Iris Flower classification

May 15, 2023

1 Dataset Information

The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other.

Attribute Information:

sepal length in cm

sepal width in cm

petal length in cm

petal width in cm

class:– Iris Setosa – Iris Versicolour – Iris Virginica

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

2 Loading the dataset

[2]:
```
data= pd.read_csv("C:/Users/MyPc/Downloads/archive/Iris.csv")
data.head(4)
```

[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

[3]:
```
# deleting a column
data = data.drop(columns = ['Id'])
data.head()
```

[3]:
```
   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
```

[4]:
```python
# to display statistics about data
data.describe()
```

[4]:
```
       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
```

[5]:
```python
# basic info about datatype
data.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
```

[6]:
```python
# to display no. of samples on each class
data['Species'].value_counts()
```

[6]:
```
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: Species, dtype: int64
```

3 Preprocessing the dataset

[7]:
```python
# check for null values
data.isnull().sum()
```

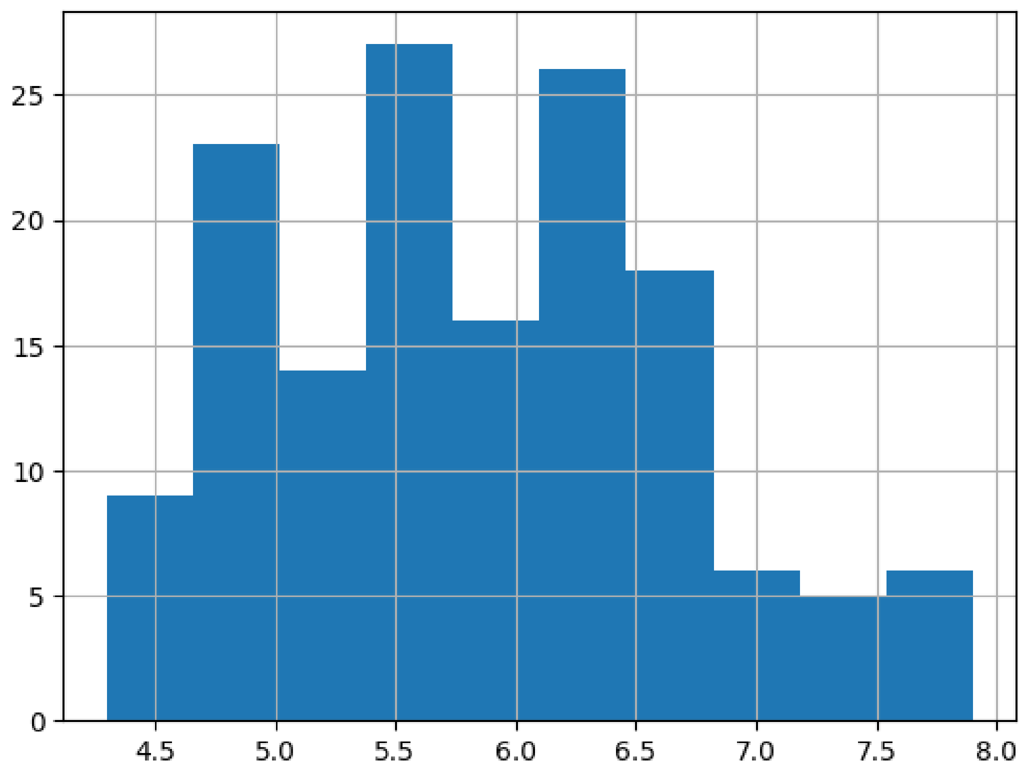
[7]:
```
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

->There are no null values

4 Exploratory Data Analysis

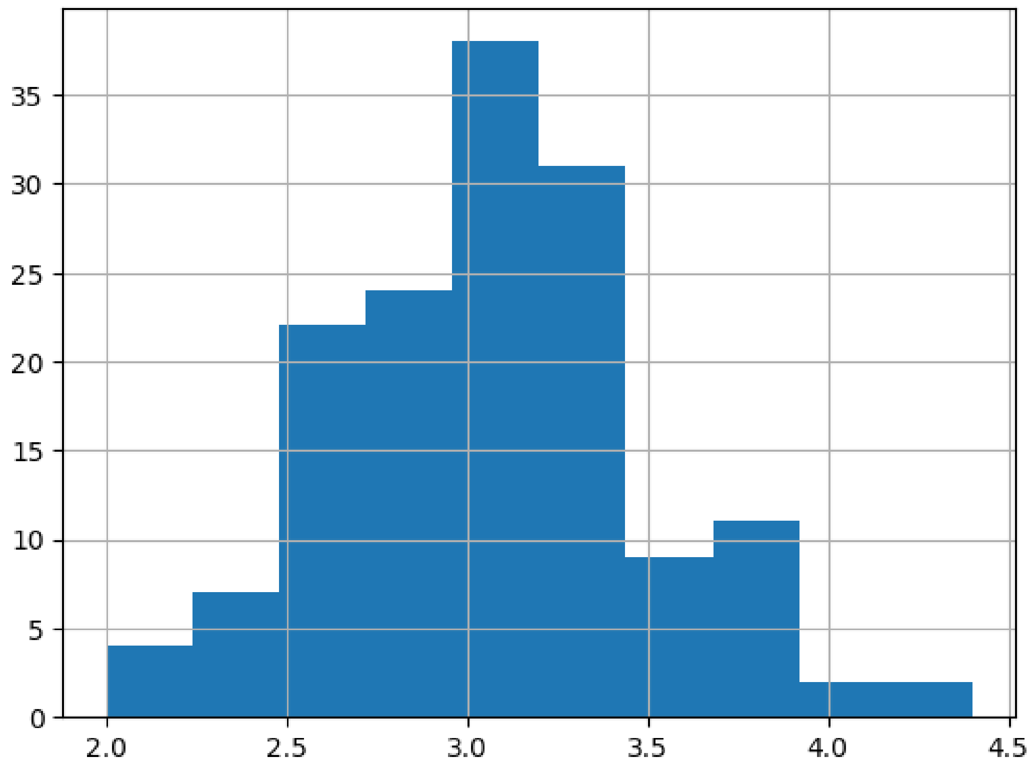
[8]:
```python
data['SepalLengthCm'].hist()
```

[8]: <AxesSubplot:>



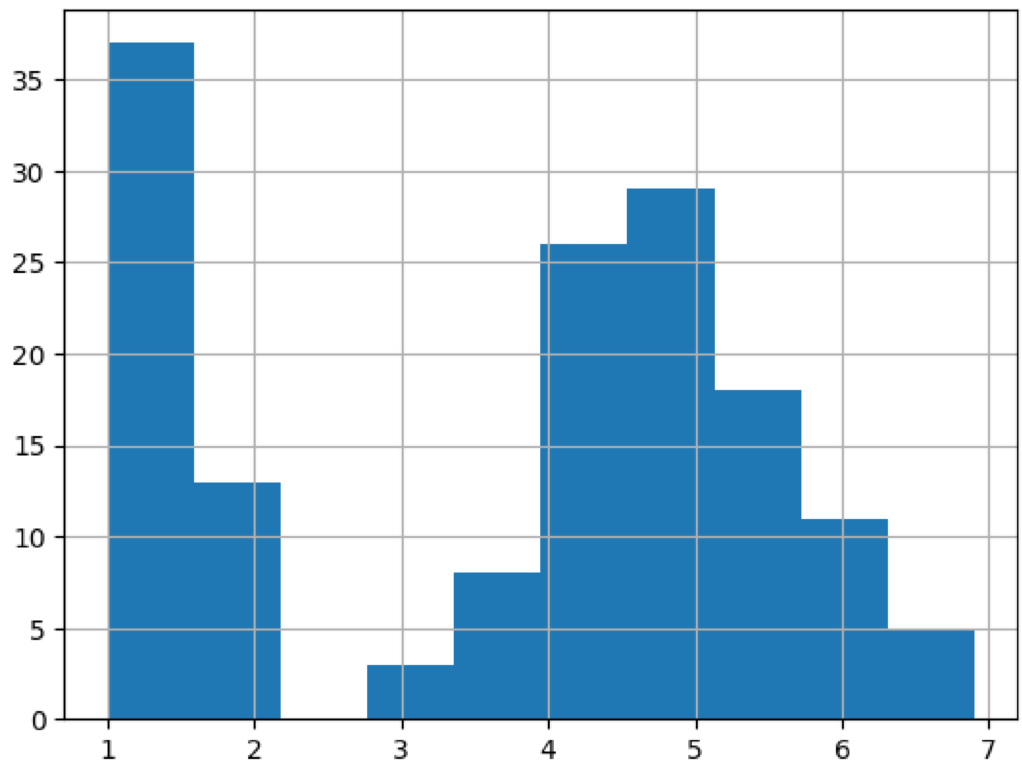
[9]:
```python
data['SepalWidthCm'].hist()
```

[9]: <AxesSubplot:>



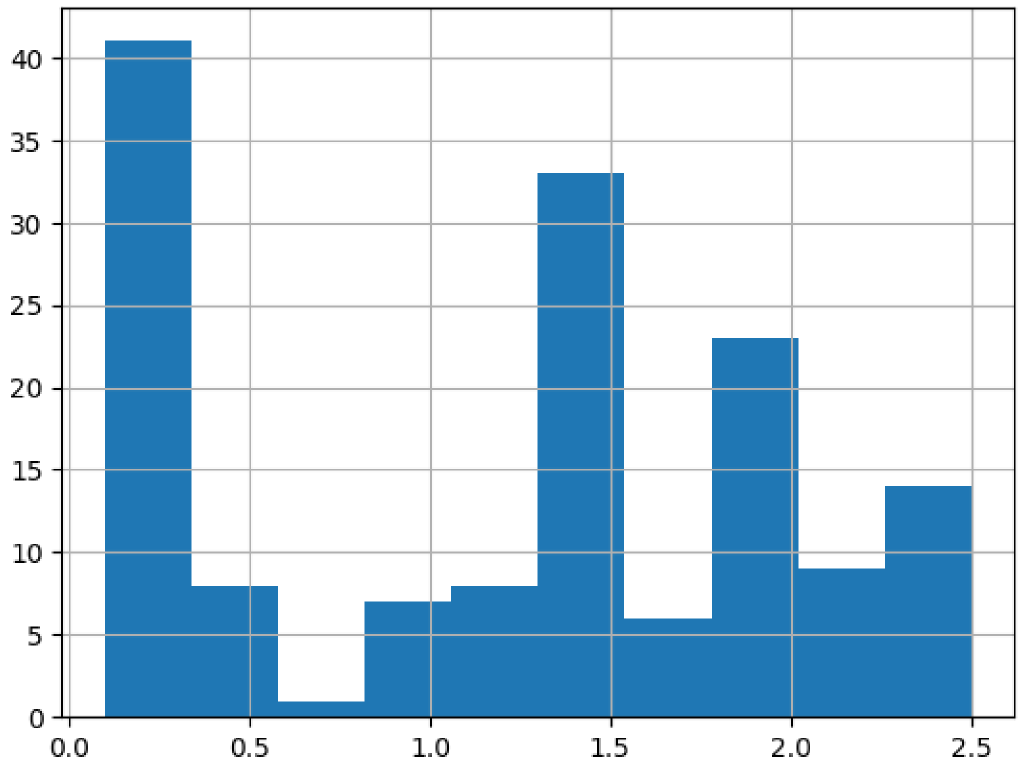

```
[10]: data['PetalLengthCm'].hist()
```

[10]: <AxesSubplot:>




```
[11]: data['PetalWidthCm'].hist()
```

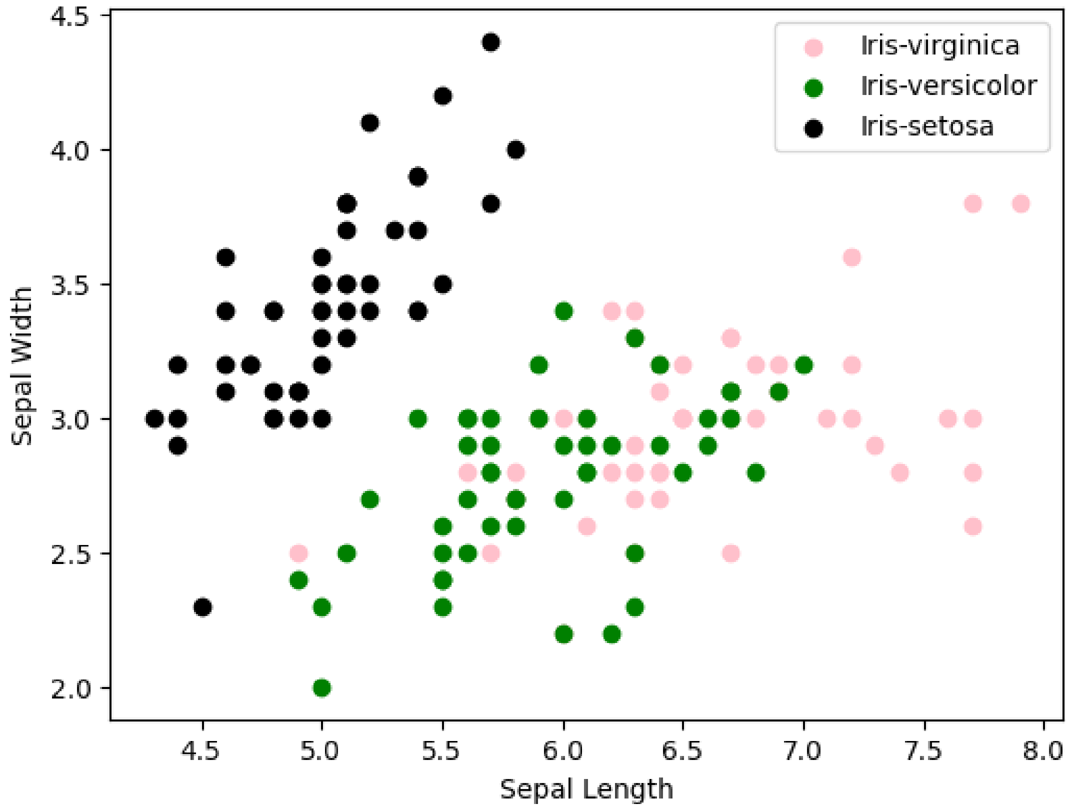
```
[11]: <AxesSubplot:>
```



[12]:
```
# scatterplot
colors = ['pink', 'green', 'black']
species = ['Iris-virginica','Iris-versicolor','Iris-setosa']
```

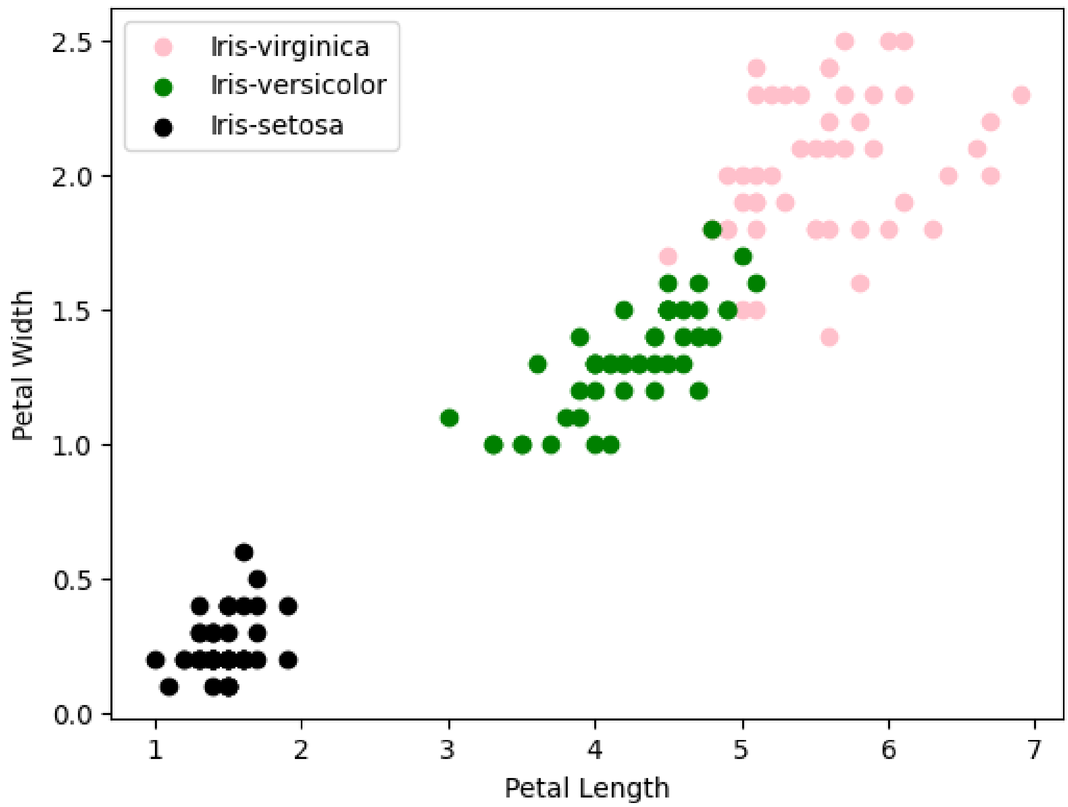
[13]:
```
for i in range(3):
    x = data[data['Species'] == species[i]]
    plt.scatter(x['SepalLengthCm'], x['SepalWidthCm'], c = colors[i],␣
 ↪label=species[i])
plt.xlabel("Sepal Length")
plt.ylabel("Sepal Width")
plt.legend()
```

[13]: <matplotlib.legend.Legend at 0x237cf99a770>



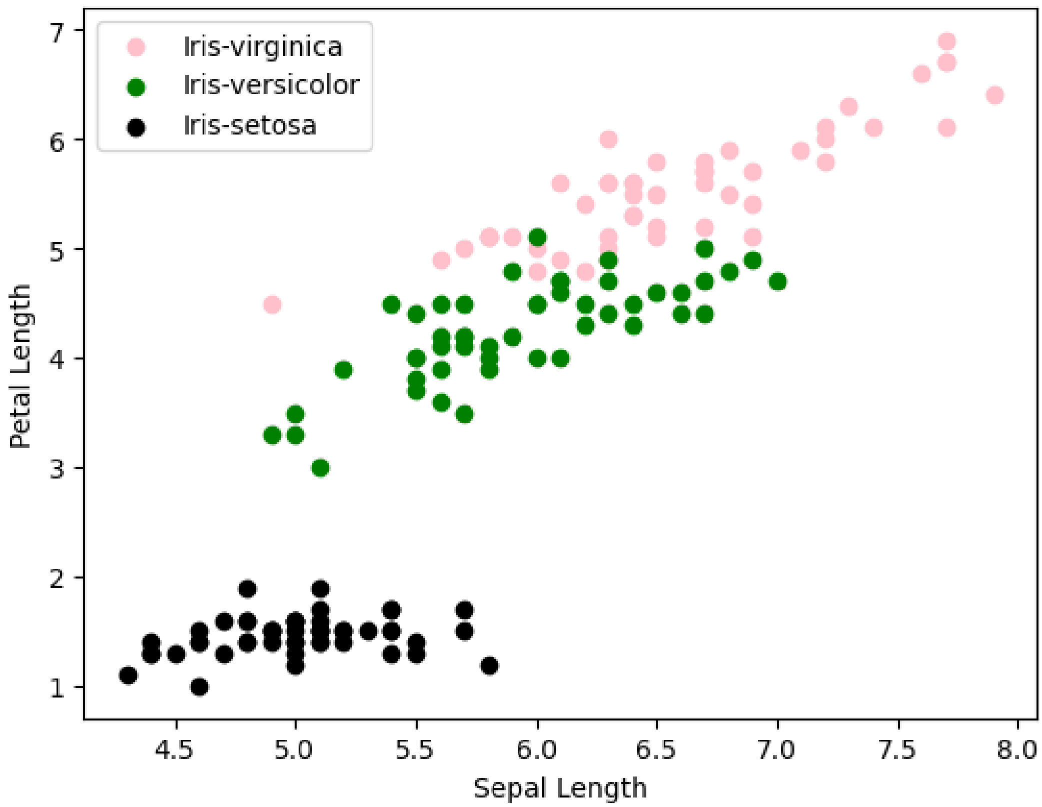
[14]:
```
for i in range(3):
    x = data[data['Species'] == species[i]]
    plt.scatter(x['PetalLengthCm'], x['PetalWidthCm'], c = colors[i],␣
 ↪label=species[i])
plt.xlabel("Petal Length")
plt.ylabel("Petal Width")
plt.legend()
```

[14]: <matplotlib.legend.Legend at 0x237cf608340>



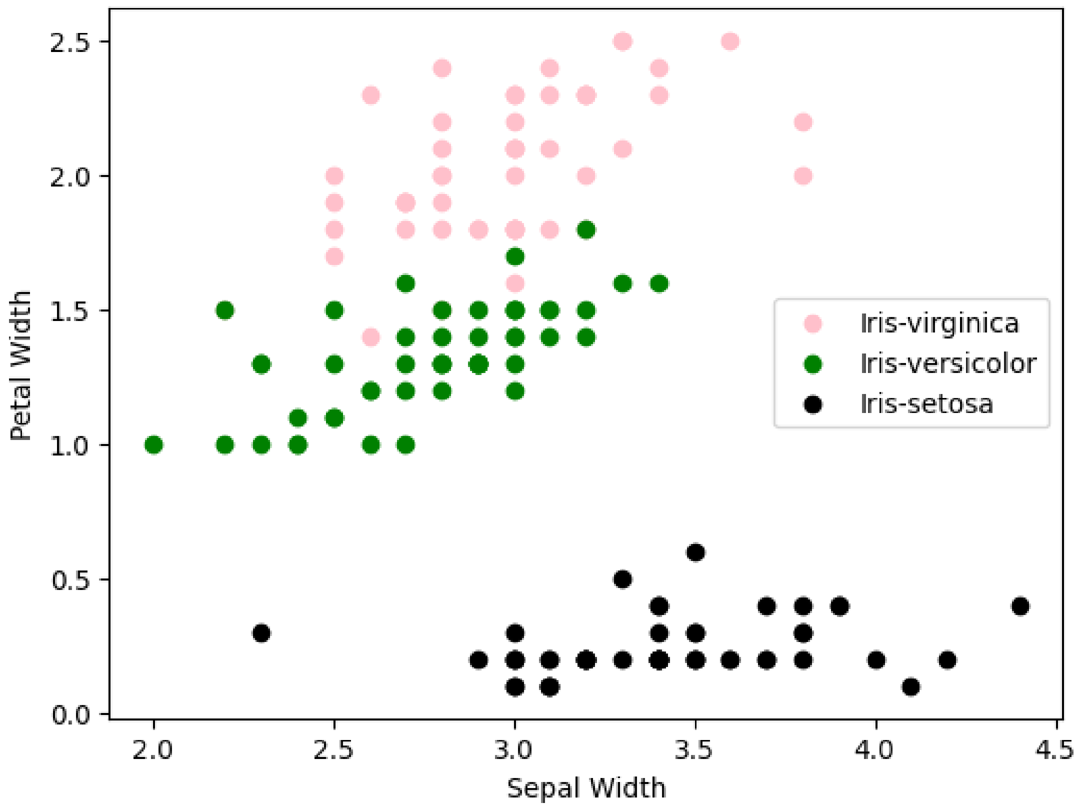
[15]:
```python
for i in range(3):
    x = data[data['Species'] == species[i]]
    plt.scatter(x['SepalLengthCm'], x['PetalLengthCm'], c = colors[i],␣
 ↪label=species[i])
plt.xlabel("Sepal Length")
plt.ylabel("Petal Length")
plt.legend()
```

[15]: <matplotlib.legend.Legend at 0x237cfa21a80>



[16]:
```
for i in range(3):
    x = data[data['Species'] == species[i]]
    plt.scatter(x['SepalWidthCm'], x['PetalWidthCm'], c = colors[i],␣
 ↪label=species[i])
plt.xlabel("Sepal Width")
plt.ylabel("Petal Width")
plt.legend()
```

[16]: <matplotlib.legend.Legend at 0x237cf670250>



5 Coorelation Matrix

A correlation matrix is a table showing correlation coefficients between variables. Each cell in the table shows the correlation between two variables. The value is in the range of -1 to 1. If two varibles have high correlation, we can neglect one variable from those two.

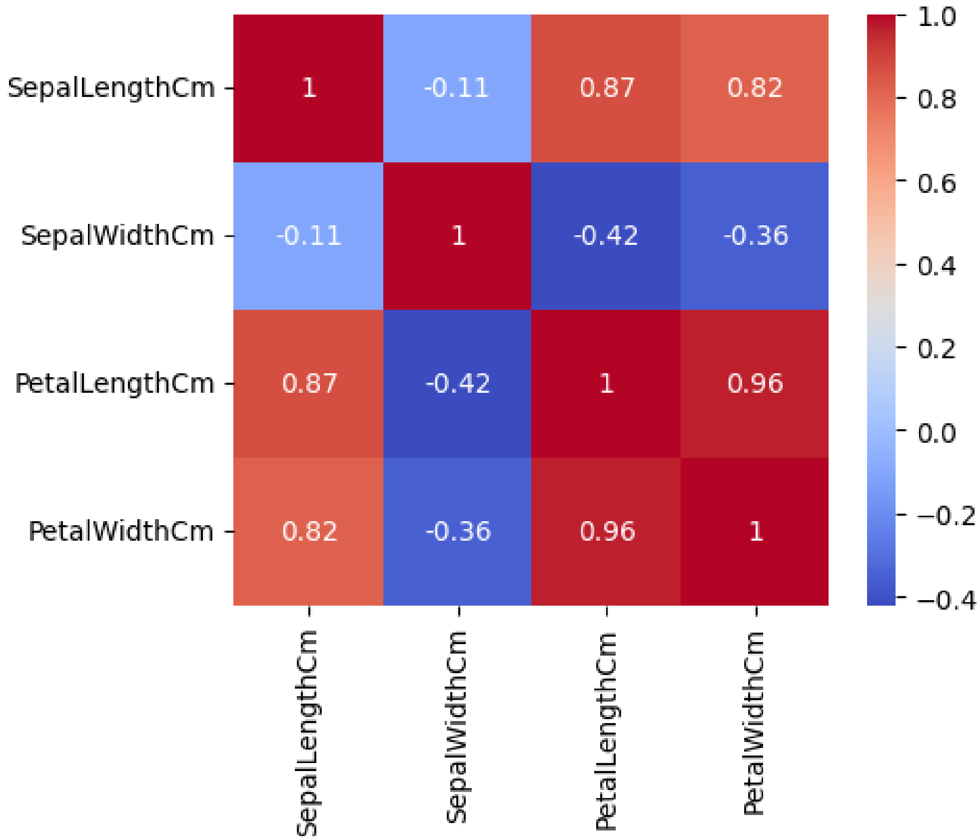
[17]:
```
data.corr()
```

[17]:

	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

[18]:
```
corr = data.corr()
fig, ax = plt.subplots(figsize=(5,4))
sns.heatmap(corr, annot=True, ax=ax, cmap = 'coolwarm')
```

[18]: <AxesSubplot:>



6 Label Encoder

In machine learning, we usually deal with datasets which contains multiple labels in one or more than one columns. These labels can be in the form of words or numbers. Label Encoding refers to converting the labels into numeric form so as to convert it into the machine-readable form.

[19]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

[20]:
```
data['Species'] = le.fit_transform(data['Species'])
data.head()
```

[20]:

	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

7 Model Training

[21]:
```
from sklearn.model_selection import train_test_split
# train - 70
# test - 30
X = data.drop(columns=['Species'])
Y = data['Species']
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.30)
```

[22]:
```
# logistic regression
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
```

[23]:
```
# model training
model.fit(x_train, y_train)
```

[23]:
```
LogisticRegression()
```

[24]:
```
# print metric to get performance
print("Accuracy: ",model.score(x_test, y_test) * 100)
```

```
Accuracy:  93.33333333333333
```

[25]:
```
# knn - k-nearest neighbours
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier()
```

[26]:
```
# knn - k-nearest neighbours
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier()
```

[27]:
```
# decision tree
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
```

[28]:
```
model.fit(x_train, y_train)
```

[28]:
```
DecisionTreeClassifier()
```

[29]:
```
# print metric to get performance
print("Accuracy: ",model.score(x_test, y_test) * 100)
```

```
Accuracy:  91.11111111111111
```

[ ]:
```

```