|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | |
| **Data Analytics III**  Implement Simple Naïve Bayes classiﬁcation algorithm using Python/R on iris.csv dataset.  Compute Confusion matrix to ﬁnd TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset.  **In [2]: import numpy as np**  **import matplotlib.pyplot as plt import pandas as pd**  **import seaborn as sns**  **from sklearn import datasets**  **from sklearn.naive\_bayes import GaussianNB**  **from sklearn.metrics import make\_scorer, accuracy\_score,preci from sklearn.metrics import classification\_report**  **from sklearn.metrics import confusion\_matrix**  **from sklearn.metrics import accuracy\_score ,precision\_score,r**  **Loading Data set**  **In [3]: # Load the iris dataset**  **df = pd.read\_csv('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.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

In [5]:

**df.isnull().sum**

Out [5]: <bound method NDFrame.\_add\_numeric\_operations.<locals>.sum of Id SepalLengthCm SepalWidthCm PetalLengthCm PetalWidthCm Species

1. **False False False False False False**
2. **False False False False False False**
3. **False False False False False False**
4. **False False False False False False**
5. **False False False False False False**

.. ... ... ... ... ...

...

1. **False False False False False False**
2. **False False False False False False**
3. **False False False False False False**
4. **False False False False False False**
5. **False False False False False False**

[150 rows x 6 columns]>

In [6]:

**=**

Out [6]:

**SepalLengthCm SepalWidthCm PetalLengthCm PetalWidthCm Species**

**df**

**df.drop(columns= ['Id'])**

**df.head()**

**0** 5.1

3.5

1.4

0.2

Iris-

setosa

**2** 4.7

3.2

1.3

0.2

Iris-

setosa

**4** 5.0

3.6

1.4

0.2

Iris-

setosa

## **1** 4.9 3.0 1.4 0.2 Iris-

setosa

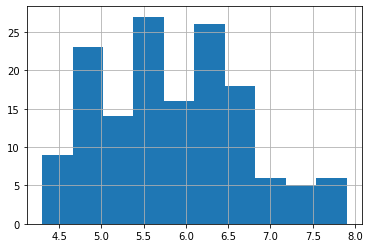
## **3** 4.6 3.1 1.5 0.2 Iris-

setosa

In [7]:

**df.describe()**

Out [7]:



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **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 [8]:

**df['Species'].value\_counts()**

Out [8]: Iris-setosa 50

Iris-versicolor 50

Iris-virginica 50

Name: Species, dtype: int64

In [9]:

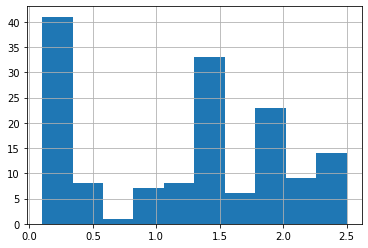
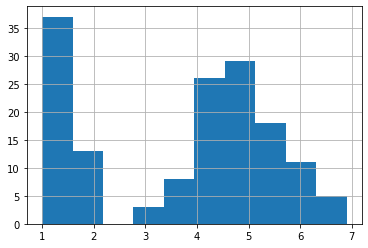
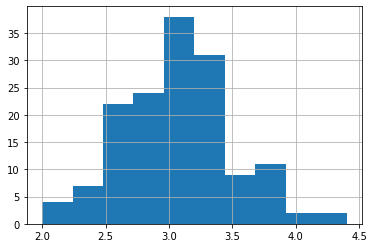
**df['SepalLengthCm'].hist()**

Out [9]: <matplotlib.axes.\_subplots.AxesSubplot at 0x7fdb8e2e2e20>

In [10]:

**df['SepalWidthCm'].hist()**

Out [10]: <matplotlib.axes.\_subplots.AxesSubplot at 0x7fdb8623e670>



**# Scatterplot**

**colors**

**['red', 'orange', 'blue']**

**species = ['Iris-virginica','Iris-versicolor','Iris-setosa']**

In [11]:

**df['PetalLengthCm'].hist()**

Out [11]: <matplotlib.axes.\_subplots.AxesSubplot at 0x7fdb85d06340>

In [12]:

**df['PetalWidthCm'].hist()**

Out [12]: <matplotlib.axes.\_subplots.AxesSubplot at 0x7fdb85c84e50>

In [13]:

**=**

In [14]:

**=**

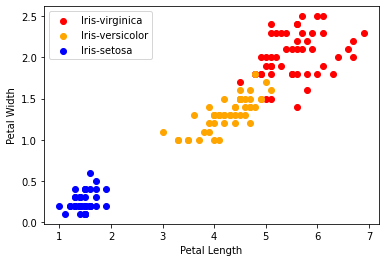
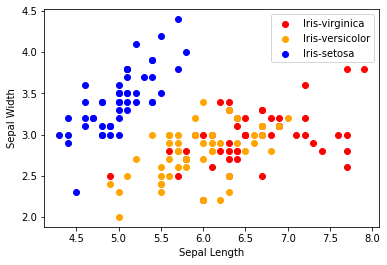
Out [14]: <matplotlib.legend.Legend at 0x7fdb85bf7790>

In [15]:

**=**

Out [15]: <matplotlib.legend.Legend at 0x7fdb85b78820>

In [16]:



**for i in range(3):**

**x = df[df['Species'] == species[i]] plt.scatter(x['PetalLengthCm'], x['PetalWidthCm'], c**

**plt.xlabel("Petal Length")**

**plt.ylabel("Petal Width") plt.legend()**

**co**

**for i in range(3):**

**x = df[df['Species'] == species[i]] plt.scatter(x['SepalLengthCm'], x['SepalWidthCm'], c**

**plt.xlabel("Sepal Length")**

**plt.ylabel("Sepal Width") plt.legend()**

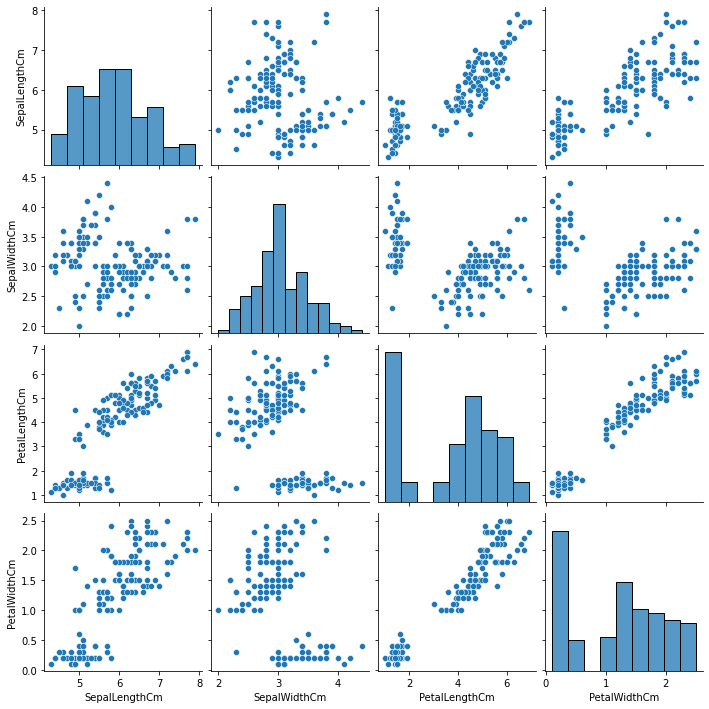
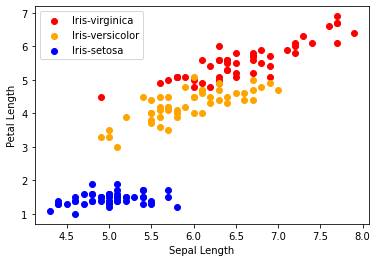
**co**

# for i in range(3):

**x = df[df['Species'] == species[i]] plt.scatter(x['SepalLengthCm'], x['PetalLengthCm'], c = c**

# plt.xlabel("Sepal Length")

**plt.ylabel("Petal Length") plt.legend()**



Out [16]: <matplotlib.legend.Legend at 0x7fdb85c805b0>

In [17]:

**sns.pairplot(df)**

Out [17]: <seaborn.axisgrid.PairGrid at 0x7fdb85c03d30>

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **In** | **[18]:** | **df.corr()** | | | | |  |
| **Out** | **[18]:** |  | **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 [19]: corr = df.corr()**  **fig, ax = plt.subplots(figsize=(5,4)) sns.heatmap(corr, annot=True, ax=ax, cmap = 'coolwarm')**  **Out [19]: <matplotlib.axes.\_subplots.AxesSubplot at 0x7fdb85345820>**    **In [20]: from sklearn.preprocessing import LabelEncoder le = LabelEncoder()**  **In [21]: df['Species'] = le.fit\_transform(df['Species']) df.head()**  **Out [21]:**  **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 | | | | | | | |

|  |  |  |
| --- | --- | --- |
| **In [33]:** | **X = df.iloc[:,:-1]** |  |
| **In [34]: X**  **Out [34]: SepalLengthCm SepalWidthCm PetalLengthCm PetalWidthCm 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 [22]: from sklearn.model\_selection import train\_test\_split # train - 70**  **# test - 30**  **X = df.drop(columns=['Species']) Y = df['Species']**  **x\_train, x\_test, y\_train, y\_test = train\_test\_split(X, Y, tes**  **In [25]:** **X**  **Out [25]:**  **SepalLengthCm SepalWidthCm PetalLengthCm PetalWidthCm**  **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 | | |

**SepalLengthCm SepalWidthCm PetalLengthCm PetalWidthCm**

**147** 6.5 3.0 5.2 2.0

**gaussian**

**GaussianNB()**

**gaussian.fit(x\_train, y\_train)**

**Y\_pred gaussian.predict(x\_test)**

**accuracy\_nb=round(accuracy\_score(y\_test,Y\_pred)\* 100, 2)**

**acc\_gaussian round(gaussian.score(x\_train, y\_train) \* 100,**

**cm**

**confusion\_matrix(y\_test, Y\_pred)**

**accuracy = accuracy\_score(y\_test,Y\_pred)**

**precision =precision\_score(y\_test, Y\_pred,average='micro') recall recall\_score(y\_test, Y\_pred,average='micro')**

**f1 f1\_score(y\_test,Y\_pred,average='micro') print('Confusion matrix for Naive Bayes\n',cm) print('accuracy\_Naive Bayes: %.3f' %accuracy) print('precision\_Naive Bayes: %.3f' %precision) print('recall\_Naive Bayes: %.3f' %recall)**

**print('f1-score\_Naive Bayes : %.3f' %f1)**

**148** 6.2 3.4 5.4 2.3

**149** 5.9 3.0 5.1 1.8

1. rows × 4 columns

|  |  |  |  |
| --- | --- | --- | --- |
| **In** | **[23]:** | **Y** | |
| **Out** | **[23]:** | **0** | **0** |
|  |  | **1** | **0** |
|  |  | **2** | **0** |
|  |  | **3** | **0** |
|  |  | **4** | **0** |
|  |  |  | **..** |
|  |  | **145** | **2** |
|  |  | **146** | **2** |
|  |  | **147** | **2** |
|  |  | **148** | **2** |
|  |  | **149** | **2** |

Name: Species, Length: 150, dtype: int64

In [35]:

**=**

**=**

In [36]:

**=**

**=**

**=**

**=**

Confusion matrix for Naive Bayes [[15 0 0]

[ 0 11 0]

[ 0 2 17]]

accuracy\_Naive Bayes: 0.956

precision\_Naive Bayes: 0.956

recall\_Naive Bayes: 0.956 f1-score\_Naive Bayes : 0.956

In [ ]:

**In [ ]:**

**In [ ]:**