Iris Flower:

Importing all module

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

Load Datasets

In [2]:
```
#from sklearn.datasets import load_iris
#iris=load_iris()
```

In [3]:
```
df=pd.read_csv('IRIS.csv')
```

In [4]:
```
df
```

Out[4]:

	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

In [5]:
```python
#iris.feature_names
df.species
```

Out[5]:
```
0          Iris-setosa
1          Iris-setosa
2          Iris-setosa
3          Iris-setosa
4          Iris-setosa
              ...
145     Iris-virginica
146     Iris-virginica
147     Iris-virginica
148     Iris-virginica
149     Iris-virginica
Name: species, Length: 150, dtype: object
```

In [6]:
```python
#df = pd.DataFrame(iris.data,columns=iris.feature_names)
```

In [7]:
```python
#To display first five rows
df.head()
```

Out[7]:

	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

In [8]:
```python
#To display last five rows
df.tail()
```

Out[8]:

	sepal_length	sepal_width	petal_length	petal_width	species
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

In [9]:
```python
#df['class']=iris.target
df.species.value_counts()
```

Out[9]:
```
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: species, dtype: int64
```

In [10]:
```
#to display datatypes
df.dtypes
```

Out[10]:
```
sepal_length    float64
sepal_width     float64
petal_length    float64
petal_width     float64
species          object
dtype: object
```

In [11]:
```
df.shape
```

Out[11]:
```
(150, 5)
```

In [12]:
```
df.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
```

In [13]:
```
#There are 150 observations with 4 features each (sepal length, sepal w
#There are no null values, so we don't have to worry about that.
#There are 50 observations of each species (setosa, versicolor, virgini
```

In [14]:
```
#to display the content of data
df.describe()
```

Out[14]:

	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

In [15]:
```
# to display no. of samples of each Sepal Length
df['sepal_length'].value_counts()
```

Out[15]:
```
5.0    10
5.1     9
6.3     9
5.7     8
6.7     8
5.8     7
5.5     7
6.4     7
4.9     6
5.4     6
6.1     6
6.0     6
5.6     6
4.8     5
6.5     5
6.2     4
7.7     4
6.9     4
4.6     4
5.2     4
5.9     3
4.4     3
7.2     3
6.8     3
6.6     2
4.7     2
7.6     1
7.4     1
7.3     1
7.0     1
7.1     1
5.3     1
4.3     1
4.5     1
7.9     1
Name: sepal_length, dtype: int64
```

In [16]:
```
# to check for null values
df.isnull().sum()
```

Out[16]:
```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

In [17]:
```
df.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
```

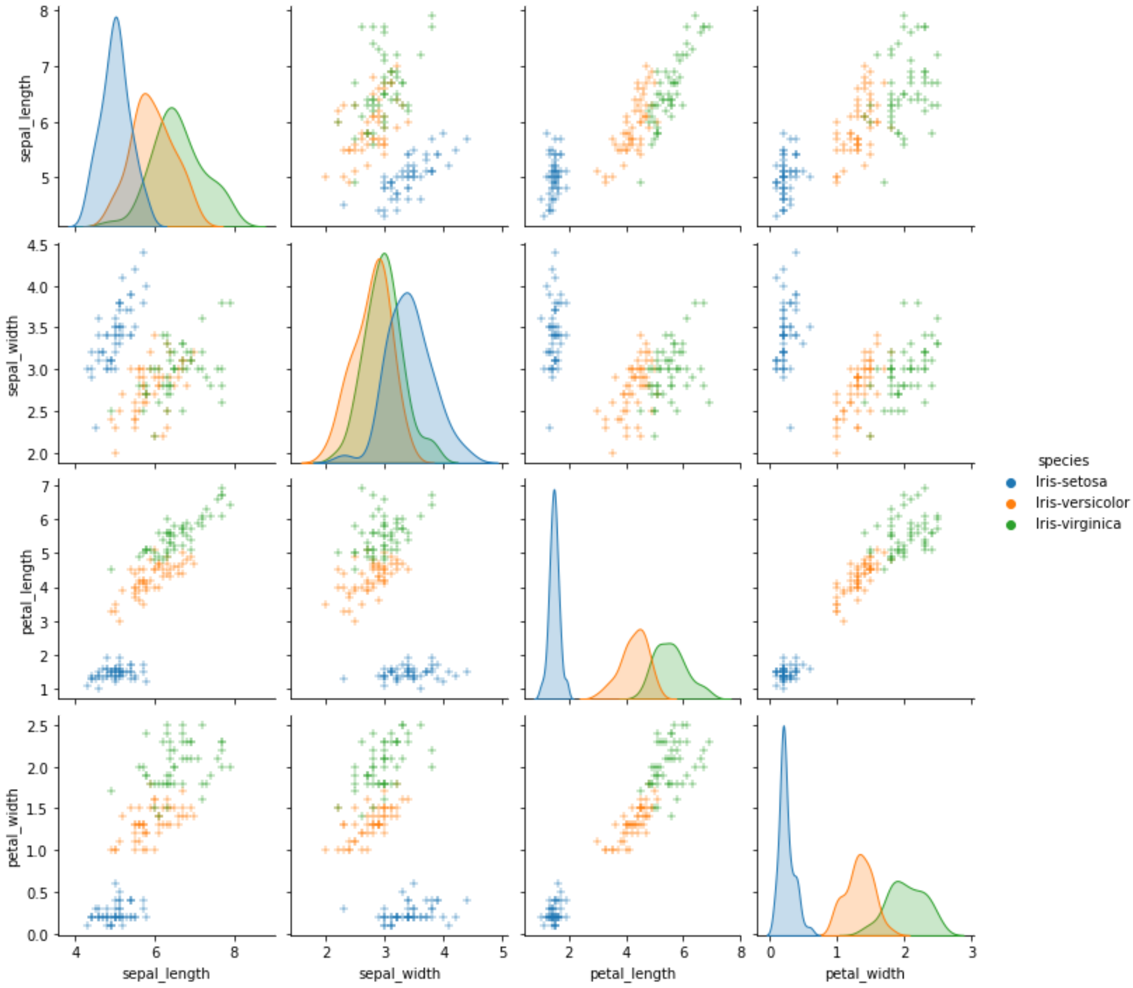
In [18]:
```
colname= df.select_dtypes('float64').columns
colname
```

Out[18]:
```
Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width'], dtyp
e='object')
```

Data Visulazation

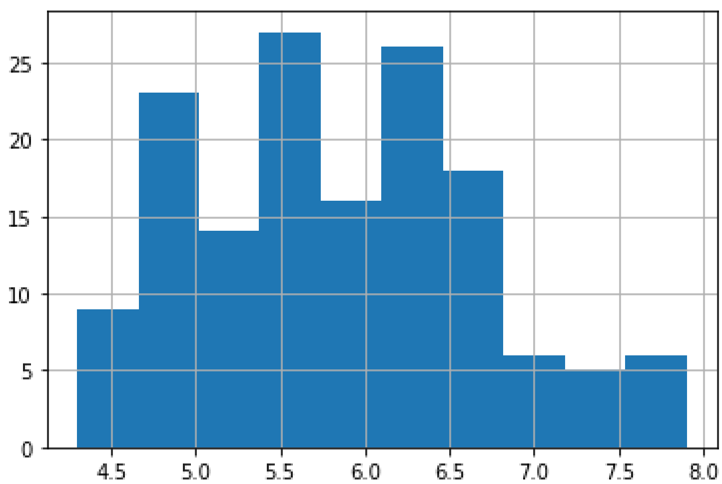
After plotting the features in a pair plot, it is obvious that the connection between pairs of traits in an iris-setosa (in pink) differs significantly from those in the other two class.The paired relationships of the other two class, iris-versicolor (brown) and iris-virginica (green), have some overlap.

In [19]:
```
g = sns.pairplot(df, hue='species', markers='+')
plt.show()
```



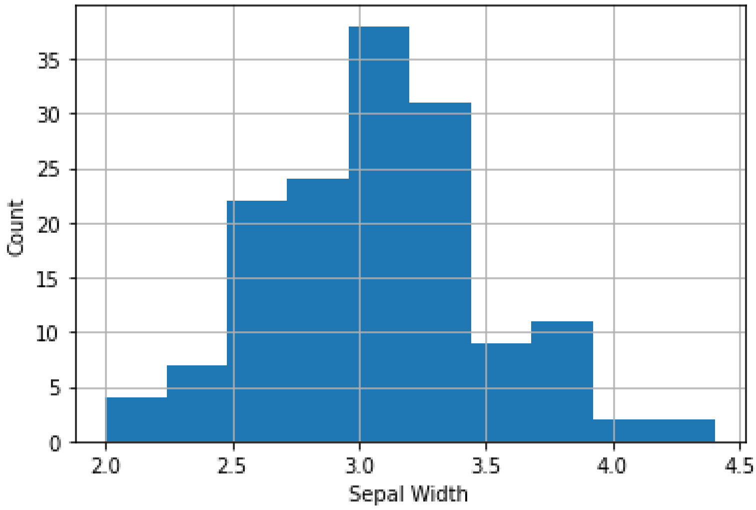
In [20]:
```
#Histogram
df["sepal_length"].hist()
```

Out[20]: <AxesSubplot:>



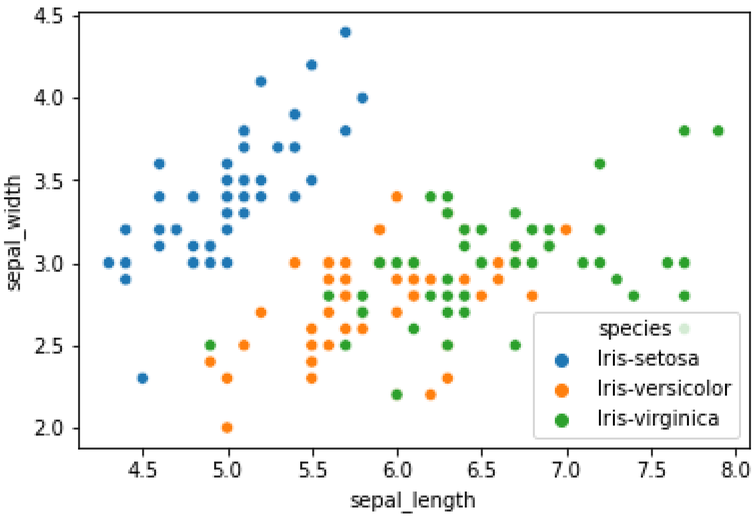
In [21]:
```python
a = df["sepal_width"].hist()
a.set_xlabel ("Sepal Width")
a.set_ylabel ("Count")
```

Out[21]: Text(0, 0.5, 'Count')



In [22]:
```python
# Plotting Scatterplot using Seaborn
sns.scatterplot(data=df, x='sepal_length', y='sepal_width', hue='specie
```

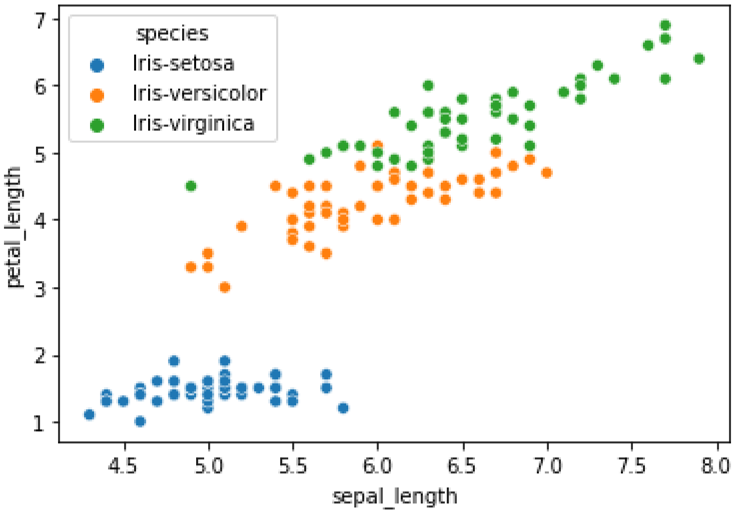
Out[22]: <AxesSubplot:xlabel='sepal_length', ylabel='sepal_width'>



In [23]:

```python
# Plotting Scatterplot using Seaborn
sns.scatterplot(data=df, x='sepal_length', y='petal_length', hue='speci
```

Out[23]: <AxesSubplot:xlabel='sepal_length', ylabel='petal_length'>



Matrix of correlation

A correlation matrix is a table that contains the coefficients of correlation between different features (attributes) in a dataset. The correlation between two variables is represented by each cell in the table. The value ranges from -1 to 1.

In [24]:

```python
# to delete Species Class from the dataset
new_df = df.drop(columns=['species'])
new_df.head()
```

Out[24]:

	sepal_length	sepal_width	petal_length	petal_width
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 [25]:

```python
# correlation matrix
corr_mat = new_df.corr()
corr_mat
```

Out[25]:

	sepal_length	sepal_width	petal_length	petal_width
sepal_length	1.000000	-0.109369	0.871754	0.817954
sepal_width	-0.109369	1.000000	-0.420516	-0.356544
petal_length	0.871754	-0.420516	1.000000	0.962757
petal_width	0.817954	-0.356544	0.962757	1.000000

In [26]:
```
# Heatmap
sns.heatmap(corr_mat, cmap='Reds', annot=True, linewidths=4, linecolor=
```

Out[26]: <AxesSubplot:>

In [27]:
```
#Label Encoder
```

In [28]:
```
#from scipy.stats import skew
```

In [29]:
```
#x=df.iloc[:,:-1]
#y=df.iloc[:,-1]
x = df.drop(columns='species')
y = df['species']
```

In [30]:
```
x
```

Out[30]:

	sepal_length	sepal_width	petal_length	petal_width
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
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns

In [31]:
```
y
```

Out[31]:
```
0         Iris-setosa
1         Iris-setosa
2         Iris-setosa
3         Iris-setosa
4         Iris-setosa
            ...
145    Iris-virginica
146    Iris-virginica
147    Iris-virginica
148    Iris-virginica
149    Iris-virginica
Name: species, Length: 150, dtype: object
```

In [32]:
```
#Split the data train and test
```

In [33]:
```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2,random_sta
```

In [34]:
```
#By building a model
def mymodel(model):

    model.fit(xtrain,ytrain)
    ypred= model.predict(xtest)

    train=model.score(xtrain,ytrain)
    test=model.score(xtest,ytest)

    print(f'Training Accuracy:- {train}\nTesting Accuracy:- {test}')

    print(classification_report(ytest,ypred))
    return model
```

Build Model

In [35]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
```

In [36]:
```
from sklearn.metrics import classification_report
```

In [37]:

```
1 knn= mymodel(KNeighborsClassifier())
```

```
Training Accuracy:- 0.95
Testing Accuracy:- 0.9666666666666667
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        11
Iris-versicolor       1.00      0.92      0.96        13
 Iris-virginica       0.86      1.00      0.92         6

       accuracy                           0.97        30
      macro avg       0.95      0.97      0.96        30
   weighted avg       0.97      0.97      0.97        30
```

In [38]:

```
1 logreg= mymodel(LogisticRegression())
```

```
Training Accuracy:- 0.9666666666666667
Testing Accuracy:- 1.0
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        11
Iris-versicolor       1.00      1.00      1.00        13
 Iris-virginica       1.00      1.00      1.00         6

       accuracy                           1.00        30
      macro avg       1.00      1.00      1.00        30
   weighted avg       1.00      1.00      1.00        30
```

In [39]:

```
1 svm= mymodel(SVC())
```

```
Training Accuracy:- 0.9583333333333334
Testing Accuracy:- 1.0
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        11
Iris-versicolor       1.00      1.00      1.00        13
 Iris-virginica       1.00      1.00      1.00         6

       accuracy                           1.00        30
      macro avg       1.00      1.00      1.00        30
   weighted avg       1.00      1.00      1.00        30
```

In [40]:

```
dt= mymodel(DecisionTreeClassifier())
```

```
Training Accuracy:- 1.0
Testing Accuracy:- 1.0
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        11
Iris-versicolor       1.00      1.00      1.00        13
 Iris-virginica       1.00      1.00      1.00         6

       accuracy                           1.00        30
      macro avg       1.00      1.00      1.00        30
   weighted avg       1.00      1.00      1.00        30
```