Exploratory Data Analysis - IRIS Data

Problem Statement: To classify a given Iris flower into setosa, versicolor or virginica based on it's features; Sepal Length, Sepal Width, Petal Length and Petal width.

In [46]:
```
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import numpy as np
```

In [49]:
```
Iris = pd.read_csv("C://Users//veene//OneDrive//Desktop//EDA PROJECTS//Iri
```

In [50]:
```
pd.set_option('display.max_rows', None)
Iris.head(150)
```

Out[50]:

	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
5	6	5.4	3.9	1.7	0.4	Iris-setosa
6	7	4.6	3.4	1.4	0.3	Iris-setosa
7	8	5.0	3.4	1.5	0.2	Iris-setosa
8	9	4.4	2.9	1.4	0.2	Iris-setosa
9	10	4.9	3.1	1.5	0.1	Iris-setosa
10	11	5.4	3.7	1.5	0.2	Iris-setosa

In [51]:
```
Iris.tail(n=8)
```

Out[51]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
142	143	5.8	2.7	5.1	1.9	Iris-virginica
143	144	6.8	3.2	5.9	2.3	Iris-virginica
144	145	6.7	3.3	5.7	2.5	Iris-virginica
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

In [52]:
```
print('The shape of the Dataset is ', Iris.shape)
print('--------------')
print('The columns of the dataset are ', Iris.columns)
print('--------------')
```

```
The shape of the Dataset is  (150, 6)
--------------
The columns of the dataset are  Index(['Id', 'SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
--------------
```

Inference: There are 150 observations in the dataset and there are 6 columns.

In [53]:
```
#creating a list that would contain column names as elements.
```

In [54]:
```
#getting the list of number of unique valuess in each column
for column in col_list:
```

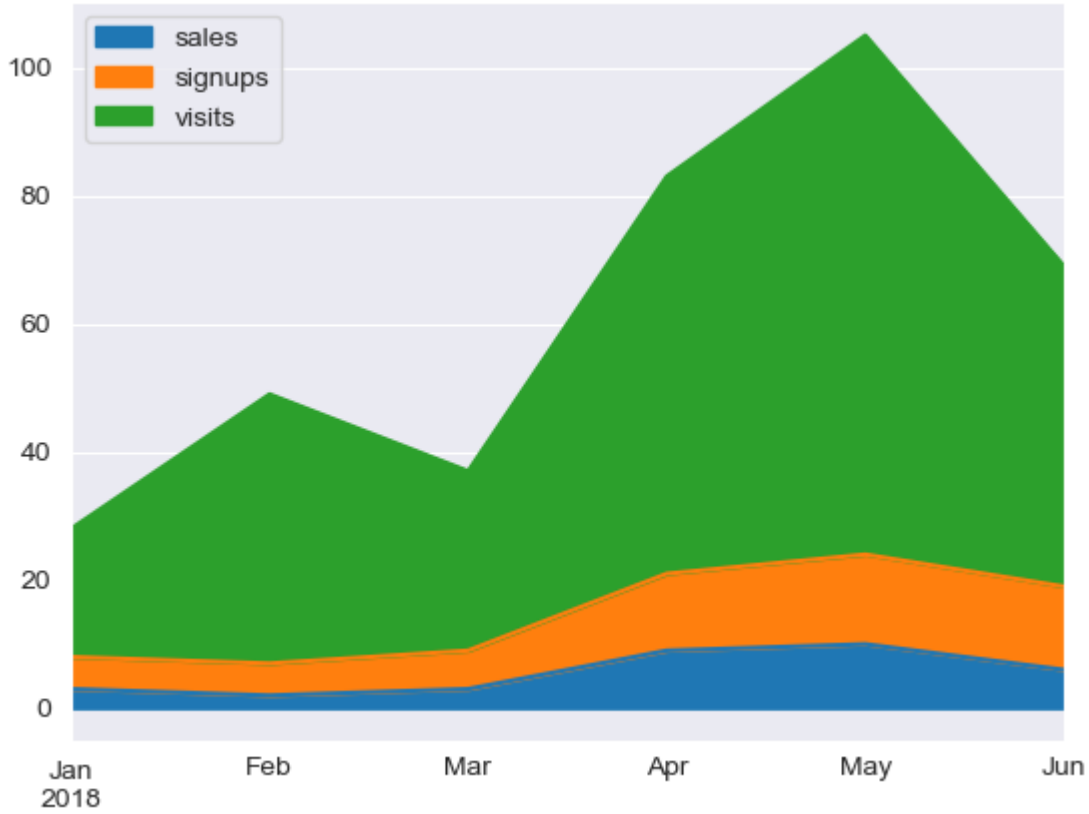
```
There are 150 unique values in the column Id
There are 35 unique values in the column SepalLengthCm
There are 23 unique values in the column SepalWidthCm
There are 43 unique values in the column PetalLengthCm
There are 22 unique values in the column PetalWidthCm
There are 3 unique values in the column Species
```

Inference:

1. Column 'Id' contains 150 unique values and ID is just a sequential number assigned to observation. Hence, It is not required for the dataset and needs to be droppped.
2. There are 3 class labels in target vector **Species**

In [55]:
```
#Dropping ID column
```

In [56]:
```
import pandas as pd
df = pd.DataFrame({
    'sales': [3, 2, 3, 9, 10, 6],
    'signups': [5, 5, 6, 12, 14, 13],
    'visits': [20, 42, 28, 62, 81, 50],
}, index=pd.date_range(start='2018/01/01', end='2018/07/01',
                       freq='M'))
ax = df.plot.area()
```



In [57]:
```
#Statistical norms
Iris.describe()
```

Out[57]:

	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

Inference:

1. All values contain 150 observations.

2. Column SepalLengthCm ranges from 4.3 to 7.9.
3. Column SepalWidthCm ranges from 2.0 to 4.4.
4. Column PetalLengthCm ranges from 1.0 to 6.9.
5. Column PetalWidthCm ranges from 0.1 to 2.5.

In [58]:
```
1 Iris.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
```

Inference:

1. There are only 150 datapoints in the dataset.
2. No column contains missing values.
3. No datatype conversion is required for any column.

In [59]:
```
1 Iris.duplicated()
```

Out[59]:
```
0      False
1      False
2      False
3      False
4      False
5      False
6      False
7      False
8      False
9      False
10     False
11     False
12     False
13     False
14     False
15     False
16     False
17     False
18     False
```

In [60]:
```
Iris[Iris.duplicated()==True]
```

Out[60]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	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

In [61]:
```
Iris.drop(34, axis=0, inplace=True)
```

In [64]:
```
Iris['Species'].nunique()
```

In [67]:
```
pie_df = pd.DataFrame(Iris['Species'].value_counts())
pie_df.rename(columns={'Species':'obs_count'}, inplace=True)
```

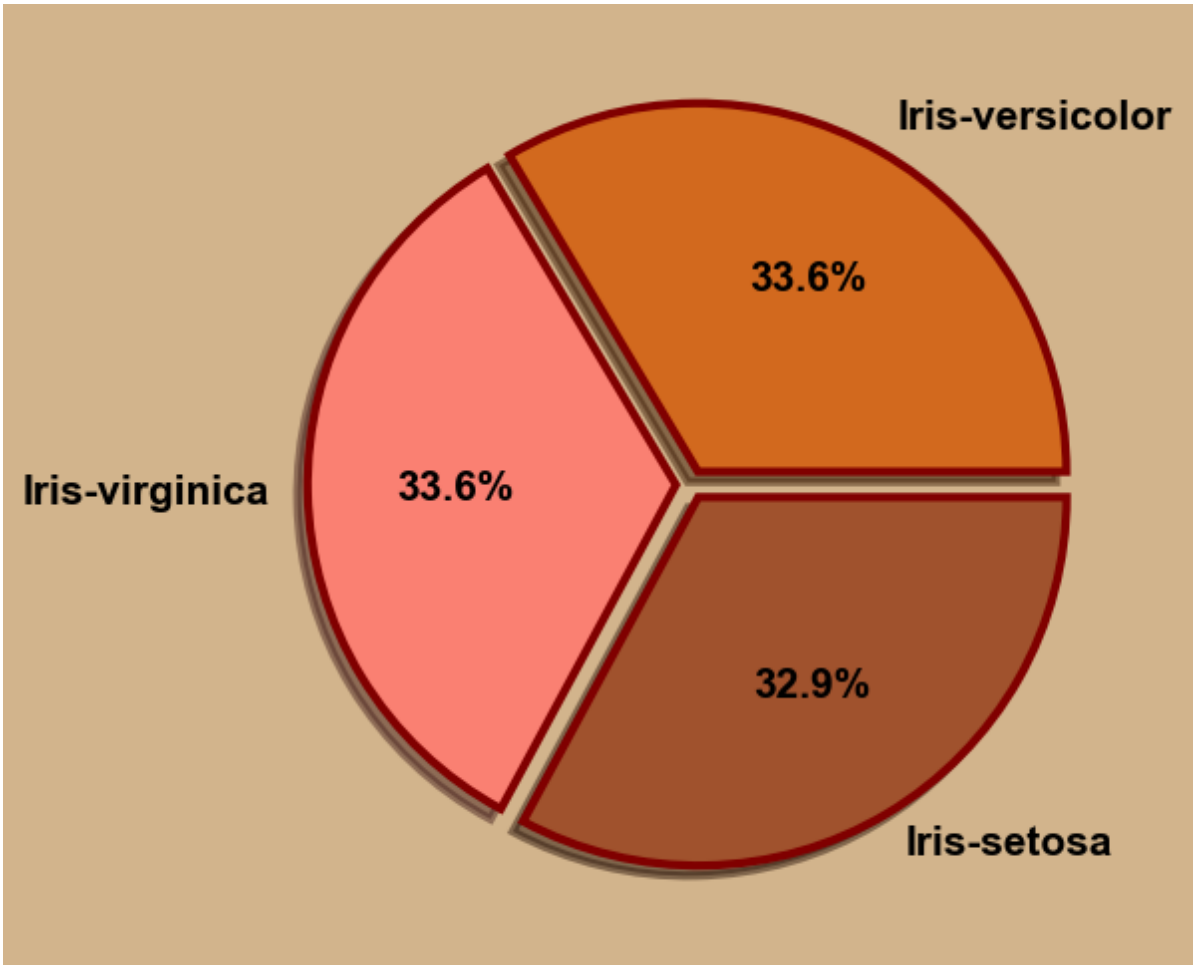
In [68]:
```
pie_df
```

Out[68]:

	obs_count
Iris-versicolor	50
Iris-virginica	50
Iris-setosa	49

In [69]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [70]:
```
plt.figure(figsize=(6,6))
plt.pie(pie_df['obs_count'],
        labels=pie_df['obs_count'].index,
        autopct='%.1f%%', shadow=True,
        explode=(0.04, 0.04, 0.04),
        colors=['chocolate','salmon','sienna'],
        wedgeprops={'linewidth':3,
                    'edgecolor':'maroon'},
        textprops={'fontweight':'bold',
                   'fontsize':15,
                   'color':'black'})
cf=plt.gcf()
cf.set_facecolor('tan')
plt.show()
```



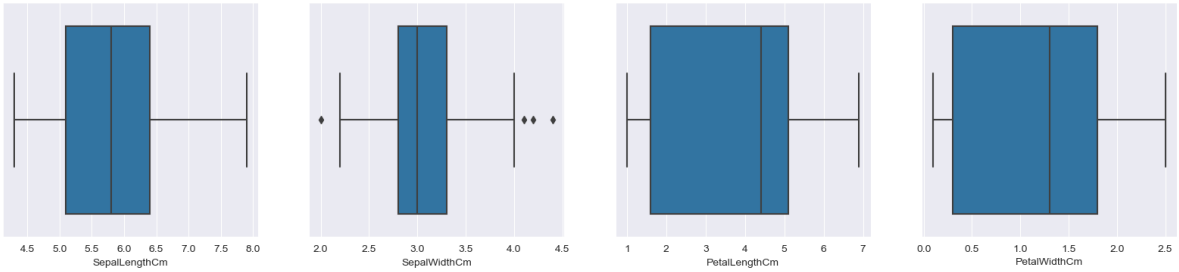
Inference:

All the class labels have almost equal observation counts. Data set is balanced.

In [71]:
```
fea_vec = col_list[1:5]
```

Out[71]:
```
['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']
```

In [72]:
```
plt.figure(figsize=(20,4))
for i in range(len(fea_vec)):
    plt.subplot(1,4,i+1)
    sns.boxplot(data=Iris, x=fea_vec[i])
```



Inference:

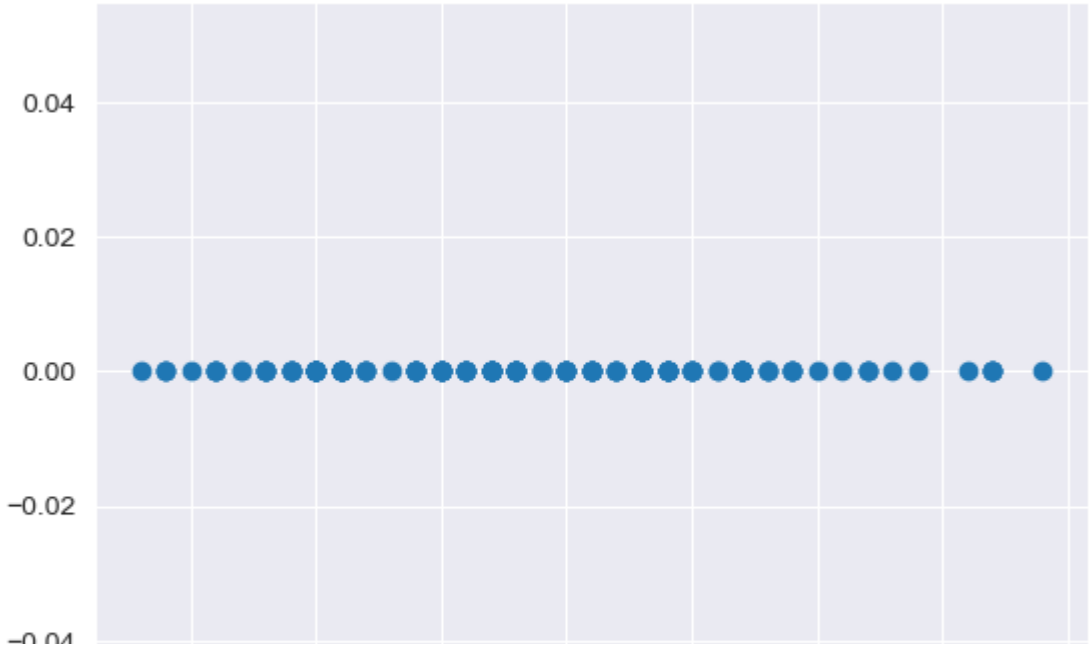
Outliers are observed only in SepalWidth

A. Univariate Analysis

1. 1D Scatter plot

Let's put 1D scatter plot by considering each f the column with respct to corresponding class label group.

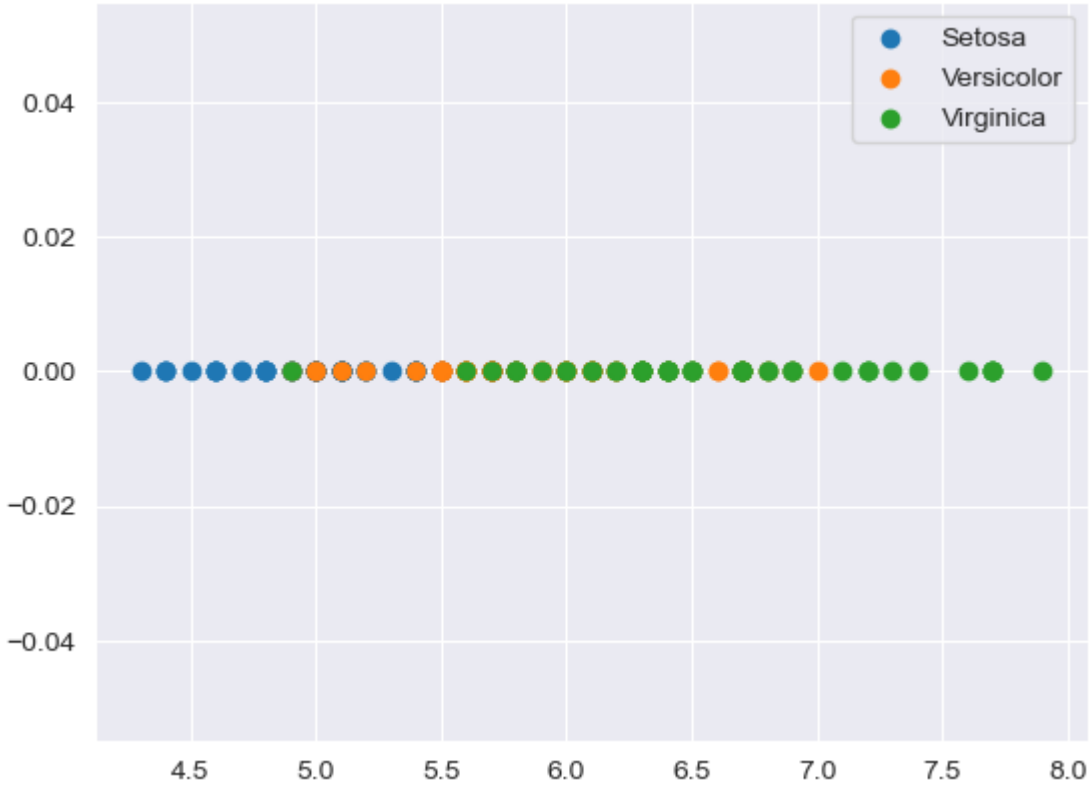
In [74]:
```
plt.scatter(Iris['SepalLengthCm'],np.zeros_like(Iris['SepalLengthCm']))
plt.show()
```



We are not able to infer anything from the above plot that would be helpful to problem statement. Hence, let's plot 1-D scatter plot in the context of target vector.

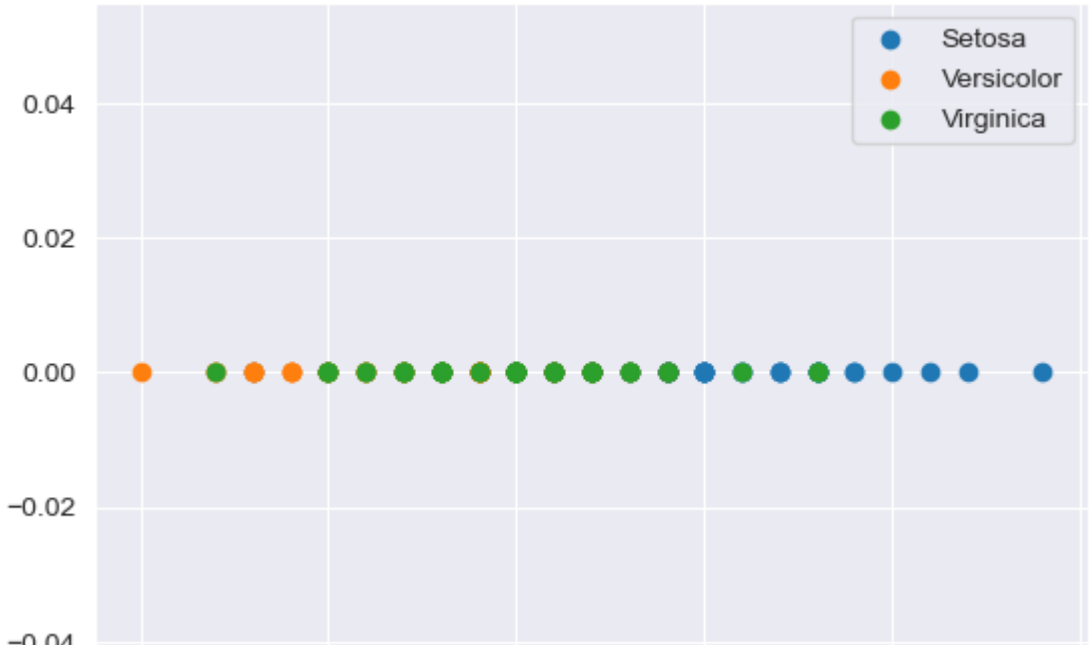
In [75]:
```
#Let me create group of datasets with respect to unique values in 'Specie'

#g_set = Iris.groupby('Species')
#Iris_setosa = g_set.get_group('Iris-setosa')
#Iris_versicolor = g_set.get_group('Iris-versicolor')
#Iris_virginica = g_set.get_group('Iris-virginica')

Iris_setosa = Iris[Iris['Species']=='Iris-setosa']
Iris_versicolor = Iris[Iris['Species']=='Iris-versicolor']
Iris_virginica = Iris[Iris['Species']=='Iris-virginica']
```

In [76]:
```
plt.scatter(Iris_setosa['SepalLengthCm'],np.zeros_like(Iris_setosa['SepalL
plt.scatter(Iris_versicolor['SepalLengthCm'],np.zeros_like(Iris_versicolor
plt.scatter(Iris_virginica['SepalLengthCm'],np.zeros_like(Iris_virginica['
plt.legend()
plt.show()
```



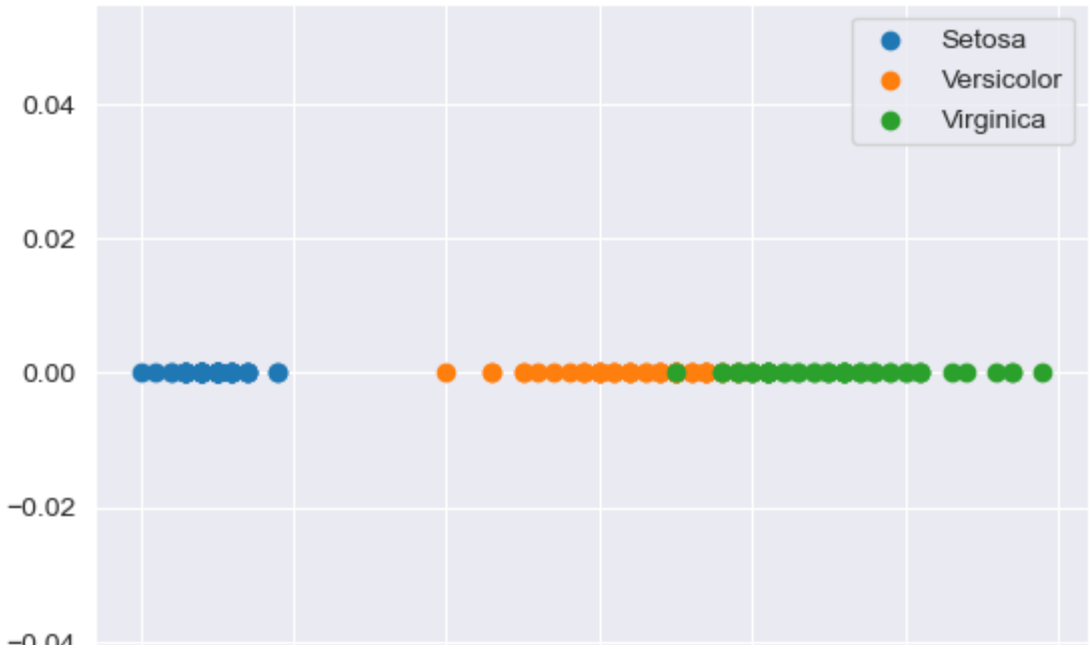
Inference: Using Sepal Length alone It is not possible to differentiate between Iris flowers. We are not able to differentiate overlap.

In [77]:
```
1 plt.scatter(Iris_setosa['SepalWidthCm'],np.zeros_like(Iris_setosa['SepalWi
2 plt.scatter(Iris_versicolor['SepalWidthCm'],np.zeros_like(Iris_versicolor[
3 plt.scatter(Iris_virginica['SepalWidthCm'],np.zeros_like(Iris_virginica['S
4 plt.legend()
5 plt.show()
```



Inference: Using Sepal Length alone It is not possible to differentiate between Iris flowers. We are not able to differentiate overlap.

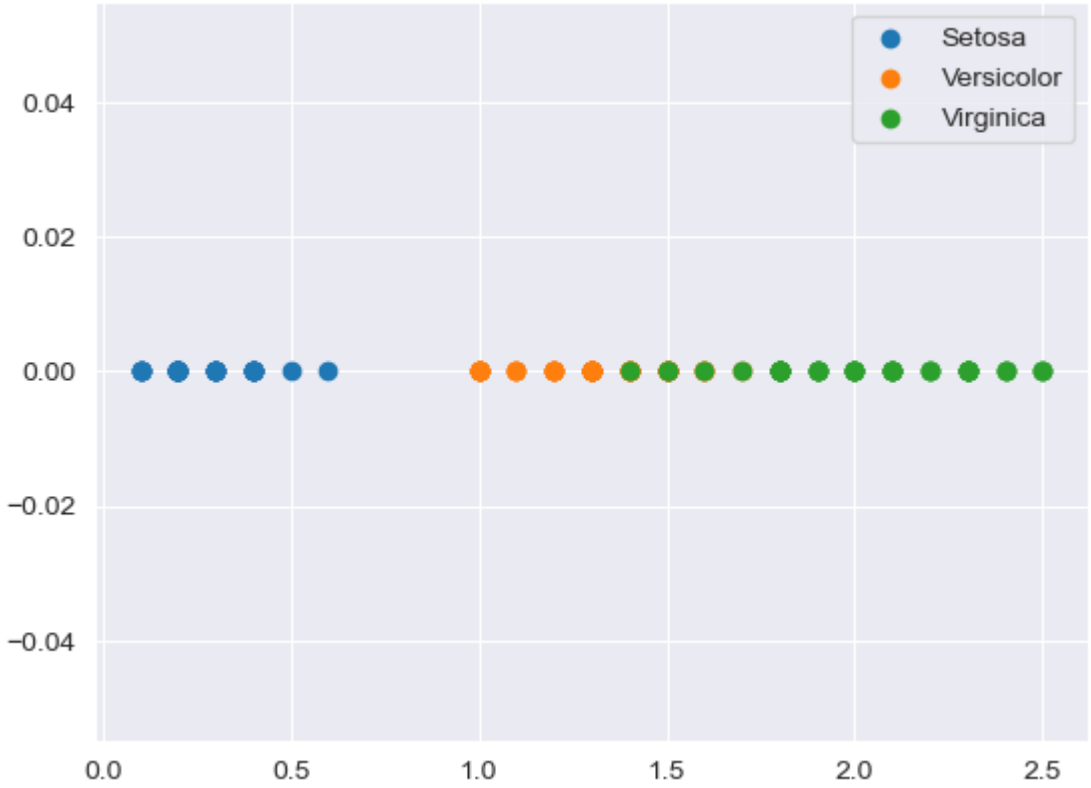
In [78]:
```
1 plt.scatter(Iris_setosa['PetalLengthCm'],np.zeros_like(Iris_setosa['PetalL
2 plt.scatter(Iris_versicolor['PetalLengthCm'],np.zeros_like(Iris_versicolor
3 plt.scatter(Iris_virginica['PetalLengthCm'],np.zeros_like(Iris_virginica['
4 plt.legend()
5 plt.show()
```



Inference:

1. If PL is less than 2, it is setosa
2. Slight overlap between versicolor and virginica but indeterminable.

In [79]:
```
plt.scatter(Iris_setosa['PetalWidthCm'],np.zeros_like(Iris_setosa['PetalWi
plt.scatter(Iris_versicolor['PetalWidthCm'],np.zeros_like(Iris_versicolor[
plt.scatter(Iris_virginica['PetalWidthCm'],np.zeros_like(Iris_virginica['P
plt.legend()
plt.show()
```

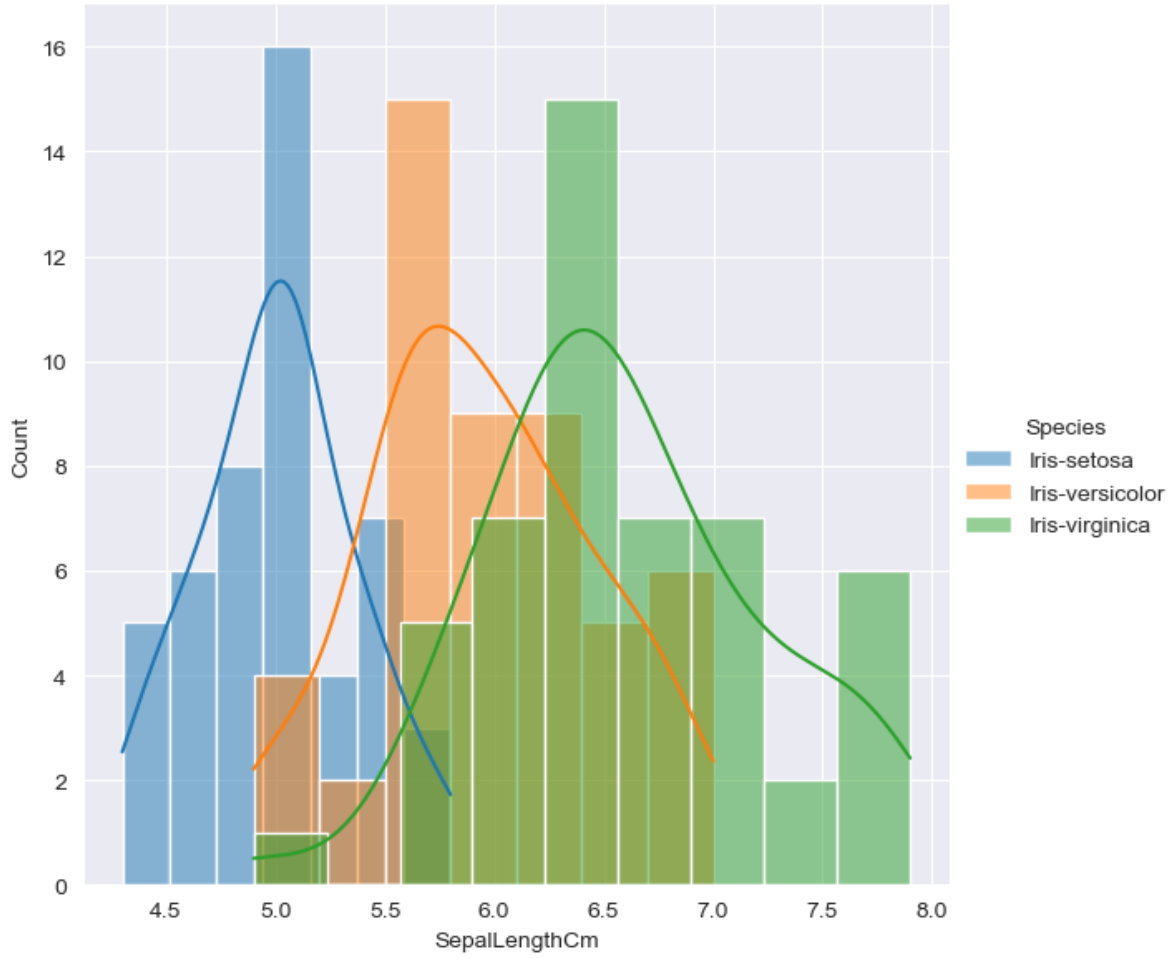


Inference:

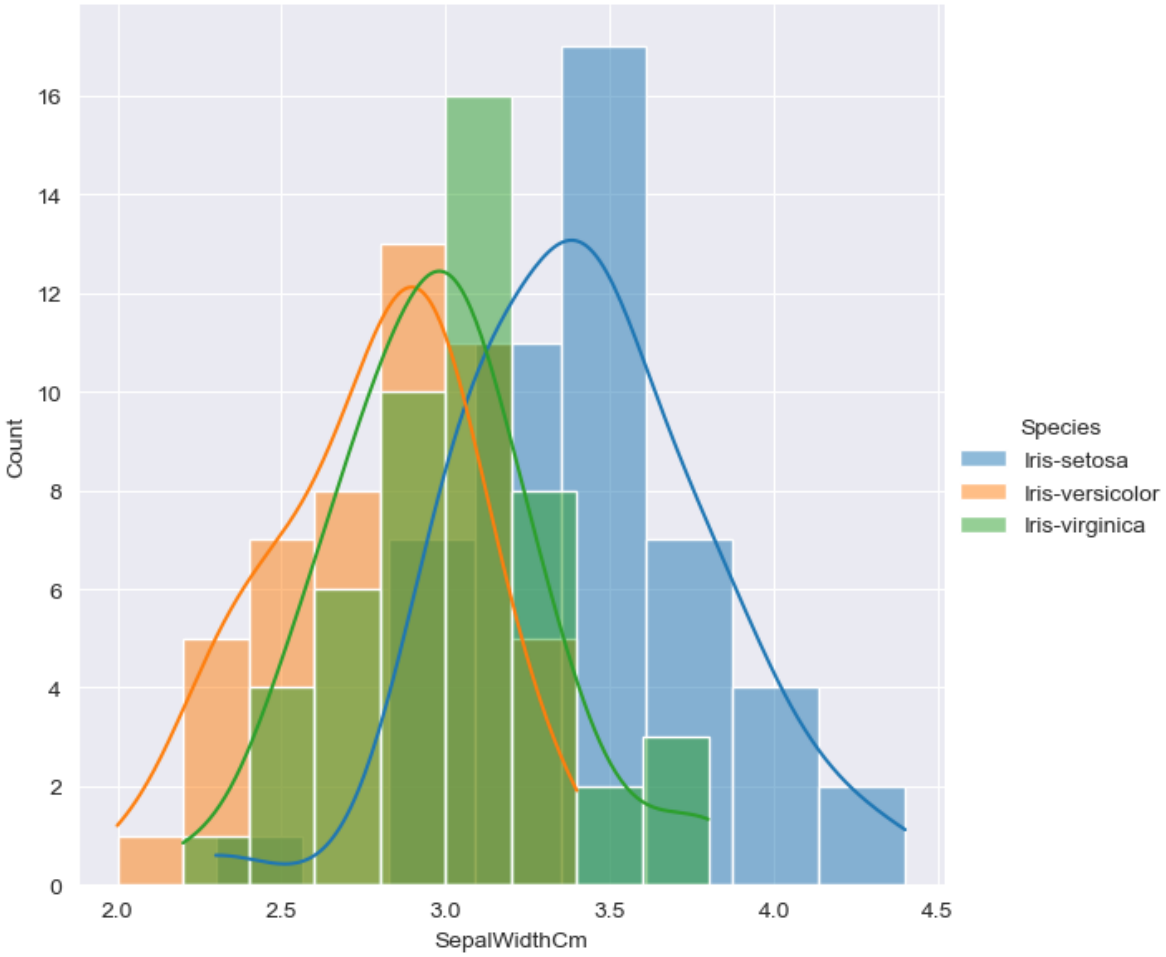
1. Setosa is separable(<0.6)
2. Overlap between virginica and versicolor is not distinguishable

2. Univariate analysis using Histogram

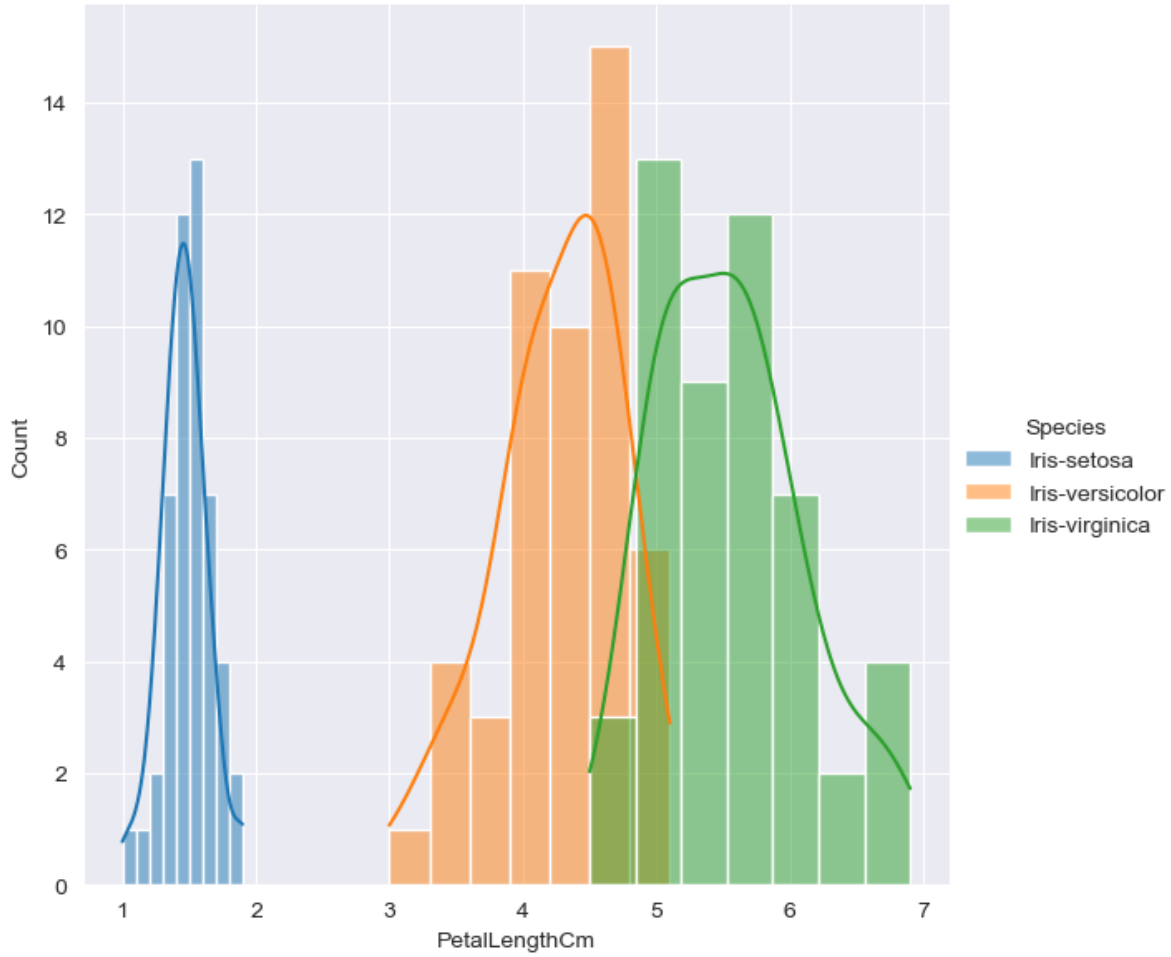
In [80]:
```
a = sns.FacetGrid(Iris, hue='Species',height=6)\
    .map(sns.histplot, 'SepalLengthCm', kde=True)\
    .add_legend();
plt.show()
```



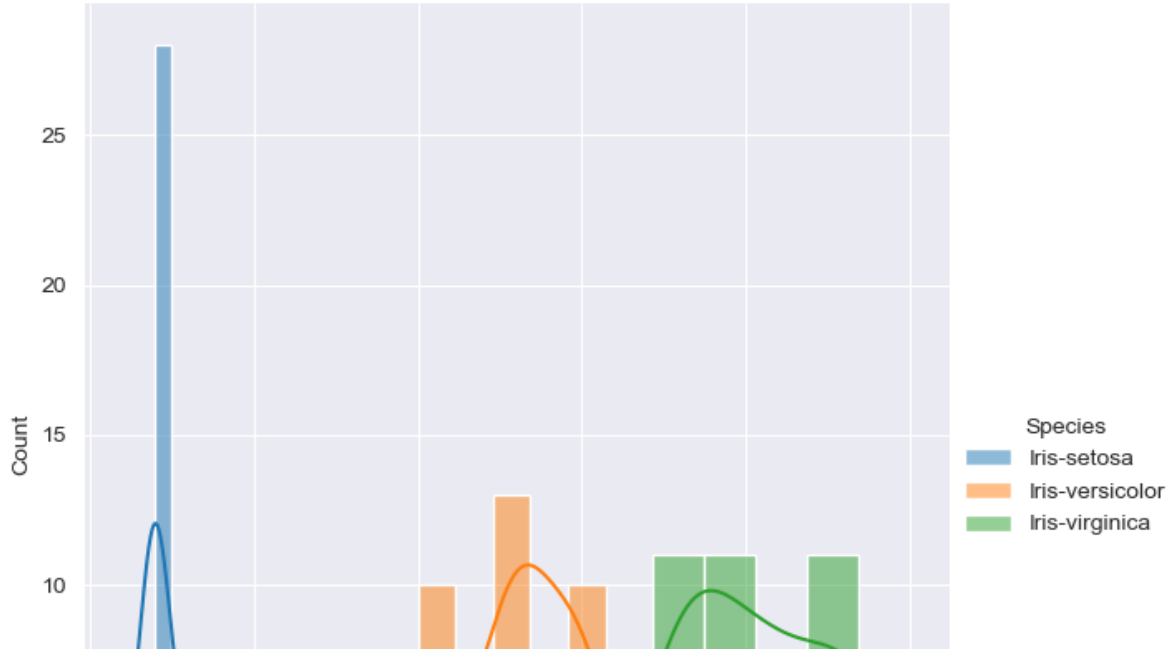
In [81]:
```
sns.FacetGrid(Iris, hue='Species',height=6)\
   .map(sns.histplot, 'SepalWidthCm', kde=True)\
   .add_legend();
plt.show()
```



In [82]:
```
sns.FacetGrid(Iris, hue='Species',height=6)\
   .map(sns.histplot, 'PetalLengthCm', kde=True)\
   .add_legend();
plt.show()
```



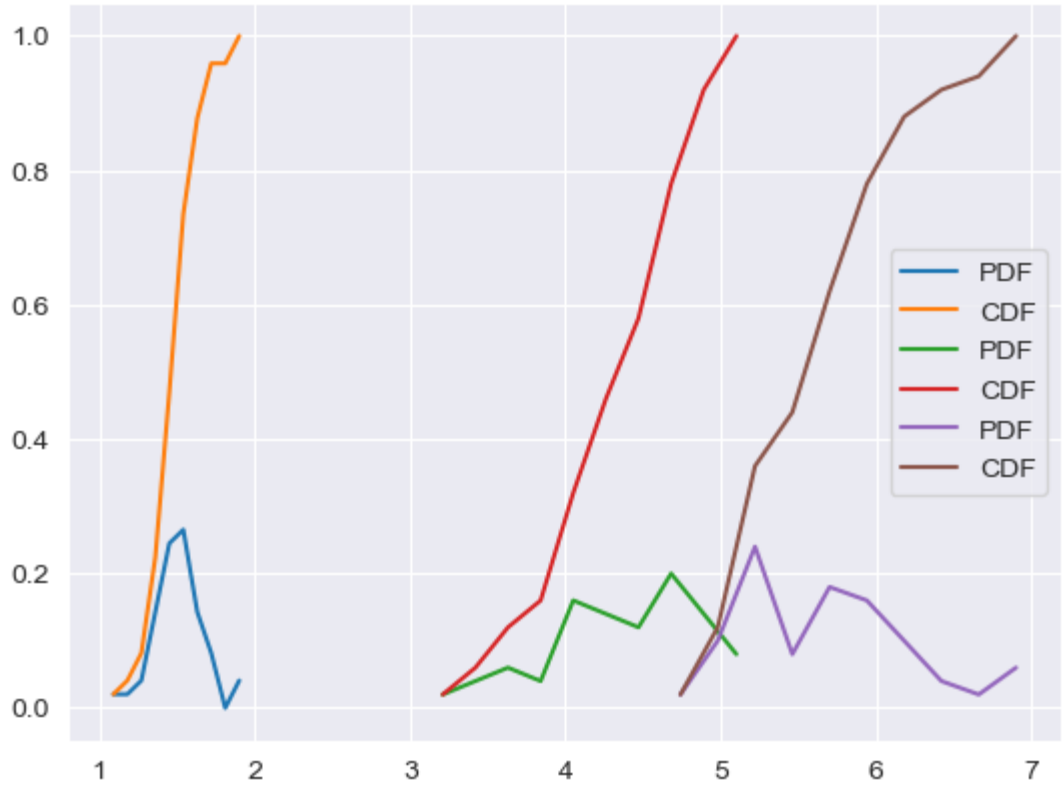
In [83]:
```
sns.FacetGrid(Iris, hue='Species',height=6)\
   .map(sns.histplot, 'PetalWidthCm', kde=True)\
   .add_legend();
plt.show()
```



Probability Density Function and Cumulative Distribution Function

In [84]:
```
counts, bin_edges = np.histogram(Iris_setosa['PetalLengthCm'], bins=10, de
pdf = counts/sum(counts)
#compute CDF
cdf = np.cumsum(pdf)
plt.plot(bin_edges[1:],pdf, label='PDF')
plt.plot(bin_edges[1:],cdf, label='CDF')
plt.legend()

counts, bin_edges = np.histogram(Iris_versicolor['PetalLengthCm'], bins=10
pdf = counts/sum(counts)
#compute CDF
cdf = np.cumsum(pdf)
plt.plot(bin_edges[1:],pdf, label='PDF')
plt.plot(bin_edges[1:],cdf, label='CDF')
plt.legend()

counts, bin_edges = np.histogram(Iris_virginica['PetalLengthCm'], bins=10,
pdf = counts/sum(counts)
#compute CDF
cdf = np.cumsum(pdf)
plt.plot(bin_edges[1:],pdf, label='PDF')
plt.plot(bin_edges[1:],cdf, label='CDF')
plt.legend()
```

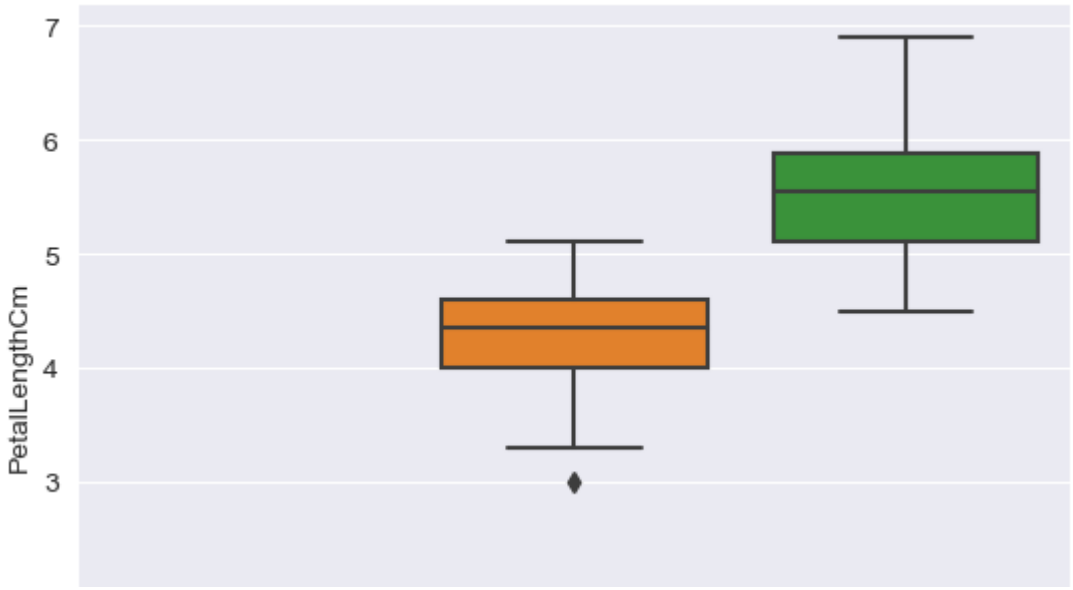
Out[84]: <matplotlib.legend.Legend at 0x11a7588ac90>



Box-Whiskers plot

In [85]:
```
1 sns.boxplot(x='Species', y='PetalLengthCm', data=Iris)
```

Out[85]: <Axes: xlabel='Species', ylabel='PetalLengthCm'>



In [86]:
```
1 sns.violinplot(x='Species', y='PetalLengthCm', data=Iris)
```

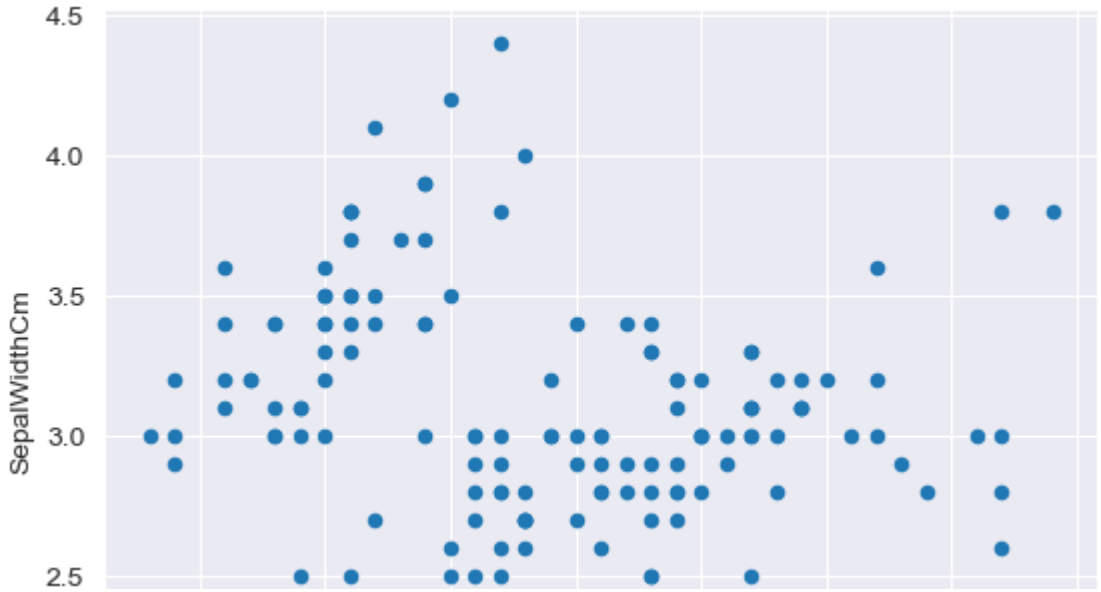
Out[86]: <Axes: xlabel='Species', ylabel='PetalLengthCm'>

B. Bivariate Analysis

1. 2D Scatter Plot

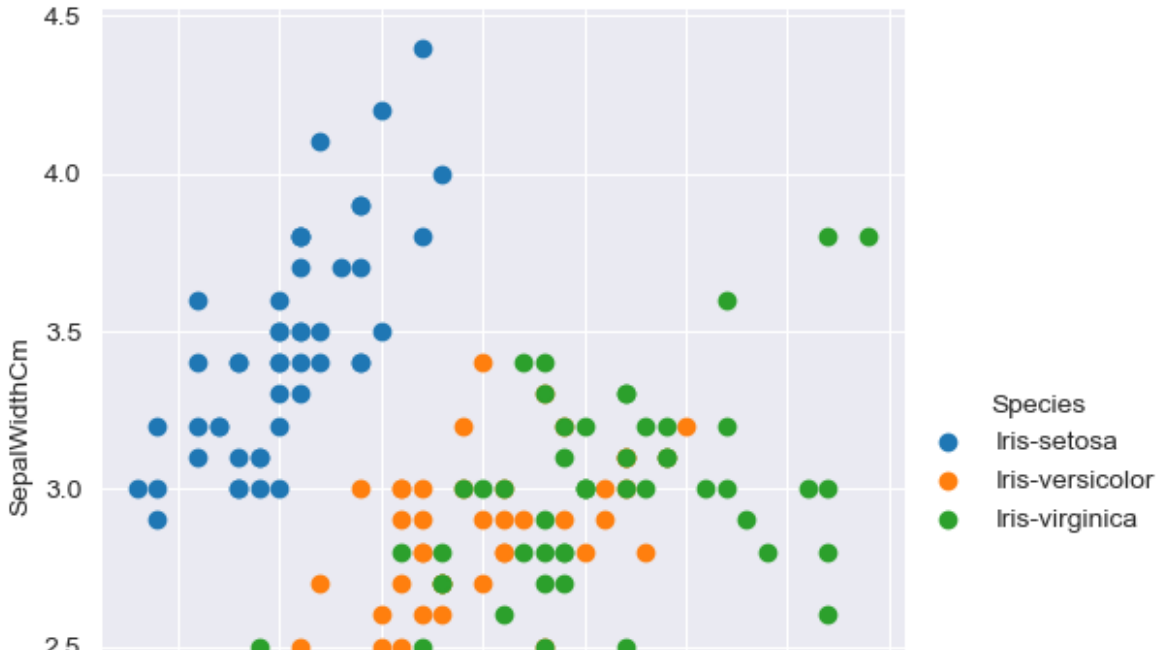
In [87]:
```
1 Iris.plot(kind='scatter', x='SepalLengthCm', y='SepalWidthCm')
```

Out[87]: <Axes: xlabel='SepalLengthCm', ylabel='SepalWidthCm'>



We are not able to infer anything from the above scatter plot which would help us to solve our problem statement, hence let's add 'hue' attribute/argument form Seaborn which would give us color segmentation based on a given column.

In [88]:
```
1 sns.set_style('darkgrid')
2 sns.FacetGrid(data=Iris, hue='Species',height=5)\
3    .map(plt.scatter,'SepalLengthCm','SepalWidthCm')\
4    .add_legend();
5 plt.show()
```

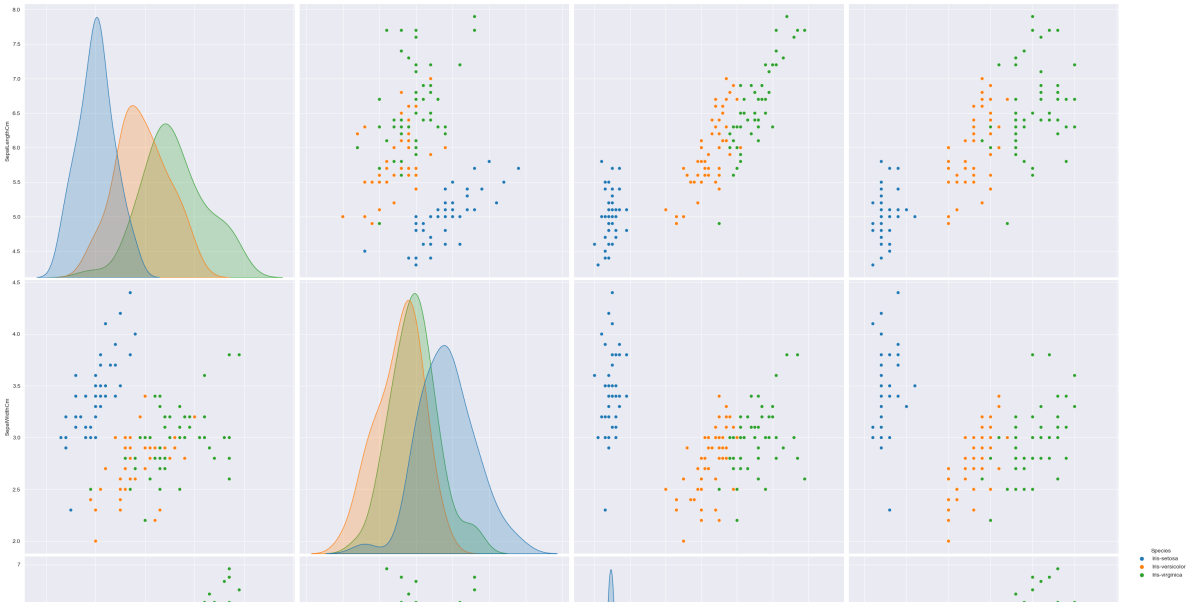


It would be difficult or not feasible to write different code for each set of columns for a 2D scatter plot. Hence let's make use of pair plot.

2. 2D Scatter plot

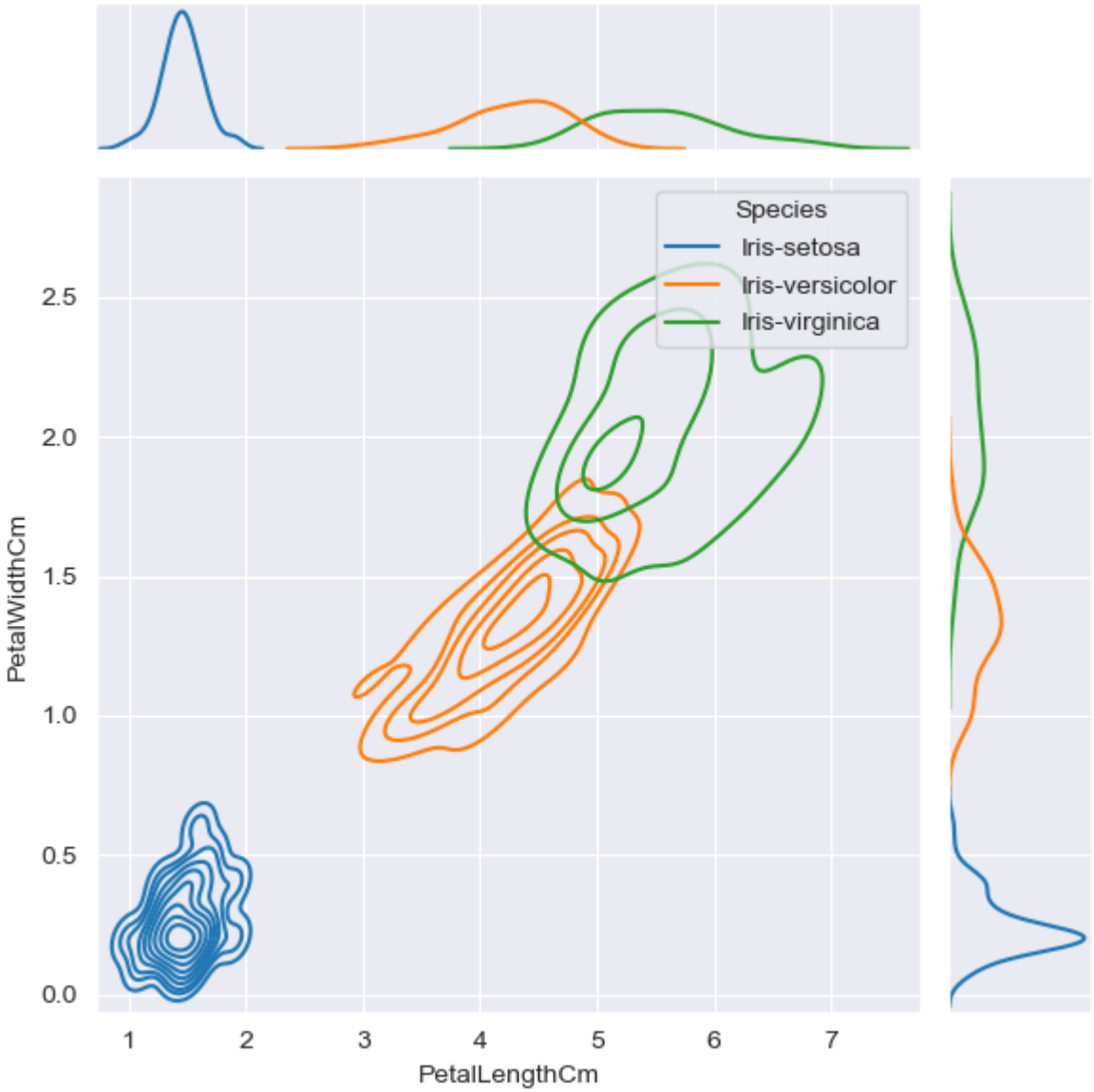
In [89]:
```
1
2 plt.figure(figsize=(50,75))
3 sns.pairplot(Iris, hue='Species', height=7)
4 plt.show()
```

```
<Figure size 5000x7500 with 0 Axes>
```



In [90]:
```
1 sns.jointplot(x='PetalLengthCm', y = 'PetalWidthCm', data = Iris, kind = '
```

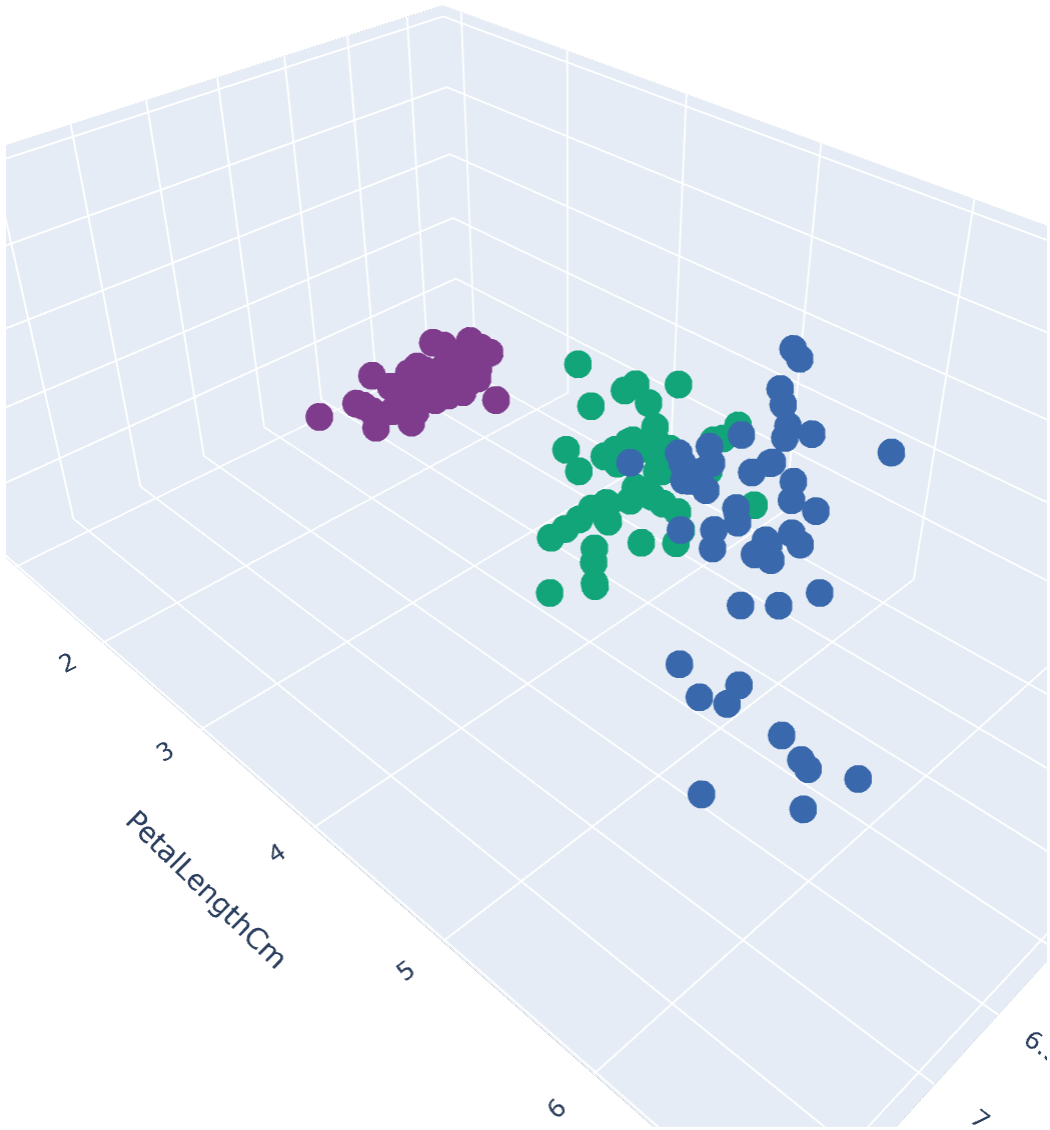
Out[90]: <seaborn.axisgrid.JointGrid at 0x11a74ada810>



C. Multivariate analysis

1. 3D scatter plot

In [91]:
```
1 import plotly.express as px
2 fig = px.scatter_3d(Iris, x='SepalLengthCm', y='PetalLengthCm', z='PetalWi
3               color='Species',color_discrete_sequence=px.colors.qualitativ
4 fig.show()
```



2. Heatmap

In [92]:
```
1 sns.heatmap(Iris.corr(), annot=True)
```

Out[92]: <Axes: >