Case Study :

Classification of Species using Logistic Regression by Yash Patel

What is Regression?

Regression analysis is a type of predictive modeling technique which is used to find the relationship between a dependent variable (usually known as the “Y” variable) and either one independent variable (the “X” variable) or a series of independent variables. When two or more independent variables are used to predict or explain the outcome of the dependent variable, this is known as multiple regression.

What is Logistic Regression?

Logistic regression is a classification algorithm. It is used to predict a binary outcome based on a set of independent variables.

Import Library :

Pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.(https://pandas.pydata.org/docs/ (https://pandas.pydata.org/docs/))

In [1]:

```
import pandas as pd
```

Accessing the Data :

Data :

We have a set of data which we will be use for prediction and classification of species. We has 4 entities to look, observe, analyse the data. Namely Sepal Length and width; and Petal Length and width.

In [2]:

```
dataset = pd.read_csv('species_data.csv')
```

Reading Data :

In [3]:

```
dataset.shape
```

Out[3]:

```
(150, 6)
```

Displaying Data :

Set of First Five Data :

In [4]:

```
dataset.head()
```

Out[4]:

	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

Exploration Of Data :

Importing Library :

Matplotlib :

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.(https://matplotlib.org/contents.html# (https://matplotlib.org/contents.html#))

In [5]:

```
import matplotlib.pyplot as plt
```

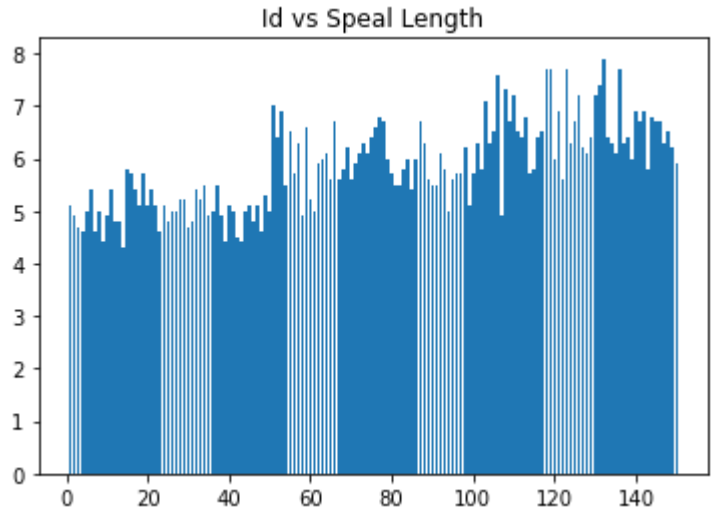
Plotting Graph :

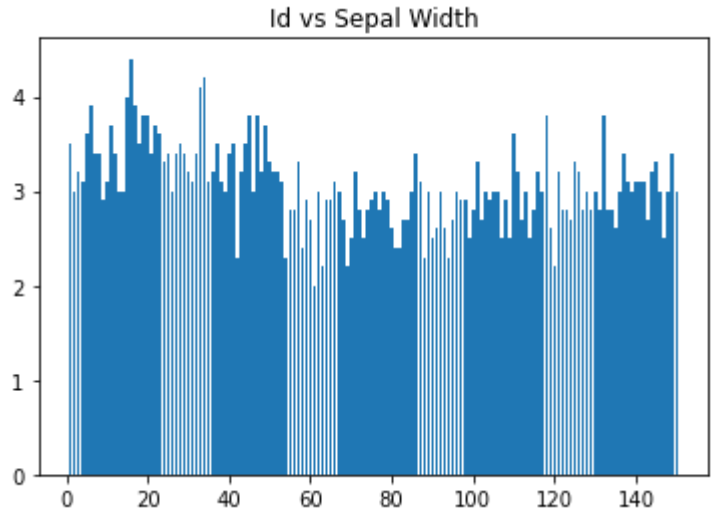
Bar Graph :

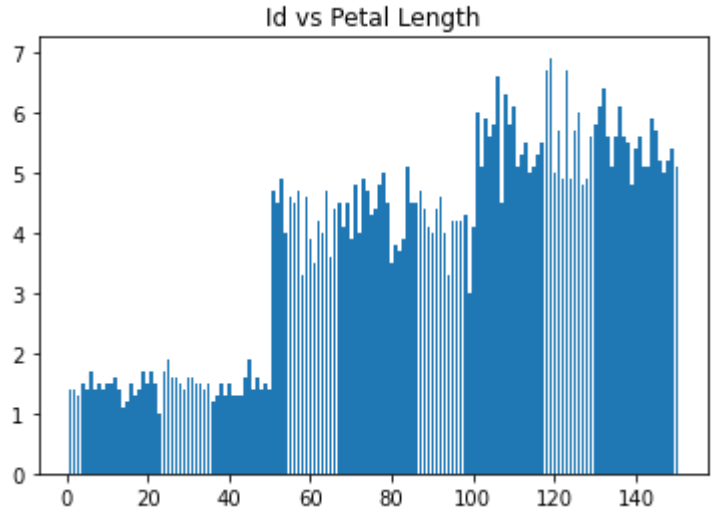
Here we are plotting a Bar Graph for various species against there Speal and Petal properties using there length and width. These criteria expresses us the value which classifies the group of data for specific species.

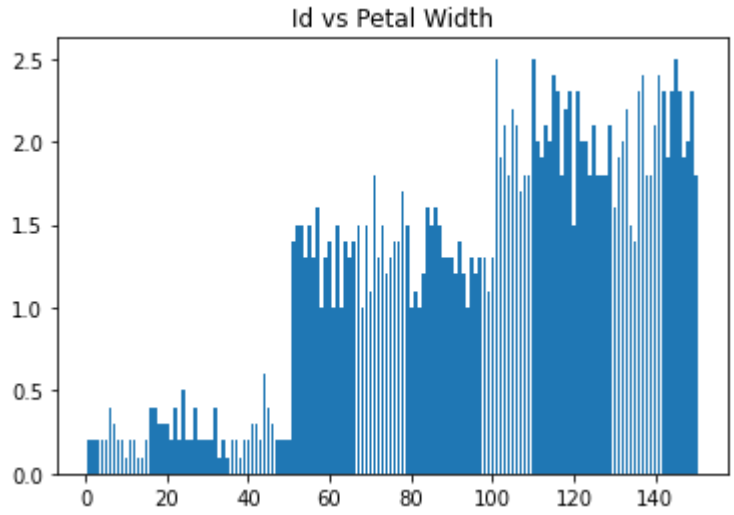
In [6]:

```
plt.bar(x = dataset['Id'] , height = dataset['SepalLengthCm'])
plt.title('Id vs Speal Length')
plt.show()

plt.bar(x = dataset['Id'], height = dataset['SepalWidthCm'])
plt.title('Id vs Sepal Width')
plt.show()

plt.bar(x = dataset['Id'] , height = dataset['PetalLengthCm'])
plt.title('Id vs Petal Length')
plt.show()

plt.bar(x = dataset['Id'], height = dataset['PetalWidthCm'])
plt.title('Id vs Petal Width')
plt.show()
```


Id vs Speal Length
0
1
2
3
4
5
6
7
8
0
20
40
60
80
100
120
140


Id vs Sepal Width
0
1
2
3
4
0
20
40
60
80
100
120
140


Id vs Petal Length
0
1
2
3
4
5
6
7
0
20
40
60
80
100
120
140

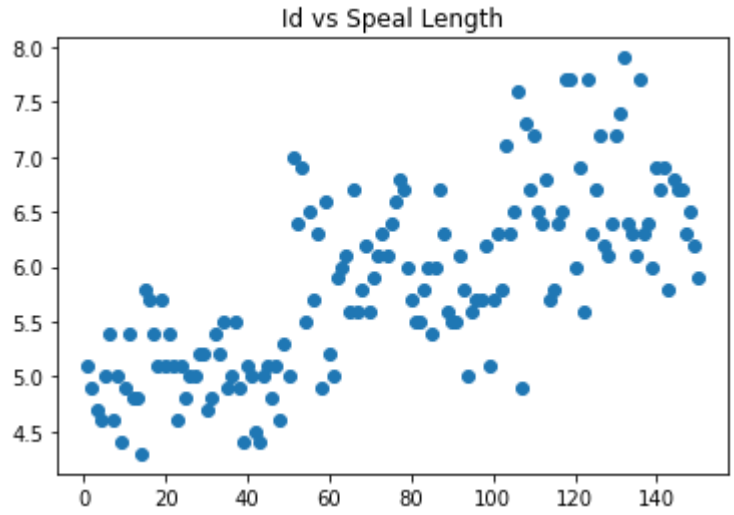


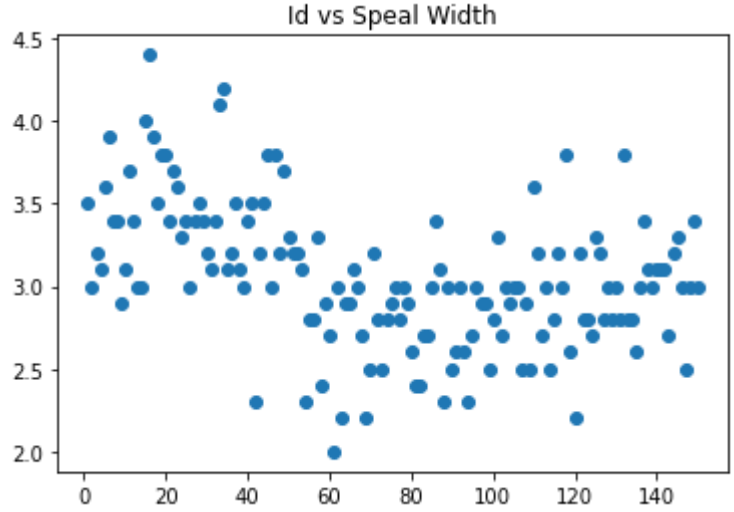
Scatter Graph :

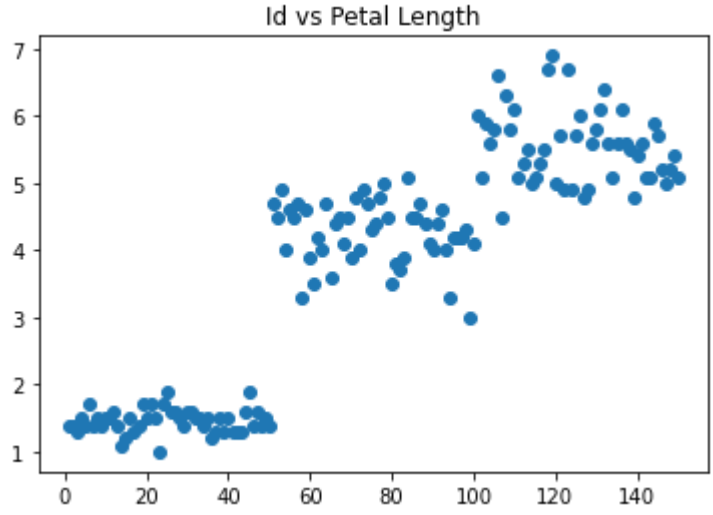
Scatter plots are used to plot data points on horizontal and vertical axis in the attempt to show how much one variable is affected by another. Each row in the data table is represented by a marker the position depends on its values in the columns set on the X and Y axes. A third variable can be set to correspond to the color or size of the markers, thus adding yet another dimension to the plot.

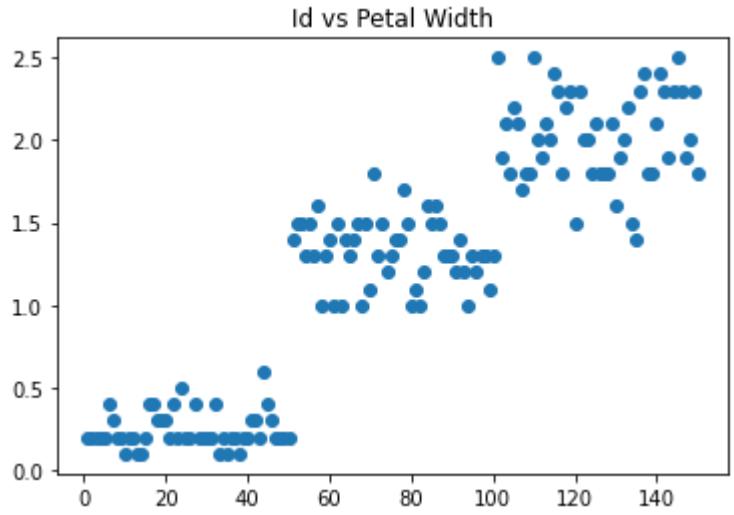
In [7]:

```
plt.scatter(dataset.Id , dataset.SepalLengthCm)
plt.title('Id vs Speal Length')
plt.show()

plt.scatter(dataset.Id , dataset.SepalWidthCm)
plt.title('Id vs Speal Width')
plt.show()

plt.scatter(dataset.Id , dataset.PetalLengthCm)
plt.title('Id vs Petal Length')
plt.show()

plt.scatter(dataset.Id , dataset.PetalWidthCm)
plt.title('Id vs Petal Width')
plt.show()
```


Id vs Speal Length
8.0
7.5
7.0
6.5
6.0
5.5
5.0
4.5
0
20
40
60
80
100
120
140


Id vs Speal Width
4.5
4.0
3.5
3.0
2.5
2.0
0
20
40
60
80
100
120
140


Id vs Petal Length
7
6
5
4
3
2
1
0
20
40
60
80
100
120
140



Box Plot Graph :

A box plot is a way of summarizing a set of data measured on an interval scale. It is often used in explanatory data analysis. This type of graph is used to show the shape of the distribution, its central value, and its variability.

A Box Plot is the visual representation of the statistical five number summary of a given data set.

A Five Number Summary includes:

Minimum

First Quartile

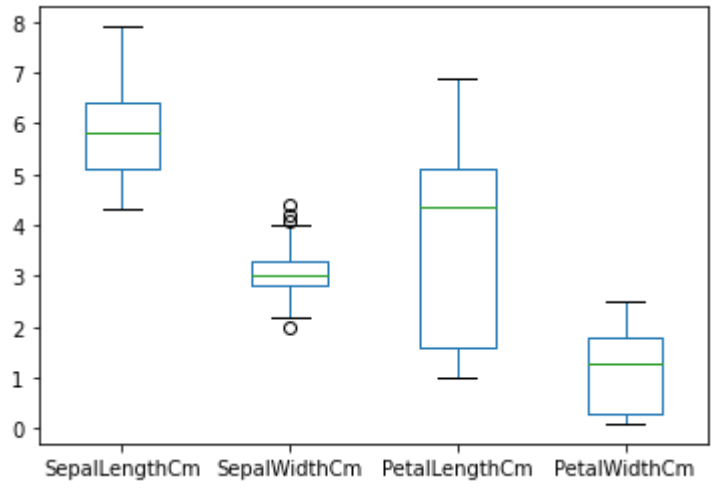
Median (Second Quartile)

Third Quartile

Maximum

In [8]:

```
box_plot = dataset[['SepalLengthCm','SepalWidthCm','PetalLengthCm','PetalWidthCm']]
box_plot.plot(kind='box')
plt.show()
```



Data Cleaning :

In [9]:

```
dataset.isna().sum()
```

Out[9]:

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

No missing Value

Data Column :

In [10]:

```
dataset.columns
```

Out[10]:

```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidth
Cm',
       'Species'],
      dtype='object')
```

Division of Data (Features and Labels) :

In [11]:

```
x = dataset.iloc[:, 1:-1].values
y = dataset.iloc[:, -1].values
print(x[:10])
print(y[:10])
```

```
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]
 [5.4 3.9 1.7 0.4]
 [4.6 3.4 1.4 0.3]
 [5.  3.4 1.5 0.2]
 [4.4 2.9 1.4 0.2]
 [4.9 3.1 1.5 0.1]]
['Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa'
 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa']
```

Splitting the data into training and test data :

Import Library :

In [12]:

```
from sklearn.model_selection import train_test_split
```

Splitting Data :

Split the Data Into Training and Test Sets In this step we will split our dataset into training and testing subsets (in proportion 80/20%). Training data set will be used for training of our model. Testing dataset will be used for validating of the model.we may check how accurate are model predictions.

In [13]:

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state
= 0)
```

Scaling the data :

Import Library :

In [14]:

```
from sklearn.preprocessing import StandardScaler
```

Scaling Data :

In [15]:

```
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

Displaying Scaled Data:

In [16]:

```
print(x_train[:10])
```

```
[[ 0.61303014  0.10850105  0.94751783  0.73603967]
 [-0.56776627 -0.12400121  0.38491447  0.34808318]
 [-0.80392556  1.03851009 -1.30289562 -1.3330616 ]
 [ 0.25879121 -0.12400121  0.60995581  0.73603967]
 [ 0.61303014 -0.58900572  1.00377816  1.25331499]
 [-0.80392556 -0.82150798  0.04735245  0.21876435]
 [-0.21352735  1.73601687 -1.19037495 -1.20374277]
 [ 0.14071157 -0.82150798  0.72247648  0.47740201]
 [ 0.02263193 -0.12400121  0.21613346  0.34808318]
 [-0.09544771 -1.05401024  0.10361279 -0.03987331]]
```

Implementation of Logistic Regression :

Import Library :

Scikit is a library which is :

Simple and efficient tools for predictive data analysis.

Accessible to everybody, and reusable in various contexts.

Built on NumPy, SciPy, and matplotlib.

Open source, commercially usable - BSD license.

In [17]:

```
from sklearn.linear_model import LogisticRegression
```

Method Implementation :

Predicting a continuous-valued attribute associated with an object. A method of Logistic Regression is used.

In [18]:

```
classifier = LogisticRegression(multi_class='ovr', random_state = 0)
classifier.fit(x_train, y_train)
```

Out[18]:

```
LogisticRegression(multi_class='ovr', random_state=0)
```

Prediction / Classification :

Importing library :

In [19]:

```
from sklearn.metrics import confusion_matrix, accuracy_score
```

Classification Accuracy :

In [20]:

```
predictions = classifier.predict(x_test)
cm = confusion_matrix(predictions, y_test)
print(cm)
accuracy_score(predictions, y_test)
```

```
[[11  0  0]
 [ 0 11  1]
 [ 0  2  5]]
```

Out[20]:

```
0.9
```

Hence the we can state that the data is 90% Accurate.