Homework Directions for Machine Learning Supervised Logistic Regression

Yog Chaudhary

ID 11727095

Professor: Dr. Bill Randall

University Of North Texas

Machine Learning Supervised Logistic Regression

Logistic regression is a supervised learning algorithm used to predict a dependent categorical target variable. In essence, if you have a large set of data that you want to categorize, logistic regression may be able to help.

Logistic regression is another technique borrowed by machine learning from the field of statistics. It is the go-to method for binary classification problems with two class values.In this post you will discover the logistic regression algorithm for machine learning.

https://www.mastersindatascience.org/learning/machine-learning-algorithms/logistic-regression/

In [2]:
```
# Import Python Libraries: NumPy and Pandas

import pandas as pd
import numpy as np
```

In [3]:
```
# Import Libraries & modules for data visualization

from pandas.plotting import scatter_matrix
import matplotlib.pyplot as plt
import seaborn as sns
```

In [4]:
```
# Import scikit-Learn module for the algorithm/modeL: Logistic Regression
from sklearn.linear_model import LogisticRegression
```

In [5]:
```
# Import scikit-Learn module to split the dataset into train/ test sub-datasets
from sklearn.model_selection import train_test_split
```

In [6]:
```
# Import scikit-Learn module for K-fold cross-validation - algorithm/modeL evaluation

from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
```

In [7]:
```
# Import scikit-Learn module classification report to later use for information about
#try to classify/lable each record

from sklearn.metrics import classification_report
```

In [8]:
```
# Import Python ALL Libraries: NumPy and Pandas
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from pandas.plotting import scatter_matrix
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.metrics import classification_report
```

In [11]:
```
# Load the data set into a pandas dataframe
# Read the Iris data set and create the dataframe df


# filepath = '../data/Iris.csv'
df = pd.read_csv ("Iris.csv")
df.head()
```

Out[11]:

	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

Performing rge Exploratory Data Analysis(EDA)

In [16]:
```
# get the dimensions or shape of the dataset
# i.e. number of records / rows X number of variables / columns

print("Shape of the dataset(rows, columns):",df.shape)
```

```
Shape of the dataset(rows, columns): (150, 6)
```

In [18]:
```
# get the data types of all the variables / attributes in the data set

print(df.dtypes)
```

```
Id                int64
SepalLengthCm    float64
SepalWidthCm     float64
PetalLengthCm    float64
PetalWidthCm     float64
Species           object
dtype: object
```

In [19]:
```
# return the summary statistics of the numeric variables/attributes in the data set

print(df.describe())
```

```
               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 [20]:
```
# Class distribution i.e. how many records are in each class

print(df.groupby('Species').size())
```

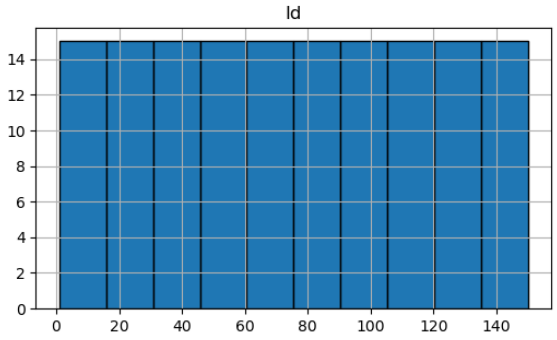
```
Species
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
dtype: int64
```

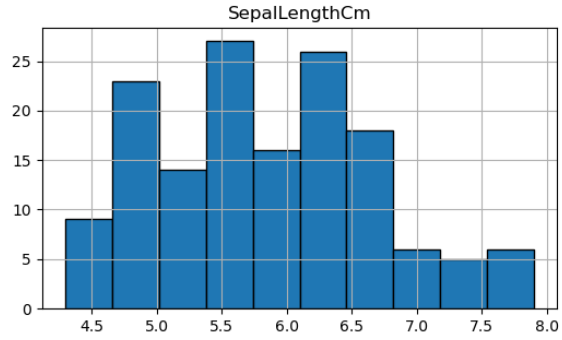
In [21]:
```
df.info()
```

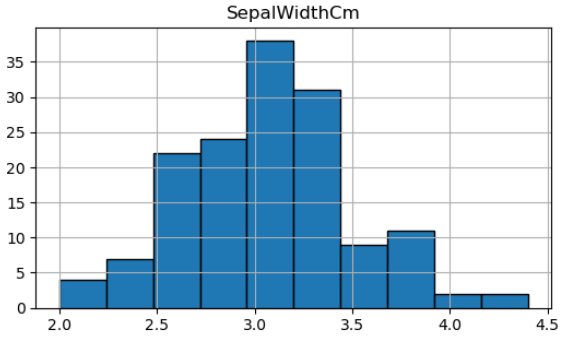
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 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
```

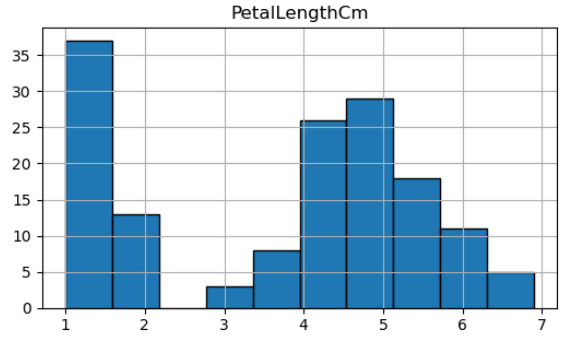
Creating a Histogram

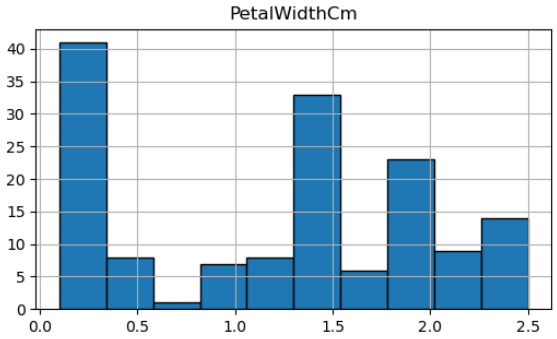
In [22]:
```
# Plot histogram for each variable. I encourage you to work with the histogram. Rememb

df.hist(edgecolor= 'black',figsize=(14,12))
plt.show()
```





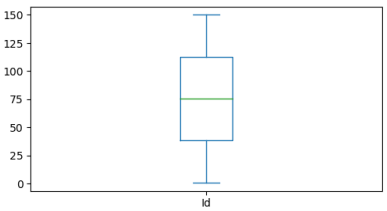


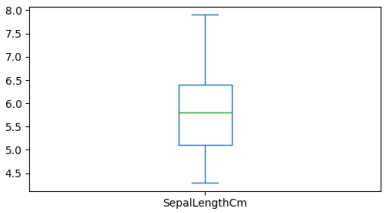


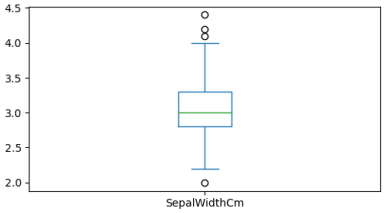


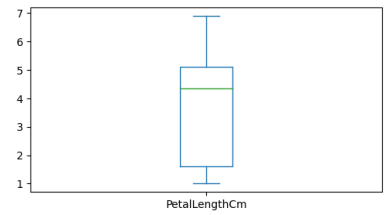
Creating a Box Plots

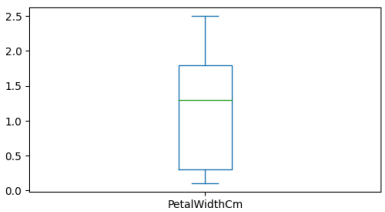
In [23]:
```
# Boxplots

df.plot(kind="box", subplots=True, layout=(5,3), sharex=False, figsize=(20,18))
plt.show()
```





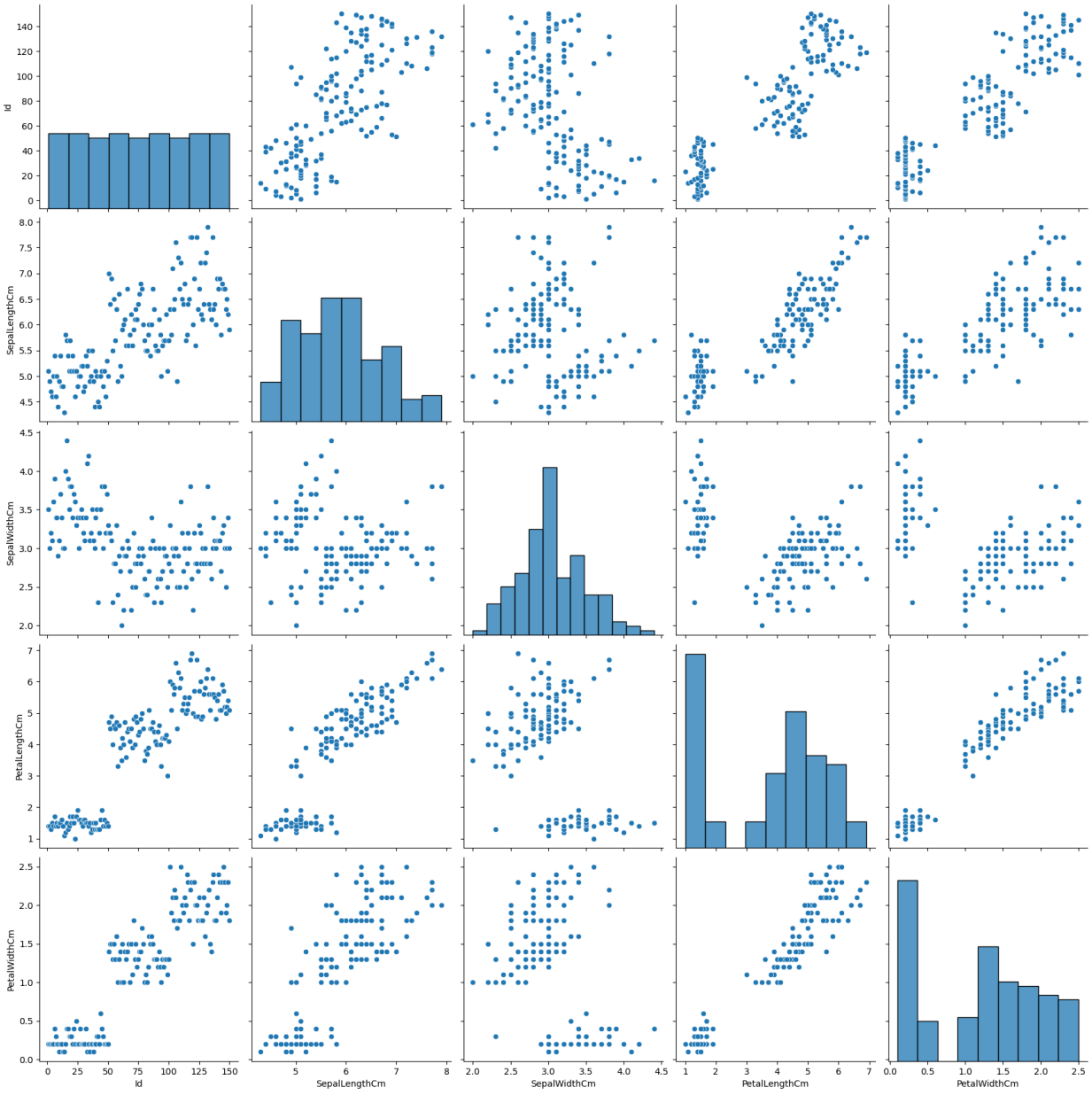






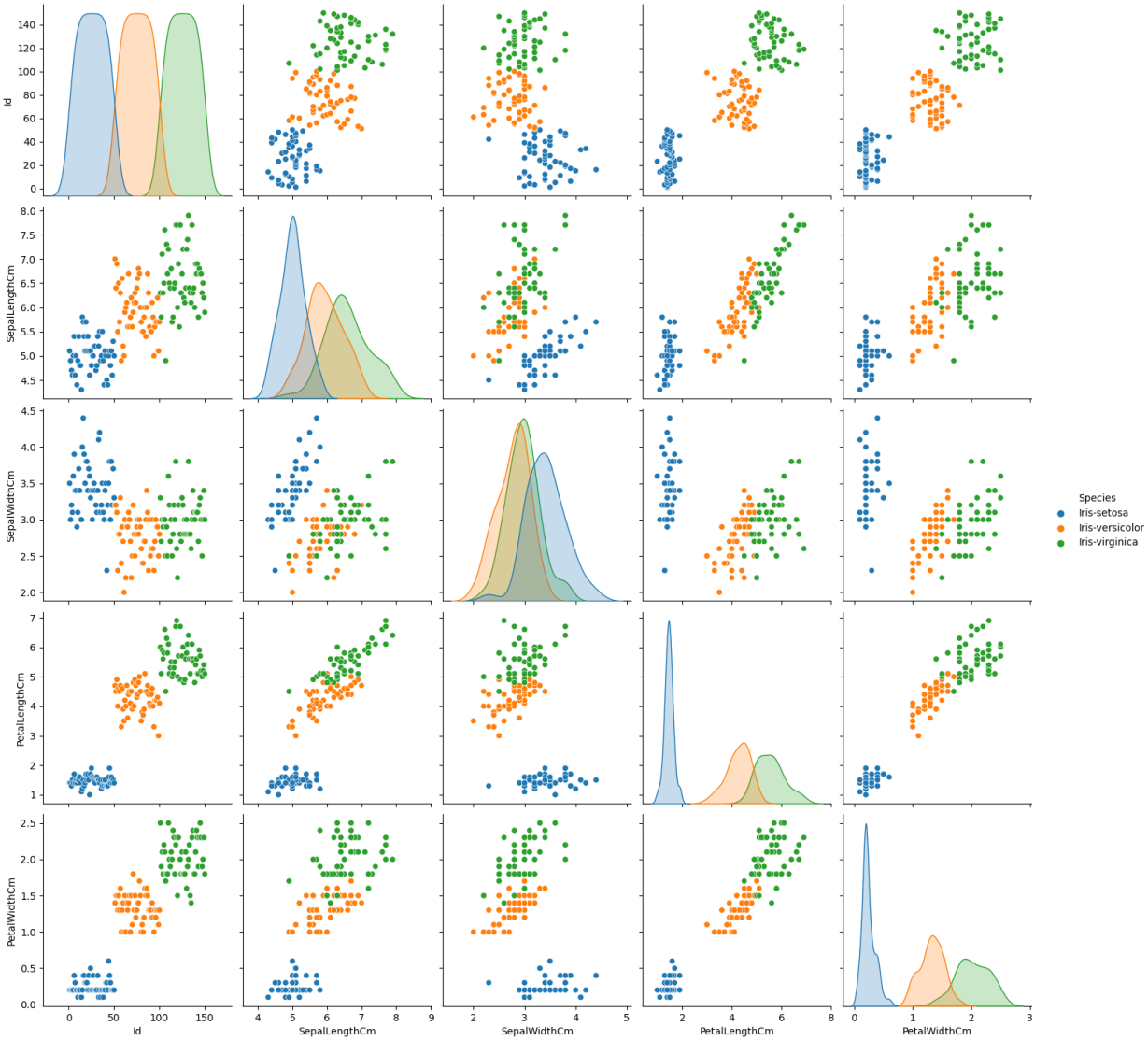
Creating a Pair plots

In [24]:
```
# Please click on the above URL to learn more about Pair Plots
# I know this is a lot of information but I wanted you to see what is possible with Se

sns.pairplot(df, height=3.5);
plt.show()
```



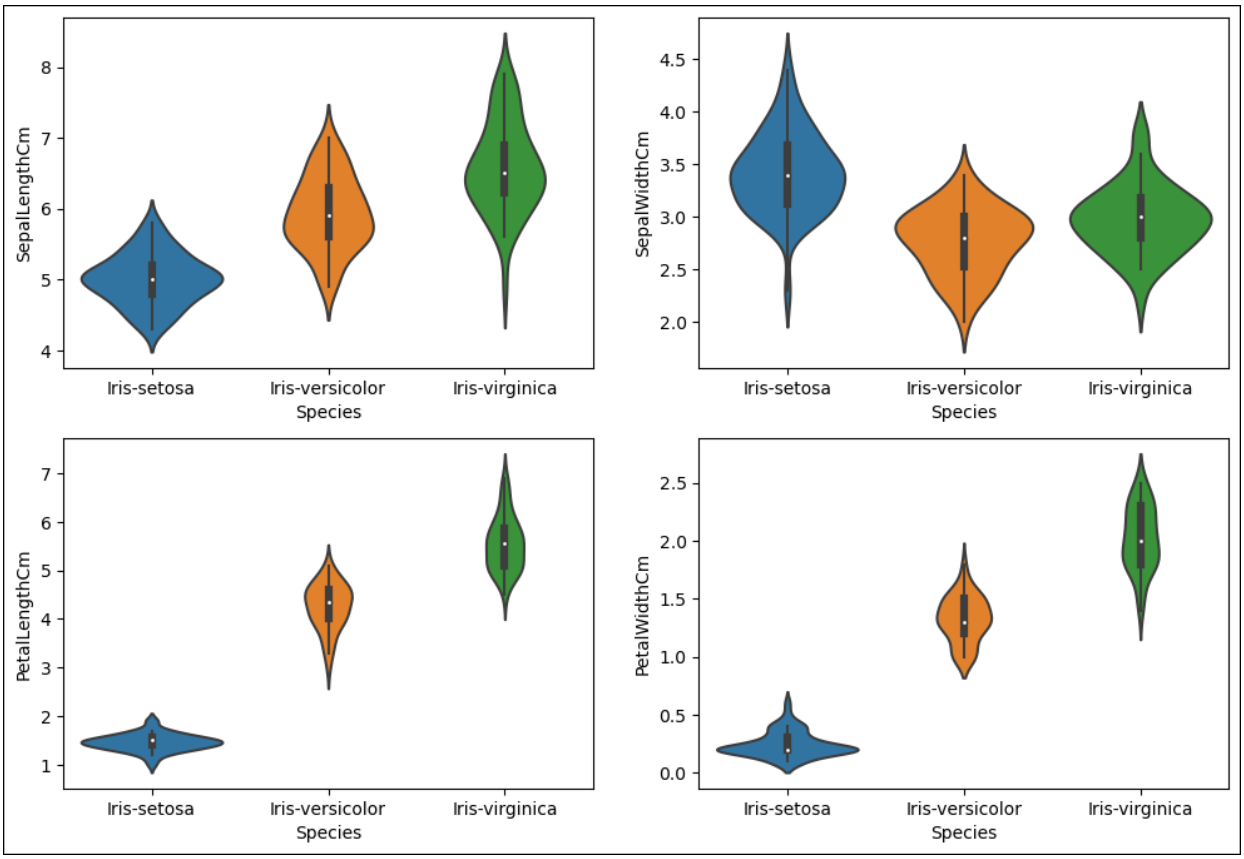
Creating a Pair Plot with color

In [25]:
```
# Let's try that again using color.  Notice: assigning a hue variable adds a semantic

sns.pairplot(df, hue='Species', height=3, aspect= 1);
```



Creating a Violin Plots

In [26]:
```
#  Please click on the URL above to learn more about Violin Plots

plt.figure(edgecolor="black", linewidth= 1.2,figsize=(12,8));
plt.subplot(2,2,1)
sns.violinplot(x='Species', y = 'SepalLengthCm', data=df)
plt.subplot(2,2,2)
sns.violinplot(x='Species', y = 'SepalWidthCm', data=df)
plt.subplot(2,2,3)
sns.violinplot(x='Species', y = 'PetalLengthCm', data=df)
plt.subplot(2,2,4)
sns.violinplot(x='Species', y = 'PetalWidthCm', data=df);
```



Separate the dataset into input and Outout Numpay Array

In [27]:
```
# store dataframe values into a numpy array

array = df.values

# separate array into input and output by slicing
# for X(input) [:, 1:5] --> all the rows, columns from 1 - 5
# these are the independent variables or predictors

X = array[:,1:5]

# for Y(input) [:, 5] --> all the rows, column 5
# this is the value we are trying to predict

Y = array[:,5]
```

Split into/Output Array into Training Or Testing Dataset.

In [28]:
```
# split the dataset --> training sub-dataset: 67%; test sub-dataset: 33%

test_size = 0.33

#selection of records to include in each data sub-dataset must be done randomly
```

```
seed = 7

#split the dataset (input and output) into training / test datasets

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=test_size,
random_state=seed)
```

Build and Train the Model

In [29]:
```
# build the model

model = LogisticRegression(random_state=seed, max_iter=1000)

# train the model using the training sub-dataset

model.fit(X_train, Y_train)

#print the classification report

predicted = model.predict(X_test)
report = classification_report(Y_test, predicted)
print("Classification Report: ", "\n", "\n",report)
```

```
Classification Report:

                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        14
Iris-versicolor       0.89      0.89      0.89        18
 Iris-virginica       0.89      0.89      0.89        18

       accuracy                           0.92        50
      macro avg       0.93      0.93      0.93        50
   weighted avg       0.92      0.92      0.92        50
```

Score the accuracy of the model

In [30]:
```
# score the accuracy level

result = model.score(X_test, Y_test)

#print out the results

print(("Accuracy: %.3f%%") % (result*100.0))
```

```
Accuracy: 92.000%
```

Classify or Prediction

In [31]:
```
model.predict([[5.3, 3.0, 4.5, 1.5]])
```

Out[31]:
```
array(['Iris-versicolor'], dtype=object)
```

In [32]:
```
model.predict([[5, 3.6, 1.4, 1.5]])
```

Out[32]:
```
array(['Iris-setosa'], dtype=object)
```

We have Trained the Model and used that trained model to predict the types of flower.

We have the listed values for each variable

Evaluate the Model useing the 10 fold cross - Validation Technique.

In [33]:
```
# Evaluate the algorithm and specify the number of times of repeated splitting, in thi

n_splits=10

#Fix the random seed.  You must use the same seed value so that the same subsets can b

seed=7

kfold=KFold(n_splits, random_state=seed, shuffle=True)

# for logistic regression, we can use the accuracy level to evaluate the model

scoring="accuracy"

#train the model and run K-fold cross validation to validate / evaluate the model

results=cross_val_score (model, X, Y, cv=kfold, scoring=scoring)

# print the evaluation results.  The result is the average of all the results obtainea

print("Accuracy: %.3f (%.3f)"% (results.mean(), results.std()))
```

```
Accuracy: 0.967 (0.054)
```

In [34]:
```
# Homework ADTA 5240 Module 5
```