LetsGrowMore Data Science Internship

Beginner Level - TASK 1

Iris Flowers Classification ML Project:

BY SMRITHIKA ANTONETTE

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.

Importing the necessary libraries

In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
```

Loading the dataset

In [2]:

```
df=pd.read_csv("Iris.csv")
df
```

Out[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
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

In [3]:
```
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]:
```
data_size=df.shape
print(f"Number of rows :{data_size[0]}")
print(f"Number of columns :{data_size[1]}")
```

```
Number of rows :150
Number of columns :6
```

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

In [6]:
```
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 [7]:
```
df.describe()
```

Out[7]:

	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

Out[8]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

In [9]:

```
df.head()
```

Out[9]:

	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

Getting the size of the dataset

In [10]:

```
data_size=df.shape
print(f"Number of rows :{data_size[0]}")
print(f"Number of columns :{data_size[1]}")
```

```
Number of rows :150
Number of columns :6
```

In [11]:

```
df.isnull().sum()
```

Out[11]:

```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

Analyzing and visualzing the dataset

In [12]:

```
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 [13]:

```
df.describe()
```

Loading [MathJax]/extensions/Safe.js

Out[13]:

	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

In [14]:

```
df.tail()
```

Out[14]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

In [15]:

```
df.head()
```

Out[15]:

	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

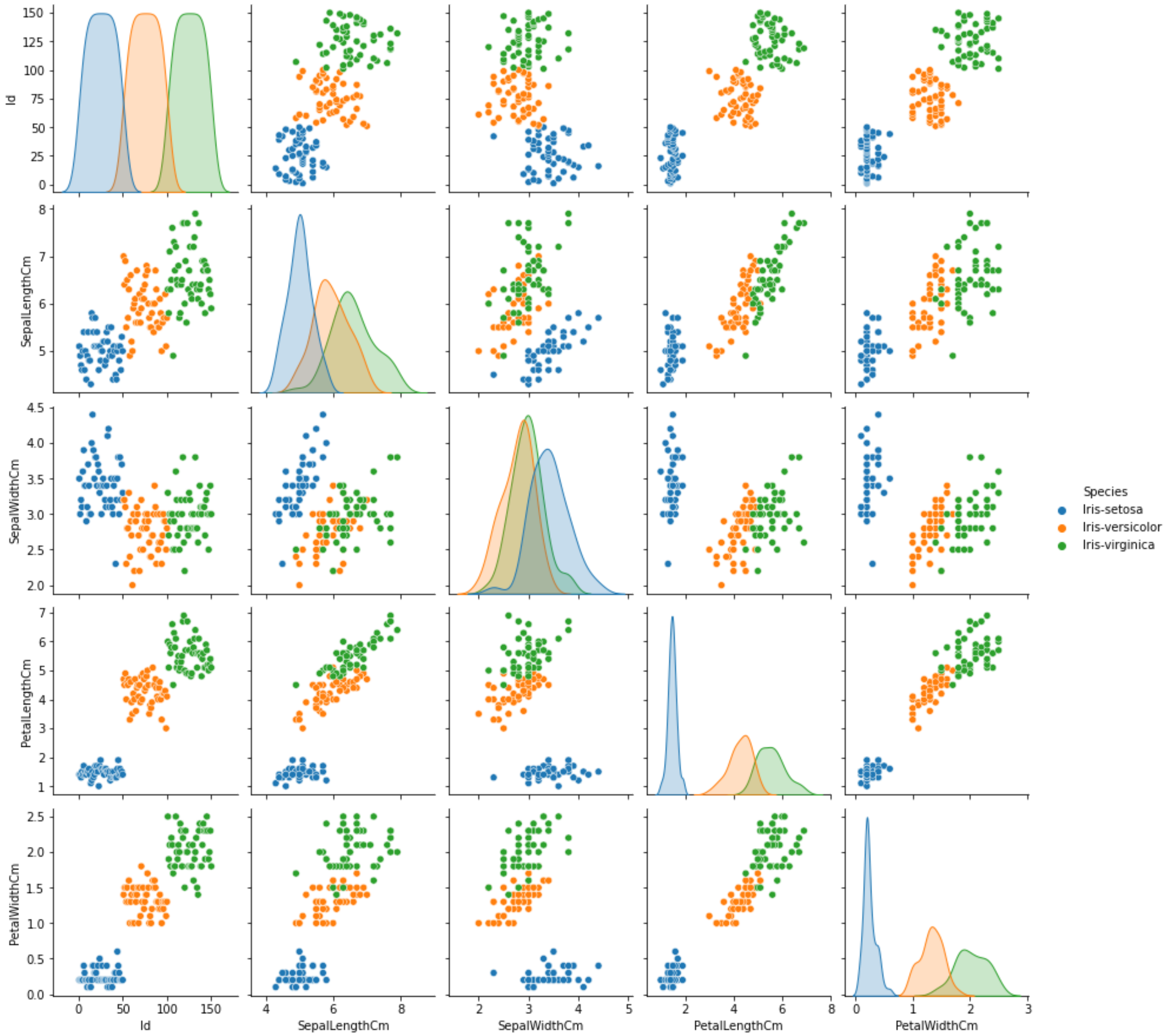
Pair Plot

In [16]:

```
sns.pairplot(df, hue='Species')
```

Out[16]:

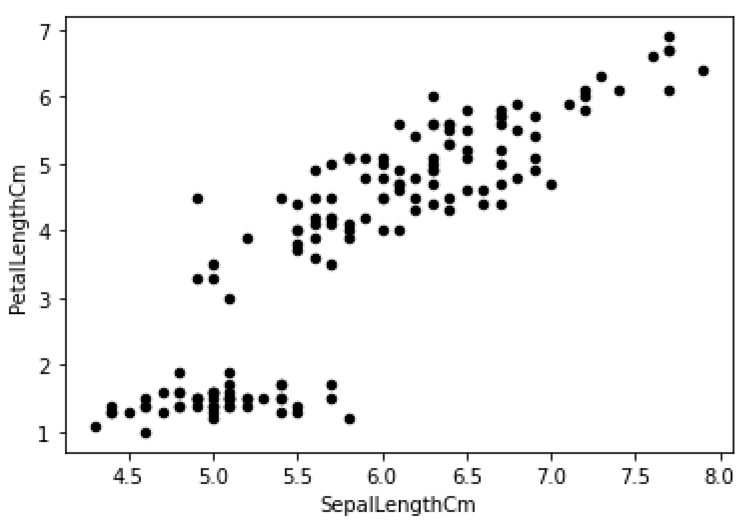
```
<seaborn.axisgrid.PairGrid at 0x2b796d83700>
```



1.Scatter Plot

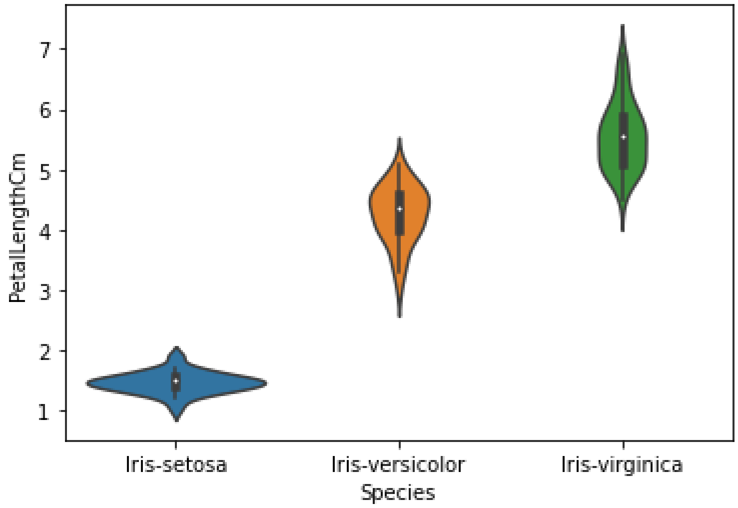
In [17]:
```
df.plot(kind="scatter", x="SepalLengthCm", y="PetalLengthCm",color="black", alpha=1)
```

Out[17]: <AxesSubplot:xlabel='SepalLengthCm', ylabel='PetalLengthCm'>



2.Violin Plot

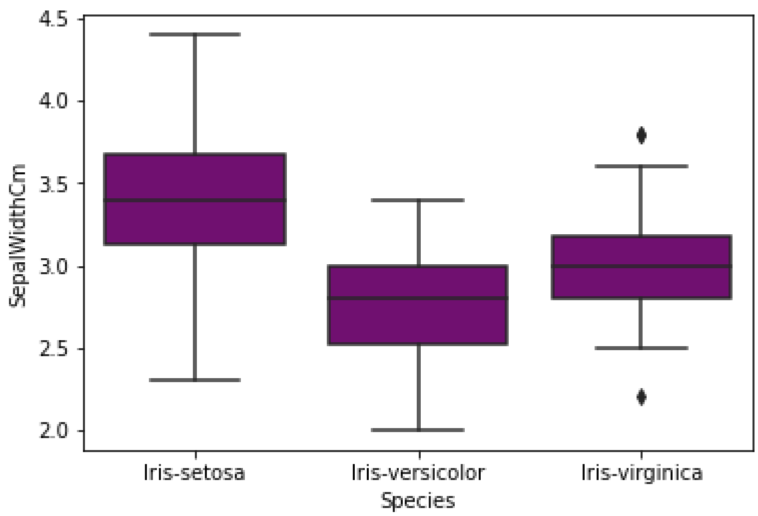
In [18]:
```
sns.violinplot(x='Species', y='PetalLengthCm', data=df)
plt.show()
```



3.Box Plot

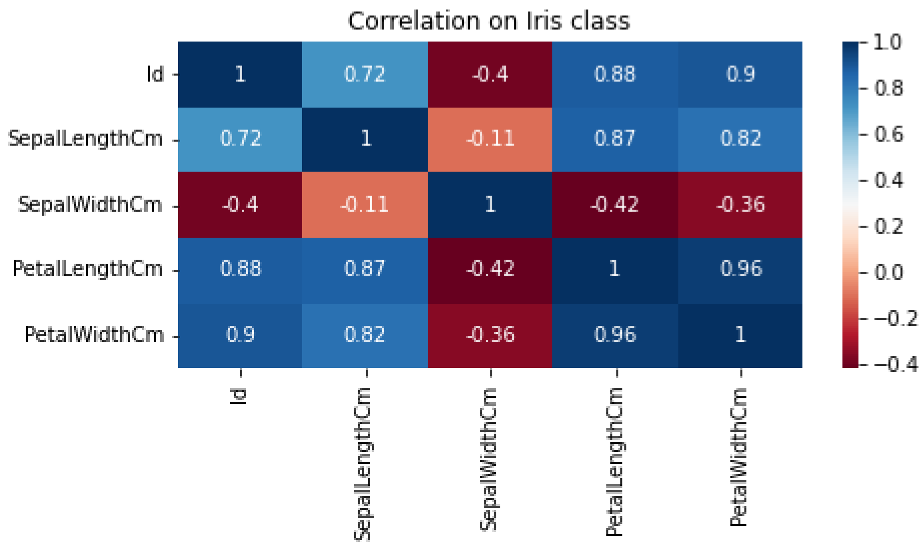
In [19]:
```
sns.boxplot(x="Species",y="SepalWidthCm",data=df,color="purple")
```

Out[19]: <AxesSubplot:xlabel='Species', ylabel='SepalWidthCm'>



4.Heat Map

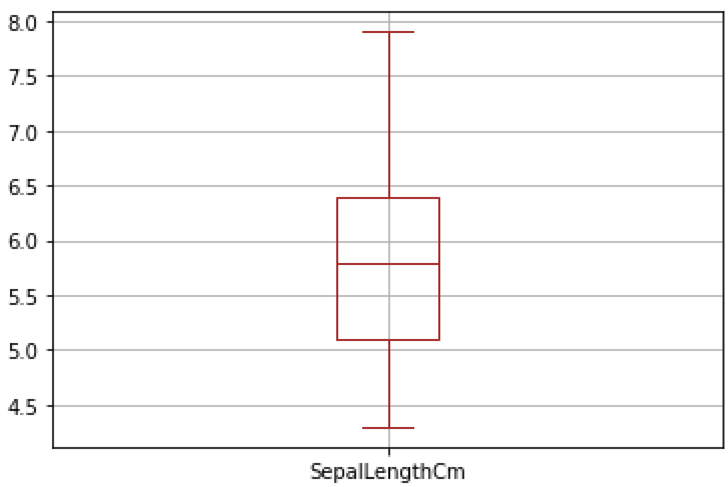
In [20]:
```
plt.subplots(figsize = (7,3))
sns.heatmap(df.corr(),annot=True,cmap="RdBu").set_title("Correlation on Iris class")
plt.show()
```



Check for Outliers

In [21]:
```
df.boxplot(column=['SepalLengthCm'],color="brown")
```

Out[21]: <AxesSubplot:>



correlation of Iris Features

In [22]:
```
#correlation of the Iris features
df.cov()
```

Out[22]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
Id	1887.500000	25.782886	-7.492282	67.667785	29.832215
SepalLengthCm	25.782886	0.685694	-0.039268	1.273682	0.516904
SepalWidthCm	-7.492282	-0.039268	0.188004	-0.321713	-0.117981
PetalLengthCm	67.667785	1.273682	-0.321713	3.113179	1.296387
PetalWidthCm	29.832215	0.516904	-0.117981	1.296387	0.582414

Splitting the Dataset

In [23]:
```
x = df.drop(['Species'], axis =1)
es']
```

Loading [MathJax]/extensions/Safe.js

In [24]:
```
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.7,random_state =0)
```

Logistic Regression

In [25]:
```
log_reg = LogisticRegression()
log_reg.fit(x_train, y_train)
predictions = log_reg.predict(x_test)
print ("Logistic Regression")
print ("The Accuracy Score ", accuracy_score(y_test, predictions))
print (confusion_matrix(y_test, predictions))
print (classification_report(y_test, predictions))
```

```
Logistic Regression
The Accuracy Score  0.9809523809523809
[[33  0  0]
 [ 1 33  0]
 [ 0  1 37]]
                 precision    recall  f1-score   support

    Iris-setosa       0.97      1.00      0.99        33
Iris-versicolor       0.97      0.97      0.97        34
 Iris-virginica       1.00      0.97      0.99        38

       accuracy                           0.98       105
      macro avg       0.98      0.98      0.98       105
   weighted avg       0.98      0.98      0.98       105
```

```
c:\users\smrithika\appdata\local\programs\python\python39\lib\site-packages\sklearn\line
ar_model\_logistic.py:814: ConvergenceWarning: lbfgs failed 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(
```

SVM

In [26]:
```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn import svm
model = SVC() # select the svm algorithm
clf=svm.SVC(gamma=0.001,C=100.)

# we train the algorithm with training data and training output
model.fit(x_train, y_train)
clf.fit(x_train, y_train)
# we pass the testing data to the stored algorithm to predict the outcome
prediction = model.predict(x_test)
print("Support Vector Machines")
print('Train-The accuracy of the SVM is: ', accuracy_score(prediction, y_test))
```

```
Support Vector Machines
Train-The accuracy of the SVM is:  0.9523809523809523
```

In [27]:
```
# train
model = SVC() # select the svm algorithm
```

Loading [MathJax]/extensions/Safe.js

```
# we train the algorithm with training data and training output
model.fit(x_train, y_train)

prediction = model.predict(x_train)
print("Support Vector Machines")
print ("Train-The accuracy of the SVM is:", accuracy_score(y_test, predictions))
print ("Train - Confusion matrix :\n",confusion_matrix(y_train, clf.predict(x_train)))

#classification report
print (classification_report(y_test, predictions))
```

```
Support Vector Machines
Train-The accuracy of the SVM is: 0.9809523809523809
Train - Confusion matrix :
 [[17  0  0]
 [ 0 16  0]
 [ 0  0 12]]
                 precision    recall  f1-score   support

    Iris-setosa       0.97      1.00      0.99        33
Iris-versicolor       0.97      0.97      0.97        34
 Iris-virginica       1.00      0.97      0.99        38

       accuracy                           0.98       105
      macro avg       0.98      0.98      0.98       105
   weighted avg       0.98      0.98      0.98       105
```

In [28]:

```
#test
print ("Test - Accuracy :", accuracy_score(y_test, clf.predict
(x_test)))
print ("Test-Confusion matrix :\n",confusion_matrix(y_test, clf.
predict(x_test)))
print (classification_report(y_test, predictions))
```

```
Test - Accuracy : 0.9809523809523809
Test-Confusion matrix :
 [[33  0  0]
 [ 1 33  0]
 [ 0  1 37]]
                 precision    recall  f1-score   support

    Iris-setosa       0.97      1.00      0.99        33
Iris-versicolor       0.97      0.97      0.97        34
 Iris-virginica       1.00      0.97      0.99        38

       accuracy                           0.98       105
      macro avg       0.98      0.98      0.98       105
   weighted avg       0.98      0.98      0.98       105
```

Using Logistic Regression the Accuracy of our Model is 95%

Using Support Vector Machines, the accuracy of our model is 98.05%