Iris Classification-LGM(Task-1)

September 1, 2022

Let's Grow More(LGMVIP)-"DATA SCIENCE INTERN"

LGMVIP August-22

SRAVANI BANDIRAJULA

BEGINNER LEVEL TASK

TASK-1-Iris Flowers Classification ML Project:

This particular ML project is usually referred to as the "Hello World" of Machine Learning.The iris flowers dataset contains numeric attributes,and it is perfect for beginners to learn about supervised ML Algorithms,mainly how to load and handle data.Also,since this is a small dataset,it can easily fit in memory without requiring special transformations or scaling capabilities.

Dataset link:http://archive.ics.uci.edu/ml/machine-learning-databases/iris

Importing Libraries

[1]:
```
import numpy as np
import pandas as pd
```

[2]:
```
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

[3]:
```
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC,LinearSVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, plot_confusion_matrix,␣
 ↪classification_report,confusion_matrix
```

[4]:
```
pwd
```

[4]:
```
'C:\\Users\\Sravani'
```

[5]:
```
df=pd.read_csv('C:\\Users\\Sravani\iris.csv')
```

[6]:
```
df
```

[6]:
```
     SepalLength  SepalWidth  PetalLength  PetalWidth         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
..           …           …            …           …               …
145          6.7         3.0          5.2         2.3  Iris-virginica
146          6.3         2.5          5.0         1.9  Iris-virginica
147          6.5         3.0          5.2         2.0  Iris-virginica
148          6.2         3.4          5.4         2.3  Iris-virginica
149          5.9         3.0          5.1         1.8  Iris-virginica

[150 rows x 5 columns]
```

[7]:
```
df.head(5)
```

[7]:
```
   SepalLength  SepalWidth  PetalLength  PetalWidth      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
```

[8]:
```
df.shape
```

[8]:
```
(150, 5)
```

[9]:
```
df.columns
```

[9]:
```
Index(['SepalLength', 'SepalWidth', 'PetalLength', 'PetalWidth', 'Species'],
dtype='object')
```

[10]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   SepalLength  150 non-null    float64
 1   SepalWidth   150 non-null    float64
 2   PetalLength  150 non-null    float64
 3   PetalWidth   150 non-null    float64
 4   Species      150 non-null    object
dtypes: float64(4), object(1)
```

```
memory usage: 6.0+ KB
```

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

[11]:
```
SepalLength    0
SepalWidth     0
PetalLength    0
PetalWidth     0
Species        0
dtype: int64
```

[12]:
```
df['Species'].unique()
```

[12]:
```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

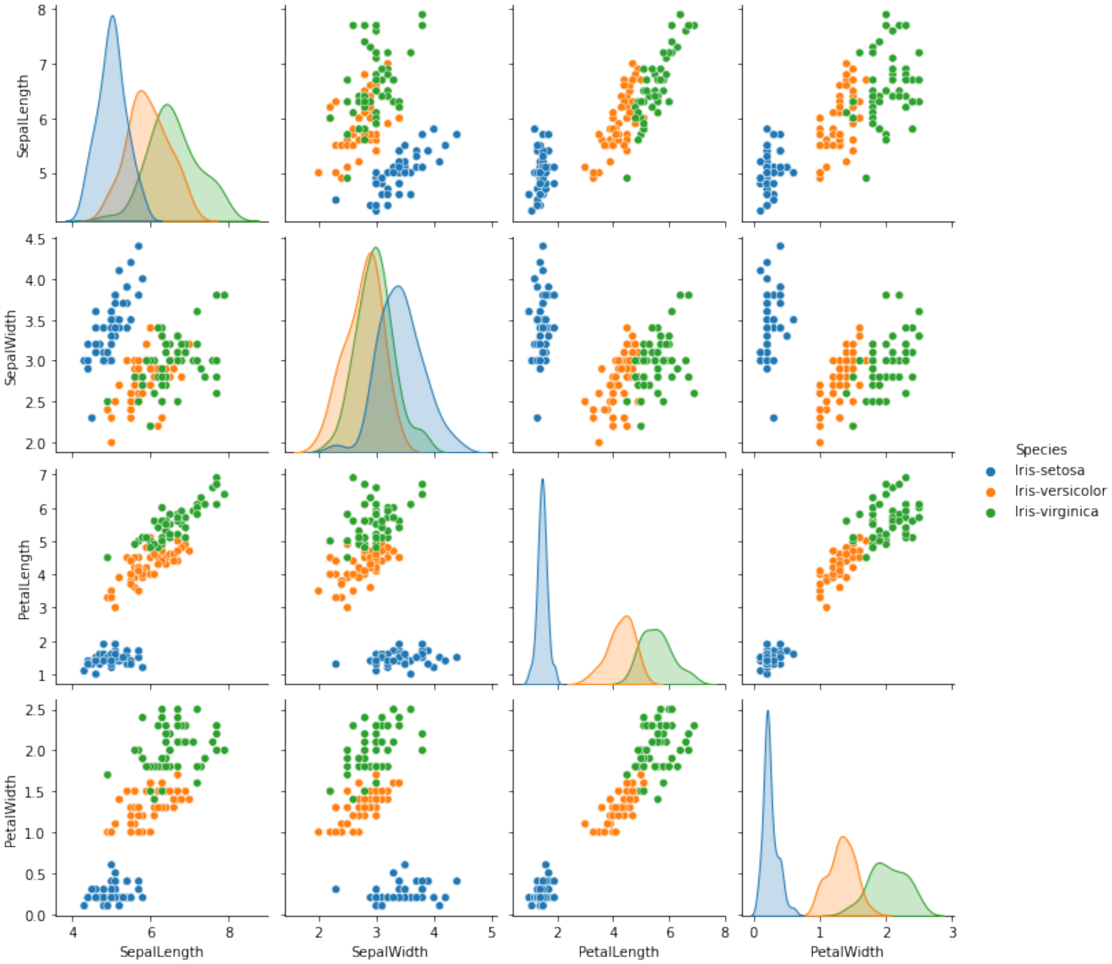
[13]:
```
df.describe()
```

[13]:
```
       SepalLength  SepalWidth  PetalLength  PetalWidth
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
```

Data Visualisation

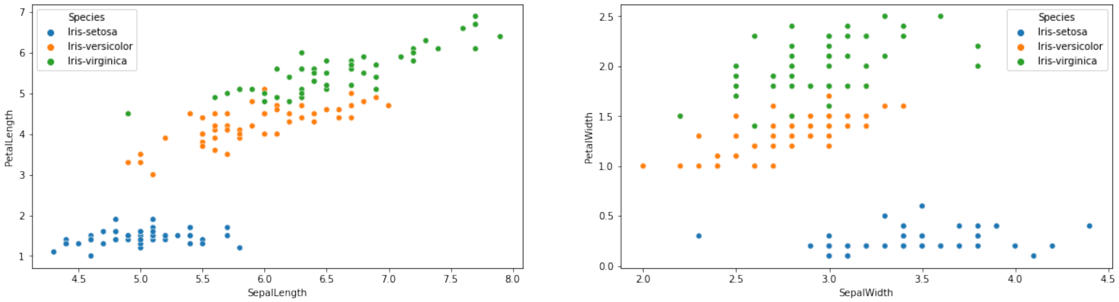
[14]:
```
sns.pairplot(df,hue="Species")
```

[14]:
```
<seaborn.axisgrid.PairGrid at 0x233b20fd5b0>
```



[15]:
```
fig,(ax1,ax2)=plt.subplots(ncols=2,figsize=(20,5))
sns.scatterplot(x='SepalLength',y='PetalLength',data=df,hue='Species',ax=ax1)
sns.scatterplot(x='SepalWidth',y='PetalWidth',data=df,hue='Species',ax=ax2)
```

[15]:
```
<AxesSubplot:xlabel='SepalWidth', ylabel='PetalWidth'>
```

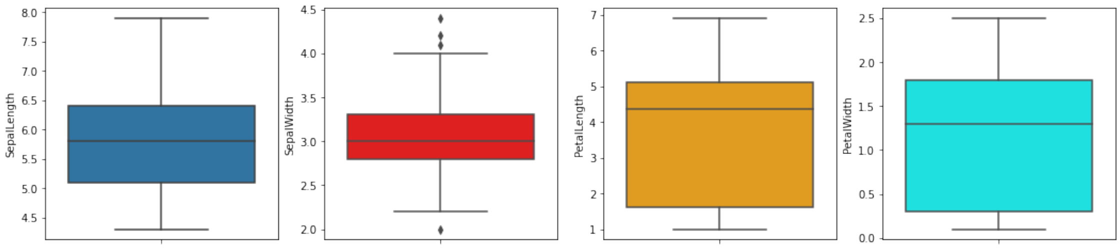


[16]:

```
plt.figure(figsize=(18,4))
plt.subplot(1,4,1)
sns.boxplot(data=df,y='SepalLength',)
plt.subplot(1,4,2)
sns.boxplot(data=df,y='SepalWidth',color='red')
plt.subplot(1,4,3)
sns.boxplot(data=df,y='PetalLength',color='orange')
plt.subplot(1,4,4)
sns.boxplot(data=df,y='PetalWidth',color='cyan')
```

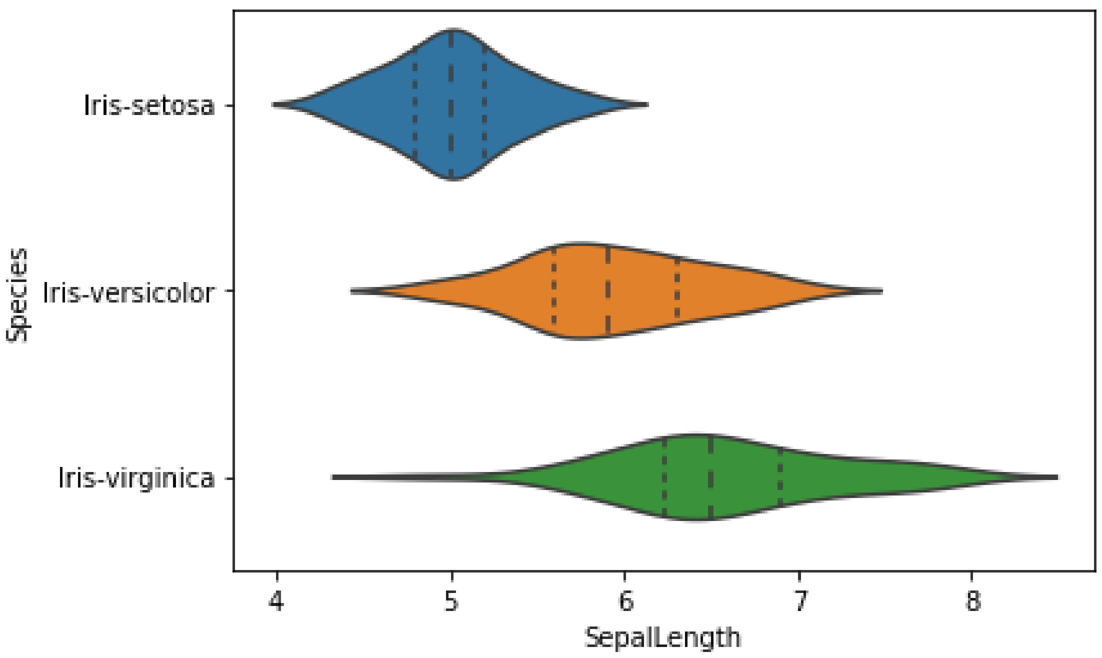
[16]:

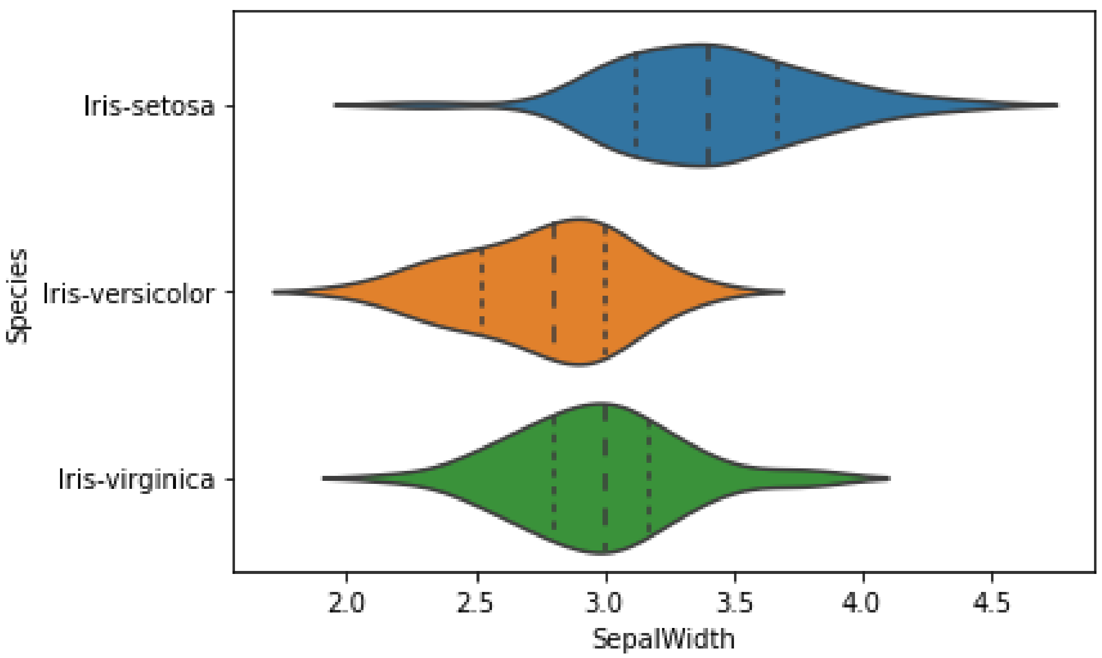
```
<AxesSubplot:ylabel='PetalWidth'>
```

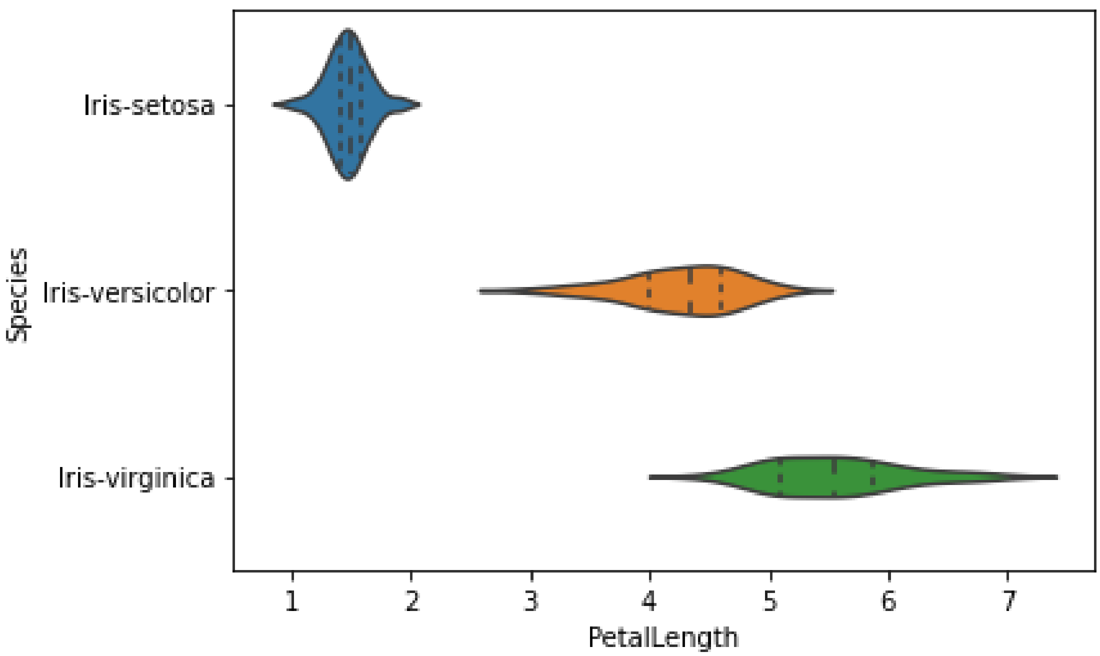


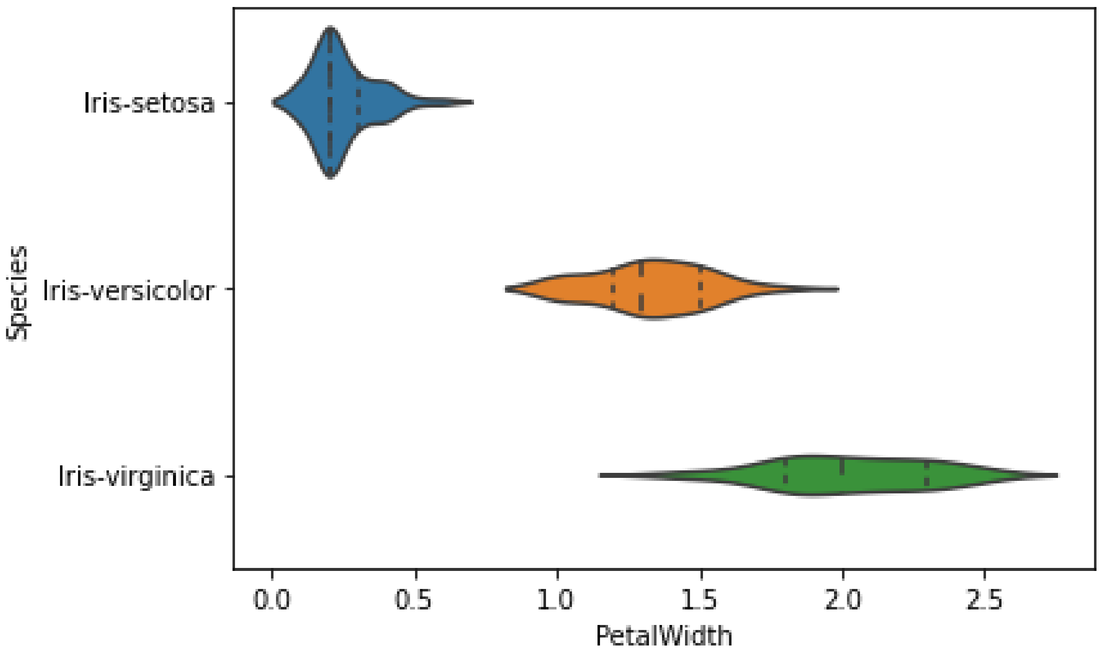
[17]:

```
sns.violinplot(y='Species', x='SepalLength', data=df, inner='quartile')
plt.show()
sns.violinplot(y='Species', x='SepalWidth', data=df, inner='quartile')
plt.show()
sns.violinplot(y='Species', x='PetalLength', data=df, inner='quartile')
plt.show()
sns.violinplot(y='Species', x='PetalWidth', data=df, inner='quartile')
plt.show()
```


Species
Iris-setosa
Iris-versicolor
Iris-virginica
4
5
6
7
8
SepalLength

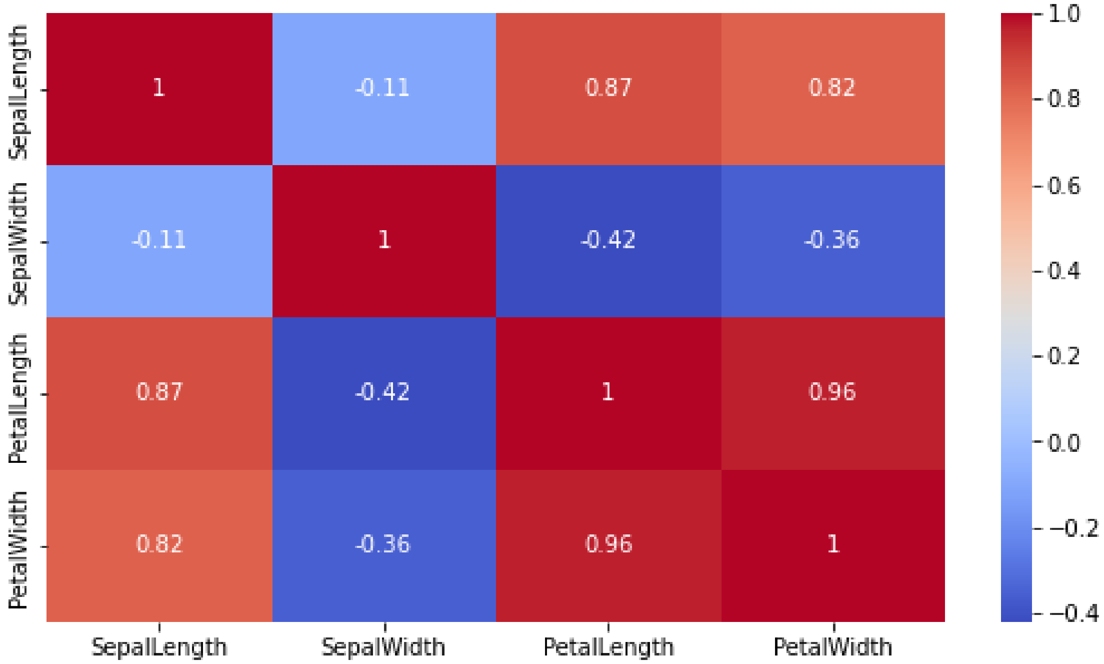

Species
Iris-setosa
Iris-versicolor
Iris-virginica
2.0
2.5
3.0
3.5
4.0
4.5
SepalWidth





[18]:

```
plt.figure(figsize=(9,5))
sns.heatmap(df.corr(), annot=True,cmap='coolwarm')
plt.show()
```



Building Model,Training and Testing

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

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

[20]:
```
    SepalLength  SepalWidth  PetalLength  PetalWidth      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
5           5.4         3.9          1.7         0.4  Iris-setosa
6           4.6         3.4          1.4         0.3  Iris-setosa
7           5.0         3.4          1.5         0.2  Iris-setosa
8           4.4         2.9          1.4         0.2  Iris-setosa
9           4.9         3.1          1.5         0.1  Iris-setosa
10          5.4         3.7          1.5         0.2  Iris-setosa
11          4.8         3.4          1.6         0.2  Iris-setosa
12          4.8         3.0          1.4         0.1  Iris-setosa
13          4.3         3.0          1.1         0.1  Iris-setosa
14          5.8         4.0          1.2         0.2  Iris-setosa
15          5.7         4.4          1.5         0.4  Iris-setosa
```

```
16           5.4          3.9           1.3          0.4  Iris-setosa
17           5.1          3.5           1.4          0.3  Iris-setosa
18           5.7          3.8           1.7          0.3  Iris-setosa
19           5.1          3.8           1.5          0.3  Iris-setosa
```

[21]:

```
from sklearn.model_selection import train_test_split
X = df.drop(columns=['Species'])
Y = df['Species']
x_train , x_test , y_train , y_test = train_test_split(X , Y , test_size = 0.3)
```

1.Logistic Regression

[22]:

```
# Initialize a Logistic Regression
lg= LogisticRegression(max_iter=1000)
```

[23]:

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

[23]:

```
LogisticRegression(max_iter=1000)
```

[24]:

```
LogisticRegression(max_iter=1000)
```

[24]:

```
LogisticRegression(max_iter=1000)
```

[25]:

```
# Predict on the test set and calculate accuracy
y_pred=lg.predict(x_test)
score=accuracy_score(y_test,y_pred)
```

[26]:

```
def report(model):
    preds=model.predict(x_test)
    print(classification_report(preds,y_test))
    plot_confusion_matrix(model,x_test,y_test)
```

[27]:

```
print('Logistic Regression')
report(lg)
print(f'Accuracy: {round(score*100,2)}%')
```

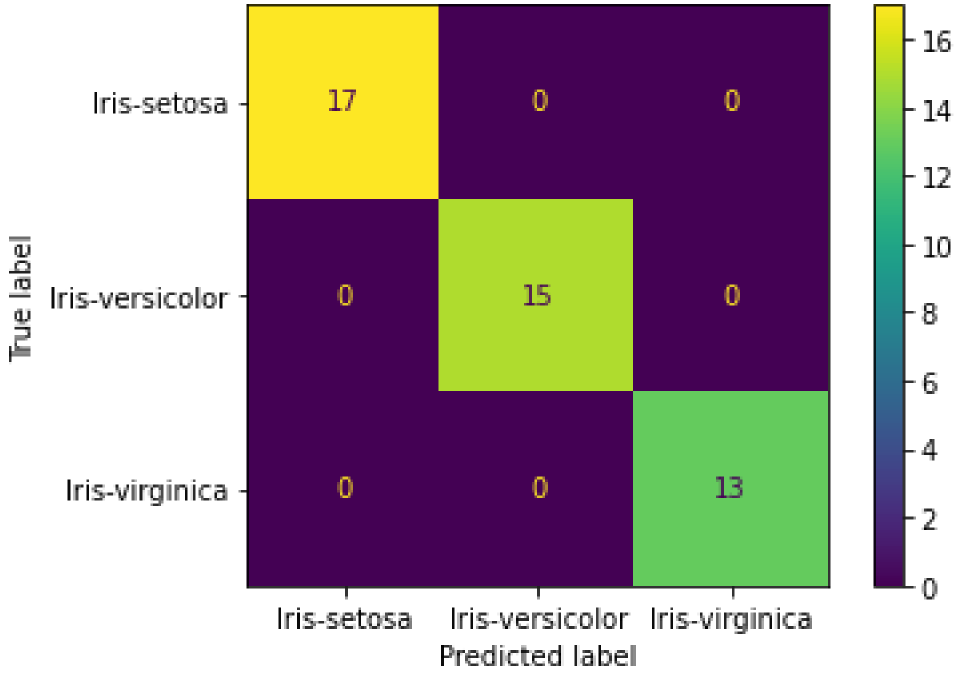
```
Logistic Regression
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        17
Iris-versicolor       1.00      1.00      1.00        15
 Iris-virginica       1.00      1.00      1.00        13

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

Accuracy: 100.0%
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87:
FutureWarning: Function plot_confusion_matrix is deprecated; Function
`plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use one
of the class methods: ConfusionMatrixDisplay.from_predictions or
ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```



2.Decision Tree Classifier

[28]:
```python
DTC = DecisionTreeClassifier()
DTC=DTC.fit(x_train,y_train)
# Predict on the test set and calculate accuracy
y_pred=DTC.predict(x_test)
score=accuracy_score(y_test,y_pred)
```

[29]:
```python
print('Decision Tree Classifier')
report(DTC)
print(f'Accuracy: {round(score*100,2)}%')
```

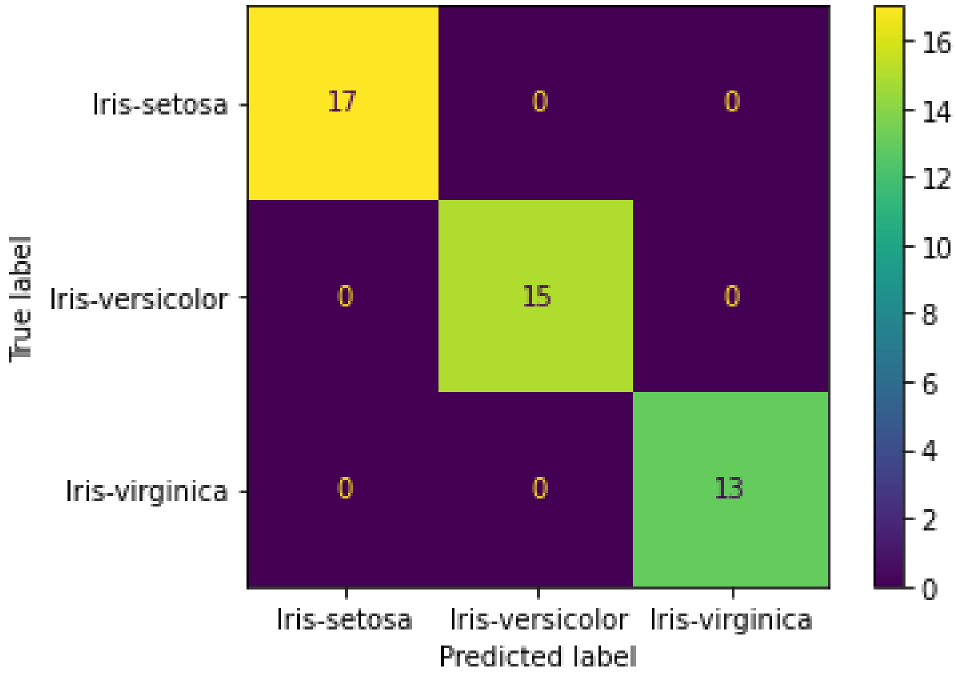
```
Decision Tree Classifier
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        17
Iris-versicolor       1.00      1.00      1.00        15
 Iris-virginica       1.00      1.00      1.00        13
```

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

Accuracy: 100.0%
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87:
FutureWarning: Function plot_confusion_matrix is deprecated; Function
`plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use one
of the class methods: ConfusionMatrixDisplay.from_predictions or
ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```



3.KNN

[30]:
```
KNN=KNeighborsClassifier(n_neighbors=6)
KNN.fit(x_train, y_train)
```

[30]:
```
KNeighborsClassifier(n_neighbors=6)
```

[31]:
```
# Predict on the test set and calculate accuracy
y_pred=KNN.predict(x_test)
score=accuracy_score(y_test,y_pred)
print('KNN')
report(KNN)
print(f'Accuracy: {round(score*100,2)}%')
```

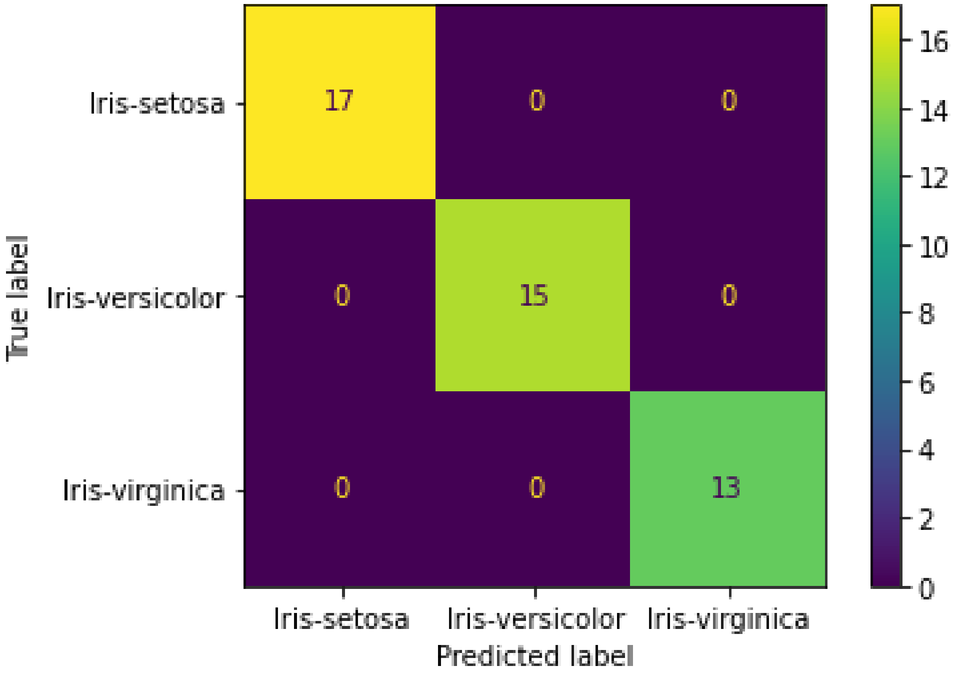
```
KNN
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        17
Iris-versicolor       1.00      1.00      1.00        15
 Iris-virginica       1.00      1.00      1.00        13

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

Accuracy: 100.0%
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87:
FutureWarning: Function plot_confusion_matrix is deprecated; Function
`plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use one
of the class methods: ConfusionMatrixDisplay.from_predictions or
ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```



4.Multinomial Naive Bayes

```
[32]: NB= MultinomialNB()
NB.fit(x_train,y_train)
MultinomialNB()
# Predict on the test set and calculate accuracy
```

```
y_pred=NB.predict(x_test)
score=accuracy_score(y_test,y_pred)
print('NB')
report(NB)
print(f'Accuracy: {round(score*100,2)}%')
```

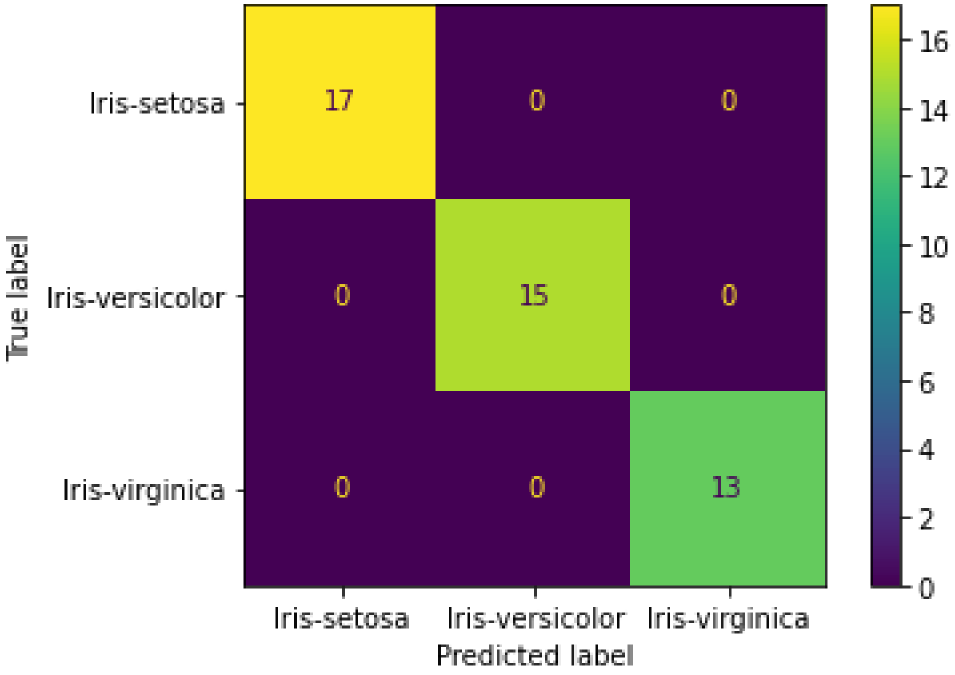
```
NB
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        17
Iris-versicolor       1.00      1.00      1.00        15
 Iris-virginica       1.00      1.00      1.00        13

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

Accuracy: 100.0%
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87:
FutureWarning: Function plot_confusion_matrix is deprecated; Function
`plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use one
of the class methods: ConfusionMatrixDisplay.from_predictions or
ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```



```
[ ]:
```