TP Bayesian Network :

A simple example of training a **Gaussian Naive Bayes** classifier on the **Iris dataset** using the scikit-learn library

GaussianNB: the **Gaussian Naive Bayes** classifier, which assumes that features follow a normal (Gaussian) distribution.

The following code builds a simple machine learning model that classifies iris flowers into species based on their measurements using the **Naive Bayes algorithm**.

[5]:
```
# Charge libraries
from sklearn import datasets
from sklearn.naive_bayes import GaussianNB

# Charge data
iris = datasets.load_iris()
features = iris.data
target = iris.target

# Create a Bayesian naif gaussian
classifer = GaussianNB()

# Train model
model = classifer.fit(features, target)
```

[7]:
```
# Create a new observation
new_observation = [[4, 4, 4, 0.4]]

# Predict class
model.predict(new_observation)
```

[7]:
```
array([1])
```

[1]:
```
# Create a bayesian classifier with probabilities a priori to each class
clf = GaussianNB(priors=[0.25, 0.25, 0.5])

# Train model
model = classifer.fit(features, target)
```

The following code is an example of **text classification** using a **Multinomial Naive Bayes** model with scikit-learn. It processes some short text samples and trains a model to distinguish between two categories (like sentiment or topic).

```python
# Charge libraries
import numpy as np
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import CountVectorizer

# Create a text
text_data = np.array(['I love Brazil. Brazil !',
                      'Brazil is best',
                      'Germany beats both'])
# creation of a bag of words
count = CountVectorizer()
bag_of_words = count.fit_transform(text_data)

# Create caracteristics matrix
features = bag_of_words.toarray()

# Create a vector
target = np.array([0,0,1])

# Create a naif bayesian classifier multinomial
classifier = MultinomialNB(class_prior=[0.25, 0.5])

# Train a model
model = classifier.fit(features, target)
```

```python
# Create an observation
new_observation = [[0, 0, 0, 1, 0, 1, 0]]

# Predict new observation
model.predict(new_observation)
```

```
array([0])
```

- **MultinomialNB:** Naive Bayes classifier for **discrete features**, often used for **text classification**.
- **CountVectorizer:** converts text to a **bag-of-words** representation (word frequency matrix).
- CountVectorizer tokenizes the text and counts word occurrences.
- fit_transform() learns the vocabulary and transforms the text into a sparse matrix.
- toarray() converts it into a regular NumPy array for easier viewing.

The following code shows how to train a **Bernoulli Naive Bayes** classifier on synthetic binary data using scikit-learn.

- Creates synthetic binary data (like yes/no or on/off features).
- Labels the data randomly as 0 or 1.
- Trains a **Bernoulli Naive Bayes classifier**, often used for things like:
    - Email spam detection (word present or not),
    - Click prediction,
    - Any classification task with binary inputs.

```
# Load libraries
import numpy as np
from sklearn.naive_bayes import BernoulliNB

# Create 3 binary features
features = np.random.randint(2, size=(100, 3))

# Create the output
target = np.random.randint(2, size=(100, 1)).ravel()

# Create a bernoulli naif bayesian classifier with a priori probabilities with each class
classifier = BernoulliNB(class_prior=[0.25, 0.5])

# Train the model
model = classifier.fit(features, target)
```

The following code trains a **Gaussian Naive Bayes** classifier on the **Iris dataset**, and then uses **probability calibration** to produce more reliable probability estimates for predictions.

- Loads and trains a GaussianNB model,
- Calibrates its probability estimates using sigmoid scaling,
- Predicts **probabilities** for a new flower measurement.

```python
# Load libraries
from sklearn import datasets
from sklearn.naive_bayes import GaussianNB
from sklearn.calibration import CalibratedClassifierCV

# charge data
iris = datasets.load_iris()
features = iris.data
target = iris.target

# create a gaussian naive bayes classifier
classifier = GaussianNB()

classifier_sigmoid = CalibratedClassifierCV(classifier, cv=2, method='sigmoid')
classifier_sigmoid.fit(features, target)

# Create a new observation
new_observation = [[2.6, 2.6, 2.6, 0.4]]

# Print calibrated probabilities
classifier_sigmoid.predict_proba(new_observation)
```

```
array([[0.31859971, 0.63663451, 0.04476578]])
```