In [1]:
```
import pandas as pd
import numpy as np
```

```
/home/rmdstic/anaconda3/lib/python3.7/site-packages/pandas/compat/_
optional.py:138: UserWarning: Pandas requires version '2.7.0' or ne
wer of 'numexpr' (version '2.6.9' currently installed).
  warnings.warn(msg, UserWarning)
```

In [2]:
```
df=pd.read_csv("/home/rmdstic/Documents/TE-A-14/iris.csv")
```

In [3]:
```
df
```

Out[3]:

	sepal_length	sepal_width	petal_length	petal_width	species
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]:
```
df.isnull().sum()
```

Out[4]:
```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

In [5]:
```
from sklearn.preprocessing import LabelEncoder
```

In [6]:
```
enc = LabelEncoder()
```

In [7]:
```
df['species'] = enc.fit_transform(df['species'])
df['species'].unique()
```

Out[7]:
```
array([0, 1, 2])
```

In [8]:
```
x = df.drop(['species'],axis=1)
x
```

Out[8]:

	sepal_length	sepal_width	petal_length	petal_width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns

In [9]:
```
y = df['species']
y
```

Out[9]:
```
0      0
1      0
2      0
3      0
4      0
      ..
145    2
146    2
147    2
148    2
149    2
Name: species, Length: 150, dtype: int64
```

In [10]:
```
from sklearn import preprocessing
```

In [11]:
```
min_max_scaler = preprocessing.MinMaxScaler()
```

In [12]:
```
a = df.iloc[:,:4]
a_scaled = min_max_scaler.fit_transform(a)
```

In [13]:
```
df_normal = pd.DataFrame(a_scaled)
```

In [14]:
```
df_normal
```

Out[14]:

	0	1	2	3
0	0.222222	0.625000	0.067797	0.041667
1	0.166667	0.416667	0.067797	0.041667
2	0.111111	0.500000	0.050847	0.041667
3	0.083333	0.458333	0.084746	0.041667
4	0.194444	0.666667	0.067797	0.041667
...	...	...	...	...
145	0.666667	0.416667	0.711864	0.916667
146	0.555556	0.208333	0.677966	0.750000
147	0.611111	0.416667	0.711864	0.791667
148	0.527778	0.583333	0.745763	0.916667
149	0.444444	0.416667	0.694915	0.708333

150 rows × 4 columns

In [15]:
```
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size = 0.
```

In [16]:
```
from sklearn.naive_bayes import GaussianNB
gaus = GaussianNB()
```

In [17]:
```
gaus.fit(xtrain, ytrain)
```

Out[17]:
```
GaussianNB(priors=None, var_smoothing=1e-09)
```

In [18]:
```
ytrain_predict = gaus.predict(xtrain)
ytest_predict = gaus.predict(xtest)
```

In [19]:
```
ytrain_predict
```

Out[19]:
```
array([0, 1, 1, 2, 1, 0, 2, 1, 2, 1, 0, 0, 2, 1, 2, 1, 0, 0, 2, 1,
1, 2,
       1, 1, 2, 2, 1, 0, 0, 1, 2, 0, 1, 2, 2, 1, 2, 2, 2, 1, 0, 1,
0, 0,
       2, 2, 1, 0, 2, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1,
2, 2,
       0, 1, 1, 0, 0, 0, 0, 1, 2, 0, 2, 0, 0, 0, 2, 1, 0, 1, 2, 1,
2, 1,
       1, 2, 2, 2, 2, 2, 0, 0, 2, 1, 0, 1, 2, 0, 2, 1, 2, 2, 2, 1,
2, 1,
       1, 2, 0, 2, 0, 1, 2, 2, 0, 2])
```

In [20]:
```
ytest_predict
```

Out[20]:
```
array([2, 2, 0, 1, 0, 1, 1, 1, 2, 1, 0, 0, 1, 0, 0, 2, 1, 0, 0, 1,
0, 2,
       0, 2, 2, 0, 2, 1, 0, 1])
```

```
In [21]: gaus.predict([[5.1,3.5,1.4,0.2]])

Out[21]: array([0])
```

```
In [31]: from sklearn.metrics import confusion_matrix, classification_report,
```

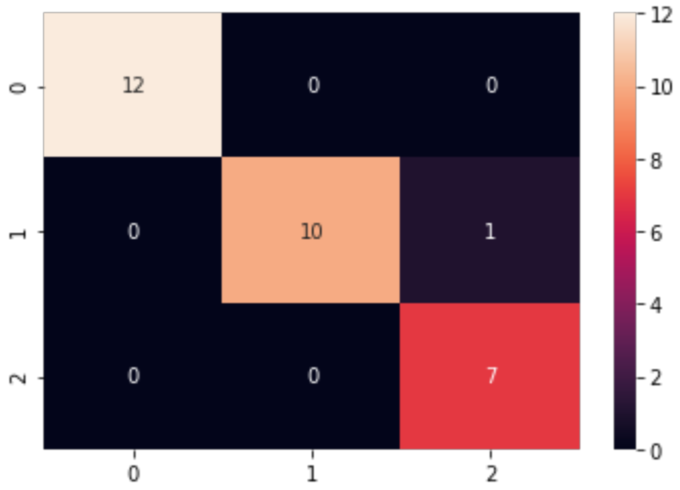
```
In [32]: matrix = confusion_matrix(ytest,ytest_predict)
         print(matrix)

         [[12  0  0]
          [ 0 10  1]
          [ 0  0  7]]
```

```
In [33]: score = accuracy_score(ytest,ytest_predict)
         print("Accuracy: ",score)

         Accuracy:  0.9666666666666667
```

```
In [34]: report = classification_report(ytest,ytest_predict)
         print(report)

                       precision    recall  f1-score   support

                    0       1.00      1.00      1.00        12
                    1       1.00      0.91      0.95        11
                    2       0.88      1.00      0.93         7

            micro avg       0.97      0.97      0.97        30
            macro avg       0.96      0.97      0.96        30
         weighted avg       0.97      0.97      0.97        30
```

```
In [35]: import seaborn as sns
```

```
In [36]: sns.heatmap(matrix,annot=True)

Out[36]: <matplotlib.axes._subplots.AxesSubplot at 0x7f1efd1c5780>
```



```
In [ ]:
```