Import Neccessary libraries

In [45]:
```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import seaborn as sns
```

Load the Dataset

In [46]:
```
1 dataset = pd.read_csv("D:\CodSoft Task\Data Science Internships\Iris Flowe
2 dataset
```

Out[46]:

	sepal_length	sepal_width	petal_length	petal_width	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
4	5.0	3.6	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

150 rows × 5 columns

Check the statistics of the dataset

In [5]:
```
1 dataset.describe()
```

Out[5]:

	sepal_length	sepal_width	petal_length	petal_width
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

Check the description of the dataset

In [6]:
```
1 dataset.info()
```

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

Checking the presence of null values

In [47]:
```
1 dataset.isnull().sum()
```

Out[47]:
```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

As we can see that there is no null value

Checking the presence of duplicate values and

if it is there delete it

In [48]:
```
1 dataset.duplicated().value_counts()
```

Out[48]:
```
False    147
True       3
dtype: int64
```

In [49]:
```
1 dataset.drop_duplicates(inplace = True)
2 dataset.duplicated().value_counts()
```

Out[49]:
```
False    147
dtype: int64
```

Check the total number of unique target values each

In [50]:
```
1 dataset["species"].value_counts()
```

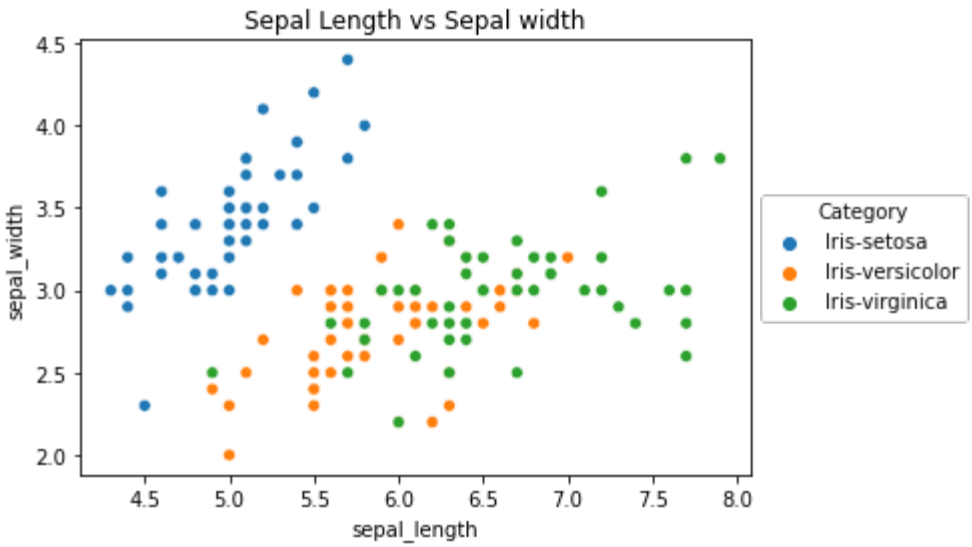
Out[50]:
```
Iris-versicolor    50
Iris-virginica     49
Iris-setosa        48
Name: species, dtype: int64
```

As we can see that the values of each target variable is almost equal and so it is evenly distributed

Visualise the relationship between "Sepal Length" vs "Sepal width" and "Petal Length" and "Petal Width"

In [71]:
```
1 sns.scatterplot(dataset["sepal_length"],dataset["sepal_width"],hue=dataset
2 plt.title("Sepal Length vs Sepal width")
3 plt.legend(loc = "center left",bbox_to_anchor=(1, 0.5),title ="Category")
4 plt.show()
```

```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarn
ing: Pass the following variables as keyword args: x, y. From version 0.12, t
he only valid positional argument will be `data`, and passing other arguments
without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



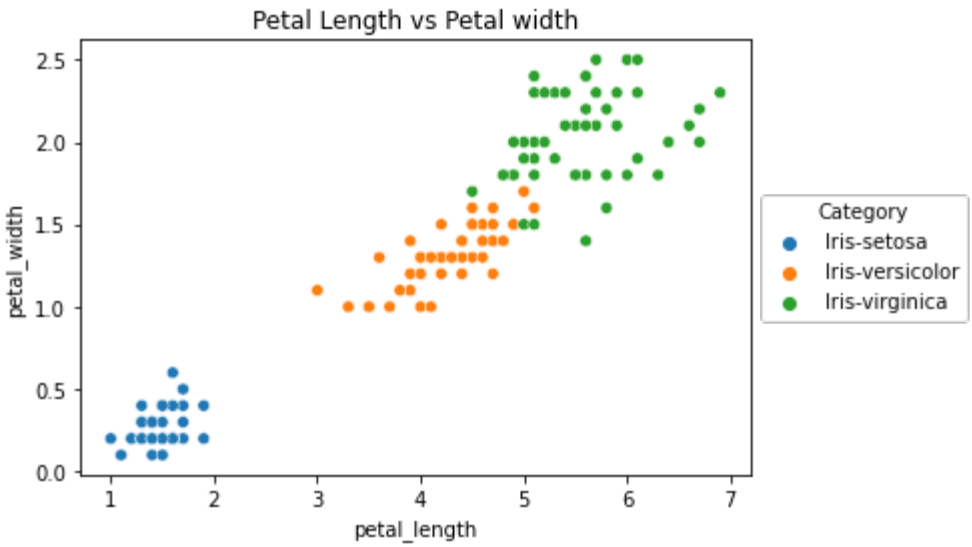
From the Graph we can see that

1. Species Virginica has larger length but smaller width.
2. Species setosa has smaller length but larger width.
3. Species versicolor is in between the Virginca and Versicolor.

So we can say that the correlation between them is negative.

In [72]:
```
sns.scatterplot(dataset["petal_length"],dataset["petal_width"],hue=dataset
plt.title("Petal Length vs Petal width")
plt.legend(loc = "center left",bbox_to_anchor=(1, 0.5),title ="Category")
plt.show()
```

```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarn
ing: Pass the following variables as keyword args: x, y. From version 0.12, t
he only valid positional argument will be `data`, and passing other arguments
without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



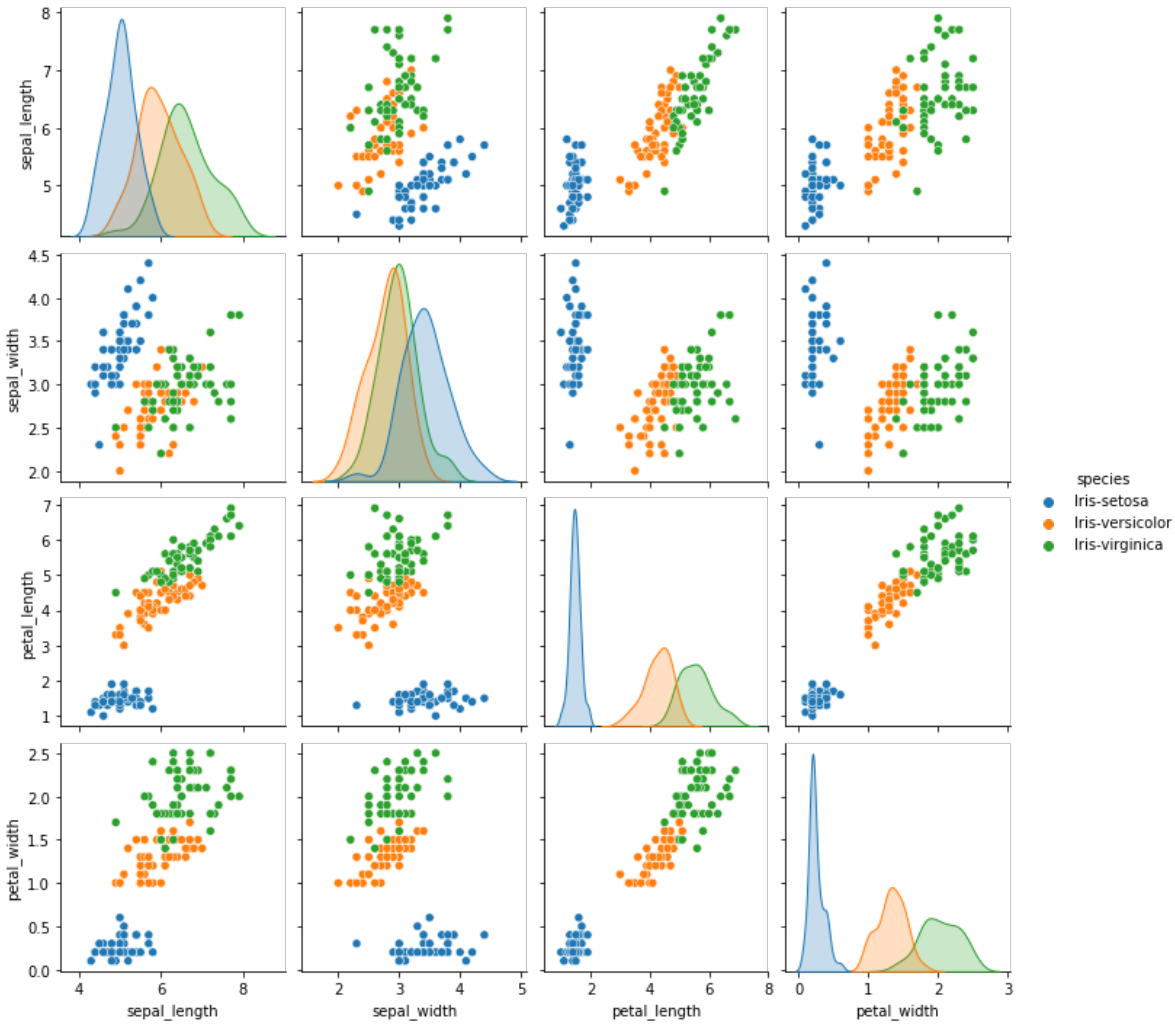
From the Graph we can see that

1. Species Virginica has larger length but Larger width.
2. Species setosa has smaller length but smaller width.
3. Species versicolor is in between the Virginca and Versicolor.

So we can say that the correlation between them is positive.

In [76]:
```
1 sns.pairplot(dataset,hue = "species")
```

Out[76]:
```
<seaborn.axisgrid.PairGrid at 0x207af2c4340>
```



As we can see from the graph that the species "Setosa" is well seperated from the other species

Draw the Heat map to check the correlation between target variable and independent variable

In [82]:
```
1 corr = dataset.corr()
2 sns.heatmap(corr, annot=True, cmap='Greens')
```

Out[82]: <AxesSubplot:>

As we can see that,

1. The relation the sepal length and sepal width is -ve. as mentioned in the above graph
2. The relation between the petal length and petal width is postive and is highly correlated 0.96

Split the dataset into dependent and independent dataset

In [87]:
```
1 x = dataset.drop(["species"],axis = 1)
2 y = dataset.loc[:,"species"]
```

In [94]:
```
1 from sklearn.model_selection import train_test_split
2 x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.3, rand
```

In [145]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
classifiers = [LogisticRegression(multi_class='multinomial',solver = 'lbfg
               DecisionTreeClassifier(),
               KNeighborsClassifier(n_neighbors=4),
              SVC()]
print("************************Training Dataset Accuracy***************
for classifier in classifiers:
    classifier = classifier
    classifier.fit(x_train,y_train)
    y_pred_train = classifier.predict(x_train)
    accu_train = accuracy_score(y_train,y_pred_train)
    print("Accuracy of {} = {}".format(classifier,accu_train))
print("\r")
print("************************Testing Dataset Accuracy****************
for classifier in classifiers:
    classifier = classifier
    classifier.fit(x_train,y_train)
    y_pred_test = classifier.predict(x_test)
    accu_test = accuracy_score(y_test,y_pred_test)
    print("Accuracy of {} = {}".format(classifier,accu_test))
```

```
************************Training Dataset Accuracy**************************
********
Accuracy of LogisticRegression(multi_class='multinomial') = 0.950980392156862
7
Accuracy of DecisionTreeClassifier() = 1.0
Accuracy of KNeighborsClassifier(n_neighbors=4) = 0.9607843137254902
Accuracy of SVC() = 0.9607843137254902

************************Testing Dataset Accuracy***************************
*******
Accuracy of LogisticRegression(multi_class='multinomial') = 1.0
Accuracy of DecisionTreeClassifier() = 1.0
Accuracy of KNeighborsClassifier(n_neighbors=4) = 0.9777777777777777
Accuracy of SVC() = 1.0

C:\Users\HP\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:81
4: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sciki
t-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
sion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
ession)
  n_iter_i = _check_optimize_result(
C:\Users\HP\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:81
4: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

```
Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

In [133]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,y_pred)
```

Out[133]:
```
array([[16,  0,  0],
       [ 0, 16,  0],
       [ 0,  0, 13]], dtype=int64)
```

In [112]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)
```

Out[112]:
```
1.0
```

In [120]:
```
a = 10
b = 20
print("{} is smaller than {}".format(a,b))
```

```
10 is smaller than 20
```

In [ ]:
```

```