

```
In [54]: #Urmila Jagdhane  
#LGM-VIP Data Science Task 1
```

```
In [123...  
  
import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns  
sns.set(style='white', color_codes=True)  
  
from sklearn.linear_model import LogisticRegression  
from sklearn.model_selection import train_test_split  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn import svm  
from sklearn import metrics  
from sklearn.tree import DecisionTreeClassifier
```

```
In [58]: df=pd.read_csv(r"C:\Users\urmil\Downloads\Iris csv.csv")
```

```
In [22]: df
```

```
Out[22]:
```

	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

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
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 [59]: df.head()
```

Out[59]:	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

```
In [60]: df.tail()
```

Out[60]:	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
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 [61]: df.shape
```

```
Out[61]: (150, 6)
```

```
In [62]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype  
---  -
 0   Id              150 non-null   int64  
 1   SepalLengthCm   150 non-null   float64
 2   SepalWidthCm    150 non-null   float64
 3   PetalLengthCm   150 non-null   float64
 4   PetalWidthCm    150 non-null   float64
 5   Species         150 non-null   object  
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

```
In [63]: df.isnull().sum()
```

```
Out[63]: Id              0
SepalLengthCm          0
SepalWidthCm           0
PetalLengthCm          0
PetalWidthCm           0
Species                0
dtype: int64
```

```
In [64]: df.describe()
```

```
Out[64]:
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

```
In [65]: df.nunique()
```

```
Out[65]: Id                150
SepalLengthCm            35
SepalWidthCm             23
PetalLengthCm            43
PetalWidthCm             22
Species                  3
dtype: int64
```

```
In [66]: df['SepalWidthCm'].value_counts()
```

```
Out[66]: 3.0      26
2.8      14
3.2      13
3.1      12
3.4      12
2.9      10
2.7       9
2.5       8
3.5       6
3.3       6
3.8       6
2.6       5
2.3       4
3.7       3
2.4       3
2.2       3
3.6       3
3.9       2
4.4       1
4.0       1
4.1       1
4.2       1
2.0       1
Name: SepalWidthCm, dtype: int64
```

```
In [67]: df.corr()
```

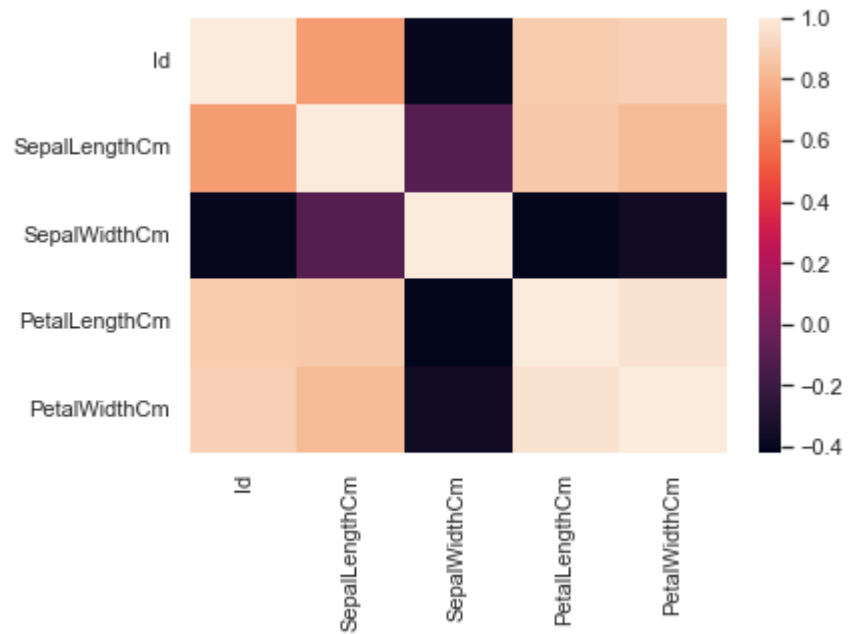
```
Out[67]:
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
--	-----------	----------------------	---------------------	----------------------	---------------------

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
Id	1.000000	0.716676	-0.397729	0.882747	0.899759
SepalLengthCm	0.716676	1.000000	-0.109369	0.871754	0.817954
SepalWidthCm	-0.397729	-0.109369	1.000000	-0.420516	-0.356544
PetalLengthCm	0.882747	0.871754	-0.420516	1.000000	0.962757

```
In [68]: sns.heatmap(df.corr())
```

```
Out[68]: <AxesSubplot:>
```



```
In [69]: df.columns
```

```
Out[69]: Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',  
              'Species'],  
              dtype='object')
```

```
In [70]: n = len(df[df['Species'] == 'Iris-versicolor'])
print("No of Versicolor in Dataset:",n)
```

No of Versicolor in Dataset: 50

```
In [71]: n1 = len(df[df['Species'] == 'Iris-versicolor'])
print("No of Versicolor in Dataset:",n1)
```

No of Versicolor in Dataset: 50

```
In [72]: n2 = len(df[df['Species'] == 'Iris-versicolor'])
print("No of Versicolor in Dataset:",n2)
```

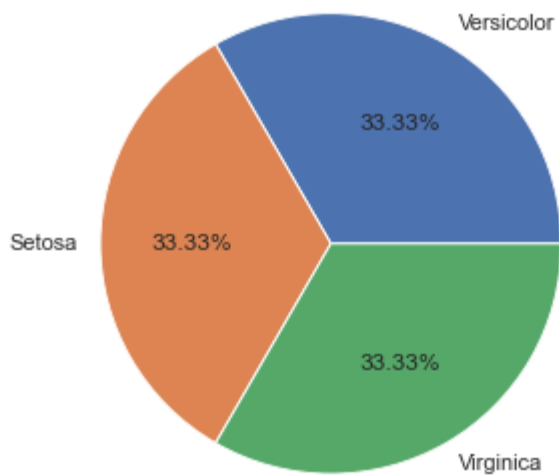
No of Versicolor in Dataset: 50

```
In [73]: df.isnull().sum()
```

```
Out[73]: Id                0
SepalLengthCm            0
SepalWidthCm             0
PetalLengthCm            0
PetalWidthCm             0
Species                 0
dtype: int64
```

```
In [78]: fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.axis('equal')
l = ['Versicolor', 'Setosa', 'Virginica']
s = [50, 50, 50]
ax.pie(s, labels = l, autopct= '%1.2f%%')

plt.show()
```

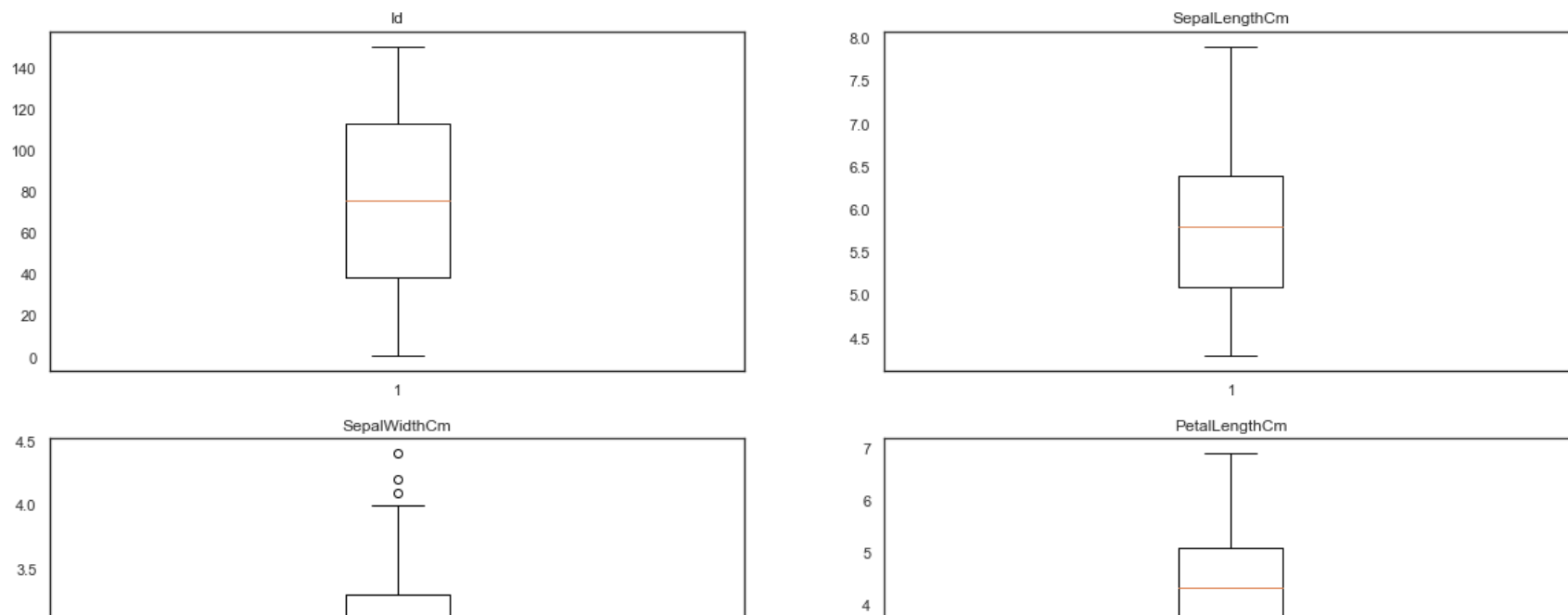


In [79]:

```
# Checking outliers
rows = 2
cols = 2
fig, axs = plt.subplots(rows,cols)
index=-1
for i in range(rows):
    for j in range(cols):
        index+=1
        axs[i,j].boxplot(df[df.columns[index]])
        axs[i,j].set_title(df.columns[index])
fig.set_size_inches(20,10)
fig.show()
```

C:\Users\urmil\AppData\Local\Temp\ipykernel_11072\1577605351.py:12: UserWarning: Matplotlib is currently using module://matplotlib_inline.backend_inline, which is a non-GUI backend, so cannot show the figure.

```
fig.show()
```



```
In [80]: sns.countplot(df['Species'])
```

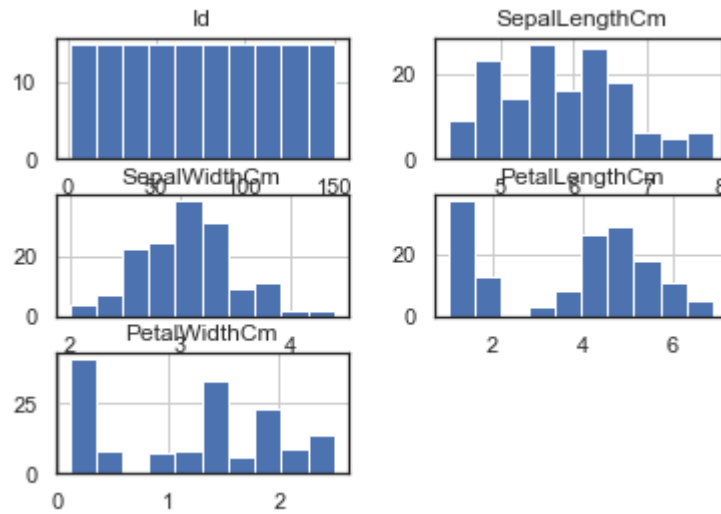
C:\Users\urmil\Documents\URMI\lib\site-packages\seaborn_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

```
warnings.warn(
```

```
Out[80]: <AxesSubplot:xlabel='Species', ylabel='count'>
```

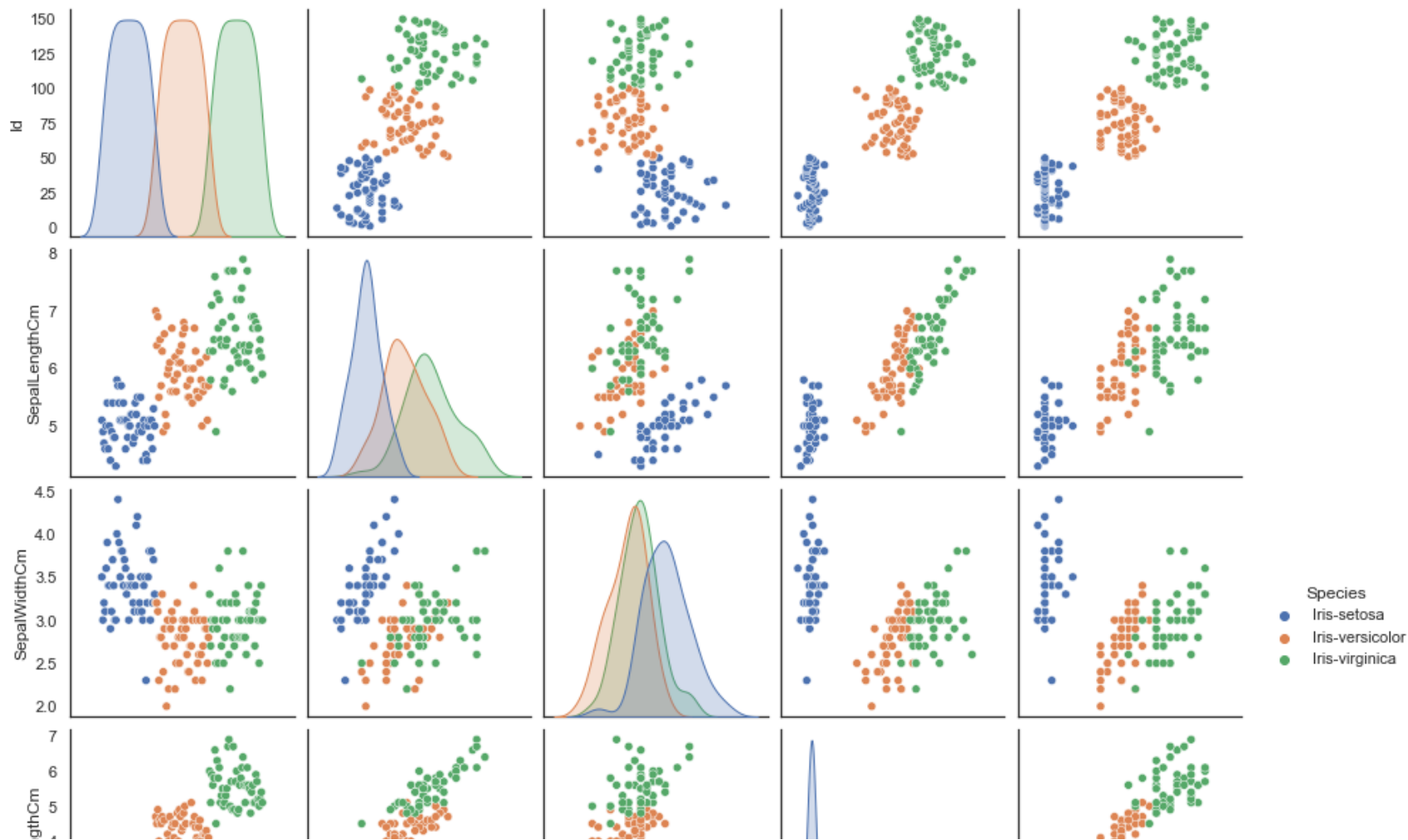



```
In [81]: df.hist()  
plt.show()
```



```
In [82]: sns.pairplot(df, hue='Species')
```

```
Out[82]: <seaborn.axisgrid.PairGrid at 0x15a1fd24220>
```



```
In [83]: train, test = train_test_split(df, test_size = 0.25)
print(train.shape)
print(test.shape)
```

```
(112, 6)
```

```
(38, 6)
```

In [106...

```
train_X = df[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',]]  
  
train_y = df.Species  
  
test_X = df[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',]]  
  
test_y = df.Species
```

In [101...

```
train_X.head()
```

Out[101...

	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 [107...

```
test_y.head()
```

Out[107...

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

In [108...

```
#Using LogisticRegression  
model = LogisticRegression()  
model.fit(train_X, train_y)  
prediction = model.predict(test_X)  
print('Accuracy:', metrics.accuracy_score(prediction, test_y))
```

```
Accuracy: 0.9733333333333334
```

In [112...

```
#Using Support Vector
from sklearn.svm import SVC
model1 = SVC()
model1.fit(train_X,train_y)

pred_y = model1.predict(test_X)

from sklearn.metrics import accuracy_score
print("Acc=",accuracy_score(test_y,pred_y))
```

Acc= 0.9733333333333334

In [113...

```
#Using KNN Neighbours
from sklearn.neighbors import KNeighborsClassifier
model2 = KNeighborsClassifier(n_neighbors=5)
model2.fit(train_X,train_y)
y_pred2 = model2.predict(test_X)

from sklearn.metrics import accuracy_score
print("Accuracy Score:,accuracy_score(test_y,y_pred2)")
```

Accuracy Score:,accuracy_score(test_y,y_pred2)

In [122...

```
result = pd.DataFrame({
    'Model': ['Logistic Regression','Support Vector Machines','KNN'],
    'Score': [0.97,0.97,0.96]})
result_df = result.sort_values(by='Score', ascending=False)
result_df = result_df.set_index('Score')
result_df.head(9)
```

Out[122...

	Model
Score	
0.97	Logistic Regression
0.97	Support Vector Machines
0.96	KNN