

```
In [4]: import io
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
df=pd.read_csv("Iris.csv")
print(df.shape)
df
```

(150, 6)

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
...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

```
In [5]: print('The DataFrame contains %d rows and %d columns'%(df.shape))
print(df.dtypes)
print(df.info())
print(df.head(7))
```

The DataFrame contains 150 rows and 6 columns

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

dtype: object

```
<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

None

	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
5	6	5.4	3.9	1.7	0.4	Iris-setosa
6	7	4.6	3.4	1.4	0.3	Iris-setosa

```
In [6]: features=df.iloc[:,0:4]
print(features.head())
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm
0	1	5.1	3.5	1.4
1	2	4.9	3.0	1.4
2	3	4.7	3.2	1.3
3	4	4.6	3.1	1.5
4	5	5.0	3.6	1.4

```
In [7]: target=df.iloc[:,5]
print(target.head())
```

```
0    Iris-setosa
1    Iris-setosa
2    Iris-setosa
3    Iris-setosa
4    Iris-setosa
Name: Species, dtype: object
```

```
In [8]: print('The initial dataframe contained %d rows and %d columns'%(df.shape))
print('The features matrix contains %d rows and %d columns'%(features.shape))
print('The target vector contains %d rows and %d columns'%(np.array(target).res
```

The initial dataframe contained 150 rows and 6 columns
The features matrix contains 150 rows and 4 columns
The target vector contains 150 rows and 1 columns

```
In [9]: from sklearn.naive_bayes import GaussianNB
algorithm=GaussianNB(priors=None,var_smoothing=1e-9)
algorithm.fit(features,target)
```

```
Out[9]: ▾ GaussianNB
GaussianNB()
```

```
In [10]: print(algorithm.classes_)

['Iris-setosa' 'Iris-versicolor' 'Iris-virginica']
```

```
In [11]: print('The Gaussain model has acheieved %.2f percent accuracy'%(algorithm.score

The Gaussain model has acheieved 0.99 percent accuracy
```

```
In [12]: observation=[[5.0,3.7,1.6,0.1]]
predictions=algorithm.predict(observation)
print(predictions)

['Iris-setosa']
```

C:\Users\vinay\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but GaussianNB was fitted with feature names
warnings.warn(

```
In [13]: print(algorithm.predict_proba(observation).round())

[[1. 0. 0.]]
```

C:\Users\vinay\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but GaussianNB was fitted with feature names
warnings.warn(

```
In [14]: x = df.iloc[:,1:5]
y = df.iloc[:,5:]
```

```
In [15]: 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,random_sta
```

```
In [16]: from sklearn.metrics import confusion_matrix,classification_report
naive_bayes = GaussianNB()
naive_bayes.fit(x_train,y_train)
pred = naive_bayes.predict(x_test)
```

C:\Users\vinay\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\utils\validation.py:1111: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
y = column_or_1d(y, warn=True)

```
In [17]: from sklearn.preprocessing import LabelEncoder
cm = confusion_matrix(y_test,pred,labels = naive_bayes.classes_)
print(cm)
```

```
[[16  0  0]
 [ 0 18  0]
 [ 0  0 11]]
```

```
In [18]: print(classification_report(y_test,pred))
```

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	16
Iris-versicolor	1.00	1.00	1.00	18
Iris-virginica	1.00	1.00	1.00	11
accuracy			1.00	45
macro avg	1.00	1.00	1.00	45
weighted avg	1.00	1.00	1.00	45

```
In [19]: from sklearn.metrics import confusion_matrix,ConfusionMatrixDisplay,classification_report
print('\nAccuracy: {:.2f}'.format(accuracy_score(y_test,pred)))
err_rate=1-accuracy_score(y_test,pred)
print('Error Rate: ',err_rate)
```

Accuracy: 1.00
Error Rate: 0.0