

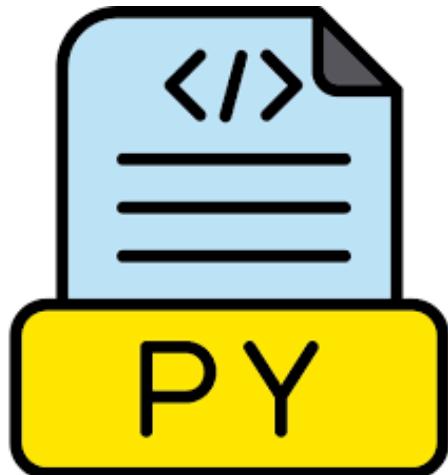
Fouille de Données

Data Mining

Classification - Partie 3

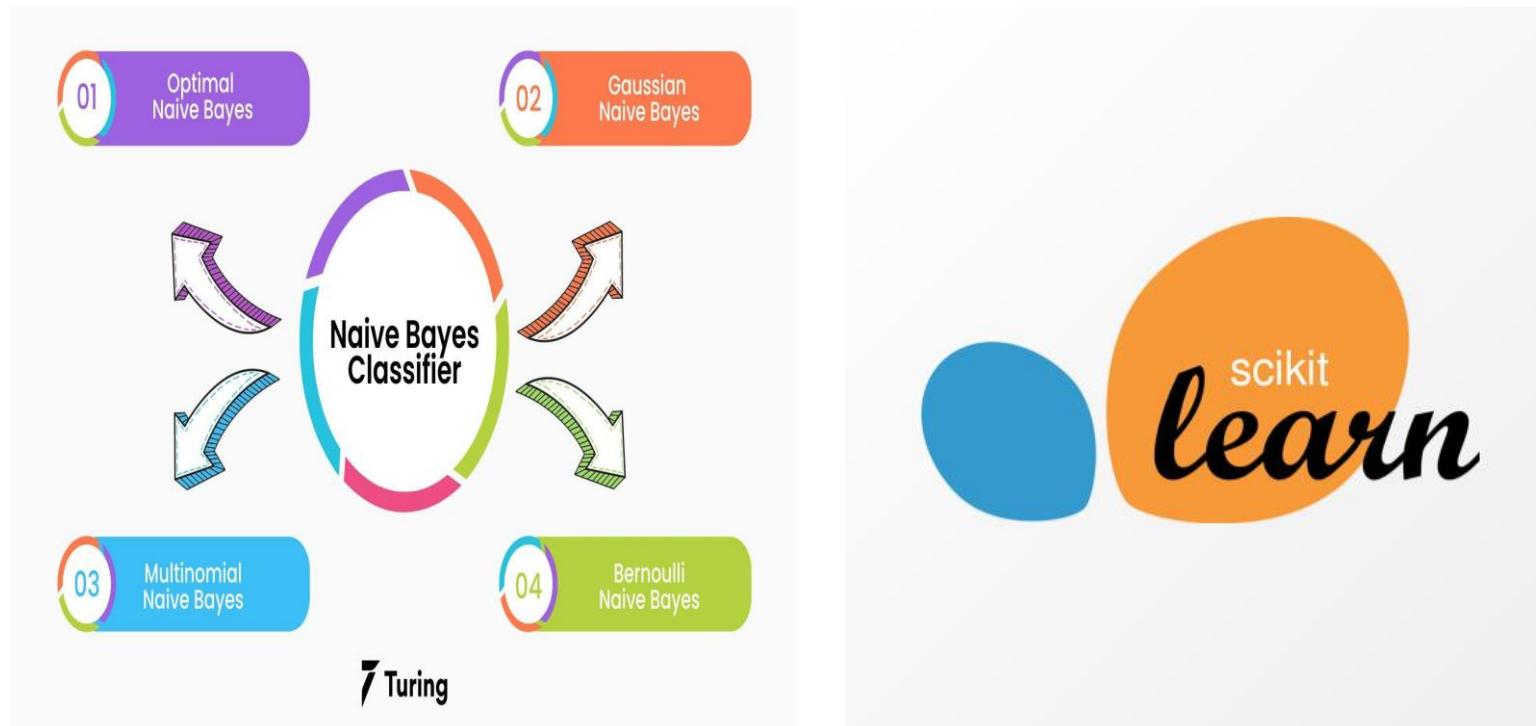
La classification naïve bayésienne

- **Série TP 4 – Naive Bayes with Scikit Learn**



La classification naïve bayésienne

- **Série TP 4 – Naive Bayes with Scikit Learn**



https://scikit-learn.org/stable/modules/naive_bayes.html

La classification naïve bayésienne

- **Série TP 4 – Naive Bayes with Scikit Learn**

1.9.1. Gaussian Naive Bayes

1.9.2. Multinomial Naive Bayes

1.9.3. Complement Naive Bayes

1.9.4. Bernoulli Naive Bayes

1.9.5. Categorical Naive Bayes



https://scikit-learn.org/stable/modules/naive_bayes.html

La classification naïve bayésienne

- **Série TP 4 – Naive Bayes with Scikit Learn**

$$P(y \mid x_1, \dots, x_n) \propto P(y) \prod_{i=1}^n P(x_i \mid y)$$
$$\hat{y} = \arg \max_y P(y) \prod_{i=1}^n P(x_i \mid y),$$

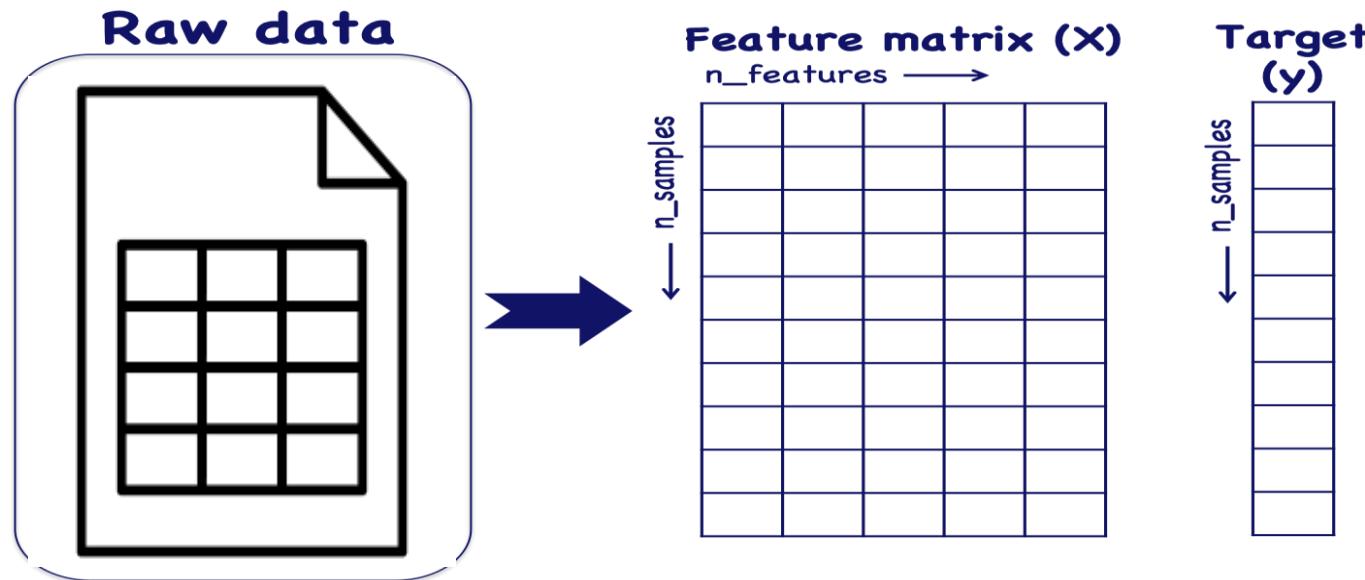
↓



https://scikit-learn.org/stable/modules/naive_bayes.html

La classification naïve bayésienne

- `sklearn.naive_bayes.CategoricalNB` implements the categorical naive Bayes algorithm for categorically distributed data.
- Takes as **input** two arrays: an **array X** of shape $(n_samples, n_features)$ holding the **training samples**, and an **array Y** of integer values, shape $(n_samples,)$, holding the **class labels** for the training samples.



La classification naïve bayésienne

- `sklearn.naive_bayes.CategoricalNB`

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

Diagram illustrating the components of the Naïve Bayes formula:

- Likelihood**: $P(x|c)$ (represented by an arrow pointing to the numerator term $P(x|c)$)
- Class Prior Probability**: $P(c)$ (represented by an arrow pointing to the numerator term $P(c)$)
- Posterior Probability**: $P(c|x)$ (represented by an arrow pointing to the entire fraction $P(c|x)$)
- Predictor Prior Probability**: $P(x)$ (represented by an arrow pointing to the denominator term $P(x)$)

$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \cdots \times P(x_n|c) \times P(c)$$

La classification naïve bayésienne

- `sklearn.naive_bayes.CategoricalNB`

$$P(c | \mathbf{X}) = P(x_1 | c) \times P(x_2 | c) \times \cdots \times P(x_n | c) \times P(c)$$

Probability calculation

The probability of category t in feature i given class c is estimated as:

$$P(x_i = t | y = c ; \alpha) = \frac{N_{tic} + \alpha}{N_c + \alpha n_i},$$

where $N_{tic} = |\{j \in J | x_{ij} = t, y_j = c\}|$ is the number of times category t appears in the samples x_i , which belong to class c , $N_c = |\{j \in J | y_j = c\}|$ is the number of samples with class c , α is a smoothing parameter and n_i is the number of available categories of feature i .

La classification naïve bayésienne

- **sklearn.naive_bayes.CategoricalNB**

In `CategoricalNB`, for each feature X_j with possible categorical values x_j , the conditional probability of a value given a class y is estimated as:

$$P(X_j = x_j \mid Y = y) = \frac{N_{y,x_j} + \alpha}{N_y + \alpha \cdot n_j}$$

where:

- N_{y,x_j} = number of samples with class y having feature $X_j = x_j$
- N_y = total number of samples with class y
- n_j = number of possible categories for feature X_j
- α = smoothing parameter (default = 1.0)

La classification naïve bayésienne

- `sklearn.naive_bayes.CategoricalNB`

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

- Prédire la classe quand le **Weather** = Overcast :

$$P(\text{Yes} \mid \text{Overcast}) = P(\text{Overcast} \mid \text{Yes}) * P(\text{Yes})$$



$$P(\text{Overcast} \mid \text{Yes}) = 4 / 9$$

La classification naïve bayésienne

- `sklearn.naive_bayes.CategoricalNB`

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

- Prédire la classe quand le **Weather** = Overcast :

$$P(\text{No} \mid \text{Overcast}) = P(\text{Overcast} \mid \text{No}) * P(\text{No})$$



$$P(\text{Overcast} \mid \text{No}) = 0 / 5$$

La classification naïve bayésienne

- `sklearn.naive_bayes.CategoricalNB`

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

- Prédire la classe quand le **Weather** = Overcast :

$$P(\text{No} \mid \text{Overcast}) = P(\text{Overcast} \mid \text{No}) * P(\text{No})$$



$$P(\text{Overcast} \mid \text{No}) = 0 / 5$$

$$P(X_j = x_j \mid Y = y) = \frac{N_{y,x_j}}{N_y}$$

La classification naïve bayésienne

- `sklearn.naive_bayes.CategoricalNB`

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

- Prédire la classe quand le **Weather** = Overcast :

$$P(\text{No} \mid \text{Overcast}) = P(\text{Overcast} \mid \text{No}) * P(\text{No})$$



$$P(\text{Overcast} \mid \text{No}) = 0 / 5$$

Problem - If a certain category never appears in training for a given class :

- its probability becomes **zero**,
- which makes the whole product of probabilities for that class **zero** during prediction.

La classification naïve bayésienne

- `sklearn.naive_bayes.CategoricalNB`

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

- Prédire la classe quand le **Weather** = Overcast :

$$P(\text{No} \mid \text{Overcast}) = P(\text{Overcast} \mid \text{No}) * P(\text{No})$$



$$P(\text{Overcast} \mid \text{No}) = 0 / 5$$

Solution – Smoothing (Lissage) : It adds a small constant (α) to each count.

$$P(X_j = x_j \mid Y = y) = \frac{N_{y,x_j} + \alpha}{N_y + \alpha \cdot n_j}$$

La classification naïve bayésienne

`sklearn.naive_bayes.CategoricalNB`

- CategoricalNB in scikit-learn uses **additive** (Laplace) **smoothing** when estimating conditional probabilities.
- This is controlled by the parameter alpha.
- It adds a small constant (α) to each count. Ensures **no zero probabilities**, and improves generalization on unseen category combinations.
- $\text{alpha}=1.0 \rightarrow \text{Laplace}$ smoothing & $\text{alpha} > 1$ is called **Lidstone** smoothing. Default = 1.
- $\text{alpha}=0.0 \rightarrow$ no smoothing.

La classification naïve bayésienne

- `sklearn.naive_bayes.CategoricalNB`

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

- Prédire la classe quand le **Weather** = Overcast :

$$P(\text{No} \mid \text{Overcast}) = P(\text{Overcast} \mid \text{No}) * P(\text{No})$$



$\alpha = 1$: $P(\text{Overcast} \mid \text{No}) = (0 + 1) / (5 + (1 * 3))$

Solution – Smoothing (Lissage) : It adds a small constant (α) to each count.

$$P(X_j = x_j \mid Y = y) = \frac{N_{y,x_j} + \alpha}{N_y + \alpha \cdot n_j}$$

La classification naïve bayésienne

- `sklearn.naive_bayes.CategoricalNB`

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

- Prédire la classe quand le **Weather** = Overcast :

$$P(\text{Yes} \mid \text{Overcast}) = P(\text{Overcast} \mid \text{Yes}) * P(\text{Yes})$$



$\alpha = 1$: $P(\text{Overcast} \mid \text{Yes}) = (4 + 1) / (9 + (1 * 3))$

Solution – Smoothing (Lissage) : It adds a small constant (α) to each count.

$$P(X_j = x_j \mid Y = y) = \frac{N_{y,x_j} + \alpha}{N_y + \alpha \cdot n_j}$$

La classification naïve bayésienne

[sklearn.naive_bayes.CategoricalNB](#)

In summary

Concept	In <code>CategoricalNB</code>
Type of smoothing	Additive (Laplace / Lidstone)
Parameter	<code>alpha</code>
Default	<code>alpha=1.0</code>
Purpose	Avoid zero probabilities for unseen categories

La classification naïve bayésienne

1. Import necessary modules
2. Load & explore the dataset
3. Split the DataFrame into features (X) and target/class (y)
4. Create training and test sets
5. Encode categorical data as numbers : LabelEncoding
6. Train the model
7. Predict and Evaluate : Accuracy & Confusion matrix

La classification naïve bayésienne

sklearn.naive_bayes.CategoricalNB

Import necessary modules : scikit-learn package

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB, CategoricalNB
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import OneHotEncoder, LabelEncoder
```

La classification naïve bayésienne

sklearn.naive_bayes.CategoricalNB

Load & explore the dataset : Exercice 2 - Série TD 2

```
lst_data = [
    ['jeune', 'f', 'v', 'faible'],
    ['jeune', 'v', 'v', 'eleve'],
    ['adulte', 'f', 'f', 'faible'],
    ['senior', 'v', 'f', 'eleve'],
    ['senior', 'f', 'v', 'moyen'],
    ['jeune', 'f', 'f', 'faible'],
    ['adulte', 'v', 'f', 'moyen'],
    ['adulte', 'v', 'v', 'moyen'],
    ['senior', 'f', 'f', 'faible'],
    ['senior', 'v', 'v', 'eleve'],
]
```

```
df = pd.DataFrame(lst_data, columns=['age', 'S1', 'S2', 'risque'])
```

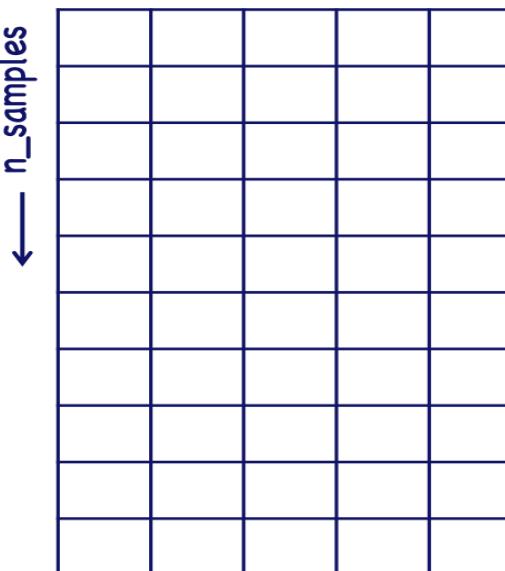
La classification naïve bayésienne

`sklearn.naive_bayes.CategoricalNB`

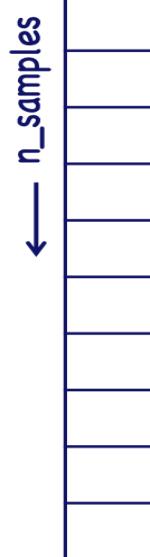
	age	S1	S2	risque
0	jeune	f	v	faible
1	jeune	v	v	eleve
2	adulte	f	f	faible
3	senior	v	f	eleve
4	senior	f	v	moyen
5	jeune	f	f	faible
6	adulte	v	f	moyen
7	adulte	v	v	moyen
8	senior	f	f	faible
9	senior	v	v	eleve



Feature matrix (X)
n_features →



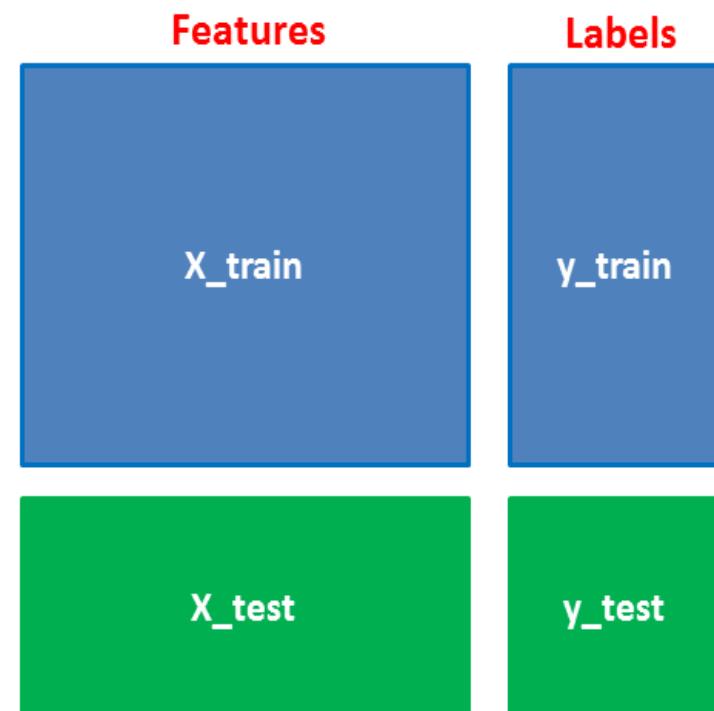
Target (y)



La classification naïve bayésienne

`sklearn.naive_bayes.CategoricalNB`

	age	S1	S2	risque
0	jeune	f	v	faible
1	jeune	v	v	eleve
2	adulte	f	f	faible
3	senior	v	f	eleve
4	senior	f	v	moyen
5	jeune	f	f	faible
6	adulte	v	f	moyen
7	adulte	v	v	moyen
8	senior	f	f	faible
9	senior	v	v	eleve



La classification naïve bayésienne

```
x = df[['age', 'S1', 'S2']]
```

```
y = df['risque']
```

La classification naïve bayésienne

```
x = df[['age', 'S1', 'S2']]
```

```
y = df['risque']
```

```
X_train, X_test, y_train, y_test =
```

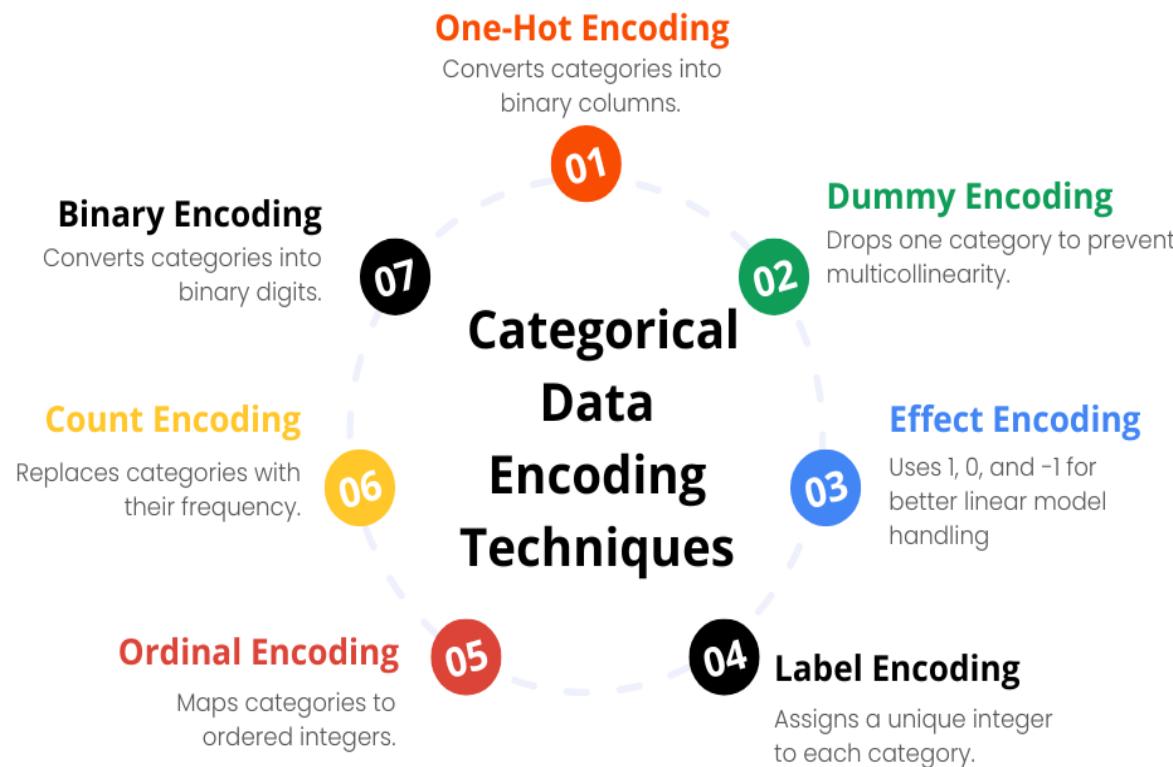
```
train_test_split(X, y,  
                  test_size = 0.3,  
                  random_state = 42)
```

La classification naïve bayésienne

1. Import necessary modules
2. Load & explore the dataset
3. Split the DataFrame into features (X) and target/class (y)
4. Create training and test sets
5. Encode categorical data as numbers : LabelEncoding
6. Train the model
7. Predict and Evaluate : Accuracy & Confusion matrix

La classification naïve bayésienne

- **Encoding : Encode categorical data as numbers**



La classification naïve bayésienne

- Encoding : **One-Hot Encoding**

Human-Readable

Pet
Cat
Dog
Turtle
Fish

Machine-Readable

Cat	Dog	Turtle	Fish
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

La classification naïve bayésienne

- Encoding : **Label Encoding**

Human-Readable

Color
Green
Red
Black
Green

Machine-Readable

Encoding
1
2
3
1

Assign unique
labels to each
category
→

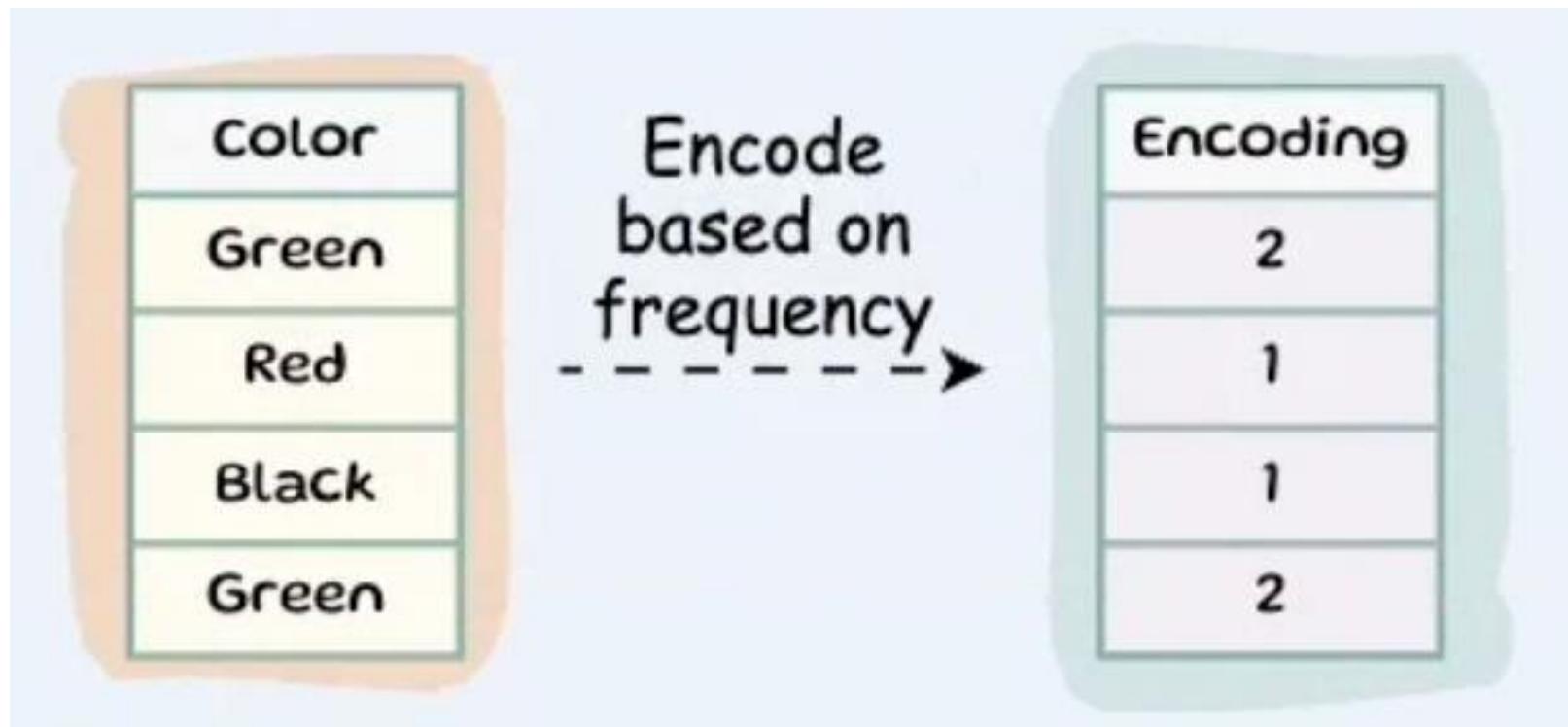
La classification naïve bayésienne

- Encoding : **Ordinal Encoding**



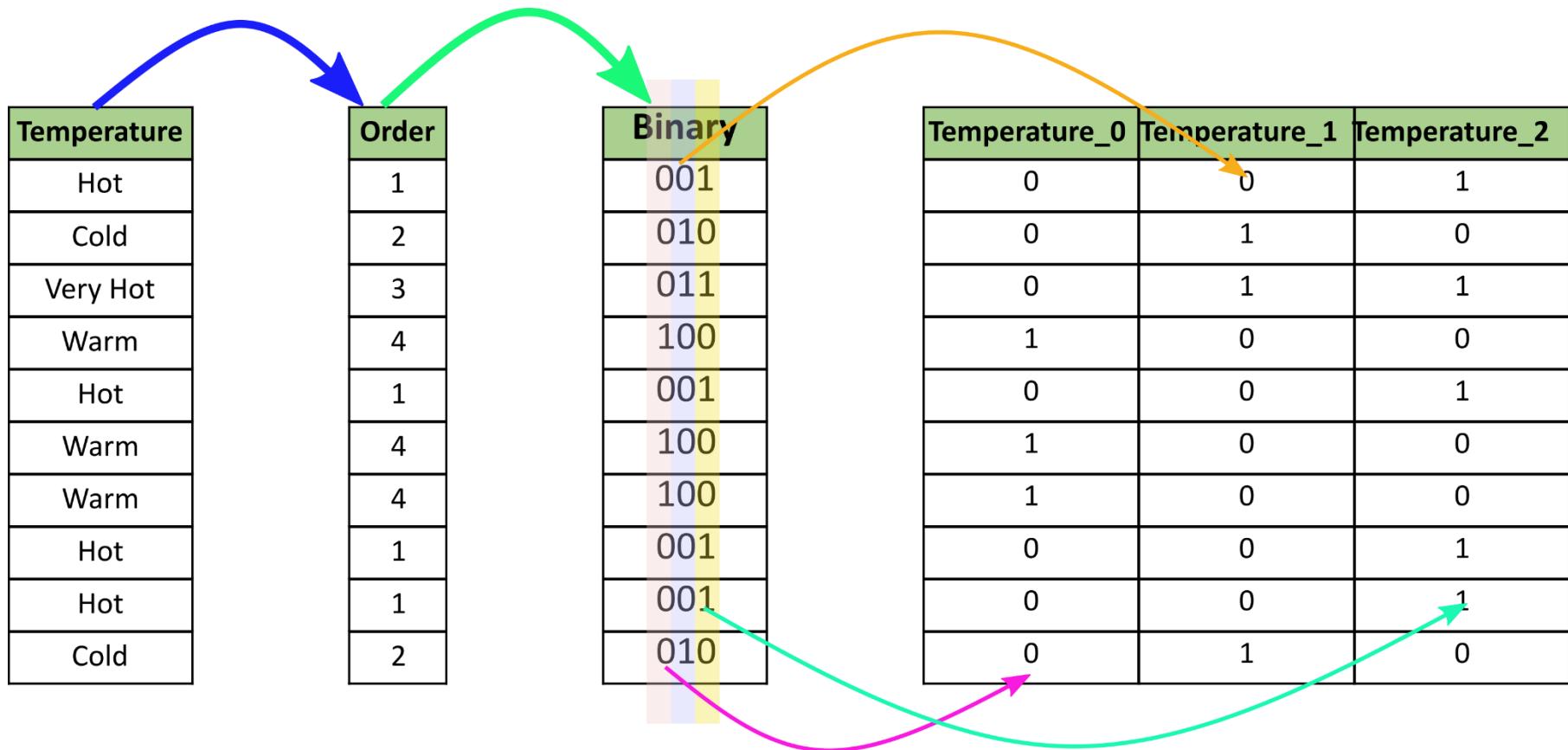
La classification naïve bayésienne

- Encoding : **Count Encoding**



La classification naïve bayésienne

- **Encoding : Binary Encoding**



La classification naïve bayésienne

- Encoding : **Label Encoding**

le = LabelEncoder()

le.fit(X_train)

X_train_le = le.transform(X_train)

La classification naïve bayésienne

- Encoding : **Label Encoding**

```
le = LabelEncoder()
```

```
le.fit(X_train)
```

```
X_train_le = le.transform(X_train)
```

```
X_test_le = le.transform(X_test)
```

La classification naïve bayésienne

`sklearn.naive_bayes.CategoricalNB`

	age	S1	S2	risque
0	jeune	f	v	faible
1	jeune	v	v	eleve
2	adulte	f	f	faible
3	senior	v	f	eleve
4	senior	f	v	moyen
5	jeune	f	f	faible
6	adulte	v	f	moyen
7	adulte	v	v	moyen
8	senior	f	f	faible
9	senior	v	v	eleve

La classification naïve bayésienne

- Encoding : **Label Encoding**

```
encoders = { }      # to store a LabelEncoder for each column
```

```
for col in X_train.columns:
```

```
    le = LabelEncoder()
```

```
    le.fit(X_train[col])
```

```
    X_train_le[col] = le.transform(X_train[col])
```

```
    encoders[col] = le          # Save encoder for later use
```

La classification naïve bayésienne

- Encoding : **Label Encoding**

```
for col in X_test.columns:
```

```
X_test_le[col] = encoders[col].transform(X_test[col])
```

La classification naïve bayésienne

- **Encoding : Label Encoding**

```
print(encoders["age"].classes_)
```

Output : array(['Adulte', 'Jeune', 'Senior'], dtype=object)

Adulte → 0

Jeune → 1

Senior → 2

La classification naïve bayésienne

- Encoding : **Label Encoding**

```
enc_values = [0, 1, 2]
```

```
decoded = encoders["age"].inverse_transform(enc_values)  
print(decoded)
```

Output : ['Adulte', 'Jeune', 'Senior']

Adulte → 0
Jeune → 1
Senior → 2

La classification naïve bayésienne

1. Import necessary modules
2. Load & explore the dataset
3. Split the DataFrame into features (X) and target/class (y)
4. Create training and test sets
5. Encode categorical data as numbers : LabelEncoding
6. Train the model
7. Predict and Evaluate : Accuracy & Confusion matrix

La classification naïve bayésienne

- **Train the model CategoricalNB**

CategoricalNB

```
class sklearn.naive_bayes.CategoricalNB(*, alpha=1.0, force_alpha=True,  
fit_prior=True, class_prior=None, min_categories=None)           [source]
```

clf = **CategoricalNB(alpha=1.0)**

clf.**fit(X_train_le, y_train)**

La classification naïve bayésienne

- **Train the model CategoricalNB**

Attributes:

category_count_ : list of arrays of shape (n_features,)

Holds arrays of shape (n_classes, n_categories of respective feature) for each feature. Each array provides the number of samples encountered for each class and category of the specific feature.

class_count_ : ndarray of shape (n_classes,)

Number of samples encountered for each class during fitting. This value is weighted by the sample weight when provided.

classes_ : ndarray of shape (n_classes,)

Class labels known to the classifier

La classification naïve bayésienne

x_train

	age	S1	S2
0	jeune	f	v
7	adulte	v	v
2	adulte	f	f
9	senior	v	v
4	senior	f	v
3	senior	v	f
6	adulte	v	f

y_train

0	faible
7	moyen
2	faible
9	eleve
4	moyen
3	eleve
6	moyen

clf.feature_names_in_

```
array(['age', 'S1', 'S2'], dtype=object)
```

clf.classes_

```
array(['eleve', 'faible', 'moyen'], dtype=
```

clf.class_count_

```
array([2., 2., 3.])
```

P(class)

La classification naïve bayésienne

x_train

age S1 S2

0	jeune	f	v
7	adulte	v	v
2	adulte	f	f
9	senior	v	v
4	senior	f	v
3	senior	v	f
6	adulte	v	f

y_train

0	faible
7	moyen
2	faible
9	eleve
4	moyen
3	eleve
6	moyen

- Array per feature
- Array (row, col) = (**cls**, **feature_values**)
- $P(x_i|cls)$

clf.category_count_

```
[array([[0., 0., 2.],  
       [1., 1., 0.],  
       [2., 0., 1.]]),  
 array([[0., 2.],  
       [2., 0.],  
       [1., 2.]]),  
 array([[1., 1.],  
       [1., 1.],  
       [1., 2.]])]
```

La classification naïve bayésienne

X_train

	age	S1	S2
--	-----	----	----

0	jeune	f	v
7	adulte	v	v
2	adulte	f	f
9	senior	v	v
4	senior	f	v
3	senior	v	f
6	adulte	v	f

y_train

0	faible
7	moyen
2	faible
9	eleve
4	moyen
3	eleve
6	moyen

- Array per feature
- Array (row, col) = (cls, feature_values)
- $P(\mathbf{x}_i|\text{cls})$

clf.category_count_

```
[array([[0., 0., 2.],  
       [1., 1., 0.],  
       [2., 0., 1.]])]
```

	Adulte	Jeune	Senior
Elevé	0	0	2
Faible	1	1	0
Moyen	2	0	1

La classification naïve bayésienne

- **Predict and Evaluate : Accuracy**

```
y_preds = clf.predict(X_test_le)
```

```
accuracy_score(y_test, y_preds)
```

```
classification_report(y_test, y_preds)
```

La classification naïve bayésienne

- Predict class of a new example : Senior, V, F, ?

```
y_pred = clf.predict([2, 1, 0]) # Based on encoded_values
```

Adulte → 0
Jeune → 1
Senior → 2

F → 0
V → 1

F → 0
V → 1

```
> array(['eleve'], dtype='<U6')
```

La classification naïve bayésienne

- Predict class of a new example : Senior, V, F, ?

```
y_pred = clf.predict_proba([2, 1, 0])
```

Adulte → 0

Jeune → 1

Senior → 2

F → 0

V → 1

F → 0

V → 1

```
array(['eleve', 'faible', 'moyen'], dtype=
```

```
> array([[0.40540541, 0.27027027, 0.32432432]])
```

La classification naïve bayésienne

- Predict class of a new example : Senior, V, F, ?

New example

```
new_sample = pd.DataFrame([["Senior", "V", "F"]],  
                           columns=["age", "S1", "S2"])
```

Encode it

```
for col in new_sample.columns:  
    new_sample_le[col] = encoders[col].transform(new_sample[col])
```

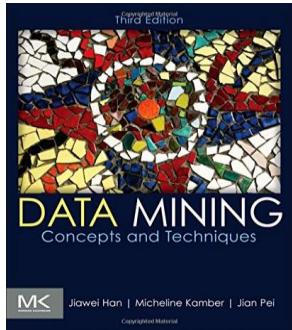
Predict

```
y_pred = clf.predict(new_sample_le)
```

La classification naïve bayésienne

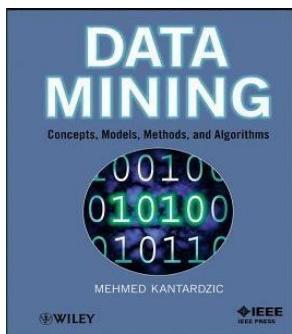
1. Import necessary modules
2. Load & explore the dataset
3. Split the DataFrame into features (X) and target/class (y)
4. Create training and test sets
5. Encode categorical data as numbers : LabelEncoding
6. Train the model
7. Predict and Evaluate : Accuracy & Confusion matrix

Ressources



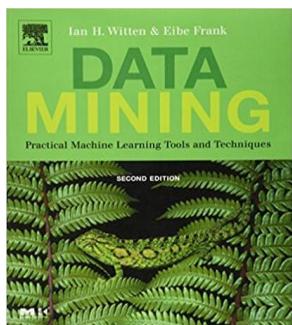
Data Mining : concepts and techniques, 3rd Edition

- ✓ Auteur : Jiawei Han, Micheline Kamber, Jian Pei
- ✓ Éditeur : Morgan Kaufmann Publishers
- ✓ Edition : Juin 2011 - 744 pages - ISBN 9780123814807



Data Mining : concepts, models, methods, and algorithms

- ✓ Auteur : Mehmed Kantardzi
- ✓ Éditeur : John Wiley & Sons
- ✓ Edition : Aout 2011 – 552 pages - ISBN : 9781118029121



Data Mining: Practical Machine Learning Tools and Techniques

- ✓ Auteur : Ian H. Witten & Eibe Frank
- ✓ Éditeur : Morgan Kaufmann Publishers
- ✓ Edition : Juin 2005 - 664 pages - ISBN : 0-12-088407-0