Naive Bayes classifier assumes that the effect of a particular feature in a class is independent of other feature For example, a loan applicant is desirable or not depending on his/her income, previous loan and transacti history, age, and location. Even if these features are interdependent, these features are still consider independently. This assumption simplifies computation, and that's why it is considered as naive. Th assumption is called class conditional independence.

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

- P(h): the probability of hypothesis h being true (regardless of the data). This is known as the pri probability of h.
- P(D): the probability of the data (regardless of the hypothesis). This is known as the prior probability.
- P(h|D): the probability of hypothesis h given the data D. This is known as posterior probability.
- P(D|h): the probability of data d given that the hypothesis h was true. This is known as posteri probability.

How Naive Bayes classifier works?

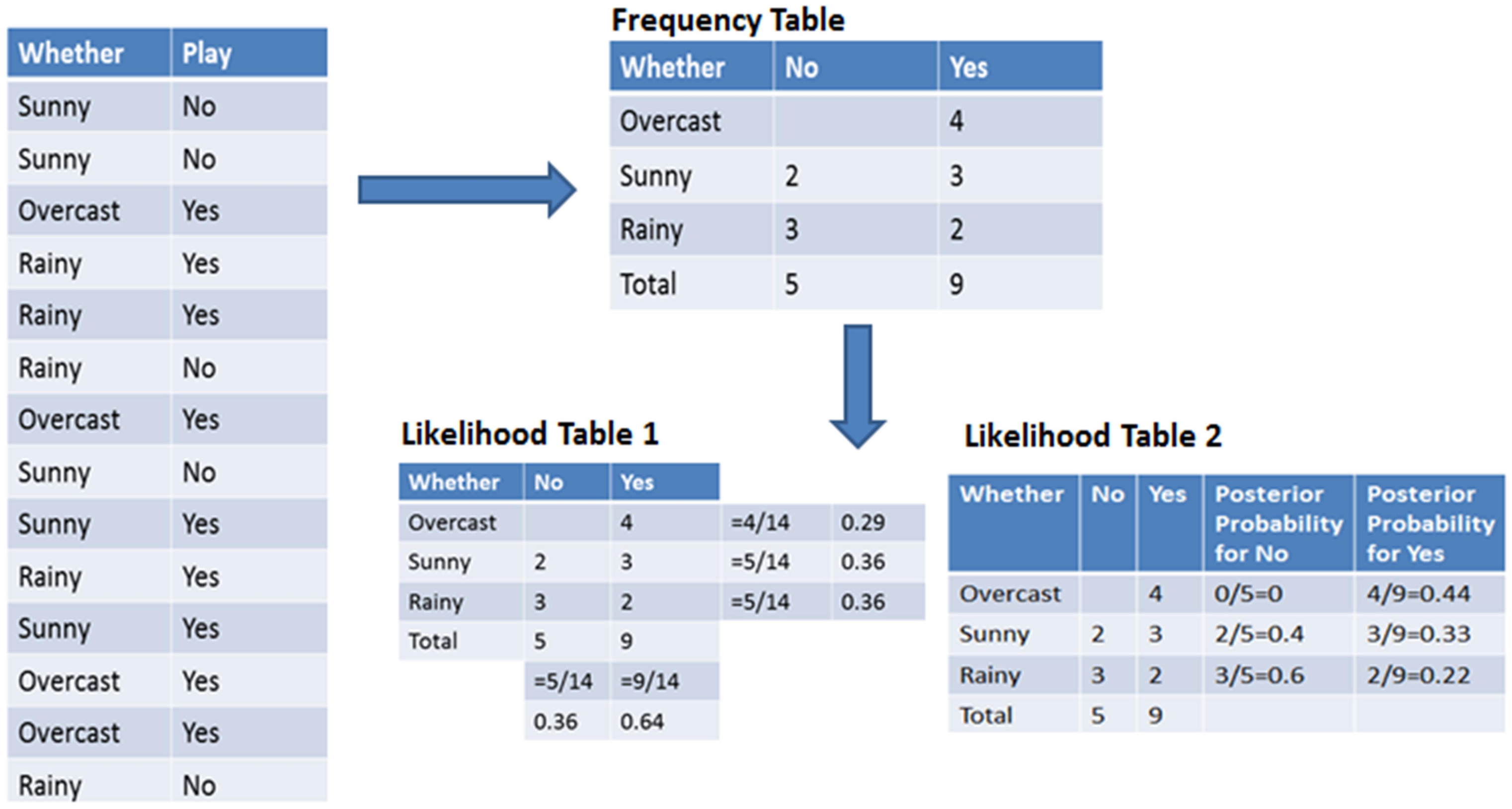
Let's understand the working of Naive Bayes through an example. Given an example of weather conditions a playing sports. You need to calculate the probability of playing sports. Now, you need to classify wheth players will play or not, based on the weather condition.

First Approach (In case of a single feature)

Naive Bayes classifier calculates the probability of an event in the following steps:

- Step 1: Calculate the prior probability for given class labels
- Step 2: Find Likelihood probability with each attribute for each class
- Step 3: Put these value in Bayes Formula and calculate posterior probability.
- Step 4: See which class has a higher probability, given the input belongs to the higher probability class.

For simplifying prior and posterior probability calculation you can use the two tables frequency and likeliho tables. Both of these tables will help you to calculate the prior and posterior probability. The Frequency tab contains the occurrence of labels for all features. There are two likelihood tables. Likelihood Table 1 is showi prior probabilities of labels and Likelihood Table 2 is showing the posterior probability.



Whether	Play
Sunny	No
Sunny	No
Overcast	Yes
Rainy	Yes
Rainy	Yes
Rainy	No
Overcast	Yes
Sunny	No
Sunny	Yes
Rainy	Yes
Sunny	Yes
Overcast	Yes
Overcast	Yes
Rainy	No

Frequency Table

Whether	No	Yes
Overcast		4
Sunny	2	3
Rainy	3	2
Total	5	9

Likelihood Table 1

Whether	No	Yes		
Overcast		4	=4/14	0.29
Sunny	2	3	=5/14	0.36
Rainy	3	2	=5/14	0.36
Total	5	9		
	=5/14	=9/14		
	0.36	0.64		

Likelihood Table 2

Whether	No	Yes	Posterior Probability for No	Posterior Probability for Yes
Overcast		4	0/5=0	4/9=0.44
Sunny	2	3	2/5=0.4	3/9=0.33
Rainy	3	2	3/5=0.6	2/9=0.22
Total	5	9		

Now suppose you want to calculate the probability of playing when the weather is overcast.

Probability of playing:

P(Yes | Overcast) = P(Overcast | Yes) P(Yes) / P (Overcast) ....(1)

1. Calculate Prior Probabilities:
   P(Overcast) = 4/14 = 0.29
   P(Yes)= 9/14 = 0.64
2. Calculate Posterior Probabilities:
   P(Overcast |Yes) = 4/9 = 0.44
3. Put Prior and Posterior probabilities in equation (1)
   P (Yes | Overcast) = 0.44 * 0.64 / 0.29 = 0.98(Higher)

Similarly, you can calculate the probability of not playing:

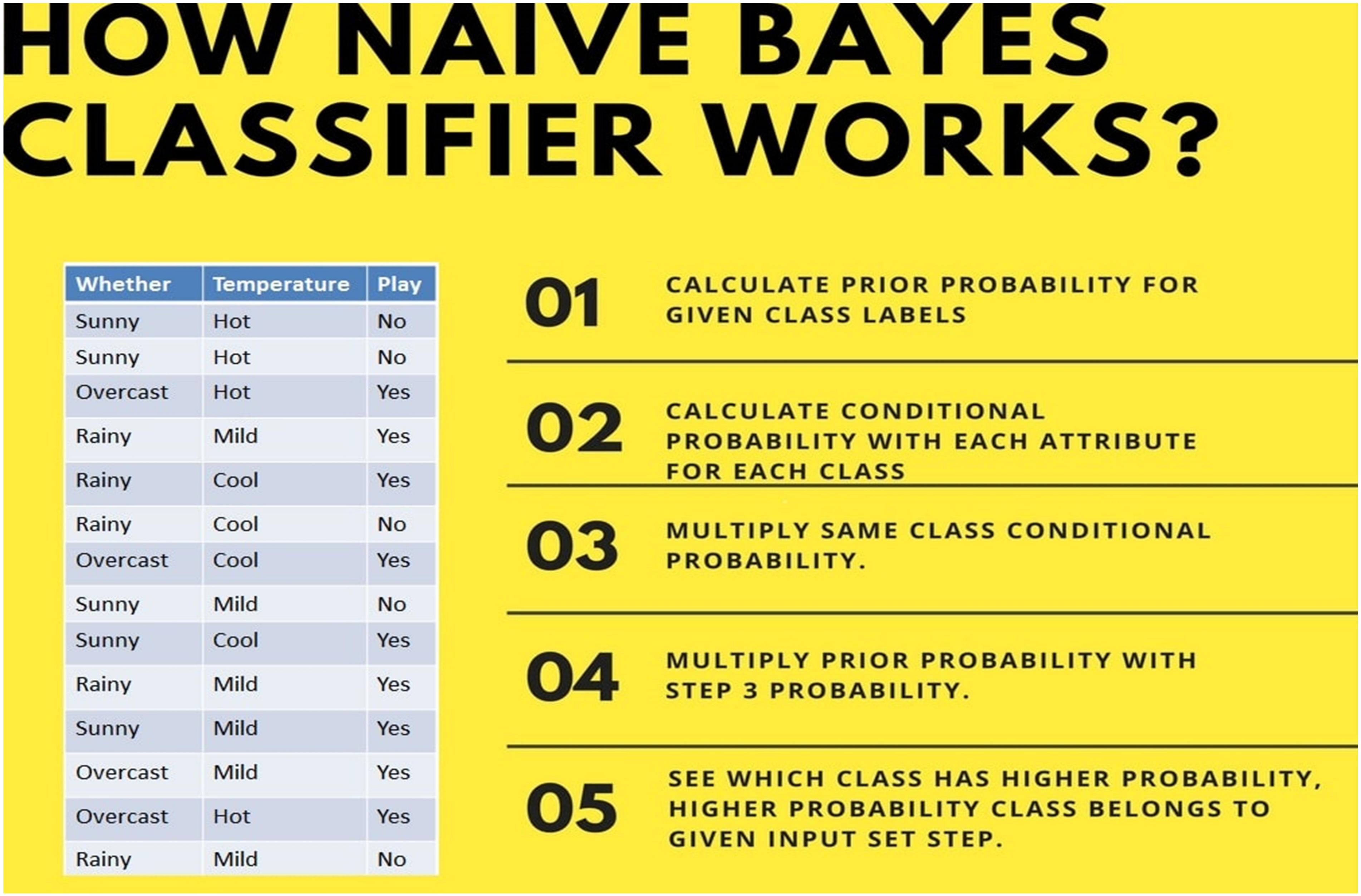
Probability of not playing:

P(No | Overcast) = P(Overcast | No) P(No) / P (Overcast) ....(2)

1. Calculate Prior Probabilities:
   P(Overcast) = 4/14 = 0.29
   P(No)= 5/14 = 0.36
2. Calculate Posterior Probabilities:
   P(Overcast |No) = 0/9 = 0
3. Put Prior and Posterior probabilities in equation (2)
   P (No | Overcast) = 0 * 0.36 / 0.29 = 0

The probability of a 'Yes' class is higher. So you can determine here if the weather is overcast than players will play the sport.

Second Approach (In case of multiple features)



HOW NAIVE BAYES CLASSIFIER WORKS?

Whether	Temperature	Play
Sunny	Hot	No
Sunny	Hot	No
Overcast	Hot	Yes
Rainy	Mild	Yes
Rainy	Cool	Yes
Rainy	Cool	No
Overcast	Cool	Yes
Sunny	Mild	No
Sunny	Cool	Yes
Rainy	Mild	Yes
Sunny	Mild	Yes
Overcast	Mild	Yes
Overcast	Hot	Yes
Rainy	Mild	No

01 CALCULATE PRIOR PROBABILITY FOR GIVEN CLASS LABELS

02 CALCULATE CONDITIONAL PROBABILITY WITH EACH ATTRIBUTE FOR EACH CLASS

03 MULTIPLY SAME CLASS CONDITIONAL PROBABILITY.

04 MULTIPLY PRIOR PROBABILITY WITH STEP 3 PROBABILITY.

05 SEE WHICH CLASS HAS HIGHER PROBABILITY, HIGHER PROBABILITY CLASS BELONGS TO GIVEN INPUT SET STEP.

Now suppose you want to calculate the probability of playing when the weather is overcast, and the temperature is mild.

Probability of playing:

*P(Play= Yes | Weather=Overcast, Temp=Mild) = P(Weather=Overcast, Temp=Mild | Play= Yes)*P(Play=Yes) ....(1)

P(Weather=Overcast, Temp=Mild | Play= Yes)= P(Overcast |Yes) P(Mild |Yes) ………..(2)

1. Calculate Prior Probabilities: P(Yes)= 9/14 = 0.64
2. Calculate Posterior Probabilities: P(Overcast |Yes) = 4/9 = 0.44 P(Mild |Yes) = 4/9 = 0.44
3. Put Posterior probabilities in equation (2) P(Weather=Overcast, Temp=Mild | Play= Yes) = 0.44 * 0.44 = 0.1936(Higher)
4. Put Prior and Posterior probabilities in equation (1) P(Play= Yes | Weather=Overcast, Temp=Mild) = 0.1936*0.64 = 0.124

Similarly, you can calculate the probability of not playing:

Probability of not playing:

*P(Play= No | Weather=Overcast, Temp=Mild) = P(Weather=Overcast, Temp=Mild | Play= No)*P(Play=No) ....(3)

P(Weather=Overcast, Temp=Mild | Play= No)= P(Weather=Overcast |Play=No) P(Temp=Mild | Play=No) ………..(4)

1. Calculate Prior Probabilities: P(No)= 5/14 = 0.36
2. Calculate Posterior Probabilities: P(Weather=Overcast |Play=No) = 0/9 = 0 P(Temp=Mild | Play=No)=2/5=0.4
3. Put posterior probabilities in equation (4) P(Weather=Overcast, Temp=Mild | Play= No) = 0 * 0.4= 0
4. Put prior and posterior probabilities in equation (3) P(Play= No | Weather=Overcast, Temp=Mild) = 0*0.36=0

The probability of a 'Yes' class is higher. So you can say here that if the weather is overcast than players will play the sport.

Naive Bayes Classifier

Definition

In machine learning, a Bayes classifier is a simple probabilistic classifier, which is based on applying Bayes' theorem. The feature model used by a naive Bayes classifier makes strong independence assumptions. This means that the existence of a particular feature of a class is independent or unrelated to the existence of every other feature.

Definition of independent events:

Two events E and F are independent, if both E and F have positive probability and if $P(E|F) = P(E)$ and $P(F|E) = P(F)$

As we have stated in our definition, the Naive Bayes Classifier is based on the Bayes' theorem. The Bayes theorem is based on the conditional probability, which we will define now:

Conditional Probability

$P(A|B)P(A|B)$ stands for "the conditional probability of A given B", or "the probability of A under the condition B", i.e. the probability of some event A under the assumption that the event B took place. When in a random experiment the event B is known to have occurred, the possible outcomes of the experiment are reduced to B, and hence the probability of the occurrence of A is changed from the unconditional probability into the conditional probability given B. The Joint probability is the probability of two events in conjunction. That is, it is the probability of both events together. There are three notations for the joint probability of A and B. It can be written as

- $P(A∩B)$
- $P(AB)$ or
- $P(A,B)$

The conditional probability is defined by

$$P(A|B)=P(A∩B)/P(B)$$

Examples for Conditional Probability

German Swiss Speaker

There are about 8.4 million people living in Switzerland. About 64 % of them speak German. There are about 7500 million people on earth.

If some aliens randomly beam up an earthling, what are the chances that he is a German speaking Swiss?

We have the events

S: being Swiss

GS: German Speaking

The probability for a randomly chosen person to be Swiss:

P(S)=8.4/7500=0.00112

If we know that somebody is Swiss, the probability of speaking German is 0.64. This corresponds to the conditional probability

P(GS|S)=0.64

So the probability of the earthling being Swiss and speaking German, can be calculated by the formula:

P(GS|S)=P(GS∩S)/P(S)

inserting the values from above gives us:

0.64=P(GS∩S)/0.00112

and

P(GS∩S)=0.0007168

So our aliens end up with a chance of **0.07168 %** of getting a German speaking Swiss person.

False Positives and False Negatives

A medical research lab proposes a screening to test a large group of people for a disease. An argument against such screenings is the problem of false positive screening results.

Suppose **0.1%** of the group suffer from the disease, and the rest is well:

P("sick")=0.1

and

P("well")=99.9.

The following is true for a screening test:

If you have the disease, the test will be positive 99% of the time, and if you don't have it, the test will be negative 99% of the time:

P("test positive" | "well") = 1 % ,and

P("test negative" | "well") = 99 %.

Finally, suppose that when the test is applied to a person having the disease, there is a 1% chance of a false negative result (and 99% chance of getting a true positive result), i.e.

P("test negative" | "sick") = 1 %, and

P("test positive" | "sick") = 99 %

	Sick	Healthy	Totals
Test result positive	**99**	**999**	**1098**
Test result negative	**1**	**98901**	**98902**
Totals	**100**	**99900**	**100000**

There are 999 False Positives and 1 False Negative.

Problem:
In many cases even medical professionals assume that "if you have this sickness, the test will be positive in 99 % of the time and if you don't have it, the test will be negative 99 % of the time. Out of the 1098 cases that report positive results only **99 (9 %) cases are correct** and **999 cases are false positives (91 %)**, i.e. if a person gets a positive test result, the **probability that he or she actually has the disease is just about 9 %.**

P("sick" | "test positive") = 99 / 1098 = 9.02 %

Iris Dataset

The Iris flower data set is a multivariate data set introduced by Ronald Fisher in his 1936 paper "The use of multiple measurements in taxonomic problems as an example of linear discriminant analysis."
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 centimetres.
Based on the combination of these four features, Fisher developed a linear discriminant model to distinguish the species from each other.

The module Scikit provides naive Bayes classifiers "off the rack".

Our first example uses the "iris dataset" contained in the model to train and test the classifier.

Gaussian Naive Bayes

GaussianNB implements the Gaussian Naive Bayes algorithm for classification. The likelihood of the features is assumed to be Gaussian:

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

The parameters σ_y and μ_y are estimated using maximum likelihood.

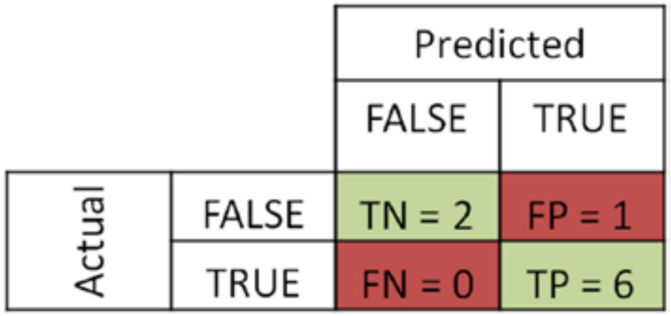
```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
X, y = load_iris(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=0)
gnb = GaussianNB()
y_pred = gnb.fit(X_train, y_train).predict(X_test)
print("Number of mislabeled points out of a total %d points : %d"% (X_test.shape[0], (y_test != y_pred).sum()))
```

OUTPUT:

```
Number of mislabeled points out of a total 75 points : 4
```

Evaluating a Classification Model Performance

The confusion matrix is the table that is used for describing the performance of the classification model.



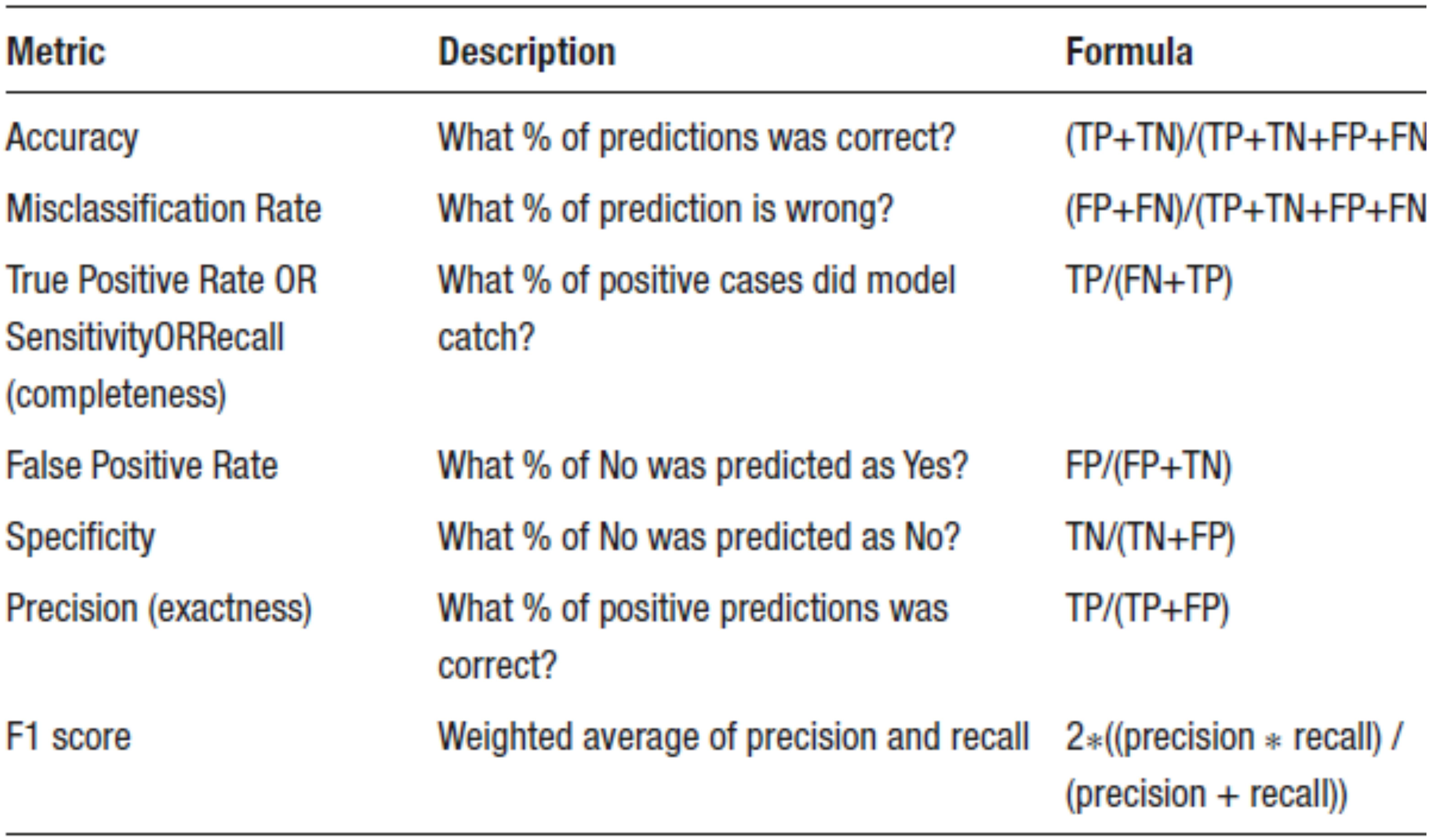
		Predicted	
		FALSE	TRUE
Actual	FALSE	TN = 2	FP = 1
	TRUE	FN = 0	TP = 6

Confusion matrix

- **True Negative (TN)**: Actual FALSE that was predicted as FALSE
- **False Positive (FP)**: Actual FALSE that was predicted as TRUE (Type I error)
- **False Negative (FN)**: Actual TRUE that was predicted as FALSE (Type II error)
- **True Positive (TP)**: Actual TRUE that was predicted as TRUE

Ideally, a good model should have high TN and TP and less of Type I & II errors.

The following table describes the key metrics derived out of a confusion matrix to measure the classification model performanc



Metric	Description	Formula
Accuracy	What % of predictions was correct?	(TP+TN)/(TP+TN+FP+FN
Misclassification Rate	What % of prediction is wrong?	(FP+FN)/(TP+TN+FP+FN
True Positive Rate OR SensitivityORRecall (completeness)	What % of positive cases did model catch?	TP/(FN+TP)
False Positive Rate	What % of No was predicted as Yes?	FP/(FP+TN)
Specificity	What % of No was predicted as No?	TN/(TN+FP)
Precision (exactness)	What % of positive predictions was correct?	TP/(TP+FP)
F1 score	Weighted average of precision and recall	2∗((precision ∗ recall) / (precision + recall))

Classification Pefromance Matrices

Naive Bayes with Multiple Labels

Till now you have learned Naive Bayes classification with binary labels. Now you will learn about multiple class classification in Naive Bayes. Which is known as multinomial Naive Bayes classification. For example, if you want to classify a news article about technology, entertainment, politics, or sports.

In model building part, you can use wine dataset which is a very famous multi-class classification problem. "This dataset is the result of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars." (UC Irvine)

Dataset comprises of 13 features (alcohol, malic_acid, ash, alcalinity_of_ash, magnesium, total_phenols, flavanoids, nonflavanoid_phenols, proanthocyanins, color_intensity, hue, od280/od315_of_diluted_wines, proline) and type of wine cultivar. This data has three type of wine Class_0, Class_1, and Class_3. Here you can build a model to classify the type of wine.

The dataset is available in the scikit-learn library.

1. ***Loading Data***

Let's first load the required wine dataset from scikit-learn datasets.

```
#Import scikit-learn dataset library
from sklearn import datasets
#Load dataset
wine = datasets.load_wine()
```

2. ***Exploring Data***

You can print the target and feature names, to make sure you have the right dataset, as such:

```
# print the names of the 13 features
print ("Features: ", wine.feature_names)


# print the label type of wine(class_0, class_1, class_2)
print ("Labels: ", wine.target_names)
```

3. **It's a good idea to always explore your data a bit, so you know what you're working with. Here, you can see the first five rows of the dataset are printed, as well as the target variable for the whole dataset.**

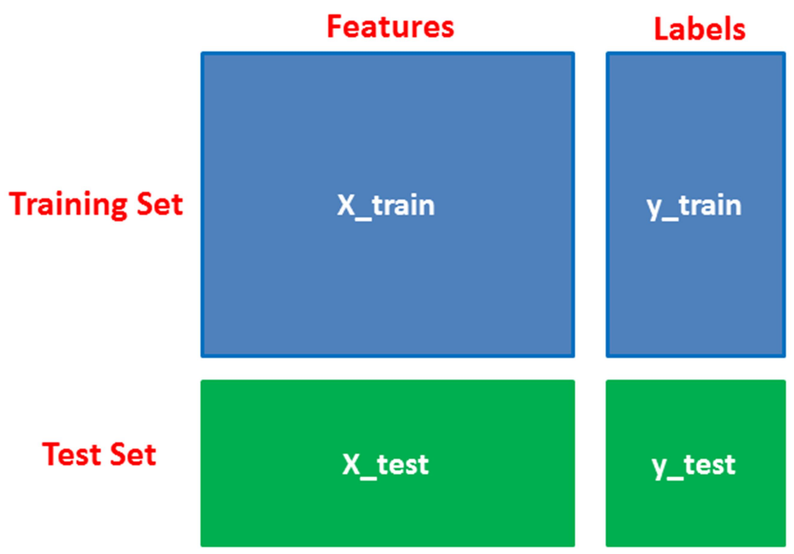
```
# print data(feature)shape
wine.data.shape

# print the wine data features (top 5 records)
print (wine.data[0:5])

# print the wine labels (0:Class_0, 1:class_2, 2:class_2)
print (wine.target)
```

4. ***Splitting Data***

   First, you separate the columns into dependent and independent variables(or features and label). Then you split those variables into train and test set.



```
# Import train_test_split function
from sklearn.model_selection import train_test_split

# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(wine.data, wine.target, test_size=0.3,random_state=109)
# 70% training and 30% test
```

5. **Model Generation**

After splitting, you will generate a random forest model on the training set and perform prediction on test set features.

```
#Import Gaussian Naive Bayes model
from sklearn.naive_bayes import GaussianNB

#Create a Gaussian Classifier
gnb = GaussianNB()
```

```
#Train the model using the training sets
gnb.fit(X_train, y_train)

#Predict the response for test dataset
y_pred = gnb.predict(X_test)
```

6. Evaluating Model

After model generation, check the accuracy using actual and predicted values.

```
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics

# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

7. Zero Probability Problem

Suppose there is no tuple for a risky loan in the dataset, in this scenario, the posterior probability will be zero, and the model is unable to make a prediction. This problem is known as Zero Probability because the occurrence of the particular class is zero.

The solution for such an issue is the Laplacian correction or Laplace Transformation. Laplacian correction is one of the smoothing techniques. Here, you can assume that the dataset is large enough that adding one row of each class will not make a difference in the estimated probability. This will overcome the issue of probability values to zero.

For Example: Suppose that for the class loan risky, there are 1000 training tuples in the database. In this database, income column has (
tuples for low income, 990 tuples for medium income, and 10 tuples for high income. The probabilities of these events, without the Laplacian correction, are 0, 0.990 (from 990/1000), and 0.010 (from 10/1000)

Now, apply Laplacian correction on the given dataset. Let's add 1 more tuple for each income-value pair. The probabilities of these events:

$$\frac{1}{1003} = 0.001, \frac{991}{1003} = 0.988, \text{ and } \frac{11}{1003} = 0.011,$$

Advantages

- **It is not only a simple approach but also a fast and accurate method for prediction.**
- **Naive Bayes has very low computation cost.**
- **It can efficiently work on a large dataset.**
- **It performs well in case of discrete response variable compared to the continuous variable.**
- **It can be used with multiple class prediction problems.**
- **It also performs well in the case of text analytics problems.**
- **When the assumption of independence holds, a Naive Bayes classifier performs better compared to other models like logistic regression.**

Disadvantages

- **The assumption of independent features. In practice, it is almost impossible that model will get a set of predictors which are entirely independent.**
- **If there is no training tuple of a particular class, this causes zero posterior probability. In this case, the model is unable to make predictions. This problem is known as Zero Probability/Frequency Problem.**

```
# Gaussian Naive Bayes
from sklearn import datasets
from sklearn import metrics
from sklearn.naive_bayes import GaussianNB
# load the iris datasets
dataset = datasets.load_iris()
# fit a Naive Bayes model to the data
model = GaussianNB()
model.fit(dataset.data, dataset.target)
print(dataset)
print(model)
# make predictions
expected = dataset.target
predicted = model.predict(dataset.data)
# summarize the fit of the model
print(metrics.classification_report(expected, predicted))
print(metrics.confusion_matrix(expected, predicted))
```

OUTPUT:

```
{'data': array([[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],
       [5.4, 3.7, 1.5, 0.2],
       [4.8, 3.4, 1.6, 0.2],
       [4.8, 3. , 1.4, 0.1],
       [4.3, 3. , 1.1, 0.1],
       [5.8, 4. , 1.2, 0.2],
       [5.7, 4.4, 1.5, 0.4],
       [5.4, 3.9, 1.3, 0.4],
       [5.1, 3.5, 1.4, 0.3],
       [5.7, 3.8, 1.7, 0.3],
       [5.1, 3.8, 1.5, 0.3],
       [5.4, 3.4, 1.7, 0.2],
       [5.1, 3.7, 1.5, 0.4],
       [4.6, 3.6, 1. , 0.2],
       [5.1, 3.3, 1.7, 0.5],
       [4.8, 3.4, 1.9, 0.2],
       [5. , 3. , 1.6, 0.2],
       [5. , 3.4, 1.6, 0.4],
       [5.2, 3.5, 1.5, 0.2],
       [5.2, 3.4, 1.4, 0.2],
       [4.7, 3.2, 1.6, 0.2],
       [4.8, 3.1, 1.6, 0.2],
       [5.4, 3.4, 1.5, 0.4],
       [5.2, 4.1, 1.5, 0.1],
       [5.5, 4.2, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.2],
       [5. , 3.2, 1.2, 0.2],
       [5.5, 3.5, 1.3, 0.2],
       [4.9, 3.6, 1.4, 0.1],
       [4.4, 3. , 1.3, 0.2],
[5.1, 3.4, 1.5, 0.2],
[5. , 3.5, 1.3, 0.3],
[4.5, 2.3, 1.3, 0.3],
[4.4, 3.2, 1.3, 0.2],
[5. , 3.5, 1.6, 0.6],
[5.1, 3.8, 1.9, 0.4],
[4.8, 3. , 1.4, 0.3],
[5.1, 3.8, 1.6, 0.2],
[4.6, 3.2, 1.4, 0.2],
[5.3, 3.7, 1.5, 0.2],
[5. , 3.3, 1.4, 0.2],
[7. , 3.2, 4.7, 1.4],
[6.4, 3.2, 4.5, 1.5],
[6.9, 3.1, 4.9, 1.5],
[5.5, 2.3, 4. , 1.3],
[6.5, 2.8, 4.6, 1.5],
[5.7, 2.8, 4.5, 1.3],
[6.3, 3.3, 4.7, 1.6],
[4.9, 2.4, 3.3, 1. ],
[6.6, 2.9, 4.6, 1.3],
[5.2, 2.7, 3.9, 1.4],
[5. , 2. , 3.5, 1. ],
[5.9, 3. , 4.2, 1.5],
[6. , 2.2, 4. , 1. ],
[6.1, 2.9, 4.7, 1.4],
[5.6, 2.9, 3.6, 1.3],
[6.7, 3.1, 4.4, 1.4],
[5.6, 3. , 4.5, 1.5],
[5.8, 2.7, 4.1, 1. ],
[6.2, 2.2, 4.5, 1.5],
[5.6, 2.5, 3.9, 1.1],
[5.9, 3.2, 4.8, 1.8],
[6.1, 2.8, 4. , 1.3],
[6.3, 2.5, 4.9, 1.5],
[6.1, 2.8, 4.7, 1.2],
[6.4, 2.9, 4.3, 1.3],
[6.6, 3. , 4.4, 1.4],
[6.8, 2.8, 4.8, 1.4],
[6.7, 3. , 5. , 1.7],
[6. , 2.9, 4.5, 1.5],
[5.7, 2.6, 3.5, 1. ],
[5.5, 2.4, 3.8, 1.1],
[5.5, 2.4, 3.7, 1. ],
[5.8, 2.7, 3.9, 1.2],
[6. , 2.7, 5.1, 1.6],
[5.4, 3. , 4.5, 1.5],
[6. , 3.4, 4.5, 1.6],
[6.7, 3.1, 4.7, 1.5],
[6.3, 2.3, 4.4, 1.3],
[5.6, 3. , 4.1, 1.3],
[5.5, 2.5, 4. , 1.3],
[5.5, 2.6, 4.4, 1.2],
[6.1, 3. , 4.6, 1.4],
[5.8, 2.6, 4. , 1.2],
[5. , 2.3, 3.3, 1. ],
[5.6, 2.7, 4.2, 1.3],
[5.7, 3. , 4.2, 1.2],
[5.7, 2.9, 4.2, 1.3],
[6.2, 2.9, 4.3, 1.3],
[5.1, 2.5, 3. , 1.1],
[5.7, 2.8, 4.1, 1.3],
[6.3, 3.3, 6. , 2.5],
[5.8, 2.7, 5.1, 1.9],
[7.1, 3. , 5.9, 2.1],
[6.3, 2.9, 5.6, 1.8],
[6.5, 3. , 5.8, 2.2],
[7.6, 3. , 6.6, 2.1],
[4.9, 2.5, 4.5, 1.7],
[7.3, 2.9, 6.3, 1.8],
[6.7, 2.5, 5.8, 1.8],
[7.2, 3.6, 6.1, 2.5],
[6.5, 3.2, 5.1, 2. ],
[6.4, 2.7, 5.3, 1.9],
[6.8, 3. , 5.5, 2.1],
[5.7, 2.5, 5. , 2. ],
[5.8, 2.8, 5.1, 2.4],
[6.4, 3.2, 5.3, 2.3],
[6.5, 3. , 5.5, 1.8],
[7.7, 3.8, 6.7, 2.2],
[7.7, 2.6, 6.9, 2.3],
[6. , 2.2, 5. , 1.5],
[6.9, 3.2, 5.7, 2.3],
[5.6, 2.8, 4.9, 2. ],
[7.7, 2.8, 6.7, 2. ],
[6.3, 2.7, 4.9, 1.8],
[6.7, 3.3, 5.7, 2.1],
[7.2, 3.2, 6. , 1.8],
[6.2, 2.8, 4.8, 1.8],
       [6.1, 3. , 4.9, 1.8],
       [6.4, 2.8, 5.6, 2.1],
       [7.2, 3. , 5.8, 1.6],
       [7.4, 2.8, 6.1, 1.9],
       [7.9, 3.8, 6.4, 2. ],
       [6.4, 2.8, 5.6, 2.2],
       [6.3, 2.8, 5.1, 1.5],
       [6.1, 2.6, 5.6, 1.4],
       [7.7, 3. , 6.1, 2.3],
       [6.3, 3.4, 5.6, 2.4],
       [6.4, 3.1, 5.5, 1.8],
       [6. , 3. , 4.8, 1.8],
       [6.9, 3.1, 5.4, 2.1],
       [6.7, 3.1, 5.6, 2.4],
       [6.9, 3.1, 5.1, 2.3],
       [5.8, 2.7, 5.1, 1.9],
       [6.8, 3.2, 5.9, 2.3],
       [6.7, 3.3, 5.7, 2.5],
       [6.7, 3. , 5.2, 2.3],
       [6.3, 2.5, 5. , 1.9],
       [6.5, 3. , 5.2, 2. ],
       [6.2, 3.4, 5.4, 2.3],
       [5.9, 3. , 5.1, 1.8]]), 'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]), 'frame': None, 'target_names': array(['setosa', 'versicolor
', 'virginica'], dtype='<U10'), 'DESCR': '.. _iris_dataset:\n\nIris plants dataset\n--------------------\n\n**Data Set Cha
racteristics:**\n\n    :Number of Instances: 150 (50 in each of three classes)\n    :Number of Attributes: 4 numeric, pred
ictive attributes and the class\n    :Attribute Information:\n        - sepal length in cm\n        - sepal width in cm\n
       - petal length in cm\n        - petal width in cm\n        - class:\n                - Iris-Setosa\n
 - Iris-Versicolour\n                - Iris-Virginica\n                \n    :Summary Statistics:\n\n    ============== ==
== ==== ======= ===== ====================\n                    Min  Max   Mean    SD   Class Correlation\n    ==========
=== ==== ==== ======= ===== ====================\n    sepal length:   4.3  7.9   5.84   0.83    0.7826\n    sepal width:
  2.0  4.4   3.05   0.43   -0.4194\n    petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)\n    petal width:    0.1
  2.5   1.20   0.76    0.9565  (high!)\n    ============== ==== ==== ======= ===== ====================\n\n    :Missing At
tribute Values: None\n    :Class Distribution: 33.3% for each of 3 classes.\n    :Creator: R.A. Fisher\n    :Donor: Michae
l Marshall (MARSHALL%PLU@io.arc.nasa.gov)\n    :Date: July, 1988\n\nThe famous Iris database, first used by Sir R.A. Fishe
r. The dataset is taken\nfrom Fisher\'s paper. Note that it\'s the same as in R, but not as in the UCI\nMachine Learning R
epository, which has two wrong data points.\n\nThis is perhaps the best known database to be found in the\npattern recogni
tion literature.  Fisher\'s paper is a classic in the field and\nis referenced frequently to this day.  (See Duda & Hart,
for example.)  The\ndata set contains 3 classes of 50 instances each, where each class refers to a\ntype of iris plant.  O
```

```
ne class is linearly separable from the other 2; the\nlatter are NOT linearly separable from each other.\n\n.. topic:: Ref
erences\n\n   - Fisher, R.A. "The use of multiple measurements in taxonomic problems"\n     Annual Eugenics, 7, Part II, 1
79-188 (1936); also in "Contributions to\n     Mathematical Statistics" (John Wiley, NY, 1950).\n   - Duda, R.O., & Hart,
P.E. (1973) Pattern Classification and Scene Analysis.\n     (Q327.D83) John Wiley & Sons.  ISBN 0-471-22361-1.  See page
218.\n   - Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New System\n     Structure and Classification Rule fo
r Recognition in Partially Exposed\n     Environments".  IEEE Transactions on Pattern Analysis and Machine\n     Intellige
nce, Vol. PAMI-2, No. 1, 67-71.\n   - Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule".  IEEE Transactions\n     on
Information Theory, May 1972, 431-433.\n   - See also: 1988 MLC Proceedings, 54-64.  Cheeseman et al"s AUTOCLASS II\n
conceptual clustering system finds 3 classes in the data.\n   - Many, many more ...', 'feature_names': ['sepal length (cm)
', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)'], 'filename': 'iris.csv', 'data_module': 'sklearn.datasets.
data'}
```

```
GaussianNB
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        50
           1       0.94      0.94      0.94        50
           2       0.94      0.94      0.94        50

    accuracy                           0.96       150

   macro avg       0.96      0.96      0.96       150

weighted avg       0.96      0.96      0.96       150


[[50  0  0]
 [ 0 47  3]
 [ 0  3 47]]
```