1) Iris Flowers Classification ML Project :

Steps to Classify Iris Flower:

1. Loading of the dataset

2. Analyzing and visualizing the dataset

3. Model training

4. Model Evaluation

5. Testing the model

In [1]:
```
# First we have to import some packages

import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
```

Step 1 – Load the data:

In [2]:
```
df = pd.read_csv("C:\\Users\\User\\Downloads\\Iris.csv")
df.head()
```

Out[2]:

	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

Step 2 – Analyze and visualize the dataset:

In [3]:
```
# Some basic statistical analysis about the data
df.describe()
```

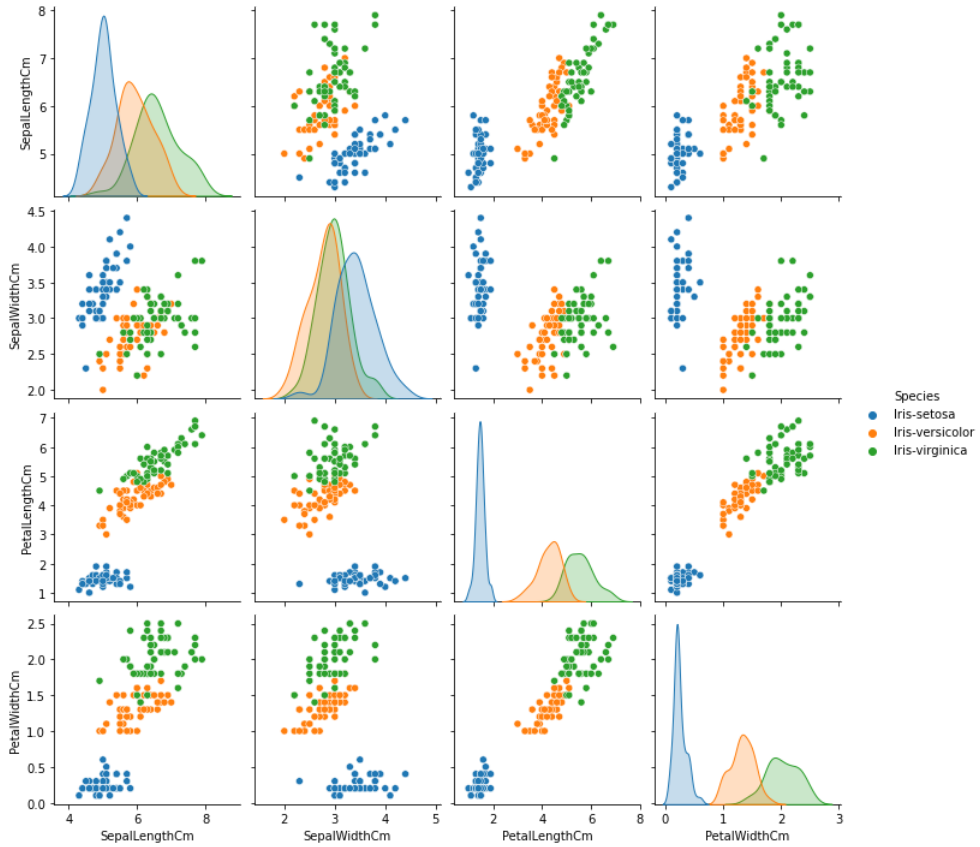
Out[3]:

	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 [4]:
```
#Drop unwanted columns
df=df.drop(columns="Id")
```

In [5]:
```
# Visualize the whole dataset
sns.pairplot(df, hue='Species')
```

Out[5]:
```
<seaborn.axisgrid.PairGrid at 0x18f79d697f0>
```



From this visualization, we can tell that iris-setosa is well separated from the other two flowers.

And iris virginica is the longest flower and iris setosa is the shortest.

Step 3 – Model training:

In [6]:
```
# Split the data to train and test dataset.
train, test = train_test_split(df, test_size = 0.3) # in this our main data is split into train and test

# the attribute test_size=0.3 splits the data into 70% and 30% ratio. train=70% and test=30%

print(train.shape)
print(test.shape)
```

```
(105, 5)
(45, 5)
```

In [7]:
```
train_X = train[['SepalLengthCm','SepalWidthCm','PetalLengthCm','PetalWidthCm']]# taking the training data features
train_y=train.Species        # output of our training data
test_X= test[['SepalLengthCm','SepalWidthCm','PetalLengthCm','PetalWidthCm']] # taking test data features
test_y =test.Species   #output value of test data
```

In [8]:
```
train_X.head(2)
```

Out[8]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
138	6.0	3.0	4.8	1.8
134	6.1	2.6	5.6	1.4

In [9]:
```
test_X.head(2)
```

Out[9]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
116	6.5	3.0	5.5	1.8
122	7.7	2.8	6.7	2.0

In [10]:
```
train_y.head()  ##output of the training data
```

Out[10]:
```
138     Iris-virginica
134     Iris-virginica
93     Iris-versicolor
15         Iris-setosa
41         Iris-setosa
Name: Species, dtype: object
```

In [11]:
```
# Support vector machine algorithm
from sklearn.svm import SVC
svn = SVC()
svn.fit(train_X, train_y)
```

Out[11]:
```
SVC()
```

Step 4 – Model Evaluation:

In [12]:
```
# Predict from the test dataset
predictions = svn.predict(test_X)
# Calculate the accuracy

accuracy_score(test_y, predictions)
```

Out[12]:
```
1.0
```

In [13]:
```
# A detailed classification report

print(classification_report(test_y, predictions))
```

```
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        11
Iris-versicolor       1.00      1.00      1.00        18
 Iris-virginica       1.00      1.00      1.00        16

       accuracy                           1.00        45
      macro avg       1.00      1.00      1.00        45
   weighted avg       1.00      1.00      1.00        45
```

Step 5 – Testing the model:

In [14]:
```
X_new = np.array([[3, 2, 1, 0.2], [  4.9, 2.2, 3.8, 1.1 ], [  5.3, 2.5, 4.6, 1.9 ]])
#Prediction of the species from the input vector
prediction = svn.predict(X_new)
print("Prediction of Species: {}".format(prediction))
```

```
Prediction of Species: ['Iris-setosa' 'Iris-versicolor' 'Iris-virginica']
```

In [15]:
```
# Save the model
import pickle
with open('SVM.pickle', 'wb') as f:
    pickle.dump(svn, f)
# Load the model
with open('SVM.pickle', 'rb') as f:
    model = pickle.load(f)
model.predict(X_new)
```

Out[15]:
```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

In [ ]: