iris dataset EDA

Life cycle of Machine learning Project

```
1.Understanding the Problem Statement
2.Data Collection
3.Exploratory data analysis
4.Data Cleaning
5.Data Pre-Processing
6.Model Training
7.Choose best model
```

1.Problem Statement:

```
We have to apply explotary data analysis on this dataset to find out which attribute is
useful to classify the diffences in flower
```

2.Data Collection:

```
Dataset and code link: https://github.com/dharavathramdas101/EDA
```

2.1 Iris Flower Dataset Description

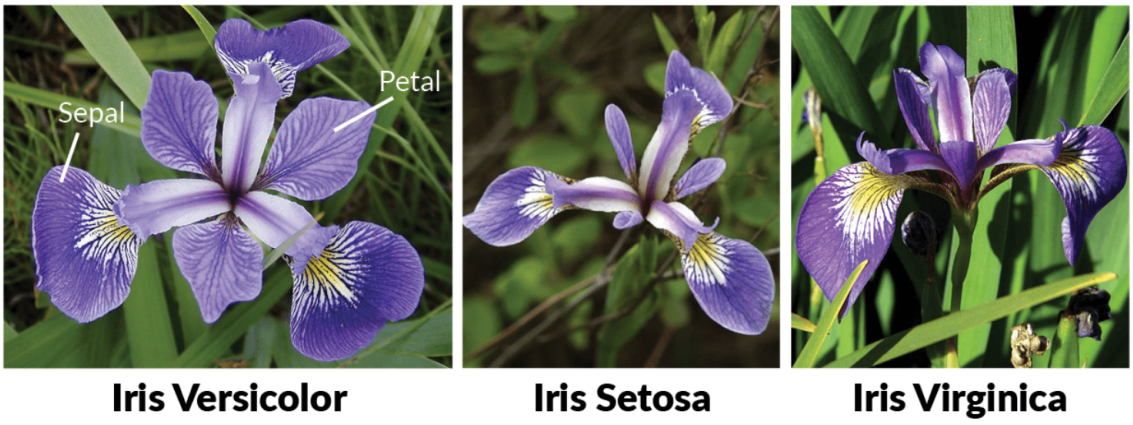
1. A simple dataset to learn the basics
2. the dataset has 150 observations, around 50 for each species of Iris - setosa, virginica and versicolor.
3. ther are 4 features measured on each sample - length and the width of sepal and petal, in centimeters.
4. fisher built a linear discriminant model to distinguish the species using 4 features

```
Attributes :-
    1. Sepal length in cm
    2. Sepal width in cm
    3. Petal length in cm
    4. Petal width in cm
    5. class: -- Iris Setosa -- Iris Versicolour -- Iris Virginica
NOTE - Class contains 3 different types of flowers
```

In [3]:

```
from IPython.display import Image
Image(filename=r"C:\Users\DHARAVATH RAMDAS\OneDrive\Pictures\iris1.png")
```

Out[3]:



2.2 importing Data and Required packages

In [119]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")

%matplotlib inline
```

2.3 Load the dataset

In [5]:

```
df = pd.read_csv(r"C:\Users\DHARAVATH RAMDAS\Downloads\Iris.csv")
```

In [6]:

```
df
```

Out[6]:

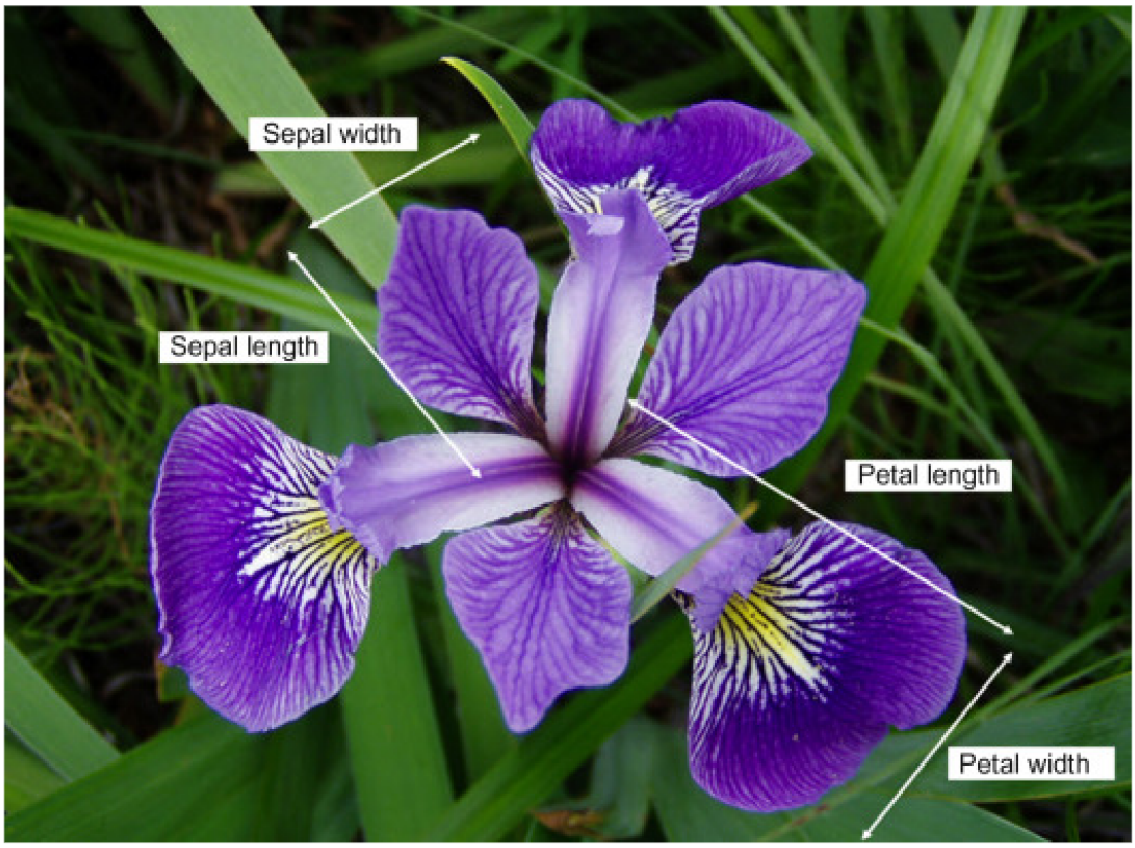
	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

In [7]:

```
Image(filename=r"C:\Users\DHARAVATH RAMDAS\OneDrive\Pictures\iris2.png")
```

Out[7]:



Observation:

1. see the data columns and the flower how we are caluculated the sepal length, sepal width, petal length, petal width
2. we have to drop the id because it is not that much use

In [113]:

```
df.head(3)
```

Out[113]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	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

we have to drop the id because it is not that much use

In [8]:

```
df.drop("Id",axis=1,inplace=True)
```

In [9]:

```
df.head(4)
```

Out[9]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	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

Use the function shape to find the dimentions of the dataframe.

In [10]:

```
df.shape
```

Out[10]:

```
(150, 5)
```

Observation: dataset have 150 rows and 5 columns

check columns information

In [11]:

```
df.columns
```

Out[11]:

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

Type of columns

In [12]:

```
type(df.columns)
```

Out[12]:

```
pandas.core.indexes.base.Index
```

Check the data balanced or imbalanced

In [13]:

```
df['Species'].value_counts()
```

Out[13]:

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

Observation: see it is balanced dataset

Dataset Description

In [14]:

```
### Describe function is used to see the statistics of the dataset such as mean, median, st
```

In [15]:

```
df.describe()
```

Out[15]:

	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

transpose the describe

In [16]:

```
df.describe().T
```

Out[16]:

	count	mean	std	min	25%	50%	75%	max
SepalLengthCm	150.0	5.843333	0.828066	4.3	5.1	5.80	6.4	7.9
SepalWidthCm	150.0	3.054000	0.433594	2.0	2.8	3.00	3.3	4.4
PetalLengthCm	150.0	3.758667	1.764420	1.0	1.6	4.35	5.1	6.9
PetalWidthCm	150.0	1.198667	0.763161	0.1	0.3	1.30	1.8	2.5

Describing the setosa class

In [17]:

```
iris_setosa = df.loc[df['Species']=="Iris-setosa"]
iris_setosa.describe()
```

Out[17]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	50.00000	50.000000	50.000000	50.00000
mean	5.00600	3.418000	1.464000	0.24400
std	0.35249	0.381024	0.173511	0.10721
min	4.30000	2.300000	1.000000	0.10000
25%	4.80000	3.125000	1.400000	0.20000
50%	5.00000	3.400000	1.500000	0.20000
75%	5.20000	3.675000	1.575000	0.30000
max	5.80000	4.400000	1.900000	0.60000

Describing the iris versicolot

In [18]:

```
iris_versicolor = df.loc[df['Species']=="Iris-versicolor"]
iris_versicolor.describe()
```

Out[18]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	50.000000	50.000000	50.000000	50.000000
mean	5.936000	2.770000	4.260000	1.326000
std	0.516171	0.313798	0.469911	0.197753
min	4.900000	2.000000	3.000000	1.000000
25%	5.600000	2.525000	4.000000	1.200000
50%	5.900000	2.800000	4.350000	1.300000
75%	6.300000	3.000000	4.600000	1.500000
max	7.000000	3.400000	5.100000	1.800000

Describing the iris virginica

In [19]:

```
iris_virginica = df.loc[df['Species']=="Iris-virginica"]
iris_virginica.describe()
```

Out[19]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	50.00000	50.000000	50.000000	50.00000
mean	6.58800	2.974000	5.552000	2.02600
std	0.63588	0.322497	0.551895	0.27465
min	4.90000	2.200000	4.500000	1.40000
25%	6.22500	2.800000	5.100000	1.80000
50%	6.50000	3.000000	5.550000	2.00000
75%	6.90000	3.175000	5.875000	2.30000
max	7.90000	3.800000	6.900000	2.50000

we need to verify the features are of which datatypes. we use info() function

In [20]:

```
df.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
```

Observation: 1.total index are 150 from 0 to 149 2.total 5 columns 3.total 4 columns are folat64 data type 4.and 1 column are object 5. memory usage is 6.0kb

Check the null values

In [21]:

```
df.isnull().sum()
```

Out[21]:

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

Observation: see we dont have any null values in dataset

we are seperating the numerical and categorical

In [22]:

```
#check the datatype of column

df['Species'].dtype
```

Out[22]:

```
dtype('O')
```

Categorical Feature

In [23]:

```
cat_col=[fea for fea in df.columns if df[fea].dtype == 'O']
df[cat_col].head()
```

Out[23]:

	Species
0	Iris-setosa
1	Iris-setosa
2	Iris-setosa
3	Iris-setosa
4	Iris-setosa

Numerical Features

In [24]:

```
num_col=[fea for fea in df.columns if df[fea].dtype != 'O']
```

In [25]:

```
df[num_col].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

observation: we have the 14 num columns

3. Exploring data

3.1 Univariate Analysis

The term univariate analysis refers to the analysis of one variable

Numerical Features

1-D Scatter plot

This plots different observations/values of the same variable corresponding to the index/observation number

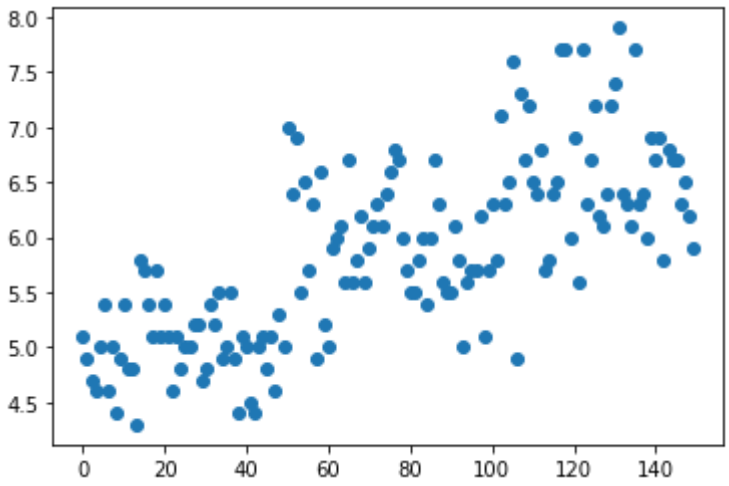
scatter plot on sepallenght

In [26]:

```
plt.scatter(df.index,df['SepalLengthCm'])
```

Out[26]:

```
<matplotlib.collections.PathCollection at 0x1809174fd90>
```



Observation: see the sepal lenght is how it is spread

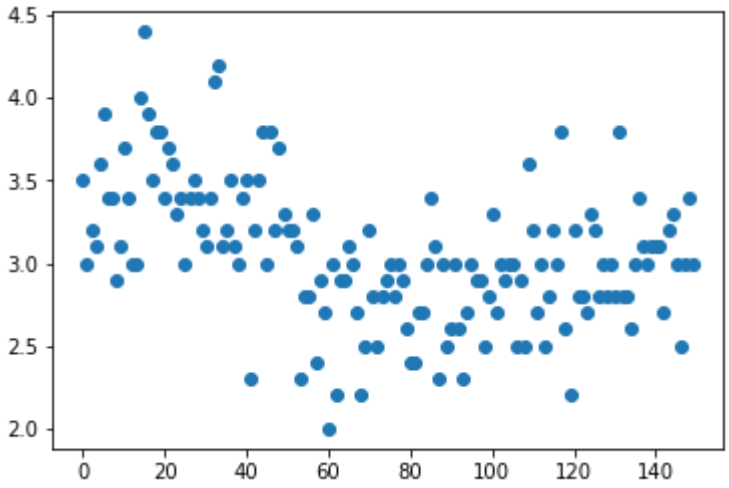
scatter plot on sepalwidth

In [27]:

```
plt.scatter(df.index,df['SepalWidthCm'])
```

Out[27]:

```
<matplotlib.collections.PathCollection at 0x18093a5ebe0>
```



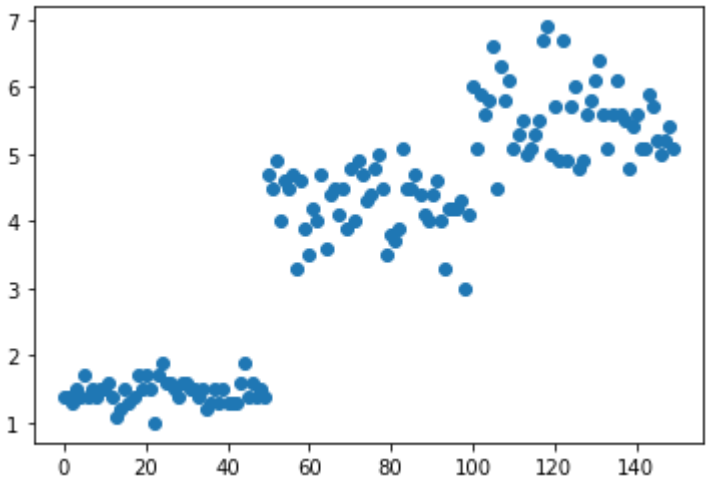
scatter plot on petallength

In [28]:

```
plt.scatter(df.index,df['PetalLengthCm'])
```

Out[28]:

```
<matplotlib.collections.PathCollection at 0x18093ad82e0>
```



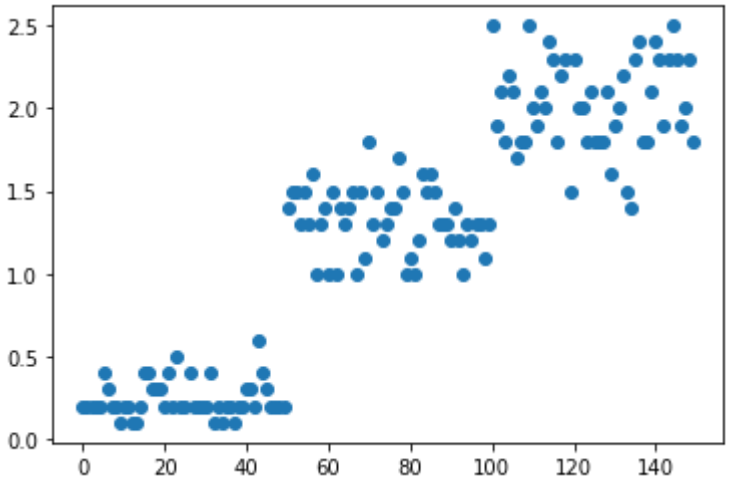
scatter plot on petalwidth

In [29]:

```
plt.scatter(df.index,df['PetalWidthCm'])
```

Out[29]:

```
<matplotlib.collections.PathCollection at 0x18093b3fa00>
```



In [ ]:

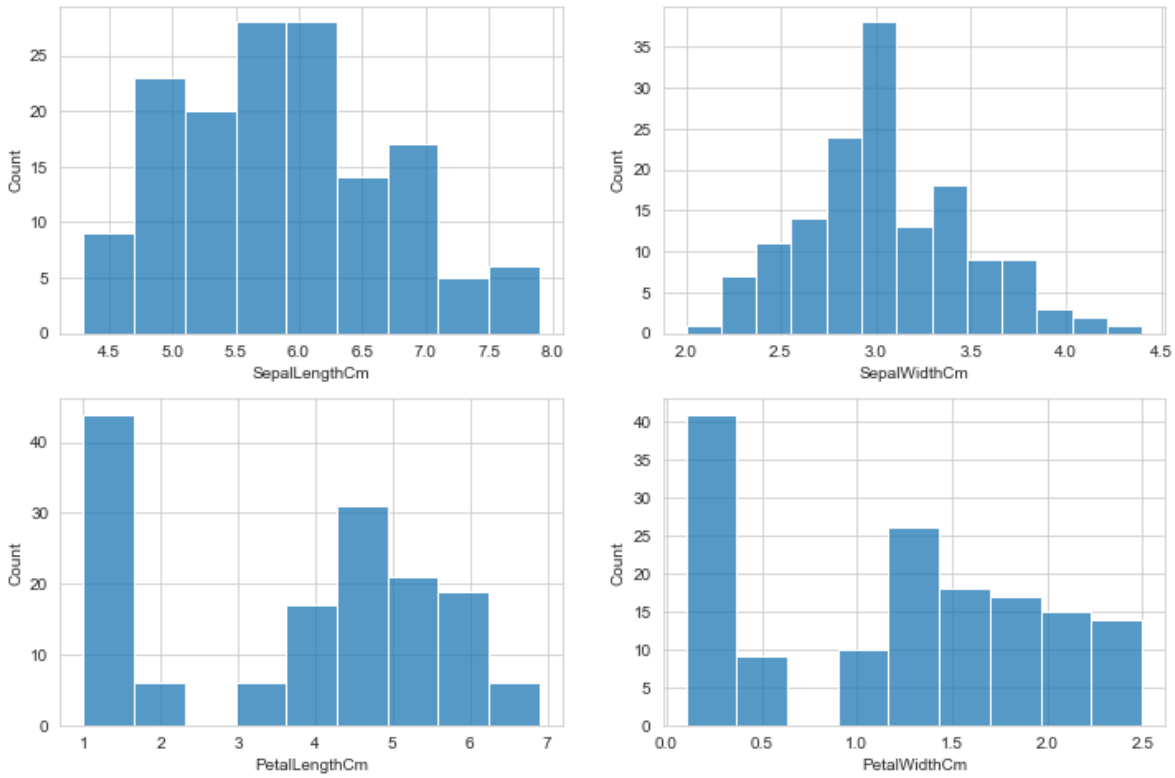
```

```

3.2 Histograms

A histogram groups values into ranges (or bins), and the height of a bar shows how many values fall in that range.

In [77]:

```
plt.figure(figsize=(12,8))
plt.title("hi")
for i in range(0,len(num_col)):
    plt.subplot(2,2,i+1)
    sns.histplot(x=num_col[i],data=df)
    plt.xlabel(num_col[i])
```

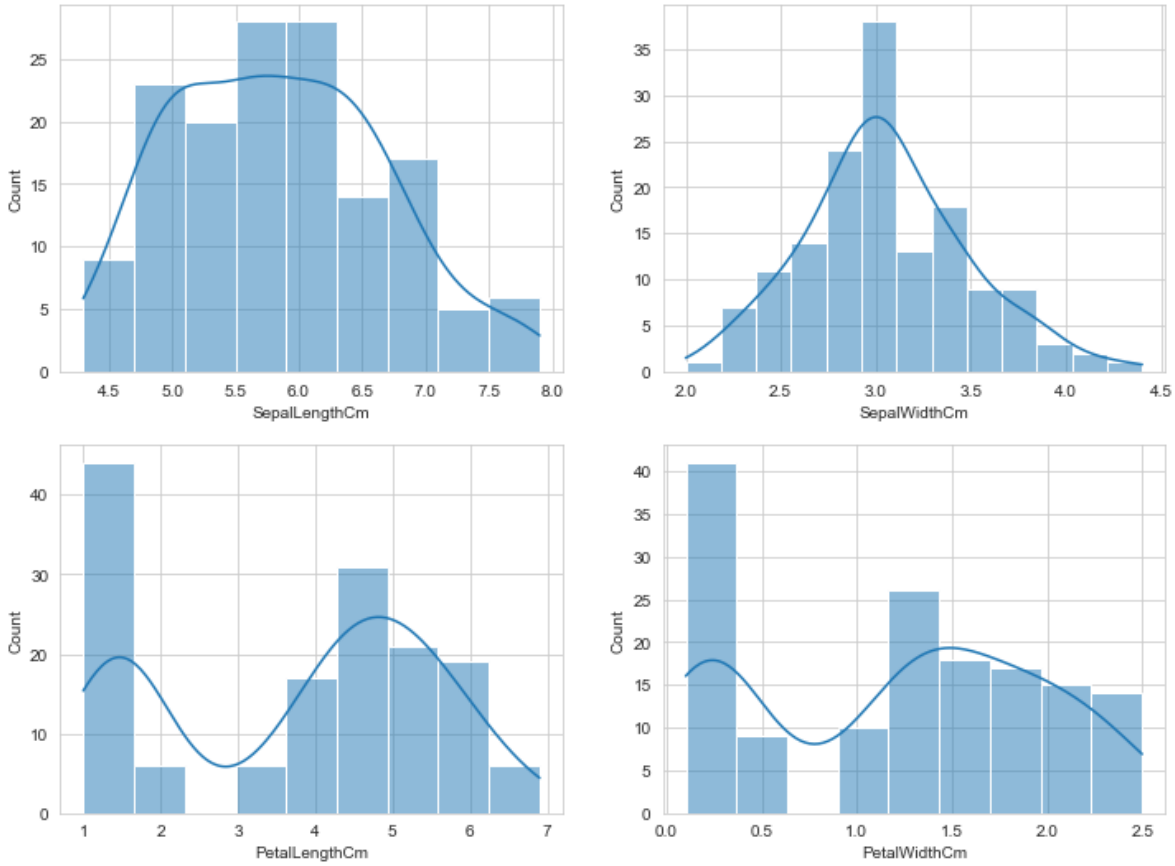


3.3 Histogram plot with KDE

We can display a histogram with a KDE curve as below

In [78]:

```
plt.figure(figsize=(12,9))
plt.title("hi")
for i in range(0,len(num_col)):
    plt.subplot(2,2,i+1)
    sns.histplot(x=num_col[i],data=df,kde=True)
    plt.xlabel(num_col[i])
```



3.4 KDE plot

kde plot shows the how much data is spreaded

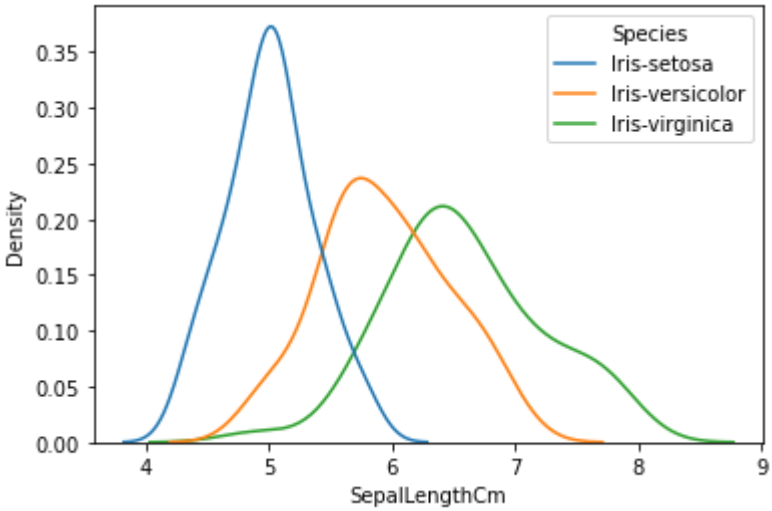
kde plot on sepallenght

In [32]:

```
sns.kdeplot(x='SepalLengthCm',data=df,hue='Species')
```

Out[32]:

```
<AxesSubplot:xlabel='SepalLengthCm', ylabel='Density'>
```



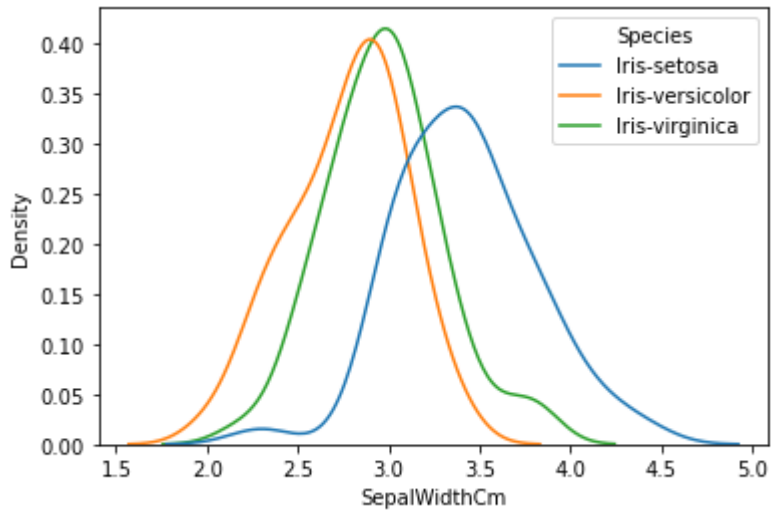
kde plot Sepallenght

In [33]:

```
sns.kdeplot(x='SepalWidthCm',data=df,hue='Species')
```

Out[33]:

```
<AxesSubplot:xlabel='SepalWidthCm', ylabel='Density'>
```



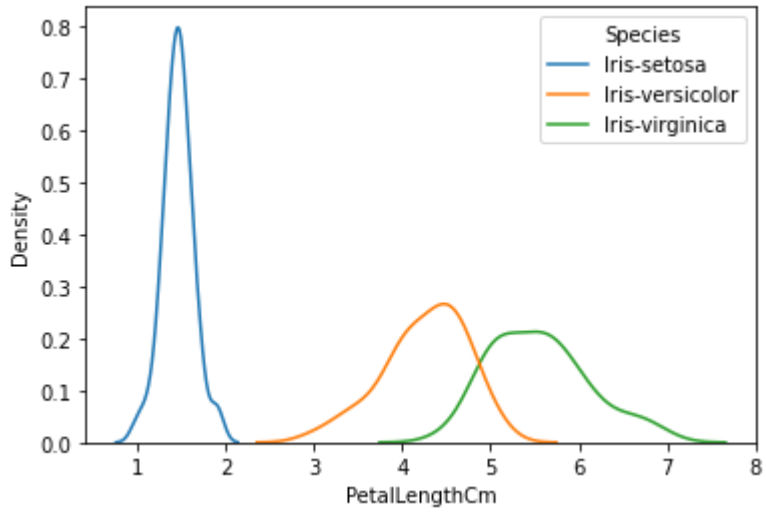
kde plot on petallength

In [34]:

```
sns.kdeplot(x='PetalLengthCm',data=df,hue='Species')
```

Out[34]:

```
<AxesSubplot:xlabel='PetalLengthCm', ylabel='Density'>
```



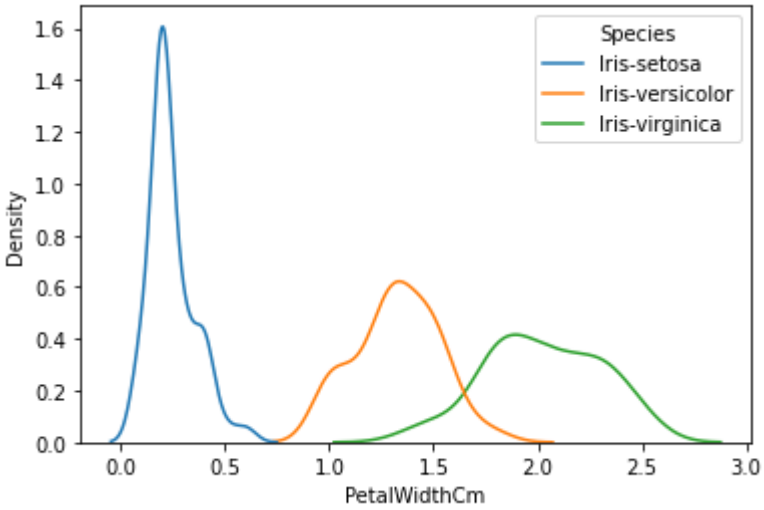
kde plot on petalwidth

In [35]:

```
sns.kdeplot(x='PetalWidthCm',data=df,hue='Species')
```

Out[35]:

```
<AxesSubplot:xlabel='PetalWidthCm', ylabel='Density'>
```



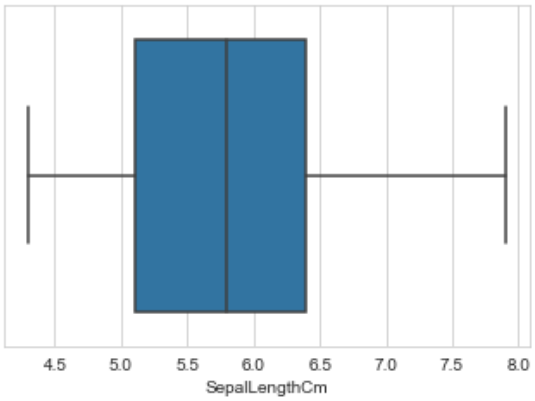
3.5 Box plots:

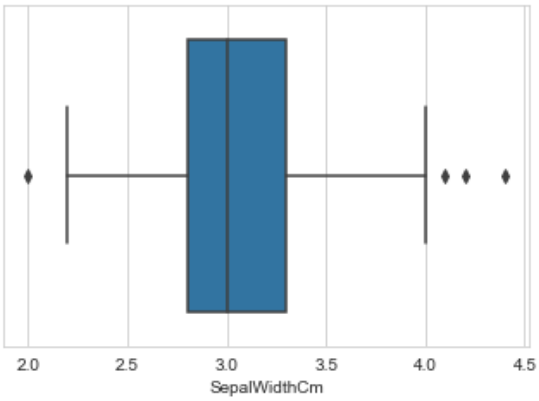
A boxplot shows the distribution, center and skewness of a numeric feature. It divides the data into sections that contain 25% of the data approximately.

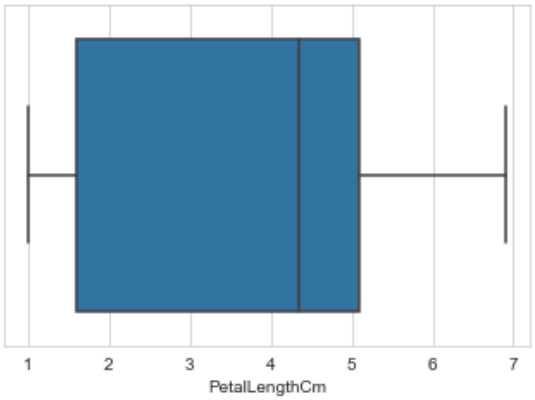
Outliers, if present, appear as dots on either end. The whiskers that extend from the box represent the minimum and maximum values. The box depicts the Interquartile range and holds 50% of the data.

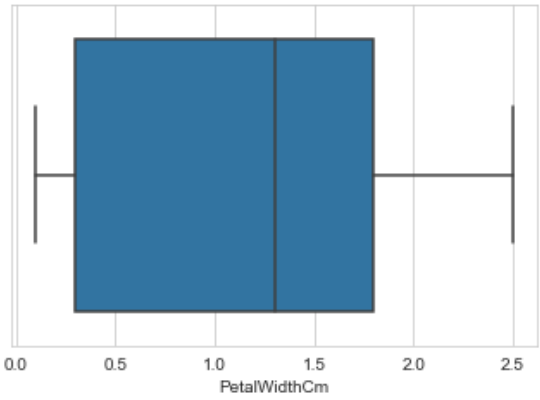
In [80]:

```
plt.figure(figsize=(12,8))
plt.title("hi")
for i in range(0,len(num_col)):
    plt.subplot(2,2,i+1)
    sns.boxplot(x=df[num_col[i]])
    plt.xlabel(num_col[i])
```









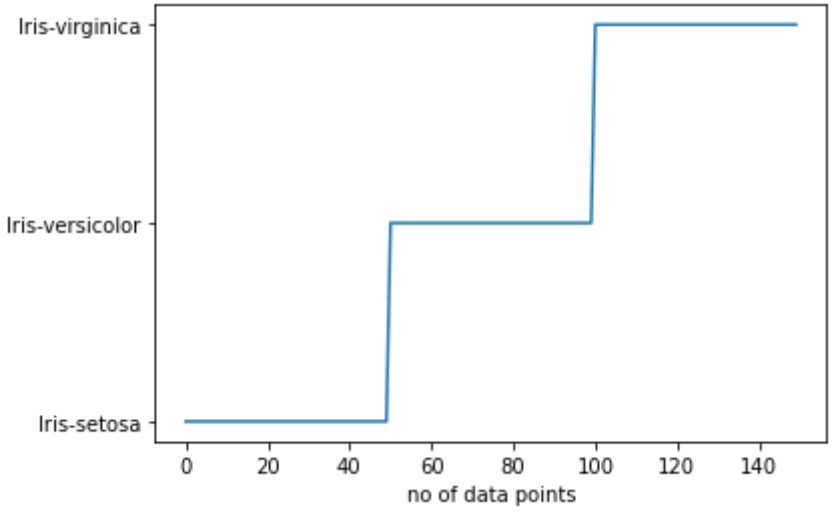

```
Observation:
    see the sepalwidth conatins the some of outlaier and other dont have the out
outlaiers
```

3.6 Visualizing Categorical Variables

Plot for species

In [37]:

```
plt.plot(df["Species"])
plt.xlabel("no of data points")
plt.show()
```

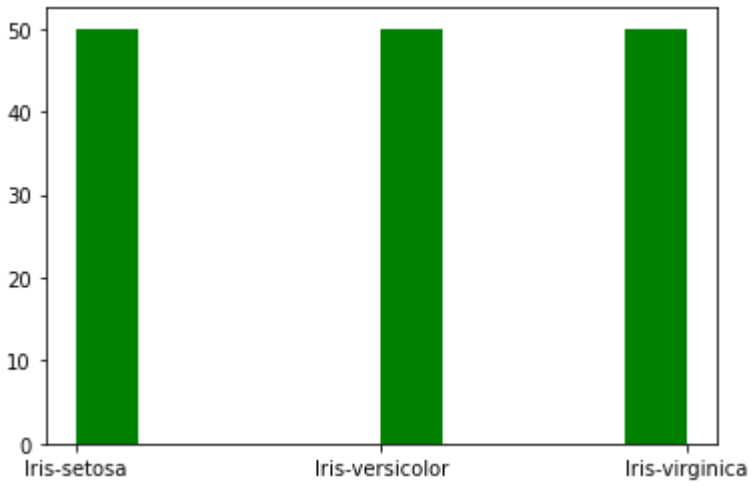


observation: see the plot it contains 50 iris-setosa, 50 iris-versicolor and 50 iris-virginica

hist plot

In [38]:

```
plt.hist(df['Species'],color="green")
plt.show()
```



observation: see total 150 in that 50 iris-setosa, 50 iris-versicolor and 50 iris-virginica

4. Bivariate Analysis

4.1 2-D Scatter Plot

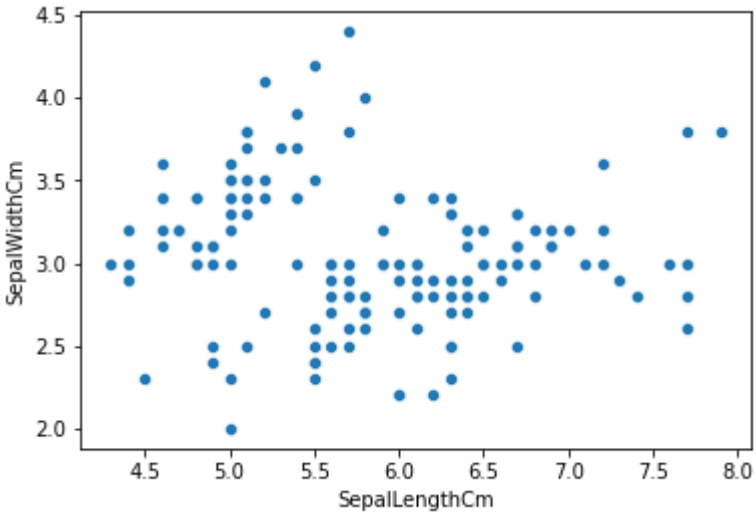
scatter plot on sepallenght and sepalwidth with out hue

In [39]:

```
sns.scatterplot(data=df,x="SepalLengthCm",y="SepalWidthCm")
```

Out[39]:

```
<AxesSubplot:xlabel='SepalLengthCm', ylabel='SepalWidthCm'>
```



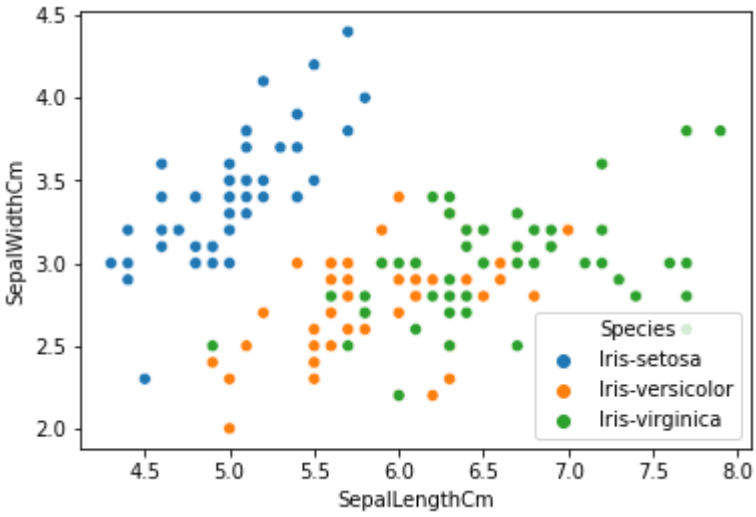
Scatter on sepallenght and sepalwidth plot with hue

In [40]:

```
sns.scatterplot(data=df,x="SepalLengthCm",y="SepalWidthCm",hue="Species")
```

Out[40]:

```
<AxesSubplot:xlabel='SepalLengthCm', ylabel='SepalWidthCm'>
```



Observation: using sepal lenght and sepal width, we can seperate setosa flower from others

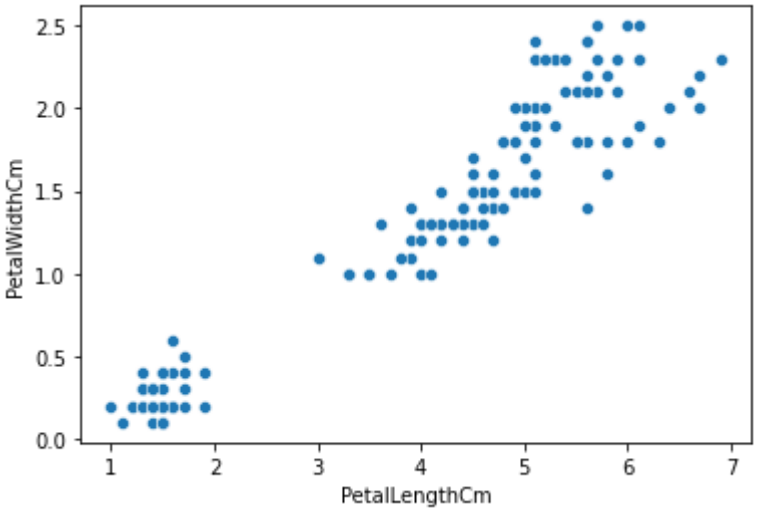
Scatter plot on petallenght and petalwidth

In [41]:

```
sns.scatterplot(data=df,x="PetalLengthCm",y="PetalWidthCm")
```

Out[41]:

```
<AxesSubplot:xlabel='PetalLengthCm', ylabel='PetalWidthCm'>
```



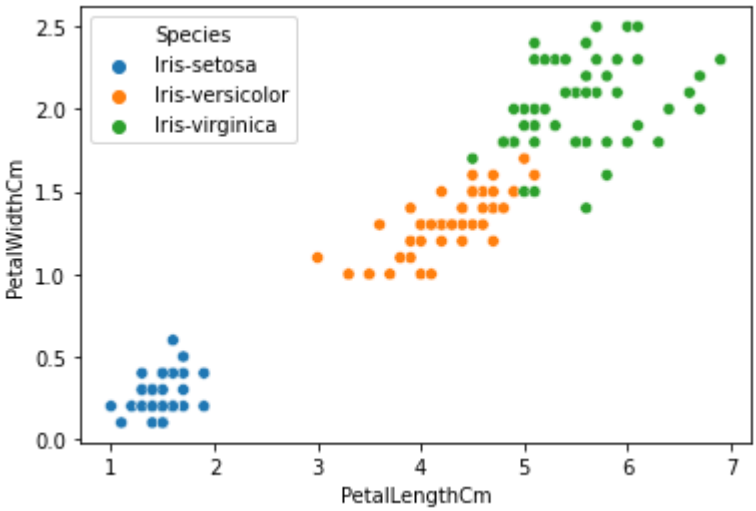
Scattet plot on petallength and petalwidth with hue

In [42]:

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

Out[42]:

```
<AxesSubplot:xlabel='PetalLengthCm', ylabel='PetalWidthCm'>
```



observation: we can seperate the setosa and also versicolor

5. Correlation

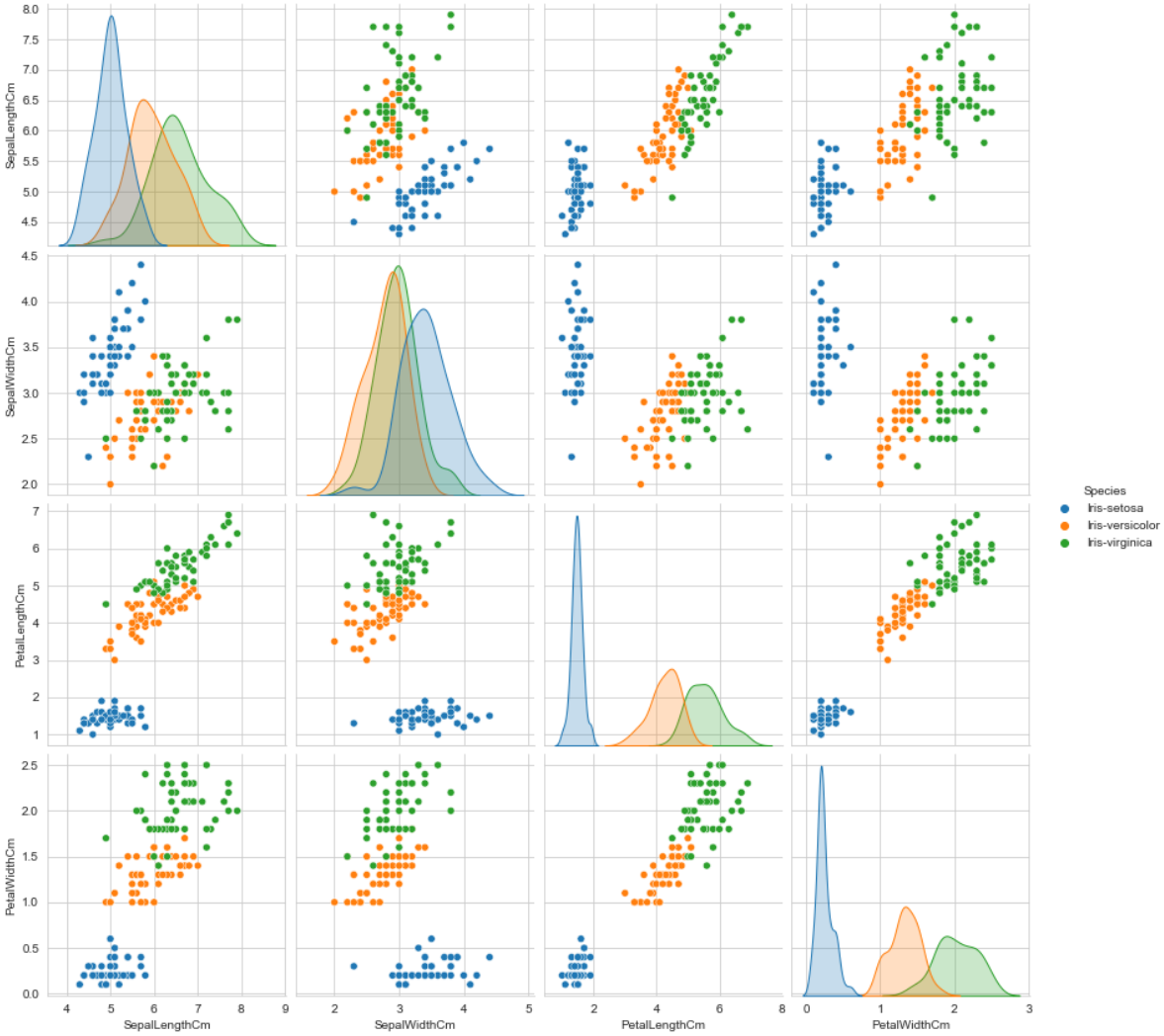
1.Now we will do some plotting/visualizing our data to understand the relation ship between the numerical features. 2.I have used seaborn library for plotting, we can also use python matplotlib library to visualize the data. 3.There are different types of plots like bar plot, box plot, scatter plot etc. 4.Scatter plot is very useful when we are analyzing the relation ship between 2 features on x and y axis. 5.In seaborn library we have pairplot function which is very useful to scatter plot all the features at once instead of plotting them individually.

5.1 Pairplot:

In [43]:

```
sns.set_style("whitegrid")
sns.pairplot(df,hue="Species",size=3)
plt.show()
```

```
C:\Users\DHARAVATH RAMDAS\Anaconda3\lib\site-packages\seaborn\axisgrid.py:20
76: UserWarning: The `size` parameter has been renamed to `height`; please u
pdate your code.
  warnings.warn(msg, UserWarning)
```




```
Observations:
 1. petal_length and petal_width are the most useful features to identify various flower
types.
 2. While Setosa can be easily identified (linearly seperable), Virginica and Versicolor
have some overlap (almost linearly           seperable).
 3. We can find "lines" and "if-else" conditions to build a simple model to classify the
flower types.
```

5.2 Heatmap:

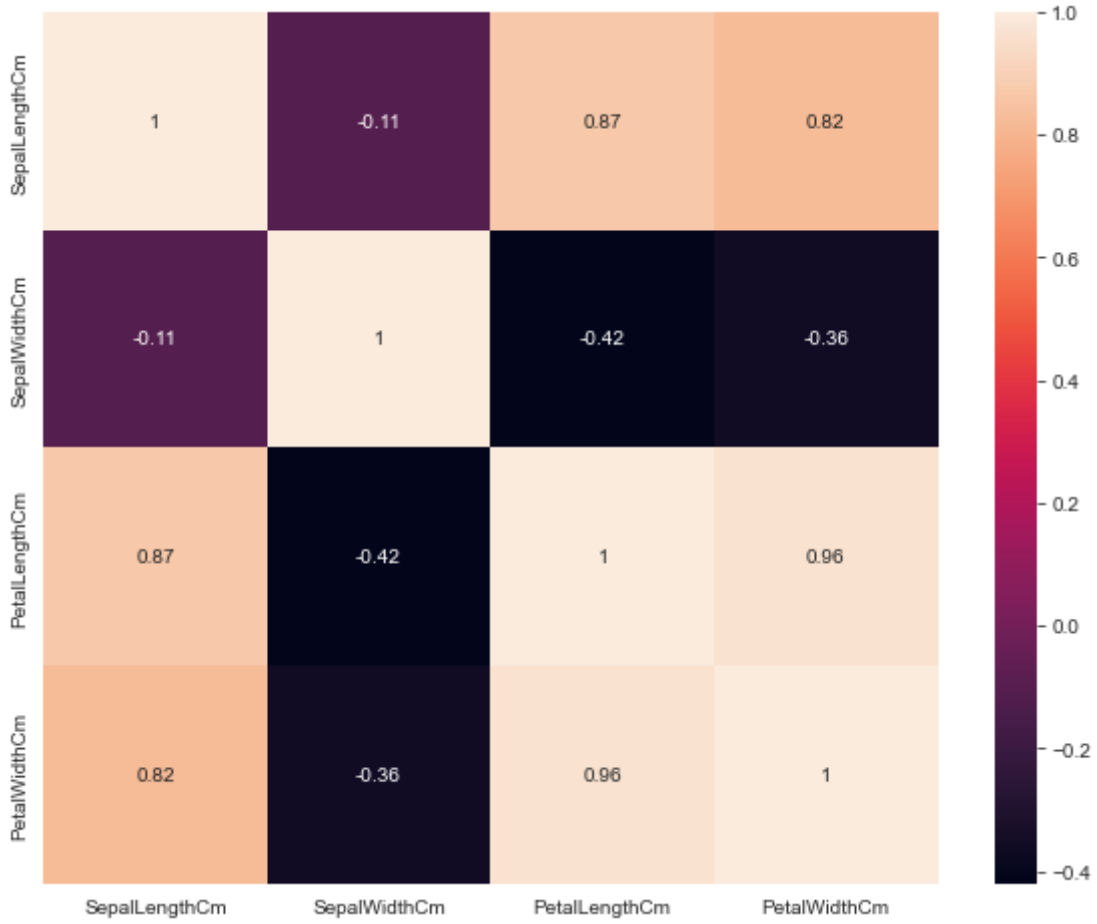
we will see how these features are correlated to each other using heatmap in seaborn library

In [44]:

```
plt.figure(figsize=(10,8))
sns.heatmap(df.corr(),annot=True)
plt.plot()
```

Out[44]:

```
[]
```




```
observation:
    1.petallenght and petalwidth has the high correlation
    2.less correlation is sepallenght and sepal width
```

6.Histogram, PDF and CDF

```
histogram shows the how much it is spreaded
```

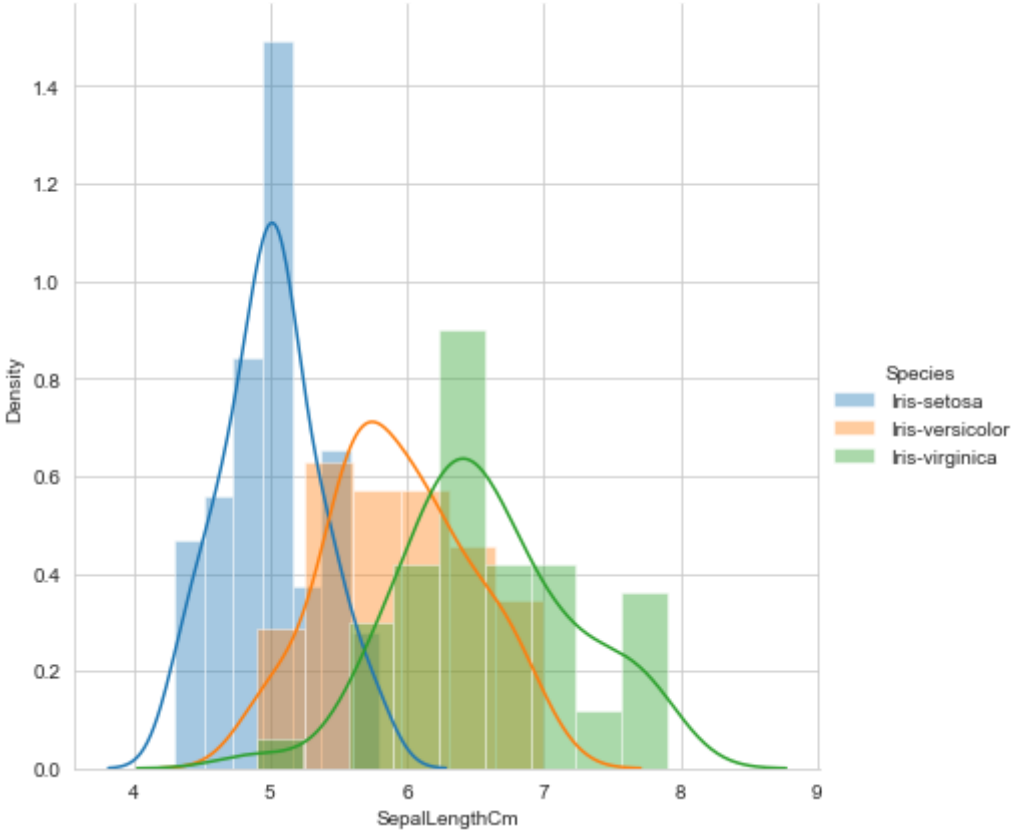
distribution plot on sepallenght

In [120]:

```
d=sns.FacetGrid(data=df,hue="Species",size=6)
d.map(sns.distplot,"SepalLengthCm")
d.add_legend()
```

Out[120]:

```
<seaborn.axisgrid.FacetGrid at 0x180979c46a0>
```



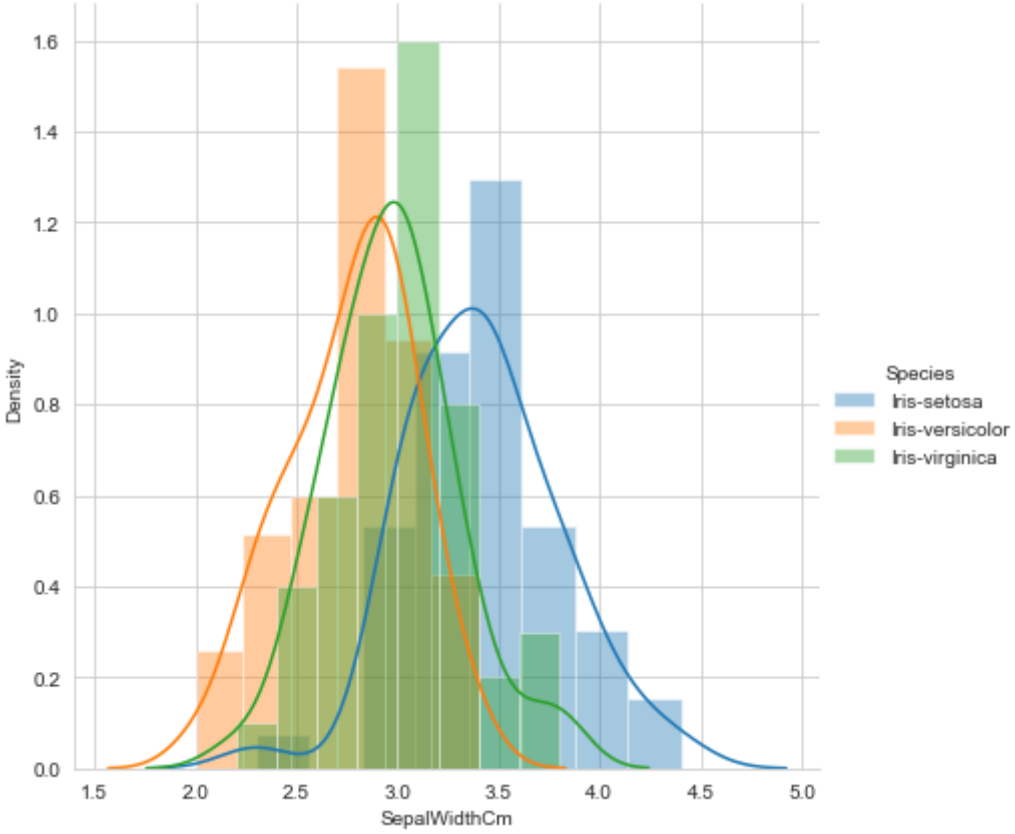
dist plot on sepal width

In [121]:

```
e=sns.FacetGrid(data=df,hue="Species",size=6)
e.map(sns.distplot,"SepalWidthCm")
e.add_legend()
```

Out[121]:

```
<seaborn.axisgrid.FacetGrid at 0x18097e49940>
```



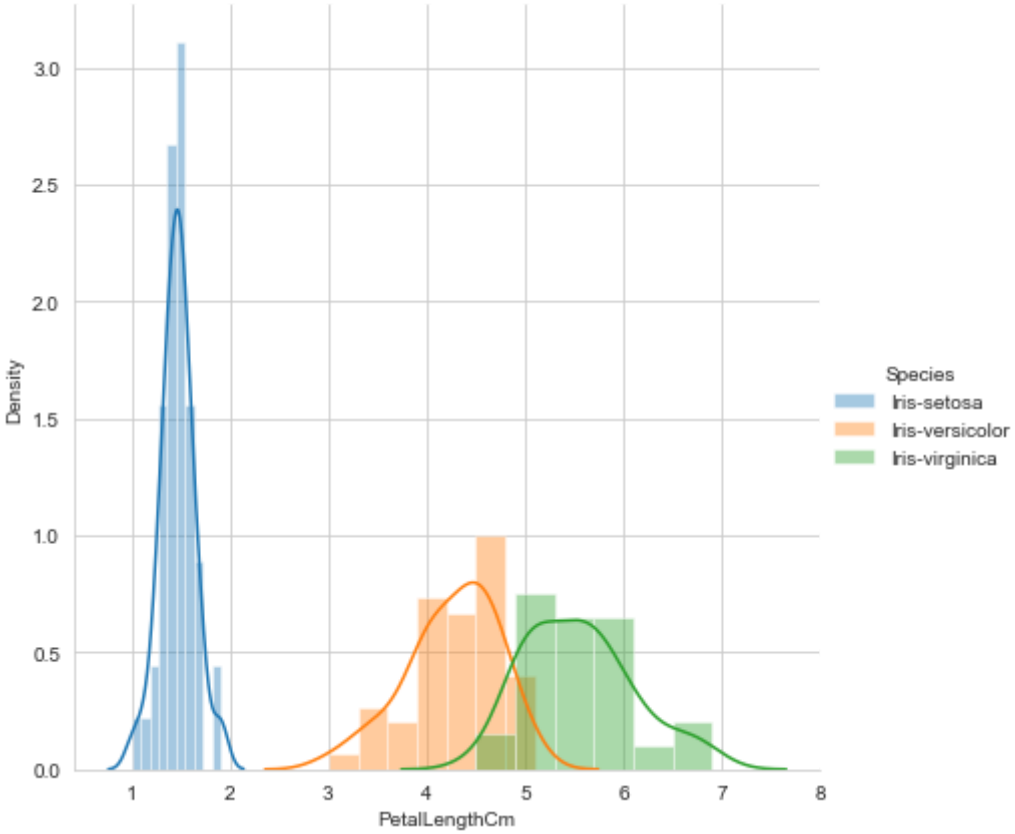
dist plot on petal lenght

In [122]:

```
f=sns.FacetGrid(data=df,hue="Species",size=6)
f.map(sns.distplot,"PetalLengthCm")
f.add_legend()
```

Out[122]:

```
<seaborn.axisgrid.FacetGrid at 0x18097c1ffa0>
```



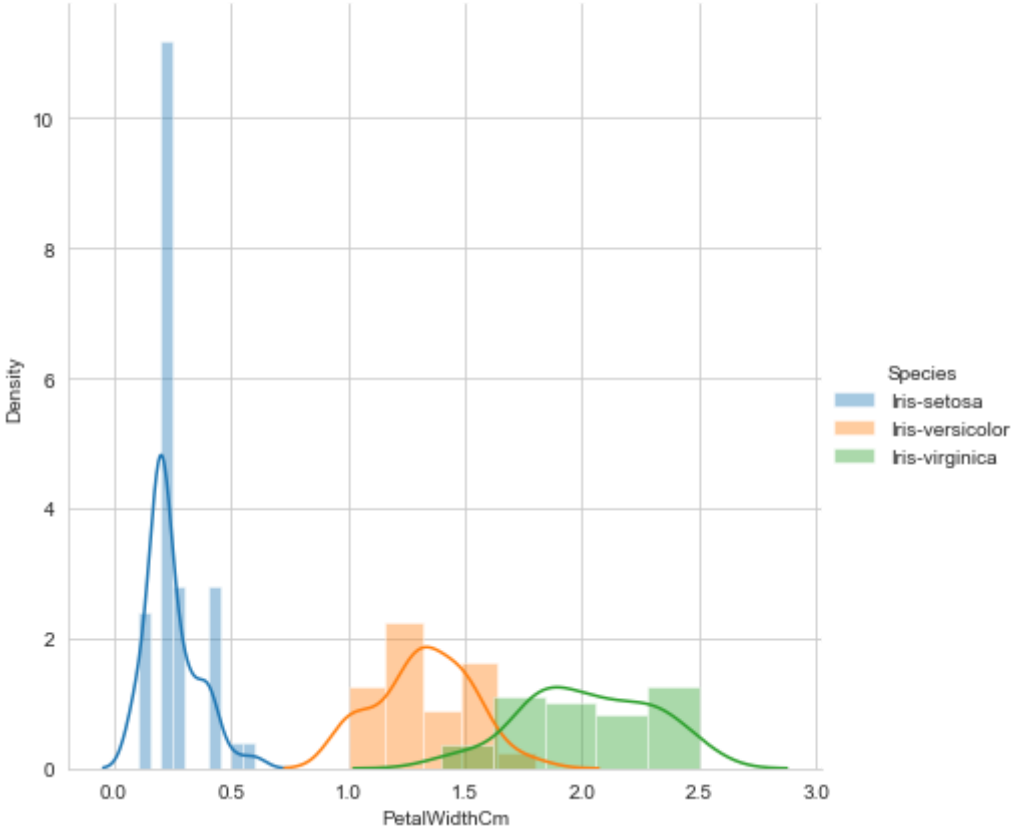
distplot on petal width

In [123]:

```
g=sns.FacetGrid(data=df,hue="Species",size=6)
g.map(sns.distplot,"PetalWidthCm")
g.add_legend()
```

Out[123]:

```
<seaborn.axisgrid.FacetGrid at 0x1809801b3a0>
```



7.Plot CDF

Plot CDF for petal length

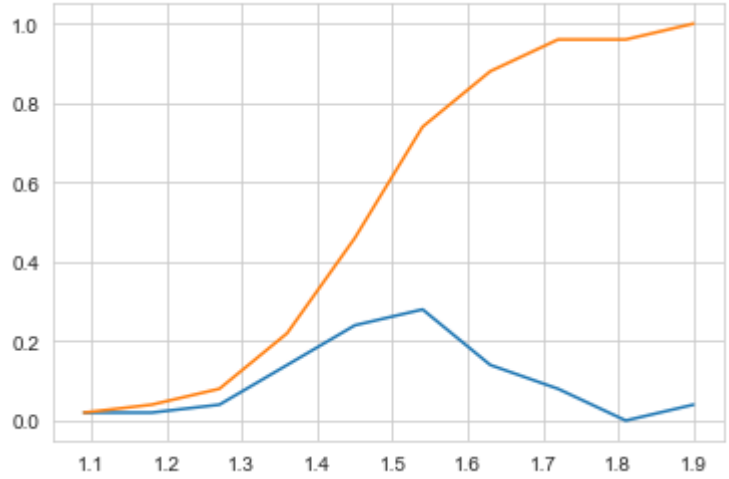
In [49]:

```
# Cumulative distribution function(cdf)
```

In [50]:

```
counts,bin_edges=np.histogram(iris_setosa["PetalLengthCm"],bins=10,density=True)
pdf=counts/(sum(counts))
print(pdf)
print(bin_edges)

# compute cdf
cdf=np.cumsum(pdf)
plt.plot(bin_edges[1:],pdf)
plt.plot(bin_edges[1:],cdf)
plt.show()
```

```
[0.02 0.02 0.04 0.14 0.24 0.28 0.14 0.08 0.   0.04]
[1.   1.09 1.18 1.27 1.36 1.45 1.54 1.63 1.72 1.81 1.9 ]
```



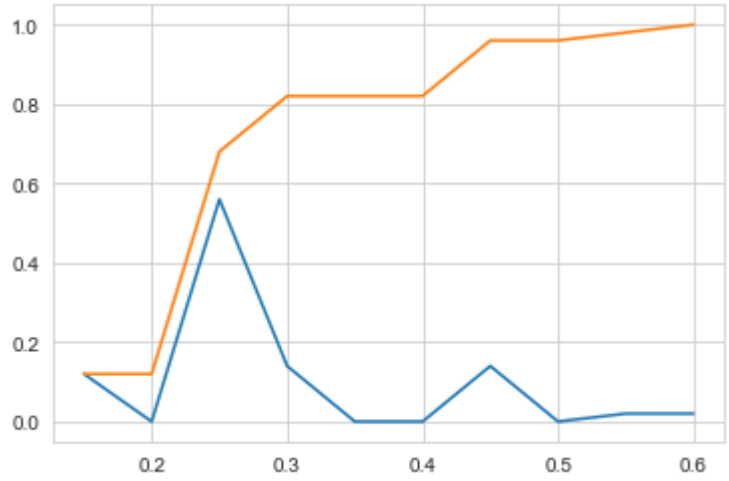

```
Observation:
    1. orange line is cdf
    2. blue line is pdf
```

plot cdf on petal lenght

In [107]:

```
counts,bin_edges=np.histogram(iris_setosa["PetalWidthCm"],bins=10,density=True)
pdf=counts/(sum(counts))
print(pdf)
print(bin_edges)

# compute cdf
cdf=np.cumsum(pdf)
plt.plot(bin_edges[1:],pdf)
plt.plot(bin_edges[1:],cdf)
plt.show()
```

```
[0.12 0.   0.56 0.14 0.   0.   0.14 0.   0.02 0.02]
[0.1  0.15 0.2  0.25 0.3  0.35 0.4  0.45 0.5  0.55 0.6 ]
```

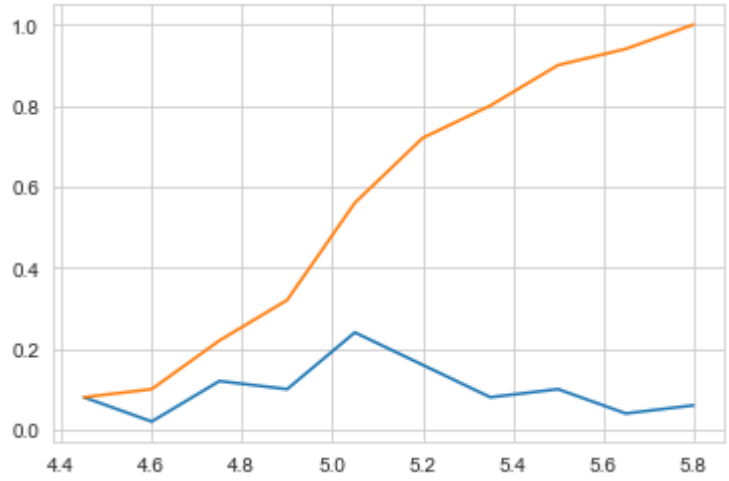


plot cdf on sepal lenght

In [108]:

```
counts,bin_edges=np.histogram(iris_setosa["SepalLengthCm"],bins=10,density=True)
pdf=counts/(sum(counts))
print(pdf)
print(bin_edges)

# compute cdf
cdf=np.cumsum(pdf)
plt.plot(bin_edges[1:],pdf)
plt.plot(bin_edges[1:],cdf)
plt.show()
```

```
[0.08 0.02 0.12 0.1  0.24 0.16 0.08 0.1  0.04 0.06]
[4.3  4.45 4.6  4.75 4.9  5.05 5.2  5.35 5.5  5.65 5.8 ]
```

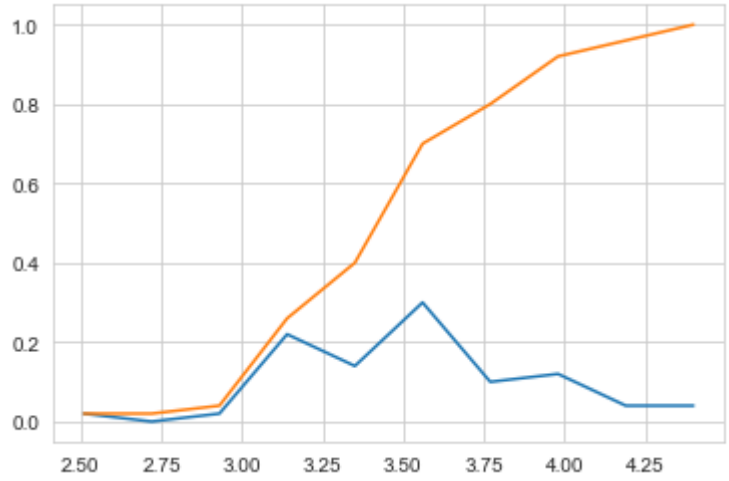


plot cdf on sepal lenght

In [110]:

```
counts,bin_edges=np.histogram(iris_setosa["SepalWidthCm"],bins=10,density=True)
pdf=counts/(sum(counts))
print(pdf)
print(bin_edges)

# compute cdf
cdf=np.cumsum(pdf)
plt.plot(bin_edges[1:],pdf)
plt.plot(bin_edges[1:],cdf)
plt.show()
```

```
[0.02 0.   0.02 0.22 0.14 0.3  0.1  0.12 0.04 0.04]
[2.3  2.51 2.72 2.93 3.14 3.35 3.56 3.77 3.98 4.19 4.4 ]
```



8.Mean, Variance and Standard Deviation

In [111]:

```
print('Means:')
print(np.mean(iris_setosa["PetalLengthCm"]))
print(np.mean(iris_versicolor["PetalLengthCm"]))
print(np.mean(iris_virginica["PetalLengthCm"]))

print('\nstandard deviation:')
print(np.std(iris_setosa["PetalLengthCm"]))
print(np.std(iris_versicolor["PetalLengthCm"]))
print(np.std(iris_virginica["PetalLengthCm"]))
```

```
Means:
1.464
4.26
5.552

standard deviation:
0.17176728442867115
0.4651881339845204
0.5463478745268441
```

Obsevations: MEAN: 51st flower length is added to our data and it is wrong i.e huge value is inserted in our case its 50, but actual length will be 1-2 so the mean is changed drastically, so following only mean is not

feasible, similarly std also

Median, Percentile, Quantile, IQR, MAD

In [112]:

```
print("\nMedians:")
print(np.median(iris_setosa["PetalLengthCm"]))
#Median with an outlier
print(np.median(np.append(iris_setosa["PetalLengthCm"],50)));
print(np.median(iris_virginica["PetalLengthCm"]))
print(np.median(iris_versicolor["PetalLengthCm"]))


print("\nQuantiles:")
print(np.percentile(iris_setosa["PetalLengthCm"],np.arange(0, 100, 25)))
print(np.percentile(iris_virginica["PetalLengthCm"],np.arange(0, 100, 25)))
print(np.percentile(iris_versicolor["PetalLengthCm"], np.arange(0, 100, 25)))

print("\n90th Percentiles:")
print(np.percentile(iris_setosa["PetalLengthCm"],90))
print(np.percentile(iris_virginica["PetalLengthCm"],90))
print(np.percentile(iris_versicolor["PetalLengthCm"], 90))

from statsmodels import robust
print ("\nMedian Absolute Deviation")
print(robust.mad(iris_setosa["PetalLengthCm"]))
print(robust.mad(iris_virginica["PetalLengthCm"]))
print(robust.mad(iris_versicolor["PetalLengthCm"]))
```

```
Medians:
1.5
1.5
5.55
4.35

Quantiles:
[1.    1.4   1.5   1.575]
[4.5   5.1   5.55  5.875]
[3.   4.   4.35 4.6 ]

90th Percentiles:
1.7
6.31
4.8

Median Absolute Deviation
0.14826022185056031
0.6671709983275211
0.5189107764769602
```

Observations: MEDIAN: Even after Adding Outlier the result is almost same.because if there are more than 50% of outliers then result will be useless.So its better than mean and std.

In [ ]:

```
```

9.Box plot and Whiskers

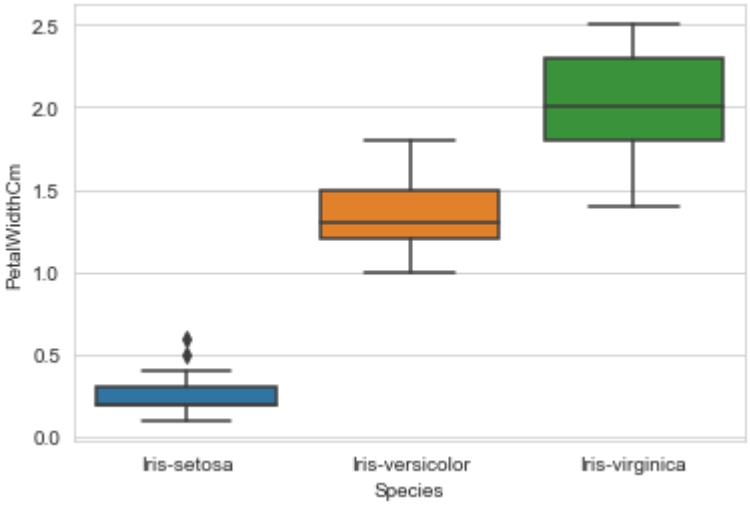
In [ ]:

```
#Box-plot with whiskers: another method of visualizing the  1-D scatter plot more intuitive


#NOTE: IN the plot below, a technique call inter-quartile range is used in plotting the whi
#Whiskers in the plot below do not correposnd to the min and max values.

#Box-plot can be visualized as a PDF on the side-ways.

sns.boxplot(x='Species',y='PetalLengthCm', data=df)
plt.show()
```

boxplot on species and petalwidth

In [114]:

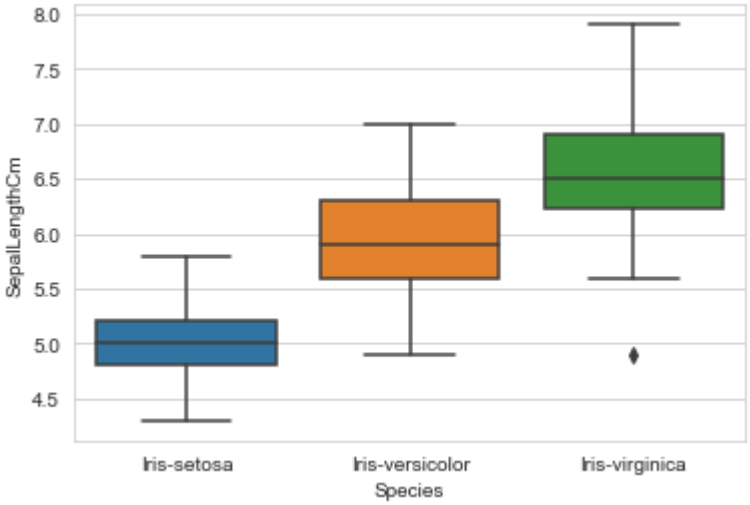
```
sns.boxplot(x='Species',y='PetalWidthCm', data=df)
plt.show()
```



boxplot on species and sepallenght

In [115]:

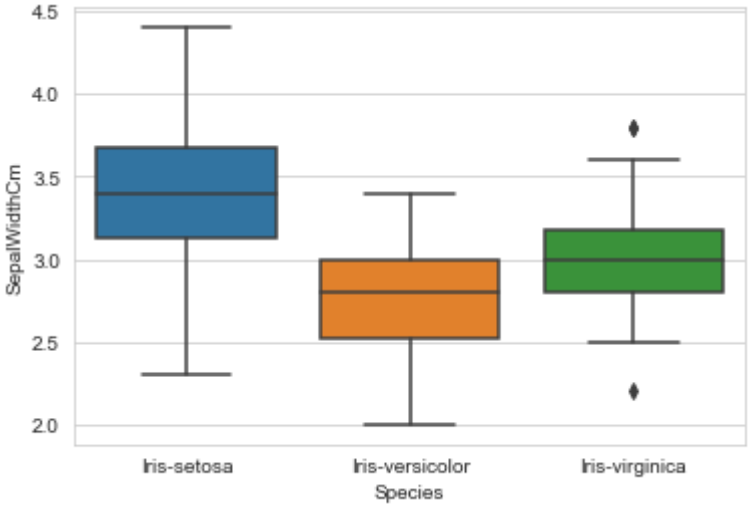
```
sns.boxplot(x='Species',y='SepalLengthCm', data=df)
plt.show()
```



boxplot on species and sepalwidth

In [116]:

```
sns.boxplot(x='Species',y='SepalWidthCm', data=df)
plt.show()
```

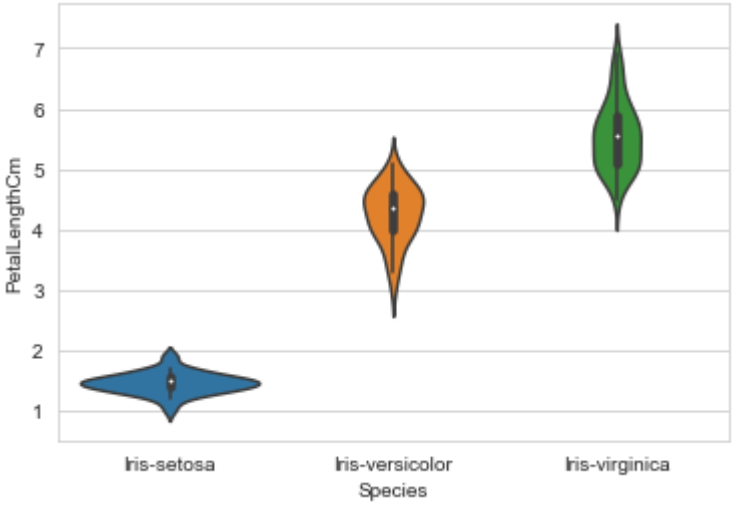


10.Violin plots

violin plot on species and petallenght

In [103]:

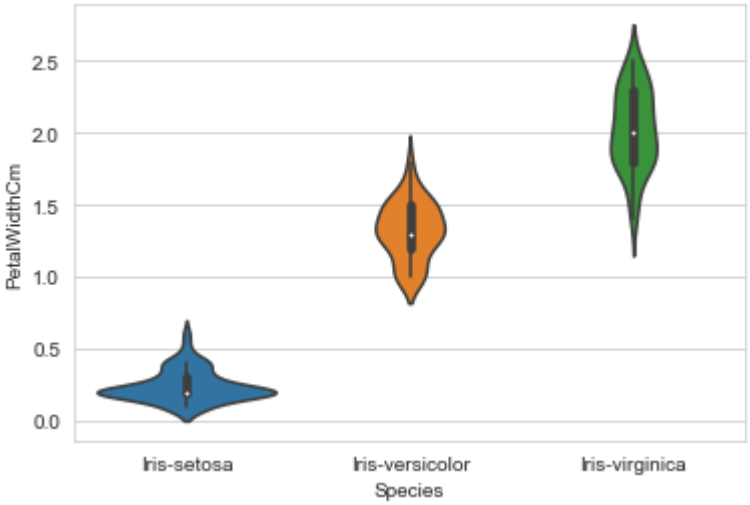
```
# A violin plot combines the benefits of the previous two plots and simplifies them

# Denser regions of the data are fatter, and sparser ones thinner in a violin plot

sns.violinplot(x="Species", y="PetalLengthCm", data=df, size=8)
plt.show()
```



violin plot on species and petalwidth

In [104]:

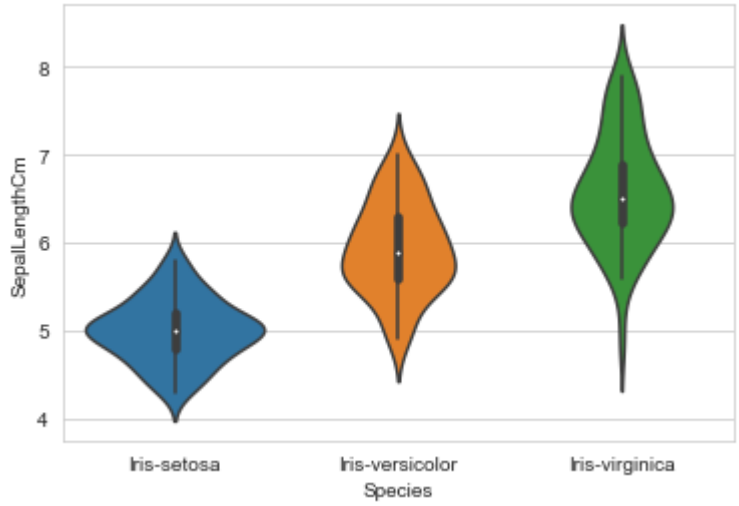
```
sns.violinplot(x="Species", y="PetalWidthCm", data=df, size=8)
plt.show()
```



violin plot on species and sepallenght

In [105]:

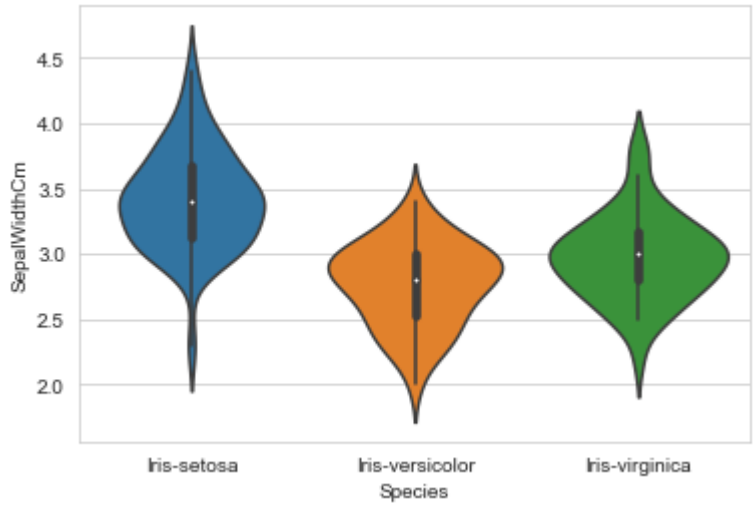
```
sns.violinplot(x="Species", y="SepalLengthCm", data=df, size=8)
plt.show()
```



violin plot on species and sepalwidth

In [106]:

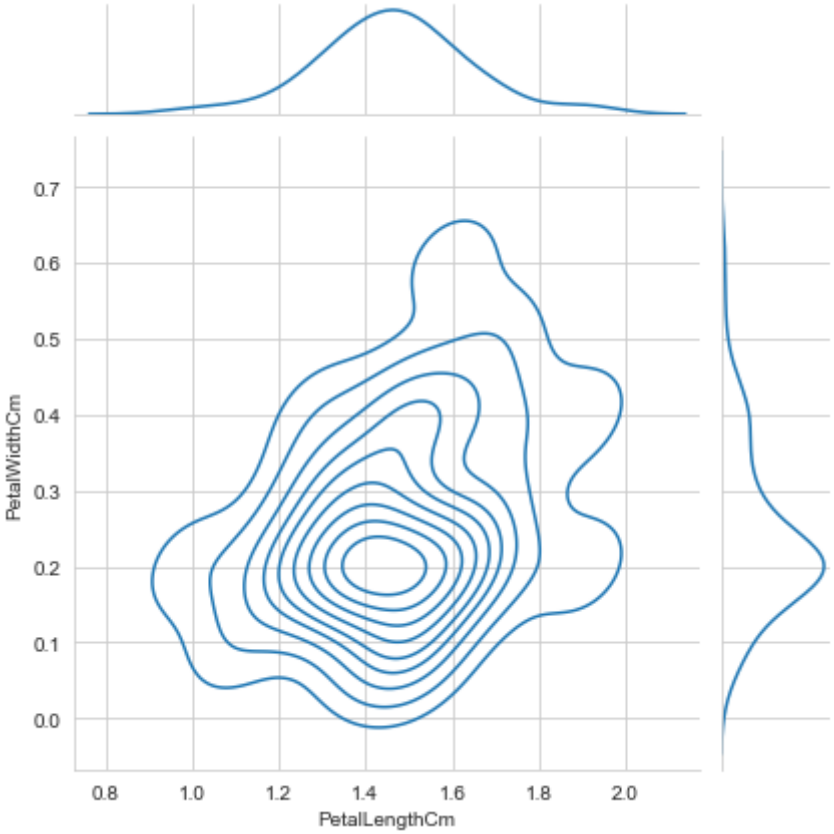
```
sns.violinplot(x="Species", y="SepalWidthCm", data=df, size=8)
plt.show()
```



11.Multivariate probability density and contour plot

In [100]:

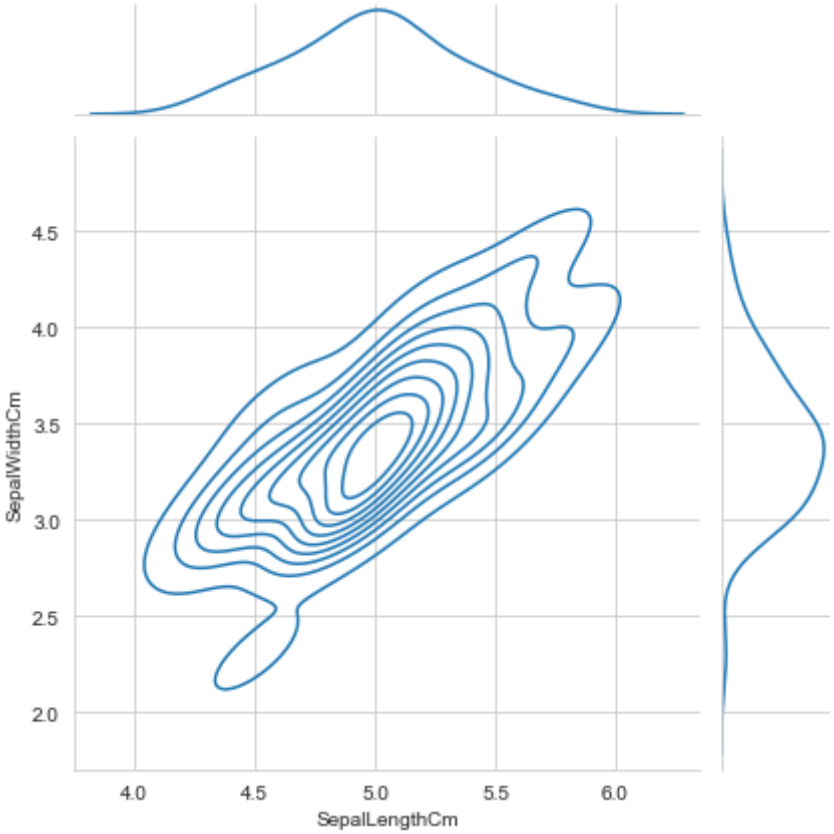
```
sns.jointplot(x="PetalLengthCm", y="PetalWidthCm", data=iris_setosa, kind="kde")
plt.show()
```



Observations: In this 2d plot Dark layer indicates more points and light layers or hills is called less points. These light to dark lines is called contours. This graph is called Contours probability density plot.

In [101]:

```
sns.jointplot(x="SepalLengthCm", y="SepalWidthCm", data=iris_setosa, kind="kde")
plt.show()
```




```
Observations:
 In this 2d plot Dark layer indicates more points and light layers or hills is called
less points.
 These light to dark lines is called contours. This graph is called Contours probability
density plot
```

Logistic regression implementation on iris

In [ ]:

```
```

In [52]:

```
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import  accuracy_score
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.preprocessing import StandardScaler
```

In [53]:

```
df.head(3)
```

Out[53]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	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

Independent and Dependent feature seperation

independent feature

In [56]:

```
X = df.drop('Species',axis=1)
```

Dependent feature

In [58]:

```
y = df['Species']
```

In [59]:

```
### Checking
```

In [60]:

```
X.head(3)
```

Out[60]:

	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

In [61]:

```
y.head(3)
```

Out[61]:

```
0    Iris-setosa
1    Iris-setosa
2    Iris-setosa
Name: Species, dtype: object
```

Spliting the data into train and test split

In [63]:

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.25,random_state= 355)
```

In [64]:

```
print(X_train.shape,y_train.shape)
```

```
(112, 4) (112,)
```

In [65]:

```
print(X_test.shape,y_test.shape)
```

```
(38, 4) (38,)
```

Standardizing or Feature scalling dataset

In [67]:

```
scaler = StandardScaler()
scaler
```

Out[67]:

```
StandardScaler
StandardScaler()
```

In [68]:

```
# Apply data
```

In [69]:

```
X_train = scaler.fit_transform(X_train)
```

In [70]:

```
X_test = scaler.transform(X_test)
```

In [71]:

```
X_train
```

Out[71]:

```
array([[ 0.20391445, -0.73732267,  0.77791956,  0.54928847],
       [-1.75017481,  0.42577788, -1.43224442, -1.35620782],
       [ 0.57030618,  0.65839799,  1.30137944,  1.77425037],
       [-1.87230539, -0.03946234, -1.54856884, -1.49231469],
       [ 0.81456734, -0.03946234,  0.83608177,  1.09371598],
       [-1.01739134,  1.12363821, -1.25775779, -0.81178031],
       [ 0.69243676,  0.19315777,  1.01056839,  0.82150223],
       [ 0.32604503, -0.03946234,  0.48710851,  0.27707471],
       [-1.26165249, -0.03946234, -1.37408221, -1.49231469],
       [ 1.18095908, -0.50470256,  0.60343293,  0.27707471],
       [-0.16247729, -0.50470256,  0.19629746,  0.14096784],
       [ 0.08178387, -0.03946234,  0.77791956,  0.82150223],
       [-1.13952192,  0.19315777, -1.31592   , -1.49231469],
       [-1.50591365,  0.42577788, -1.37408221, -1.35620782],
       [ 1.54735081, -0.03946234,  1.24321723,  1.22982286],
       [-0.52886902, -0.03946234,  0.4289463 ,  0.41318159],
       [-0.40673844, -1.20256289,  0.13813525,  0.14096784],
       [-1.75017481, -0.27208245, -1.37408221, -1.35620782],
```

In [72]:

```
X_test
```

Out[72]:

```
array([[ 1.30308965,  0.19315777,  0.95240618,  1.22982286],
       [ 0.93669792, -0.27208245,  0.48710851,  0.14096784],
       [-0.16247729, -0.50470256,  0.4289463 ,  0.14096784],
       [ 0.32604503, -0.50470256,  0.13813525,  0.14096784],
       [ 0.20391445, -0.27208245,  0.4289463 ,  0.41318159],
       [ 1.66948139,  0.42577788,  1.30137944,  0.82150223],
       [-0.28460787, -0.03946234,  0.4289463 ,  0.41318159],
       [ 0.57030618,  0.8910181 ,  1.0687306 ,  1.63814349],
       [ 0.69243676, -0.50470256,  1.0687306 ,  1.22982286],
       [ 0.69243676,  0.42577788,  0.89424398,  1.50203661],
       [ 0.32604503, -0.27208245,  0.54527072,  0.27707471],
       [ 0.32604503, -0.96994278,  1.0687306 ,  0.27707471],
       [-0.04034671, -0.50470256,  0.77791956,  1.63814349],
       [-0.77313018,  2.51935886, -1.31592   , -1.49231469],
       [-0.89526076,  1.12363821, -1.37408221, -1.22010094],
       [-1.13952192,  0.19315777, -1.31592   , -1.49231469],
       [-1.50591365,  1.35625832, -1.60673104, -1.35620782],
       [-1.38378307,  0.42577788, -1.43224442, -1.35620782],
       [-0.04034671, -0.96994278,  0.13813525,  0.00486096],
       [-0.04034671, -0.73732267,  0.77791956,  0.9576091 ],
       [-1.75017481, -0.03946234, -1.43224442, -1.35620782],
       [-0.77313018,  1.12363821, -1.31592   , -1.35620782],
       [-0.77313018,  0.8910181 , -1.37408221, -1.35620782],
       [ 0.44817561, -1.90042321,  0.4289463 ,  0.41318159],
       [-0.89526076,  0.65839799, -1.19959558, -0.94788718],
       [ 1.0588285 ,  0.19315777,  1.0687306 ,  1.63814349],
       [ 2.28013428,  1.82149853,  1.70851491,  1.36592974],
       [ 1.0588285 ,  0.65839799,  1.12689281,  1.77425037],
       [-1.26165249,  0.19315777, -1.25775779, -1.35620782],
       [ 1.0588285 ,  0.19315777,  0.37078409,  0.27707471],
       [-0.40673844,  2.75197897, -1.37408221, -1.35620782],
       [ 2.52439544,  1.82149853,  1.53402828,  1.09371598],
       [-1.13952192, -0.03946234, -1.37408221, -1.35620782],
       [-1.01739134,  1.12363821, -1.43224442, -1.22010094],
       [ 0.81456734, -0.03946234,  1.01056839,  0.82150223],
       [-0.04034671, -0.73732267,  0.19629746, -0.2673528 ],
       [ 0.81456734,  0.42577788,  0.77791956,  1.09371598],
       [-0.89526076,  1.82149853, -1.25775779, -1.35620782]])
```

Model Training

In [74]:

```
logistic_regre = LogisticRegression()
logistic_regre
```

Out[74]:

▾ LogisticRegression

LogisticRegression()

In [88]:

```
logistic_regre.fit(X_train,y_train)
```

Out[88]:

```
LogisticRegression()
```

Prediction

In [89]:

```
logistic_regre_pred = logistic_regre.predict(X_test)
logistic_regre_pred
```

Out[89]:

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

Performance Metrics

Confusion Matrix

In [90]:

```
from sklearn.metrics import confusion_matrix

confusion_mat = confusion_matrix(y_test,logistic_regre_pred)
confusion_mat
```

Out[90]:

```
array([[14,  0,  0],
       [ 0, 10,  0],
       [ 0,  0, 14]], dtype=int64)
```

In [91]:

```
truly_positive=confusion_mat[0][0]
falsely_positive=confusion_mat[0][1]
falsely_negative=confusion_mat[1][0]
truly_negative=confusion_mat[1][1]
```

Accuracy Score

In [92]:

```
accuracy= round(accuracy_score(y_test,logistic_regre_pred),4)
accuracy
```

Out[92]:

```
1.0
```

In [93]:

```
### manual caluculation for accuracy

accuracy_manual=round(((truly_positive+truly_negative)/(truly_positive+falsely_positive+fal
print("Accuracy of our model is {}".format(accuracy_manual))
```

```
Accuracy of our model is 1.0
```

Precision Score

In [95]:

```
precision_manual= round(truly_positive/(truly_positive+falsely_positive),4)
print("Precision of our model is : ",precision_manual)
```

```
Precision of our model is :  1.0
```

Recall Score

In [96]:

```
recall_manual=round(truly_positive/(truly_positive+falsely_negative),4)
print("Recall of our model is {}".format(recall_manual))
```

```
Recall of our model is 1.0
```

F-1 Score

In [98]:

```
f1_score=2*(precision_manual*recall_manual)/(precision_manual+recall_manual)
print("F-1 Score of our model is {} ".format(round(f1_score,4)))
```

```
F-1 Score of our model is 1.0
```

Classification Report

In [99]:

```
print(classification_report(y_test, logistic_regre_pred))
```

```
                precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        14
Iris-versicolor       1.00      1.00      1.00        10
 Iris-virginica       1.00      1.00      1.00        14

       accuracy                           1.00        38
      macro avg       1.00      1.00      1.00        38
   weighted avg       1.00      1.00      1.00        38
```

follow me on :-
Linkedin link: https://www.linkedin.com/in/dharavath-ramdas-a283aa213/
GitHub link: https://github.com/dharavathramdas101