In [1]:
```
pip install ydata-profiling
```

```
Note: you may need to restart the kernel to use updated packages.
ERROR: pip's dependency resolver does not currently take into account all the package requirements. Whether or not this matter depends on your use case.
ERROR: No matching distribution found for collections
```

In [2]:
```
# Importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, classification_report
```

In [3]:
```
# to read the dataflie
df = pd.read_csv('iris.data')
df.head()
df.tail()
df.info()
df.describe()
```

Out[3]:

	5.1	3.5	1.4	0.2	Iris-setosa
0	4.9	3.0	1.4	0.2	Iris-setosa
1	4.7	3.2	1.3	0.2	Iris-setosa
2	4.6	3.1	1.5	0.2	Iris-setosa
3	5.0	3.6	1.4	0.2	Iris-setosa
4	5.4	3.9	1.7	0.4	Iris-setosa

	5.1	3.5	1.4	0.2	Iris-setosa
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 149 entries, 0 to 148
Data columns (total 5 columns):
 #   Column       Non-Null Count  dtype
---  ------       --------------  -----
 0   5.1          149 non-null    float64
 1   3.5          149 non-null    float64
 2   1.4          149 non-null    float64
 3   0.2          149 non-null    float64
 4   Iris-setosa  149 non-null    object
dtypes: float64(4), object(1)
memory usage: 5.9 KB
```

In [4]:
```
# to get summarized statistical info / desc analysis
df.describe()
```

Out[4]:

	5.1	3.5	1.4	0.2
count	149.000000	149.000000	149.000000	149.000000
mean	5.848322	3.051007	3.774497	1.205369
std	0.828066	0.433594	1.764420	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.400000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

```
# to check the dataflie is balanced or not
df.iloc['Species'].value_counts()
```

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

```
# to count nul values
df.iloc[nul].value_counts()/ #The above data tells us that there are no missing values in the dataset. So, we can continue with eda and then prediction using
```

Out[5]:

```
SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  Species
False          False         False          False         150     150
Name: count, dtype: int64
```

In [6]:
```
type(df.nul)
```

```
pandas.core.frame.DataFrame
```

In [7]:
```
df.iloc[nul].sum()
```

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

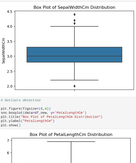
In [8]:
```
# to get summarized statistical info / desc analysis
df.iloc[desc]
```

Out[8]:

	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

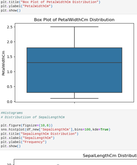
```
# Outliers detection
```

In [9]:
```
plt.figure(figsize=(8,6))
sns.boxplot(dataframe, y='SepalLengthCm')
plt.title('Box Plot of SepalLengthCm Distribution')
plt.xlabel('SepalLengthCm')
plt.show()
```



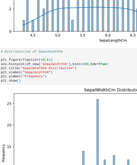
In [10]:
```
# Outliers detection
```

In [11]:
```
plt.figure(figsize=(8,6))
sns.boxplot(dataframe, y='SepalWidthCm')
plt.title('Box Plot of SepalWidthCm Distribution')
plt.xlabel('SepalWidthCm')
plt.show/ #outliers detection
```



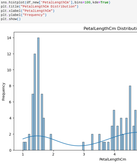
In [12]:
```
# Outliers detection
```

In [13]:
```
plt.figure(figsize=(8,6))
sns.boxplot(dataframe, y='PetalLengthCm')
plt.title('Box Plot of PetalLengthCm Distribution')
plt.xlabel('PetalLengthCm')
plt.show()
```



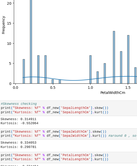
In [14]:
```
# Outliers detection
```

In [15]:
```
plt.figure(figsize=(8,6))
sns.boxplot(dataframe, y='PetalWidthCm')
plt.title('Box Plot of PetalWidthCm Distribution')
plt.xlabel('PetalWidthCm')
plt.show()
```



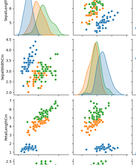
In [16]:
```
#Histogram
# Distribution of SepalLengthCm

plt.figure(figsize=(10,6))
sns.histplot(df_new['SepalLengthCm'],bins=100,kde=True)
plt.title('SepalLengthCm Distribution')
plt.xlabel('SepalLengthCm')
plt.ylabel('Frequency')
plt.show()
```



In [17]:
```
# Distribution of SepalWidthCm

plt.figure(figsize=(10,6))
sns.histplot(df_new['SepalWidthCm'],bins=100,kde=True)
plt.title('SepalWidthCm Distribution')
plt.xlabel('SepalWidthCm')
plt.ylabel('Frequency')
plt.show()
```



In [18]:
```
# Distribution of PetalLengthCm

plt.figure(figsize=(10,6))
sns.histplot(df_new['PetalLengthCm'],bins=100,kde=True)
plt.title('PetalLengthCm Distribution')
plt.xlabel('PetalLengthCm')
plt.ylabel('Frequency')
plt.show()
```

In [19]:
```
# Distribution of PetalWidthCm

plt.figure(figsize=(10,6))
sns.histplot(df_new['PetalWidthCm'],bins=100,kde=True)
plt.title('PetalWidthCm Distribution')
plt.xlabel('PetalWidthCm')
plt.ylabel('Frequency')
plt.show()
```


In [20]:
```
#Skewness checking
print("Skewness: %f" % df_new['SepalLengthCm'].skew())
print("Kurtosis: %f" % df_new['SepalLengthCm'].kurt())
```

```
Skewness: 0.314911
Kurtosis: -0.552064
```

In [21]:
```
print("Skewness: %f" % df_new['SepalWidthCm'].skew())
print("Kurtosis: %f" % df_new['SepalWidthCm'].kurt()) #around 0 , so normal distribution
```

```
Skewness: 0.334053
Kurtosis: 0.290781
```

In [22]:
```
print("Skewness: %f" % df_new['PetalLengthCm'].skew())
print("Kurtosis: %f" % df_new['PetalLengthCm'].kurt())
```

```
Skewness: -0.274021
Kurtosis: -1.402103
```

In [23]:
```
print("Skewness: %f" % df_new['PetalWidthCm'].skew())
print("Kurtosis: %f" % df_new['PetalWidthCm'].kurt()))
```

```
Skewness: -0.104997
Kurtosis: -1.340604
```

In [24]:
```
sns.pairplot(df_new, hue='Species')
plt.show()
```

```
C:\Users\DEBASHRITI\AnacondaFiles\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has changed to tight
  self.figure.tight_layout(*args, **kwargs)
```


In [25]:
```
# Dropping Species column
df_new.drop('Species', axis = 1, inplace = True)
df_new.head()
```

Out[25]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
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

In [26]:
```
# Correlation matrix
corr = df_new.corr()
sns.heatmap(corr, annot = True)
```

```
<Axes: >
```


The data above tells us that both petal parameters are positively correlated with each other unlike the sepal length and width. Sepal length is more correlated to petal length and width than sepal width.