Oasis Infobytes - Internship

Task 1 - IRIS Flower Classification Using Machine Learning Problem Statement - IRIS flower has three species; setosa, versicolor and virginica which differs according to their measurements. Now assume that you have the measurement of the IRIS flower according to their species and here your task is to train a machine learning model that can learn from the measurements of the IRIS species and classify them.

In [1]:
```
#import library
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [3]:
```
#upload dataset
df =pd.read_csv('C:/Users/cws/Downloads/Iris.csv')
df.head()
```

Out[3]:

	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 [4]:
```
df.shape
```

Out[4]:
```
(150, 6)
```

In [9]:
```
#delete unnecessary coloumn
df = df.drop(columns=['Id'])
df.head()
```

Out[9]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	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 [10]:
```
#Dataset discriptive statistics
df.describe()
```

Out[10]:

	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

In [11]:
```
#basic info of dataset
df.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
```

In [13]:
```
#number of sample each class
df['Species'].value_counts()
```

Out[13]:
```
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: Species, dtype: int64
```

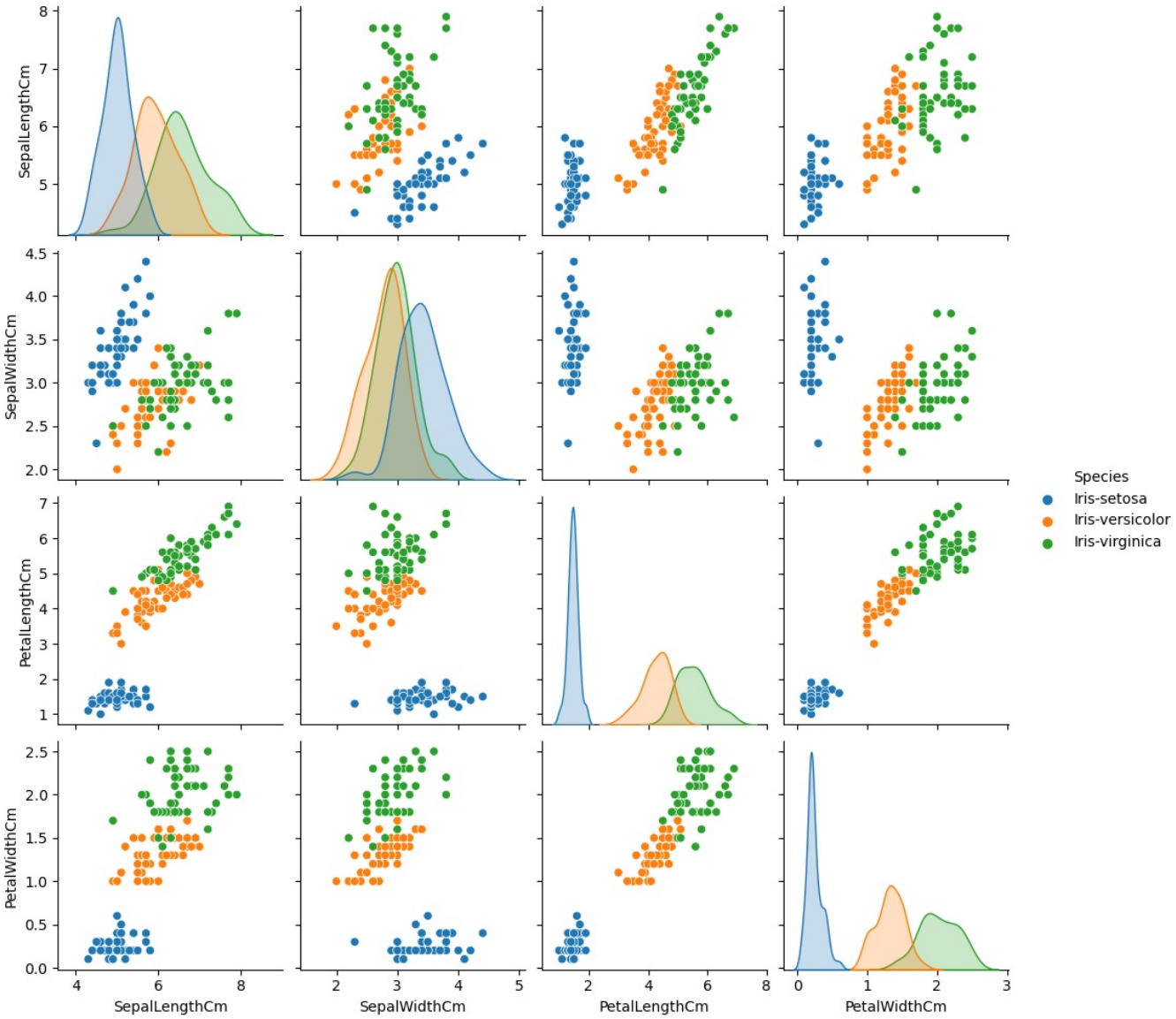
In [14]:
```
#Null value in dataset
df.isna().sum()
```

Out[14]:
```
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

Exploratry Data analysis

In [15]:
```
#visulization of dataset
sns.pairplot(df, hue='Species')
```

Out[15]:
```
<seaborn.axisgrid.PairGrid at 0x21ea5cdcd90>
```



Correlation Matrics

In [21]:
```
df.corr()
```

```
C:\Users\cws\AppData\Local\Temp\ipykernel_12576\1134722465.py:1: FutureWarning: The default value of numeric_on
ly in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or
specify the value of numeric_only to silence this warning.
  df.corr()
```

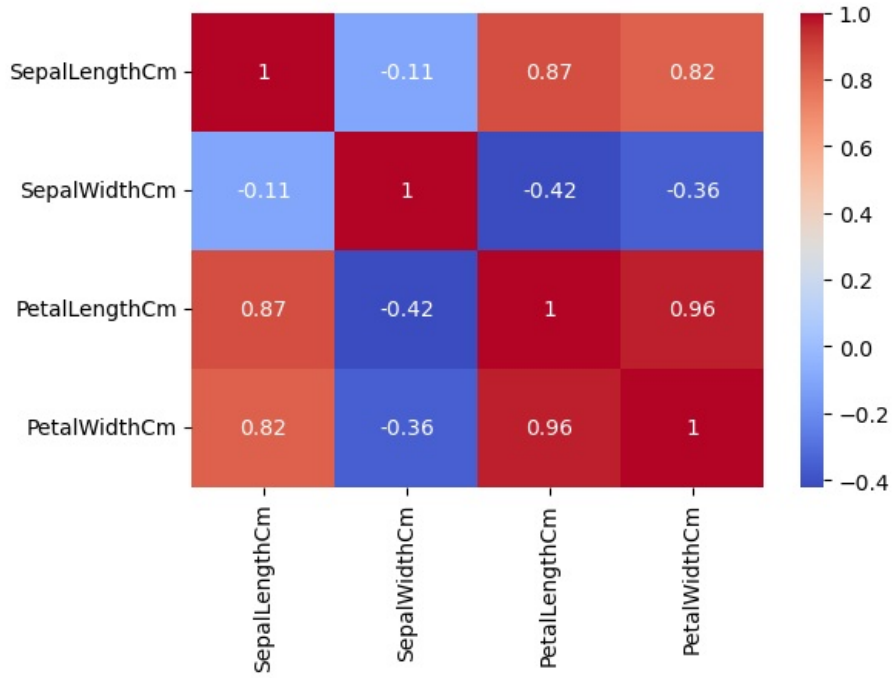
Out[21]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
SepalLengthCm	1.000000	-0.109369	0.871754	0.817954
SepalWidthCm	-0.109369	1.000000	-0.420516	-0.356544
PetalLengthCm	0.871754	-0.420516	1.000000	0.962757
PetalWidthCm	0.817954	-0.356544	0.962757	1.000000

In [22]:
```
corr = df.corr()
fig, ax = plt.subplots(figsize=(6,4))
sns.heatmap(corr, annot=True, ax=ax, cmap = 'coolwarm')
```

```
C:\Users\cws\AppData\Local\Temp\ipykernel_12576\457449378.py:1: FutureWarning: The default value of numeric_onl
y in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or
specify the value of numeric_only to silence this warning.
  corr = df.corr()
```

Out[22]:
```
<Axes: >
```



Label Encoder

In [23]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

In [24]:
```
df['Species'] = le.fit_transform(df['Species'])
df.head()
```

Out[24]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

Model Training

In [25]:
```
from sklearn.model_selection import train_test_split
#train - 70
#test - 30
X = df.drop(columns=['Species','Species'])
Y = df['Species']
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.30)
```

Model 1 : Logistic Regression

In [44]:
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
```

```
model.fit(x_train, y_train)
```

```
C:\Users\cws\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning: lbfgs faile
d 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
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[44]:
```
LogisticRegression()
```

In [45]:
```
#print metic to get performance
print("Accuracy: ", model.score(x_test, y_test)*100)
```

```
Accuracy:  100.0
```

Model 2: KNN (K Nearest Neighbors)

In [29]:
```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier()
```

In [30]:
```
model.fit(x_train, y_train)
```

Out[30]:
```
KNeighborsClassifier()
```

In [31]:
```
#print metic to get performance
print("Accuracy: ", model.score(x_test, y_test)*100)
```

```
Accuracy:  100.0
```

Model 3: Decision Tree

In [32]:
```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
```

In [33]:
```
model.fit(x_train, y_train)
```

Out[33]:
```
DecisionTreeClassifier()
```

In [34]:
```
#print metic to get performance
print("Accuracy: ", model.score(x_test, y_test)*100)
```

```
Accuracy:  100.0
```

Model 4: Support Vector Machine Algorithm

In [36]:
```
from sklearn.svm import SVC
model_svc = SVC()
model_svc.fit(x_train, y_train)
```

Out[36]:
```
SVC()
```

In [39]:
```
prediction1 = model_svc.predict(x_test)

#print metic to get performance
print("Accuracy: ", model.score(x_test, y_test)*100)
```

```
Accuracy:  100.0
```

Thanks