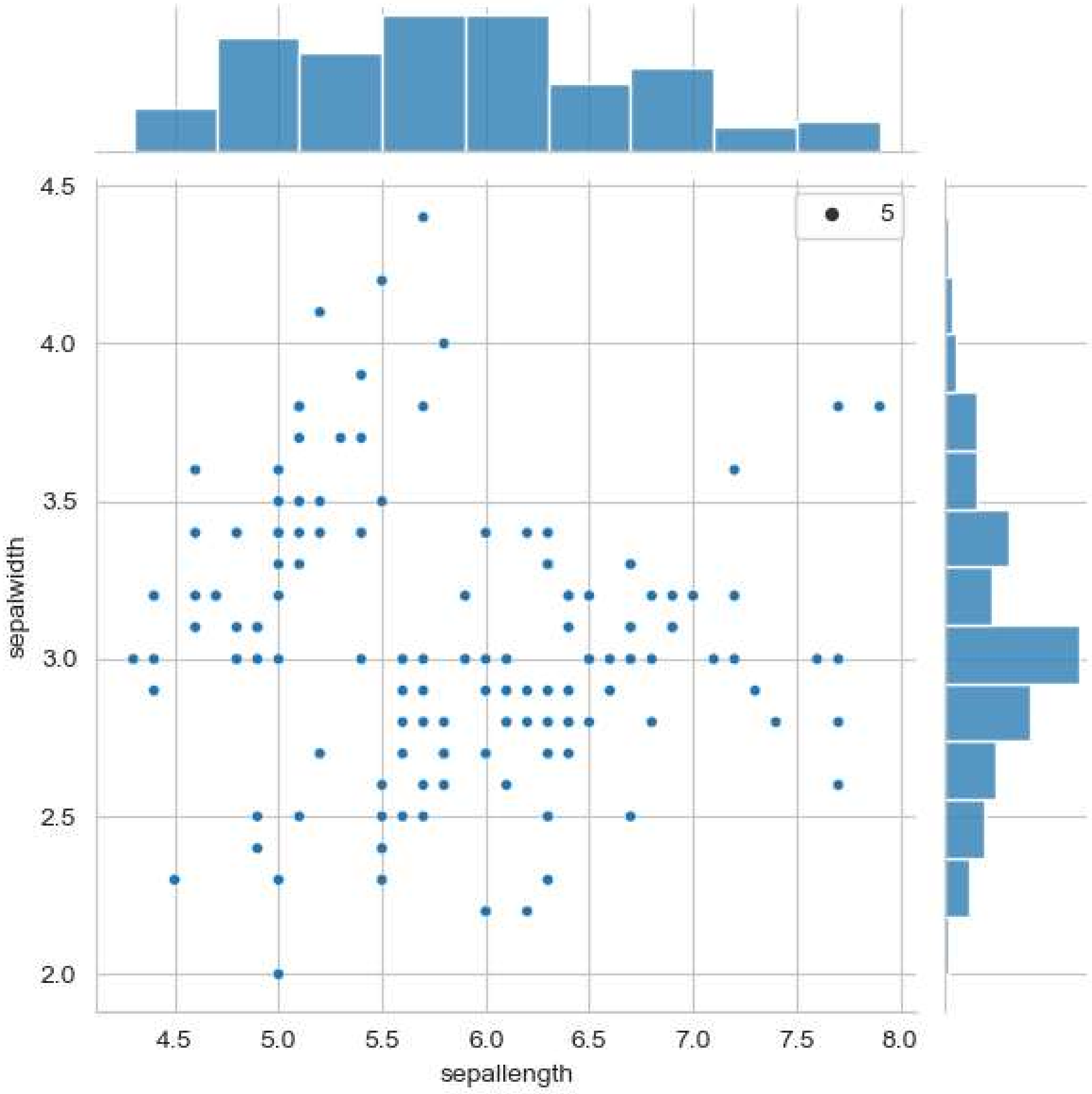
In [44]:

```
iris_data.plot(kind='scatter',x='sepallength',y='sepalwidth')
plt.show()
```



In [45]:
```
sns.jointplot(x="sepallength",y="sepalwidth",data=iris_data,size=5)
P
```

Out[45]:
```
<seaborn.axisgrid.JointGrid at 0x1a54aeb35b0>
```



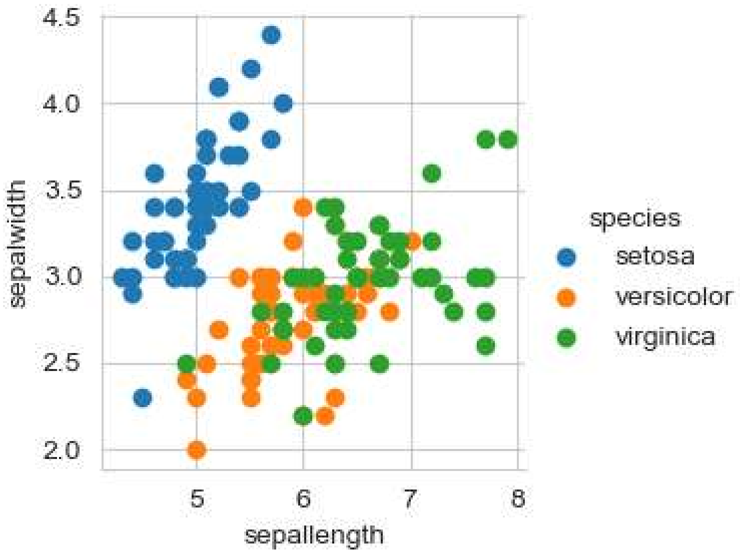
In [46]:
```
# We cannot make much sense out of it
```

In [47]:

```
sns.set_style("whitegrid");
sns.FacetGrid(iris_data,hue="species") \
   .map(plt.scatter, "sepallength","sepalwidth")\
   .add_legend()
```

Out[47]:

```
<seaborn.axisgrid.FacetGrid at 0x1a54b0b7a00>
```

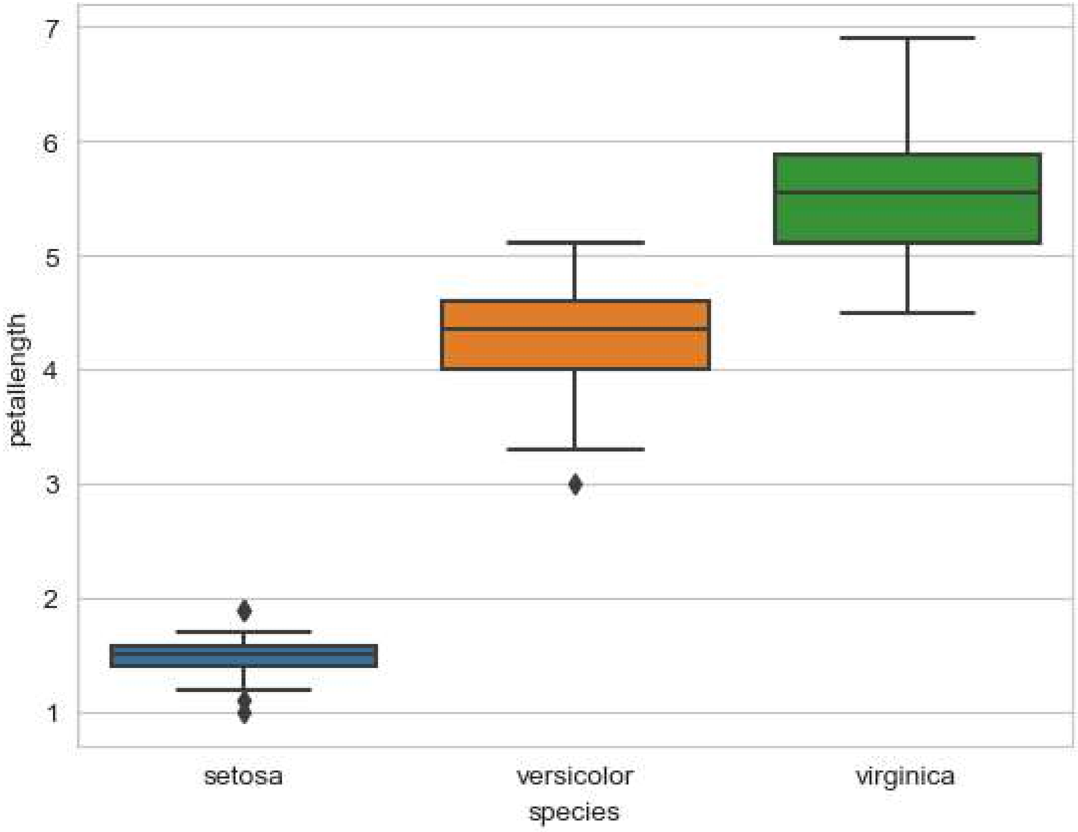


In [48]:

```
sns.boxplot(x="species",y="petallength",data=iris_data)
```

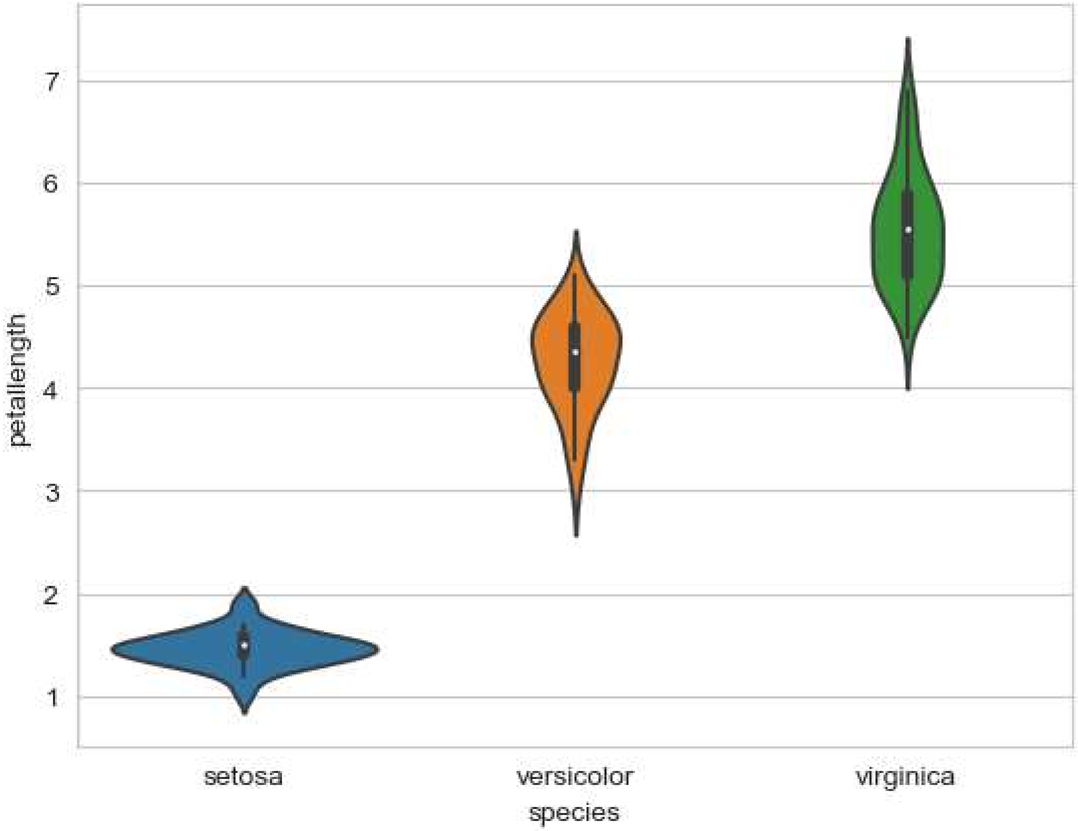
Out[48]:

```
<Axes: xlabel='species', ylabel='petallength'>
```



In [49]:
```
sns.violinplot(x="species",y="petallength",data=iris_data,size=6)
```

Out[49]:
```
<Axes: xlabel='species', ylabel='petallength'>
```

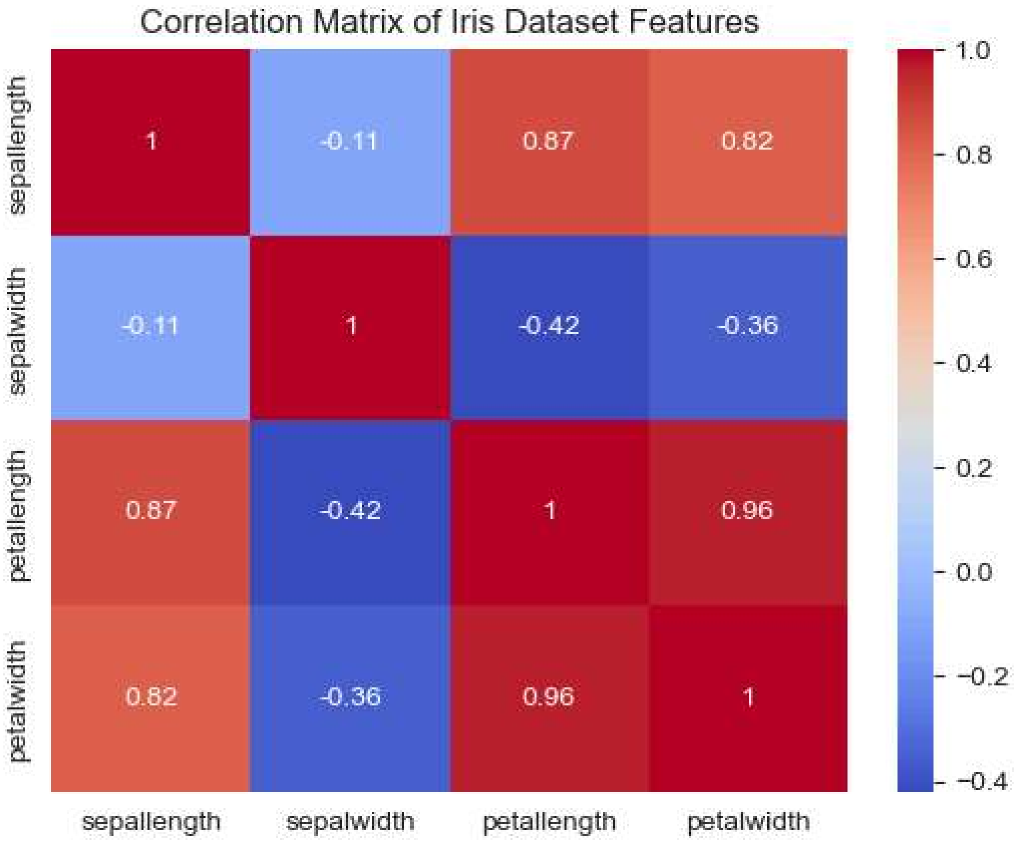


In [54]:
```
#Calculate correlations
correlation_matrix = iris_data.corr()
```

```
C:\Users\C ZONE\AppData\Local\Temp\ipykernel_11108\301716142.py:2: FutureWarn
ing: The default value of numeric_only in DataFrame.corr is deprecated. In a
future version, it will default to False. Select only valid columns or specif
y the value of numeric_only to silence this warning.
  correlation_matrix = iris_data.corr()
```

In [55]:

```python
# Visualize the correlation matrix using a heatmap
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix of Iris Dataset Features')
plt.show()
```



conclusion

- The Exploratory Data Analysis (EDA) of the Iris dataset in Python provides invaluable insights into the dataset's structure, distribution, and relationships. The EDA process involves various techniques, including visualizing the target column, understanding relationships between variables, creating histograms, handling correlation, and managing outliers.
- The use of Python and its libraries like Pandas, Seaborn, and Matplotlib, makes EDA an efficient and insightful process. This process is not only applicable to the Iris dataset but also serves as a blueprint for analyzing other datasets in various fields. EDA remains a cornerstone in data science, providing the groundwork for informed decision-making and advanced analytical studies.