iris-flower-classification-oasis

March 4, 2024

1 IRIS FLOWER CLASSIFICATION (OASIS)

Problem Statement:

Iris flower has three species; setosa, versicolor, and virginica, which differs according to their measurements. Now assume that you have the measurements of the iris flowers 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.

[1]:
```
# Import Necessary Libraries
```

[2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier

import warnings
warnings.filterwarnings('ignore')
```

[3]:
```
# Loading Dataset
```

[4]:
```
Data = pd.read_csv('Iris.csv')
```

[5]:
```
# Exploratory Data Analysis
```

[6]:
```
Data.head()
```

[6]:
```
   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
```

[7]:
```
Data.tail()
```

[7]:
```
      Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  \
145  146            6.7           3.0            5.2           2.3
146  147            6.3           2.5            5.0           1.9
147  148            6.5           3.0            5.2           2.0
148  149            6.2           3.4            5.4           2.3
149  150            5.9           3.0            5.1           1.8

            Species
145  Iris-virginica
146  Iris-virginica
147  Iris-virginica
148  Iris-virginica
149  Iris-virginica
```

[8]:
```
Data.columns
```

[8]:
```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

[9]:
```
Data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

[10]:
```
Data.describe()
```

[10]:
```
               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
```

```
[11]: Data.isnull().sum()
```

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

```
[12]: Data.isnull().sum().sum()
```

```
[12]: 0
```

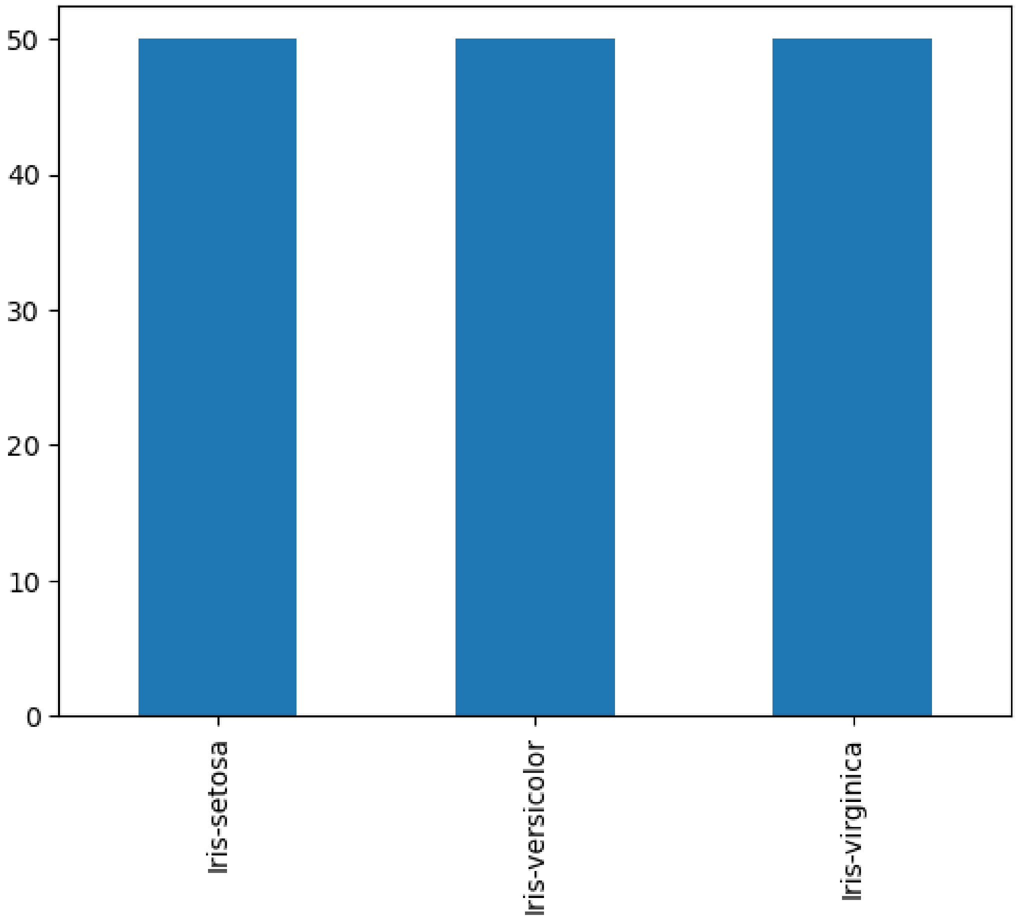
```
[13]: Data[Data.duplicated()]
```

```
[13]: Empty DataFrame
      Columns: [Id, SepalLengthCm, SepalWidthCm, PetalLengthCm, PetalWidthCm, Species]
      Index: []
```

```
[14]: # Data Visualization
```

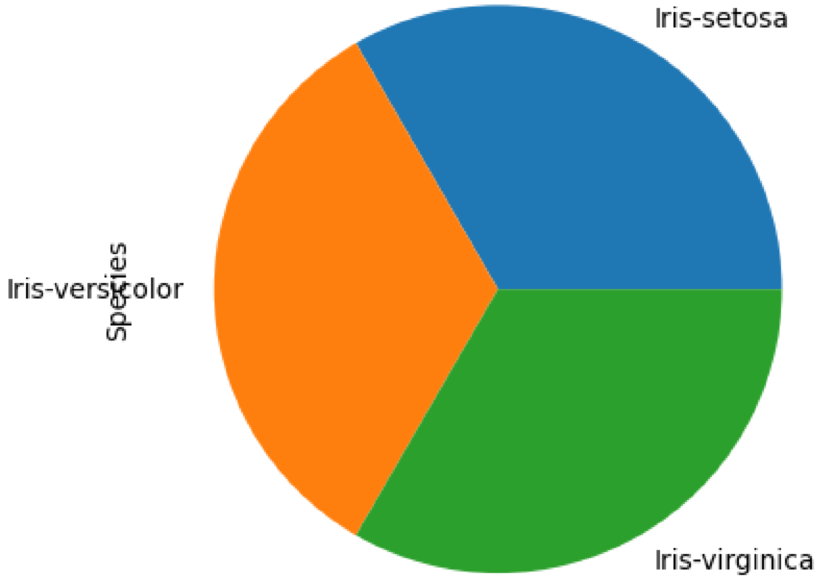
```
[15]: Data['Species'].value_counts().plot(kind='bar')
```

```
[15]: <Axes: >
```



```
[16]: Data['Species'].value_counts().plot.pie()
```

```
[16]: <Axes: ylabel='Species'>
```

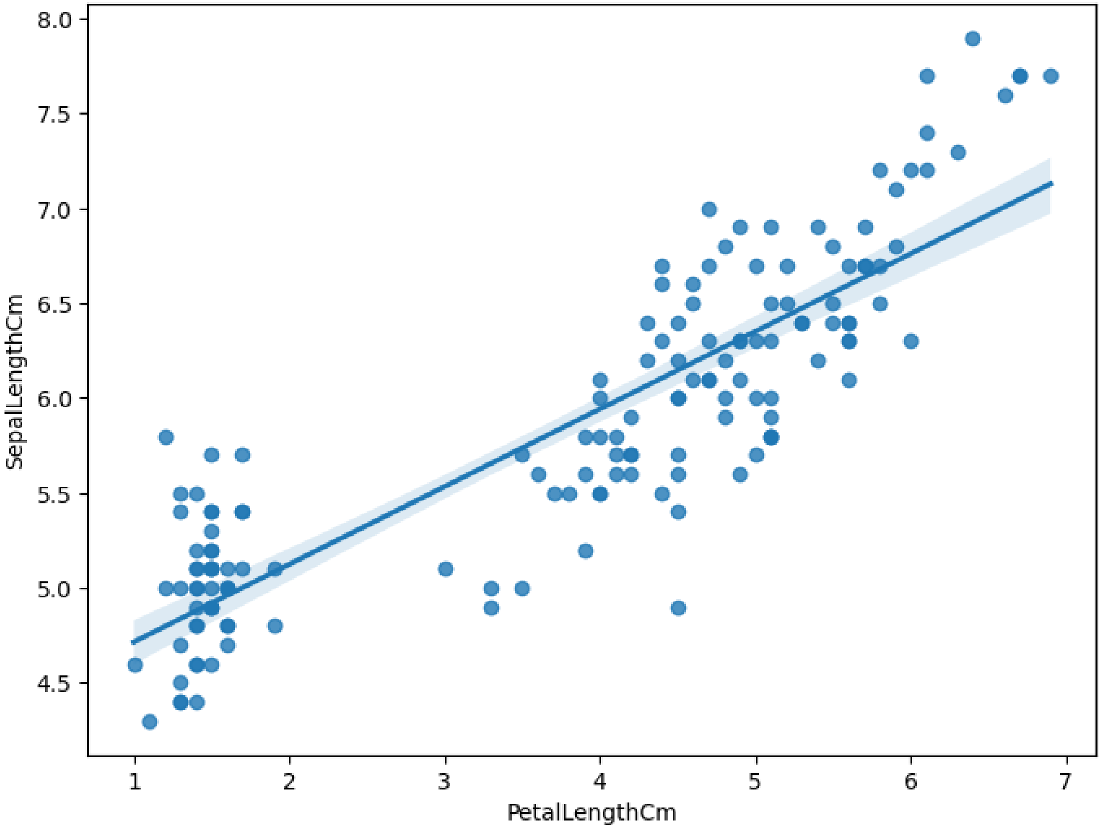


[17]:

```
plt.figure(figsize=(8,6))
sns.regplot(x='PetalLengthCm',y='SepalLengthCm',data=Data)
```

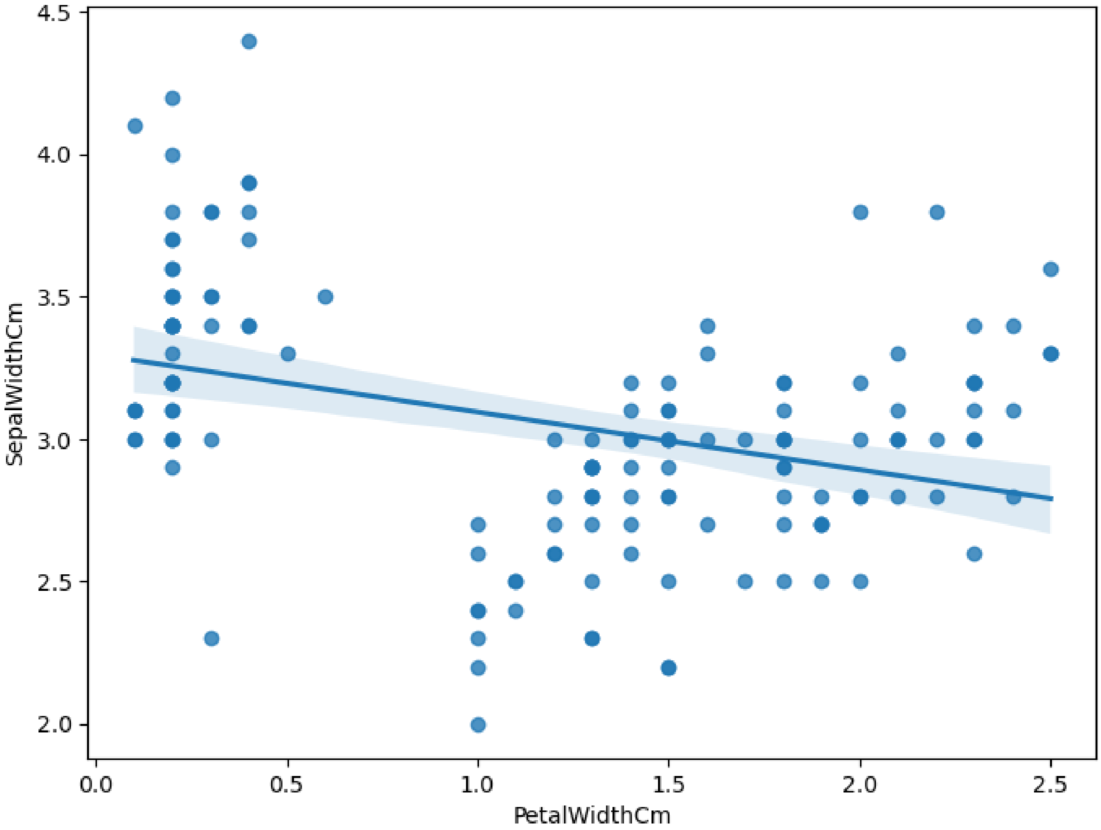
[17]:

```
<Axes: xlabel='PetalLengthCm', ylabel='SepalLengthCm'>
```



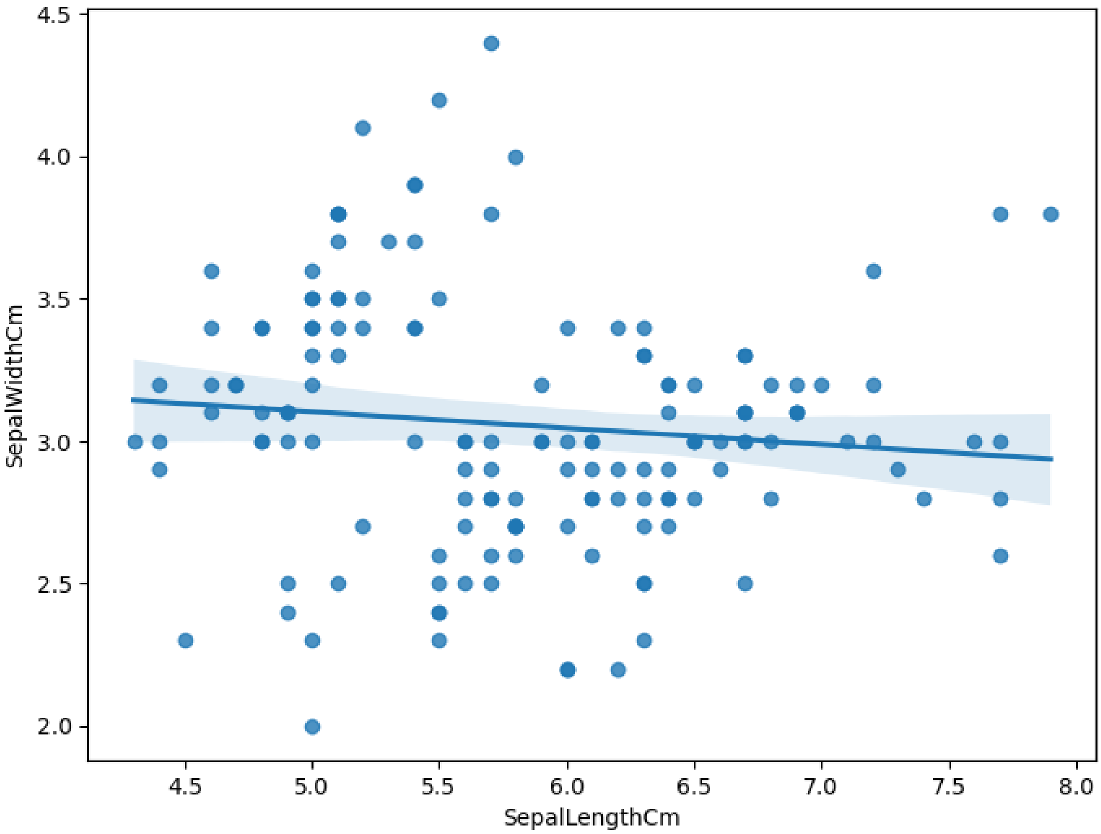
```
[18]: plt.figure(figsize=(8,6))
      sns.regplot(x='PetalWidthCm',y='SepalWidthCm',data=Data)
```

```
[18]: <Axes: xlabel='PetalWidthCm', ylabel='SepalWidthCm'>
```



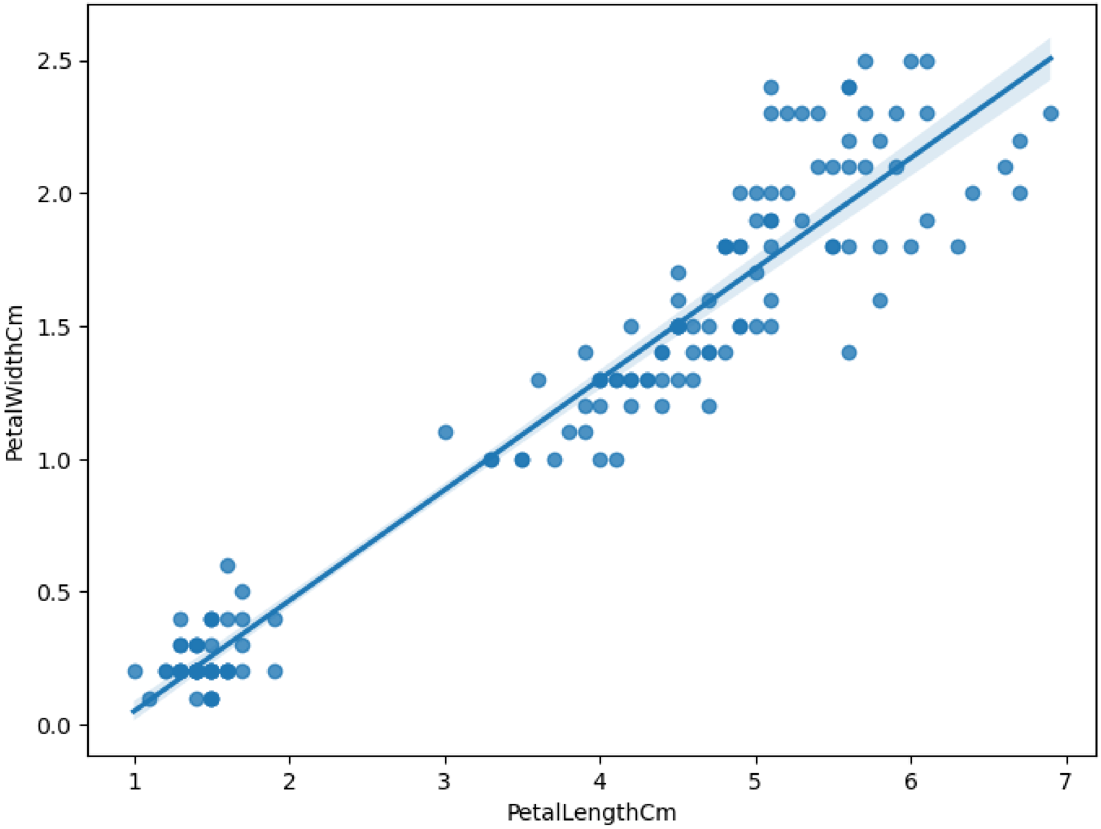
```
[19]: plt.figure(figsize=(8,6))
      sns.regplot(x='SepalLengthCm',y='SepalWidthCm',data=Data)
```

```
[19]: <Axes: xlabel='SepalLengthCm', ylabel='SepalWidthCm'>
```



[20]:
```
plt.figure(figsize=(8,6))
sns.regplot(x='PetalLengthCm',y='PetalWidthCm',data=Data)
```

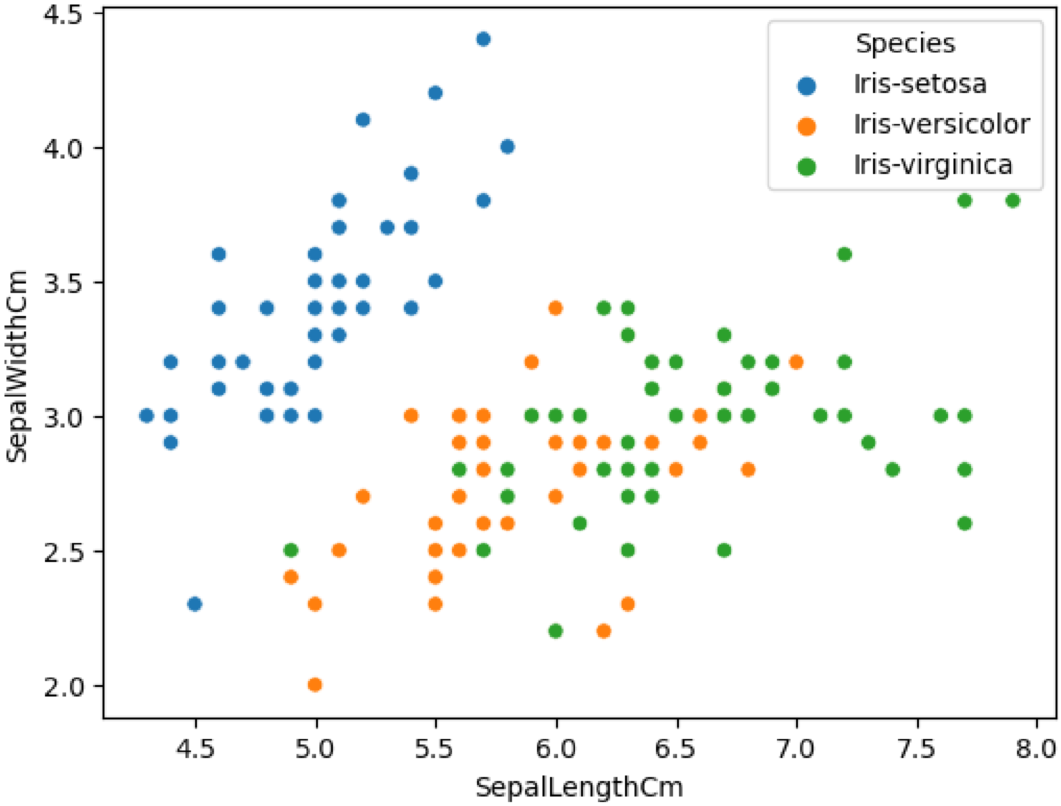
[20]:
```
<Axes: xlabel='PetalLengthCm', ylabel='PetalWidthCm'>
```



```
[21]: # Bivariate analysis
```

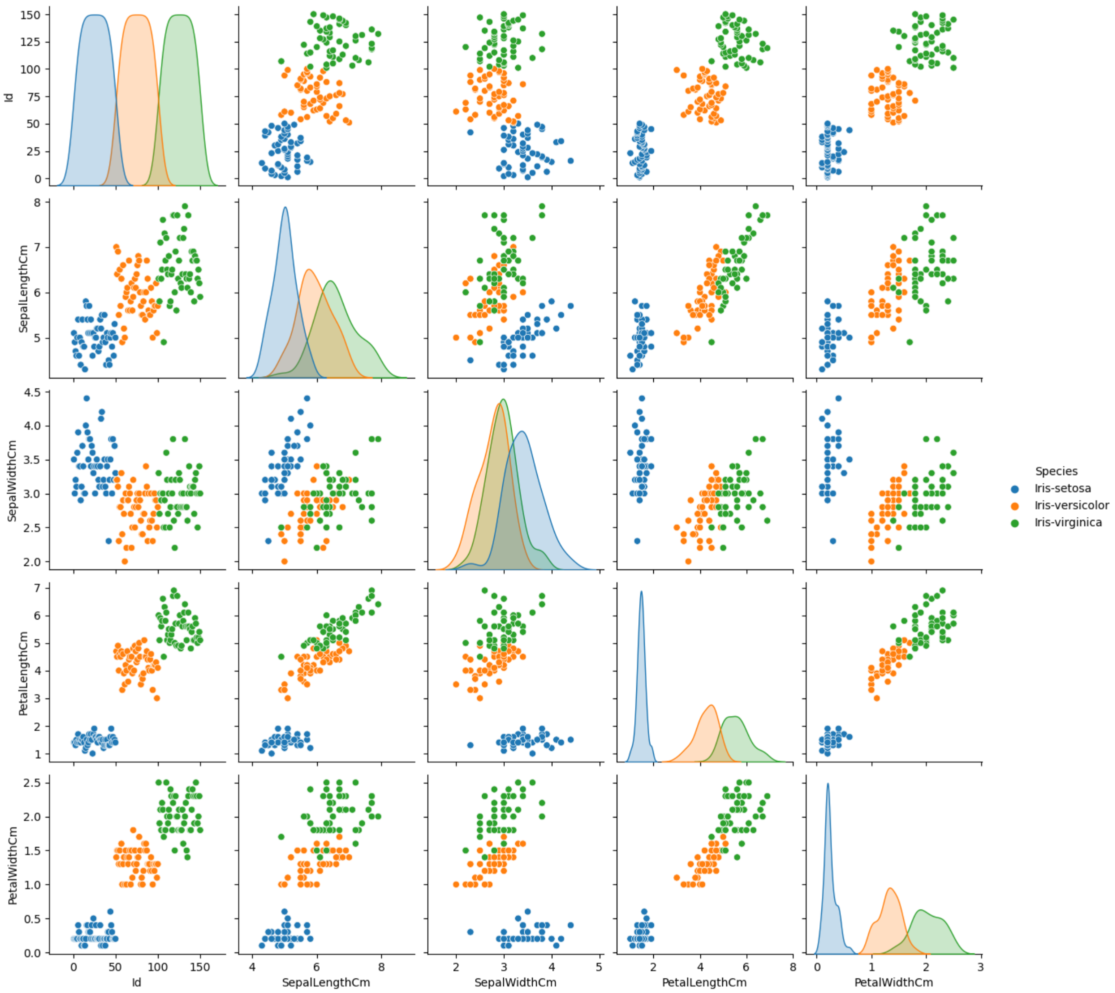
```
[22]: sns.
        ↪scatterplot(x=Data["SepalLengthCm"],y=Data["SepalWidthCm"],hue=Data['Species'])
```

```
[22]: <Axes: xlabel='SepalLengthCm', ylabel='SepalWidthCm'>
```



```
[23]: sns.pairplot(data = Data , hue = 'Species')
```

```
[23]: <seaborn.axisgrid.PairGrid at 0x18a39f8ce20>
```



[24]:
```
#  Machine Learning Models
```

[25]:
```
x = Data.iloc[: , 0:4].values
y = Data.iloc[:,4].values
```

[26]:
```
y
```

[26]:
```
array([0.2, 0.2, 0.2, 0.2, 0.2, 0.4, 0.3, 0.2, 0.2, 0.1, 0.2, 0.2, 0.1,
       0.1, 0.2, 0.4, 0.4, 0.3, 0.3, 0.3, 0.2, 0.4, 0.2, 0.5, 0.2, 0.2,
       0.4, 0.2, 0.2, 0.2, 0.2, 0.4, 0.1, 0.2, 0.1, 0.2, 0.2, 0.1, 0.2,
       0.2, 0.3, 0.3, 0.2, 0.6, 0.4, 0.3, 0.2, 0.2, 0.2, 0.2, 1.4, 1.5,
       1.5, 1.3, 1.5, 1.3, 1.6, 1. , 1.3, 1.4, 1. , 1.5, 1. , 1.4, 1.3,
       1.4, 1.5, 1. , 1.5, 1.1, 1.8, 1.3, 1.5, 1.2, 1.3, 1.4, 1.4, 1.7,
       1.5, 1. , 1.1, 1. , 1.2, 1.6, 1.5, 1.6, 1.5, 1.3, 1.3, 1.3, 1.2,
       1.4, 1.2, 1. , 1.3, 1.2, 1.3, 1.3, 1.1, 1.3, 2.5, 1.9, 2.1, 1.8,
       2.2, 2.1, 1.7, 1.8, 1.8, 2.5, 2. , 1.9, 2.1, 2. , 2.4, 2.3, 1.8,
       2.2, 2.3, 1.5, 2.3, 2. , 2. , 1.8, 2.1, 1.8, 1.8, 1.8, 2.1, 1.6,
       1.9, 2. , 2.2, 1.5, 1.4, 2.3, 2.4, 1.8, 1.8, 2.1, 2.4, 2.3, 1.9,
       2.3, 2.5, 2.3, 1.9, 2. , 2.3, 1.8])
```

[27]:

```
from sklearn.preprocessing import LabelEncoder
```

[28]:

```
encode = LabelEncoder()
y = encode.fit_transform(y)
y
```

[28]:

```
array([ 1,  1,  1,  1,  1,  3,  2,  1,  1,  0,  1,  1,  0,  0,  1,  3,  3,
        2,  2,  2,  1,  3,  1,  4,  1,  1,  3,  1,  1,  1,  1,  3,  0,  1,
        0,  1,  1,  0,  1,  1,  2,  2,  1,  5,  3,  2,  1,  1,  1,  1, 10,
       11, 11,  9, 11,  9, 12,  6,  9, 10,  6, 11,  6, 10,  9, 10, 11,  6,
       11,  7, 14,  9, 11,  8,  9, 10, 10, 13, 11,  6,  7,  6,  8, 12, 11,
       12, 11,  9,  9,  9,  8, 10,  8,  6,  9,  8,  9,  9,  7,  9, 21, 15,
       17, 14, 18, 17, 13, 14, 14, 21, 16, 15, 17, 16, 20, 19, 14, 18, 19,
       11, 19, 16, 16, 14, 17, 14, 14, 14, 17, 12, 15, 16, 18, 11, 10, 19,
       20, 14, 14, 17, 20, 19, 15, 19, 21, 19, 15, 16, 19, 14],
      dtype=int64)
```

[29]:

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# Encode the species column
label_encoder = LabelEncoder()
Data['Species'] = label_encoder.fit_transform(Data['Species'])

# Split the data into features and target
X = Data.drop(['Species', 'Id'], axis=1)
y = Data['Species']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

# Train a Random Forest Classifier
rf_classifier = RandomForestClassifier()
rf_classifier.fit(X_train, y_train)

# Predict on the test set
y_pred = rf_classifier.predict(X_test)

# Calculate the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
```

```
accuracy
```

[29]:

```
1.0
```

[30]:

```
X_train , X_test , y_train , y_test = train_test_split(x,y,test_size=.20)
```

[31]:

```
model = KNeighborsClassifier(n_neighbors=4)
model.fit(X_train , y_train)
```

[31]:

```
KNeighborsClassifier(n_neighbors=4)
```

[32]:

```
y_pred = model.predict(X_test)
y_pred
```

[32]:

```
array([0, 2, 1, 0, 1, 2, 1, 2, 1, 2, 1, 0, 1, 1, 2, 2, 2, 0, 1, 1, 1, 0,
       0, 1, 2, 0, 0, 1, 0, 0])
```

[33]:

```
y_test
```

[33]:

```
15     0
111    2
61     1
29     0
52     1
144    2
87     1
108    2
59     1
129    2
75     1
8      0
65     1
74     1
142    2
100    2
106    2
40     0
60     1
72     1
71     1
36     0
1      0
70     1
137    2
31     0
20     0
92     1
```

```
7      0
27     0
Name: Species, dtype: int32
```

[34]:

```python
from sklearn.metrics import accuracy_score

accuracy_score(y_test , y_pred)
```

[34]:

```
1.0
```

[35]:

```python
test_errors = []

for k in range(1,10):
    model = KNeighborsClassifier(n_neighbors=k)
    model.fit(X_train , y_train)


    y_pred_test = model.predict(X_test)
    error = 1- accuracy_score(y_test , y_pred_test)
    test_errors.append(error)

test_errors
```

[35]:

```
[0.0, 0.0, 0.0, 0.0, 0.0, 0.033333333333333326, 0.0, 0.033333333333333326, 0.0]
```