Institute of Computer Technology
B. Tech Computer Science and Engineering
Sub: Data Mining and Warehousing (2CSE60E27)

PRACTICAL 8: NAÏVE BAYES

The Iris flower data set is a multivariate data set introduced by the British statistician and biologist Ronald Fisher in his 1936 paper 'The use of multiple measurements in taxonomic problems'. It is sometimes called Anderson's Iris data set because Edgar Anderson collected the data to quantify the morphologic variation of Iris flowers of three related species. The data set consists of 50 samples from each of three species of Iris (Iris Setosa, Iris virginica, and Iris versicolor). Four features were measured from each sample: the length and the width of the sepals and petals, in centimeters.

This dataset became a typical test case for many statistical classification techniques. Kindly implement the Naive Bayes Classification for famous Iris Flower. Dataset that consists of 3 classes of flowers. In this, there are 4 independent variables namely the, sepal_length, sepal_width, petal_length and petal_width. The dependent variable is the species which we will predict using the four independent features of the flowers.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

dataset = pd.read_csv(r'C:\Users\admin\Desktop\dishwa\dmw\Practical 8\iris.csv')
X = dataset.iloc[:,:4]. values
y = dataset['species' ]. values
dataset.head(5)

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train , X_test , y_train , y_test = train_test_split(X,y,test_size=0.3)

# Feature Scaling
from sklearn. preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc. fit_transform(X_train)
X_test = sc. transform(X_test)

# Training the Naive Bayes Classification model on the Training Set
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier. predict(X_test)
```

```
y_pred

# Confusion Matrix
from sklearn. metrics import confusion_matrix
import seaborn as sns
cm = confusion_matrix(y_test, y_pred)
cm_df=pd.DataFrame(cm,index=['Iris-setosa','Iris-versicolor','Iris-virginica'],columns=['Iris-setosa','Iris-versicolor','Iris-virginica'])
sns.heatmap(cm_df, annot=True)
plt.title('Confusion Matrix')
plt.ylabel('Actual Values')
plt.xlabel('Predicted Values')
plt.show()

from sklearn. metrics import accuracy_score
print ("Accuracy : ", accuracy_score(y_test, y_pred))

# Comparing the Real Values with Predicted Values
df = pd. DataFrame({'Real Values' : y_test, 'Predicted Values' : y_pred})
df

dataset

temp = dataset.loc[dataset['species']=='Iris-virginica']
temp_1 = dataset.loc[dataset['species']=='Iris-setosa']
temp_2 = dataset.loc[dataset['species']=='Iris-versicolor']
plt.scatter(temp['sepal_length'],temp['sepal_width'])
plt.scatter(temp_1['sepal_length'],temp_1['sepal_width'])
plt.scatter(temp_2['sepal_length'],temp_2['sepal_width'])
plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')
plt.legend(['Iris-virginica','Iris-setosa','Iris-versicolor'])
plt.show()

temp = dataset.loc[dataset['species']=='Iris-virginica']
temp_1 = dataset.loc[dataset['species']=='Iris-setosa']
temp_2 = dataset.loc[dataset['species']=='Iris-versicolor']
plt.scatter(temp['petal_length'],temp['petal_width'])
plt.scatter(temp_1['petal_length'],temp_1['petal_width'])
plt.scatter(temp_2['petal_length'],temp_2['petal_width'])
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
plt.legend(['Iris-virginica','Iris-setosa','Iris-versicolor'])
plt.show()

plt.scatter(df['Real Values']=='Iris-virginica',df['Predicted Values']=='Iris-virginica')
plt.show()
plt.scatter(df['Real Values']=='Iris-setosa',df['Predicted Values']=='Iris-setosa')
```

```
plt.show()
plt.scatter(df['Real Values']=='Iris-versicolor',df['Predicted Values']=='Iris-versicolor')
plt.show()

plt.figure(figsize=(6,6))
plt.scatter(df['Real Values'], df['Predicted Values'], c='crimson')
plt.yscale('log')
plt.xscale('log')

p1 = max(max(df['Predicted Values']), max(df['Real Values']))
p2 = min(min(df['Predicted Values']), min(df['Real Values']))
print(p1)
print(p2)

plt.plot([p1, p2], [p1, p2], 'b-')
plt.xlabel('True Values', fontsize=15)
plt.ylabel('Predictions', fontsize=15)
plt.axis('equal')
plt.show()
```

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [3]:
```
dataset = pd.read_csv(r'C:\Users\admin\Desktop\dishwa\dmw\Practical 8\iris.csv')
X = dataset.iloc[:,:4]. values
y = dataset['species' ]. values
dataset.head(5)
```

Out[3]:

	sepal_length	sepal_width	petal_length	petal_width	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

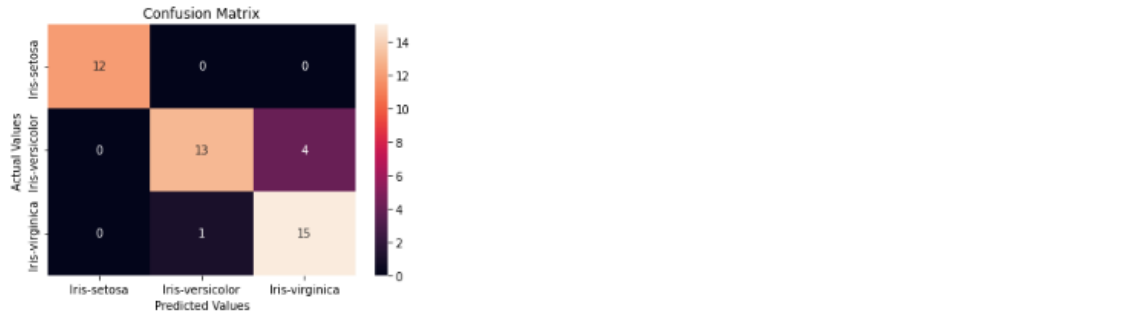
In [19]:
```
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train , X_test , y_train , y_test = train_test_split(X,y,test_size=0.3)

# Feature Scaling
from sklearn. preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc. fit_transform(X_train)
X_test = sc. transform(X_test)

# Training the Naive Bayes Classification model on the Training Set
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)
```

Out[19]:
```
GaussianNB()
```

In [7]:
```
# Predicting the Test set results
y_pred = classifier. predict(X_test)
y_pred
```

Out[7]:
```
array(['Iris-versicolor', 'Iris-versicolor', 'Iris-virginica',
       'Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-setosa',
       'Iris-versicolor', 'Iris-virginica', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-setosa', 'Iris-virginica',
       'Iris-virginica', 'Iris-virginica', 'Iris-versicolor',
       'Iris-virginica', 'Iris-virginica', 'Iris-setosa',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-virginica',
       'Iris-virginica', 'Iris-setosa', 'Iris-virginica',
       'Iris-virginica', 'Iris-versicolor', 'Iris-virginica',
       'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
       'Iris-setosa', 'Iris-virginica', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa',
       'Iris-virginica', 'Iris-virginica', 'Iris-setosa',
       'Iris-virginica', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-versicolor'], dtype='<U15')
```

In [8]:
```
# Confusion Matrix
from sklearn. metrics import confusion_matrix
import seaborn as sns
cm = confusion_matrix(y_test, y_pred)
cm_df=pd.DataFrame(cm,index=['Iris-setosa','Iris-versicolor','Iris-virginica'],columns=['Iris-setosa','Iris-versicolor','Iris-v
sns.heatmap(cm_df, annot=True)
plt.title('Confusion Matrix')
plt.ylabel('Actual Values')
plt.xlabel('Predicted Values')
plt.show()
```



In [9]:
```
from sklearn. metrics import accuracy_score
print ("Accuracy : ", accuracy_score(y_test, y_pred))
```

```
Accuracy :  0.8888888888888888
```

In [10]:
```
# Comparing the Real Values with Predicted Values
df = pd. DataFrame({'Real Values' : y_test, 'Predicted Values' : y_pred})
df
```

Out[10]:

	Real Values	Predicted Values
0	Iris-versicolor	Iris-versicolor
1	Iris-versicolor	Iris-versicolor
2	Iris-virginica	Iris-virginica
3	Iris-setosa	Iris-setosa
4	Iris-setosa	Iris-setosa
5	Iris-versicolor	Iris-versicolor
6	Iris-setosa	Iris-setosa
7	Iris-versicolor	Iris-versicolor
8	Iris-virginica	Iris-virginica
9	Iris-versicolor	Iris-versicolor
10	Iris-versicolor	Iris-versicolor
11	Iris-setosa	Iris-setosa
12	Iris-versicolor	Iris-virginica
13	Iris-virginica	Iris-virginica
14	Iris-versicolor	Iris-virginica
15	Iris-versicolor	Iris-versicolor
16	Iris-virginica	Iris-virginica
17	Iris-virginica	Iris-virginica

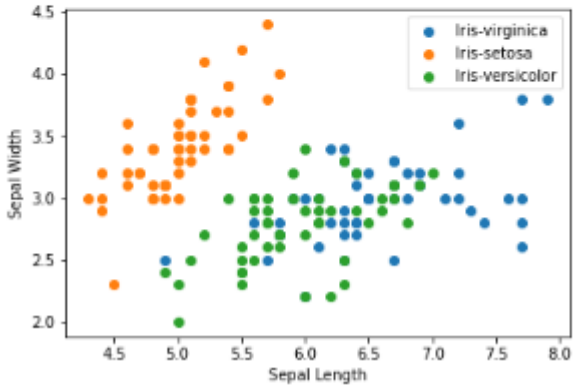
In [11]:
```
dataset
```

Out[11]:

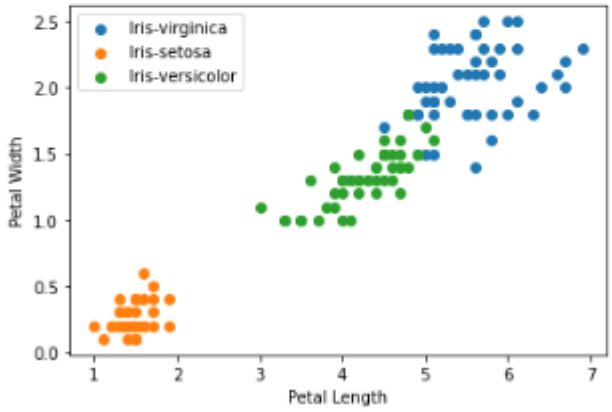
	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
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 × 5 columns

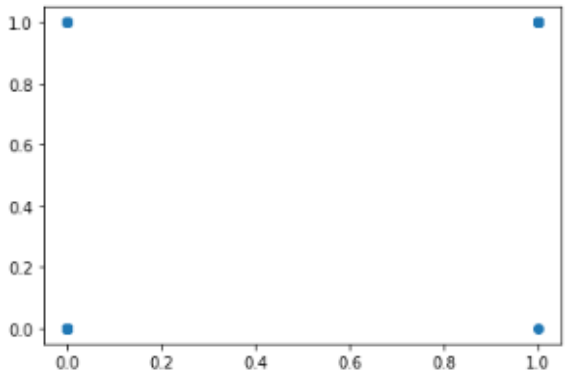
In [12]:
```
temp = dataset.loc[dataset['species']=='Iris-virginica']
temp_1 = dataset.loc[dataset['species']=='Iris-setosa']
temp_2 = dataset.loc[dataset['species']=='Iris-versicolor']
plt.scatter(temp['sepal_length'],temp['sepal_width'])
plt.scatter(temp_1['sepal_length'],temp_1['sepal_width'])
plt.scatter(temp_2['sepal_length'],temp_2['sepal_width'])
plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')
plt.legend(['Iris-virginica','Iris-setosa','Iris-versicolor'])
plt.show()
```



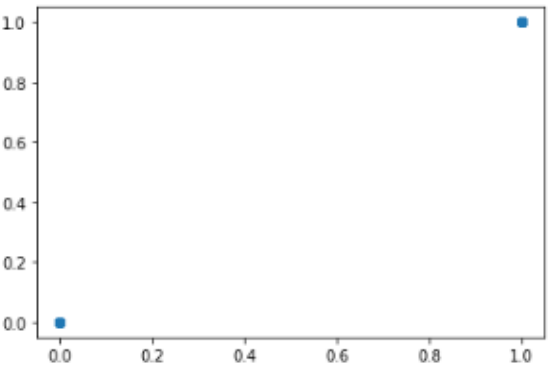
In [13]:
```
temp = dataset.loc[dataset['species']=='Iris-virginica']
temp_1 = dataset.loc[dataset['species']=='Iris-setosa']
temp_2 = dataset.loc[dataset['species']=='Iris-versicolor']
plt.scatter(temp['petal_length'],temp['petal_width'])
plt.scatter(temp_1['petal_length'],temp_1['petal_width'])
plt.scatter(temp_2['petal_length'],temp_2['petal_width'])
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
plt.legend(['Iris-virginica','Iris-setosa','Iris-versicolor'])
plt.show()
```



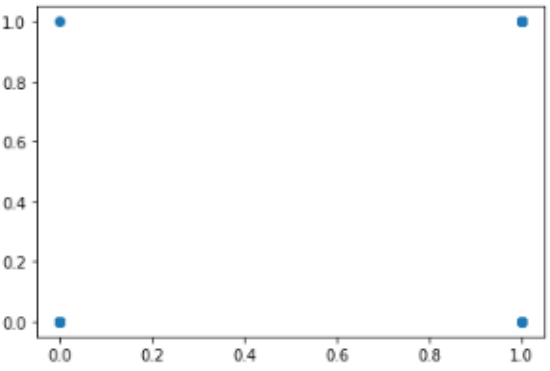
In [15]:
```
plt.scatter(df['Real Values']=='Iris-virginica',df['Predicted Values']=='Iris-virginica')
plt.show()
```



In [16]:
```
plt.scatter(df['Real Values']=='Iris-setosa',df['Predicted Values']=='Iris-setosa')
plt.show()
```



In [17]:
```
plt.scatter(df['Real Values']=='Iris-versicolor',df['Predicted Values']=='Iris-versicolor')
plt.show()
```



In [21]:
```
plt.figure(figsize=(6,6))
plt.scatter(df['Real Values'], df['Predicted Values'], c='crimson')
plt.yscale('log')
plt.xscale('log')

p1 = max(max(df['Predicted Values']), max(df['Real Values']))
p2 = min(min(df['Predicted Values']), min(df['Real Values']))
print(p1)
print(p2)

plt.plot([p1, p2], [p1, p2], 'b-')
plt.xlabel('True Values', fontsize=15)
plt.ylabel('Predictions', fontsize=15)
plt.axis('equal')
plt.show()
```

```
Iris-virginica
Iris-setosa
```