Experiment No. 4

Aim: To implement Bayesian algorithm.

Requirements: Windows O.S, Weka tool, Python and Python libraries: Pandas, Numpy, Sklearn and Matplot.

Problem Statement: To implement Naïve Bayes algorithm on iris data set using Weka tool and Python.

Theory:

Naive Bayes: Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable. Bayes' theorem states the following relationship, given class variable y and dependent feature vector x1 through xn, :

$$P(y|x1,\ldots,xn) = P(y)P(x1,\ldots,xn|y)/P(x1,\ldots,xn)$$

Using the naive conditional independence assumption that

$$P(xi|y,x1,\ldots,xi-1,xi+1,\ldots,xn) = P(xi|y),$$

for all i, this relationship is simplified to

$$P(y|x1,\ldots,xn) = P(y)\prod i=1nP(xi|y)/P(x1,\ldots,xn)$$

Since P(x1,…,xn) is constant given the input, we can use the following classification rule:

$$P(y|x1,\ldots,xn) \propto P(y)\prod i=1nP(xi|y)\Downarrow$$

$$y\hat{} = \text{argmax}(y)(P(y)\prod i=1nP(xi|y)),$$

and we can use Maximum A Posteriori (MAP) estimation to estimate P(y) and P(xi|y); the former is then the relative frequency of class y in the training set.

The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of P(xi|y).

In spite of their apparently over-simplified assumptions, naive Bayes classifiers have worked quite well in many real-world situations, famously document classification and spam filtering. They require a small amount of training data to estimate the necessary parameters. (For theoretical reasons why naive Bayes works well, and on which types of data it does, see the references below.)

Naive Bayes learners and classifiers can be extremely fast compared to more sophisticated methods. The decoupling of the class conditional feature distributions means that each distribution can be independently estimated as a one dimensional distribution. This in turn helps to alleviate problems stemming from the curse of dimensionality.

Gaussian Naive Bayes: When working with continuous data, an assumption often taken is that the continuous values associated with each class are distributed according to a normal (or Gaussian) distribution. The likelihood of the features is assumed to be-

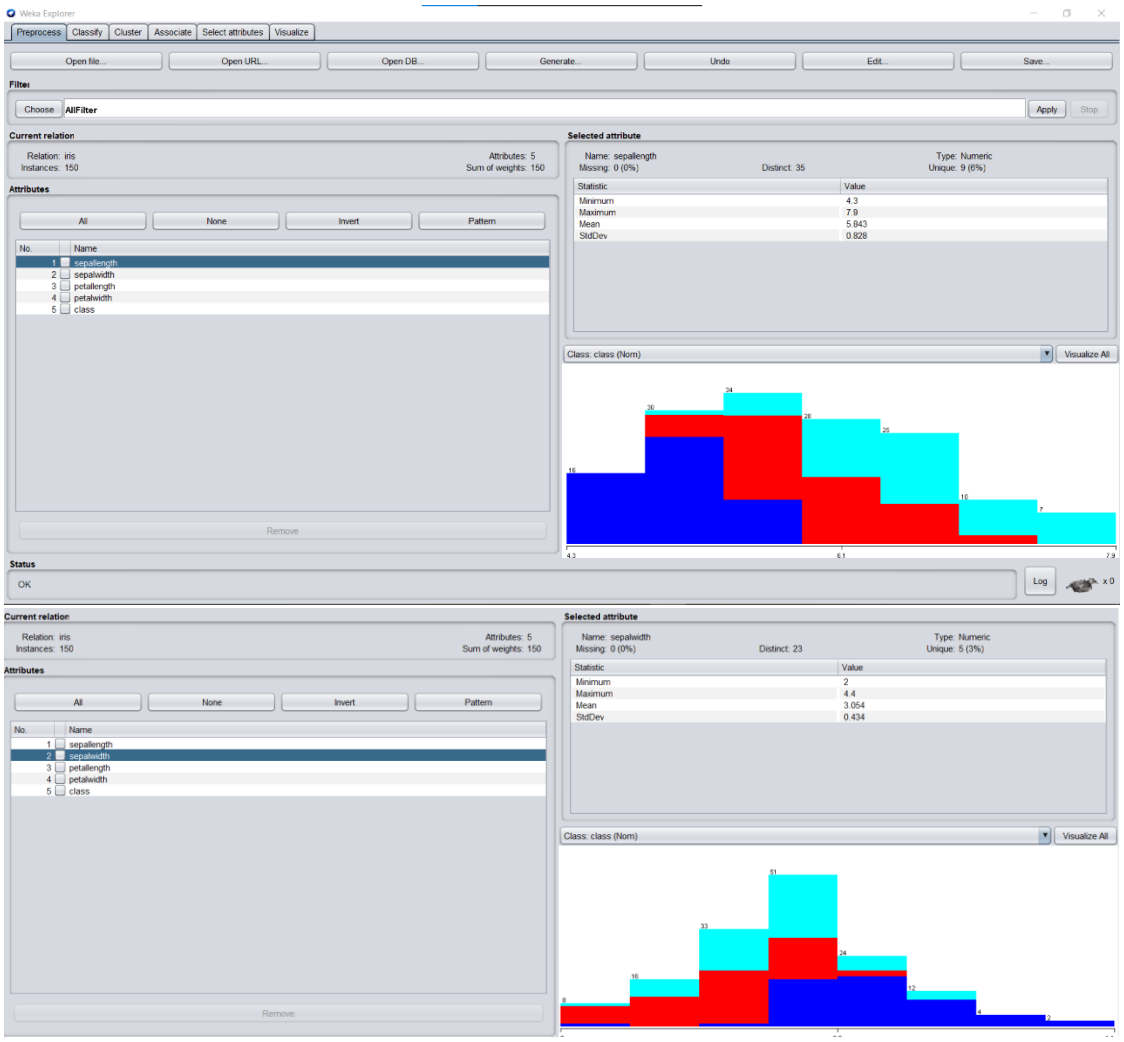
$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

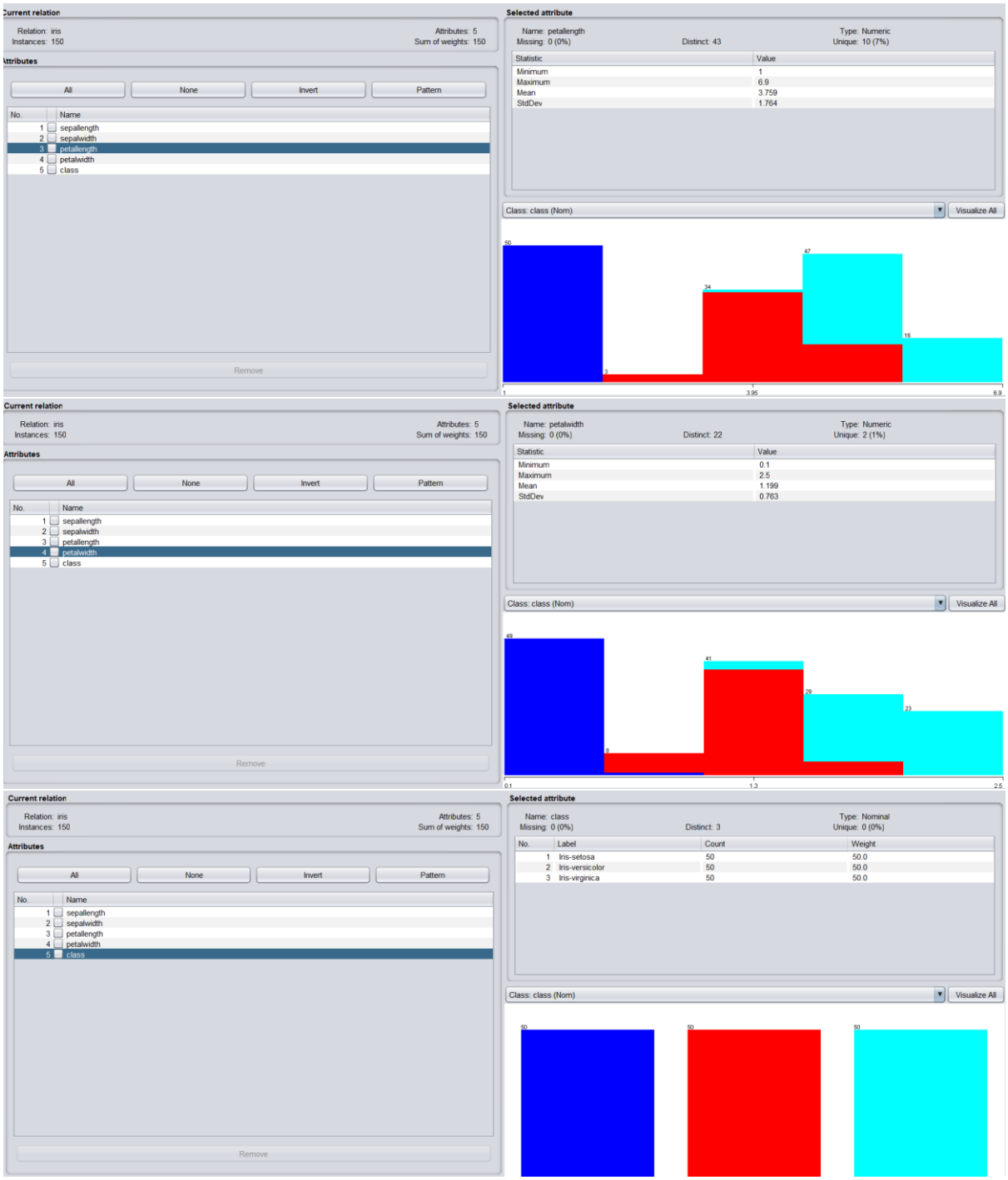
Sometimes assume variance

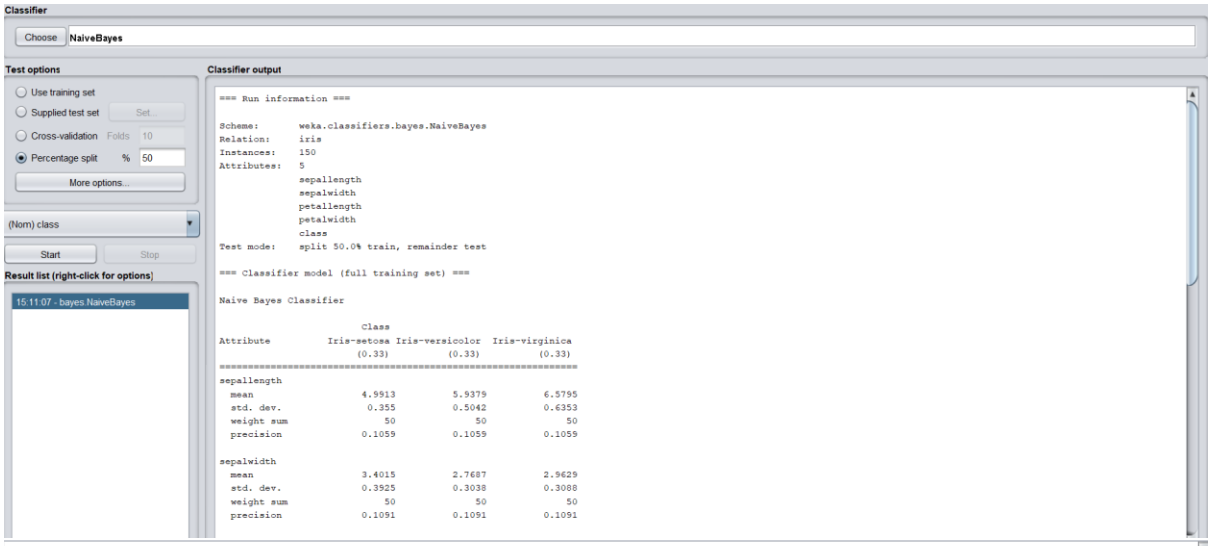
- is independent of Y (i.e., σi),
- or independent of Xi (i.e., σk)
- or both (i.e., σ)

Gaussian Naive Bayes supports continuous valued features and models each as conforming to a Gaussian (normal) distribution.

Weka tool output:




Current relation
Relation: iris
Instances: 150
Attributes: 5
Sum of weights: 150
Attributes
All
None
Invert
Pattern
No.
Name
1 sepallength
2 sepalwidth
3 petallength
4 petalwidth
5 class
Remove
Selected attribute
Name: petallength
Missing: 0 (0%)
Distinct: 43
Type: Numeric
Unique: 10 (7%)
Statistic
Value
Minimum 1
Maximum 6.9
Mean 3.759
StdDev 1.764
Class: class (Nom)
Visualize All
50
3
24
47
18
1
3.95
6.9
Current relation
Relation: iris
Instances: 150
Attributes: 5
Sum of weights: 150
Attributes
All
None
Invert
Pattern
No.
Name
1 sepallength
2 sepalwidth
3 petallength
4 petalwidth
5 class
Remove
Selected attribute
Name: petalwidth
Missing: 0 (0%)
Distinct: 22
Type: Numeric
Unique: 2 (1%)
Statistic
Value
Minimum 0.1
Maximum 2.5
Mean 1.199
StdDev 0.763
Class: class (Nom)
Visualize All
49
8
41
29
23
0.1
1.3
2.5
Current relation
Relation: iris
Instances: 150
Attributes: 5
Sum of weights: 150
Attributes
All
None
Invert
Pattern
No.
Name
1 sepallength
2 sepalwidth
3 petallength
4 petalwidth
5 class
Remove
Selected attribute
Name: class
Missing: 0 (0%)
Distinct: 3
Type: Nominal
Unique: 0 (0%)
No.
Label
Count
Weight
1 Iris-setosa 50 50.0
2 Iris-versicolor 50 50.0
3 Iris-virginica 50 50.0
Class: class (Nom)
Visualize All
50
50
50




```
=== Run information ===

Scheme:       weka.classifiers.bayes.NaiveBayes
Relation:     iris
Instances:    150
Attributes:   5
              sepallength
              sepalwidth
              petallength
              petalwidth
              class
Test mode:    split 50.0% train, remainder test

=== Classifier model (full training set) ===

Naive Bayes Classifier

                          Class
Attribute          Iris-setosa Iris-versicolor  Iris-virginica
                        (0.33)          (0.33)          (0.33)
============================================================
sepallength
  mean                 4.9913          5.9379          6.5795
  std. dev.             0.355          0.5042          0.6353
  weight sum               50              50              50
  precision            0.1059          0.1059          0.1059

sepalwidth
  mean                 3.4015          2.7687          2.9629
  std. dev.            0.3925          0.3038          0.3088
  weight sum               50              50              50
  precision            0.1091          0.1091          0.1091

petallength
  mean                 1.4694          4.2452          5.5516
  std. dev.            0.1782          0.4712          0.5529
  weight sum               50              50              50
  precision            0.1405          0.1405          0.1405

petalwidth
  mean                 0.2743          1.3097          2.0343
  std. dev.            0.1096          0.1915          0.2646
  weight sum               50              50              50
  precision            0.1143          0.1143          0.1143



Time taken to build model: 0 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0 seconds

=== Summary ===

Correctly Classified Instances          72               96      %
Incorrectly Classified Instances         3                4      %
Kappa statistic                          0.9398
Mean absolute error                      0.0336
Root mean squared error                  0.1606
Relative absolute error                  7.5238 %
Root relative squared error             33.8795 %
Total Number of Instances               75

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Iris-setosa
                 1.000    0.061    0.897      1.000    0.945      0.917    0.987     0.975     Iris-versicolor
                 0.889    0.000    1.000      0.889    0.941      0.915    0.988     0.982     Iris-virginica
Weighted Avg.    0.960    0.021    0.964      0.960    0.960      0.941    0.991     0.985

=== Confusion Matrix ===

  a  b  c   <-- classified as
 22  0  0 |  a = Iris-setosa
  0 26  0 |  b = Iris-versicolor
  0  3 24 |  c = Iris-virginica
```

Python output:

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix
```

In [2]:
```
data = load_iris()
X,Y = load_iris(return_X_y=True)
```

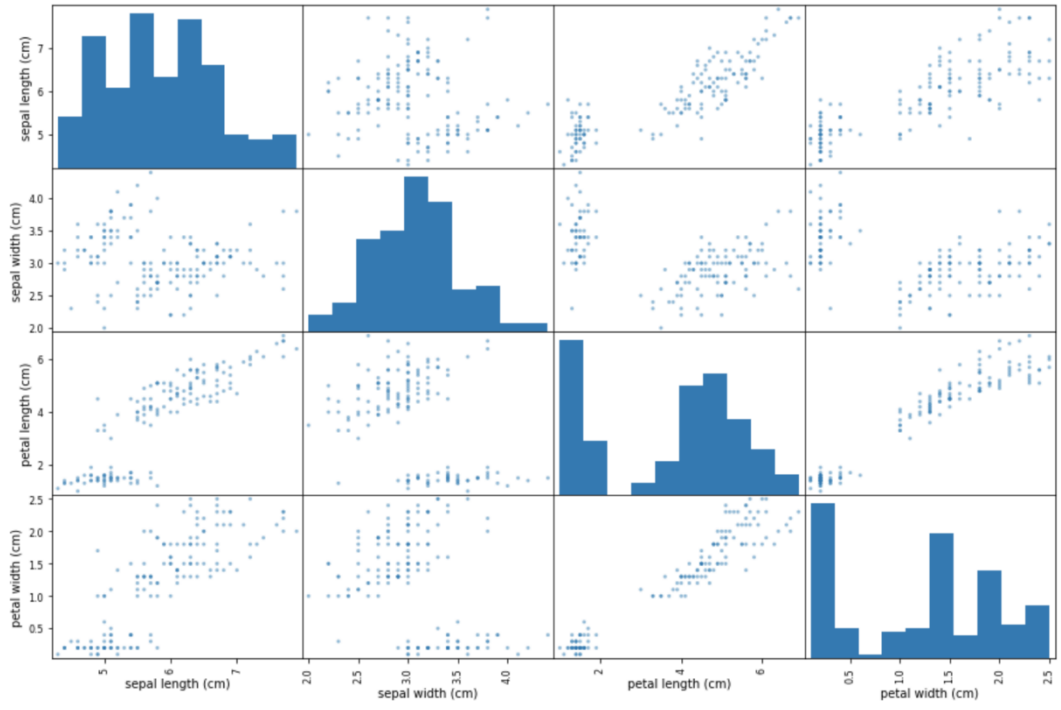
In [3]:
```
iris = pd.DataFrame(data = np.column_stack((data.data,data.target)),columns = [*data.feature_names,"Class"])
iris.Class.replace([0,1,2],['setosa', 'versicolor', 'virginica'],inplace = True)
iris
```

Out[3]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	Class
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

In [4]:
```
pd.plotting.scatter_matrix(iris,figsize=(15,10),diagonal="hist")
plt.show()
```



In [6]:
```
iris.groupby("Class").describe()
```

Out[6]:

	sepal length (cm)								sepal width (cm)		...	petal length (cm)		petal width (cm)						
	count	mean	std	min	25%	50%	75%	max	count	mean	...	75%	max	count	mean	std	min	25%	50%	75%	max
Class																					
setosa	50.0	5.006	0.352490	4.3	4.800	5.0	5.2	5.8	50.0	3.428	...	1.575	1.9	50.0	0.246	0.105386	0.1	0.2	0.2	0.3	0.6
versicolor	50.0	5.936	0.516171	4.9	5.600	5.9	6.3	7.0	50.0	2.770	...	4.600	5.1	50.0	1.326	0.197753	1.0	1.2	1.3	1.5	1.8
virginica	50.0	6.588	0.635880	4.9	6.225	6.5	6.9	7.9	50.0	2.974	...	5.875	6.9	50.0	2.026	0.274650	1.4	1.8	2.0	2.3	2.5

3 rows × 32 columns

In [7]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,train_size=0.5,random_state=0)
```

In [8]:
```
nav_bay = GaussianNB()
model = nav_bay.fit(X_train,Y_train)
y_pred = model.predict(X_test)
```

In [9]:
```
prediction = pd.DataFrame(np.column_stack((X_test,Y_test,y_pred)),
                          columns = [*data.feature_names,"Original_Class","Predicted_Class"])
prediction.Original_Class.replace([0,1,2],['setosa', 'versicolor', 'virginica'],inplace = True)
prediction.Predicted_Class.replace([0,1,2],['setosa', 'versicolor', 'virginica'],inplace = True)
prediction
```

Out[9]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	Original_Class	Predicted_Class
0	5.8	2.8	5.1	2.4	virginica	virginica
1	6.0	2.2	4.0	1.0	versicolor	versicolor
2	5.5	4.2	1.4	0.2	setosa	setosa
3	7.3	2.9	6.3	1.8	virginica	virginica
4	5.0	3.4	1.5	0.2	setosa	setosa
...	...	...	...	...	...	...
70	6.4	2.7	5.3	1.9	virginica	virginica
71	5.7	3.0	4.2	1.2	versicolor	versicolor
72	5.4	3.4	1.7	0.2	setosa	setosa
73	5.7	4.4	1.5	0.4	setosa	setosa
74	6.9	3.1	4.9	1.5	versicolor	versicolor

In [10]:
```
accuracy = prediction.loc[prediction["Original_Class"] == prediction["Predicted_Class"]].count()[0]/prediction.count()[0]
print(f"Accuracy of model = {accuracy*100}%")
```

```
Accuracy of model = 94.66666666666667%
```

In [12]:
```
conf_mat = pd.DataFrame(confusion_matrix(prediction["Original_Class"],prediction["Predicted_Class"]),
                        index = ['setosa', 'versicolor', 'virginica'],columns = ['setosa', 'versicolor', 'virginica'])
conf_mat.index.name = "Original Class"
conf_mat
```

Out[12]:

Original Class	setosa	versicolor	virginica
setosa	21	0	0
versicolor	0	30	0
virginica	0	4	20

Conclusion: We have successfully implemented Bayes Algorithm on iris data set using Weka tool and Python libraries.