# Naïve Bayes Classifier

## Python Programming Lab

05506231 Statistics and Probability

Asst. Prof. Dr.Anantaporn Hanskunatai

# Outline

- Naïve Bayes Classifier
- Case study on Loan Prediction
- Python Programming for Loan Prediction

# Naïve Bayes Classifier

- apply Bayes theorem to classification problem

$$P(A|B) = \frac{P(A)P(B|A)}{P(B)} \Rightarrow P(class|attribute) = \frac{P(class)P(attribute|class)}{P(attribute)}$$

- given a training set

$$P(class|a_1, a_2, ..., a_n) = \frac{P(class)P(a_1, a_2, ..., a_n|class)}{P(a_1, a_2, ..., a_n)}$$

where $a_i$ is the attribute value of the query instance

$$P(a_1, a_2, ..., a_n|class|) = \prod_{i=1}^{n} P(a_i|class)$$

$$= P(a_i|class)P(a_2|class)...P(a_n|class)$$

$$v_{NB} = \underset{v_j \in v}{\arg\max}\, P(v_j)\prod_{i=1}^{n} P(a_i|v_j)$$

# Case study on Loan Prediction

- Use simple_loan.csv dataset to compute all conditional probabilities (Naïve bayes model)
- Input: age, employed, own_house, credit
- Output: Loan prediction of each customer (target)
- Manual Computing
- Coding with Google Colab

# Dataset: `simple_loan.csv`

| age | employed | own_house | credit | target |
|---|---|---|---|---|
| young | FALSE | n | fair | no |
| young | FALSE | n | good | no |
| young | TRUE | n | good | yes |
| young | TRUE | y | fair | yes |
| young | FALSE | n | fair | no |
| middle | FALSE | n | fair | no |
| middle | FALSE | n | good | no |
| middle | TRUE | y | good | yes |
| middle | FALSE | y | excellent | yes |
| middle | FALSE | y | excellent | yes |
| old | FALSE | y | excellent | yes |
| old | FALSE | y | good | yes |
| old | TRUE | n | good | yes |
| old | TRUE | n | excellent | yes |
| old | FALSE | n | fair | no |
| old | FALSE | n | excellent | yes |
| young | TRUE | y | fair | yes |

# Naïve Bayes Classifier (Manual Computing)

| age | employed | own_house | credit | target |
|---|---|---|---|---|
| young | FALSE | n | fair | no |
| young | FALSE | n | good | no |
| young | TRUE | n | good | yes |
| young | TRUE | y | fair | yes |
| young | FALSE | n | fair | no |
| middle | FALSE | n | fair | no |
| middle | FALSE | n | good | no |
| middle | TRUE | y | good | yes |
| middle | FALSE | y | excellent | yes |
| middle | FALSE | y | excellent | yes |
| old | FALSE | y | excellent | yes |
| old | FALSE | y | good | yes |
| old | TRUE | n | good | yes |
| old | TRUE | n | excellent | yes |
| old | FALSE | n | fair | no |
| old | FALSE | n | excellent | yes |
| young | TRUE | y | fair | yes |

**P(target = “no”) = 6/17 = 0.3529**

**P(target = “yes”) = 11/17= 0.6471**

P(age = “middle” | target = “no”) = 2/6

P(age = “old” | target = “no”) = 1/6

P(age = “young” | target = “no”) = 3/6

P(age = “middle” | target = “yes”)= 3/11

P(age = “old” | target = “yes”) = 5/11

P(age = “young” | target = “yes”) = 3/11

P(employed=“false” | target=“no”) = 6/6

P(employed=“true” | target=“no”) = 0/6

P(employed=“false” | target=“yes”) = 5/11

P(employed=“true” | target=“yes”) = 6/11

P(own_house = “n” | target=“no”) = 6/6

P(own_house = “y” | target=“no”) = 0/6

P(own_house = “n” | target= “yes”) = 4/11

P(own_house = “y” | target=“yes”) = 7/11

P(credit= “excellent” | target=“no”) = 0/6

P(credit= “fair” | target=“no”) = 4/6

P(credit= “good” | target=“no”) = 2/6

P(credit= “excellent” | target=“yes”) = 5/11

P(credit= “fair” | target=“yes”) = 2/11

P(credit= “good” | target=“yes”) = 4/11

# Prediction a New Customer

- a new customer X
- X = (age ="old", employed = "false", own_house = "n", credit= "good")

**P(target = "no") = 6/17 = 0.3529** **P(target = "yes") = 11/17= 0.6471**

P(age = "old" | target = "no") = 1/6

P(age = "old" | target = "yes") = 5/11

P(employed="false" | target="no") = 6/6

P(employed="false" | target="yes") = 5/11

P(own_house = "n" | target="no") = 6/6

P(own_house = "n" | target= "yes") = 4/11

P(credit= "good" | target="no") = 2/6

P(credit= "good" | target="yes") = 4/11

$\hat{P}(v_j)\prod_{i=1}^{n} P(a_i|v_j)$ When $v_j$= target="no"

= (6/17) x (1/6) x (6/6) x (6/6) x (2/6) = 0.019608

$\hat{P}(v_j)\prod_{i=1}^{n} P(a_i|v_j)$ When $v_j$= target="yes"

= (11/17) x (5/11) x (5/11) x (4/11) x (4/11) = 0.017678

**Therefore, X belongs to class ("target= no")**

# Prediction a New Customer

- a new customer X
- X = (age ="middle", employed = "true", own_house = "y", credit= "fair")

**P(target = "no") = 6/17 = 0.3529** **P(target = "yes") = 11/17= 0.6471**

P(age = "middle" | target = "no") = 2/6

P(age = "middle" | target = "yes")= 3/11

P(employed="true" | target="no") = 0/6

P(employed="true" | target="yes") = 6/11

P(own_house = "y" | target="no") = 0/6

P(own_house = "y" | target="yes") = 7/11

P(credit= "fair" | target="no") = 4/6

P(credit= "fair" | target="yes") = 2/11

$\hat{P}(v_j)\prod_{i=1}^{n} P(a_i|v_j)$ When $v_j$= target="no"

= (6/17) x (2/6) x 0 x 0 x (4/6) = 0

$\hat{P}(v_j)\prod_{i=1}^{n} P(a_i|v_j)$ When $v_j$= target="yes"

= (11/17) x (3/11) x (6/11) x (7/11) x (2/11) = 0.011137

**Therefore, X belongs to class ("target= yes")**

# Flowchart

**Upload and Read Data File**

- Separate data between dependent and independent variables

**Label Encoding**

- Transform string to numeric

**Model Construction**

- Compute all conditional probabilities

**Model Prediction**

- Predict unknown data

# Upload and Read Data File

```python
from google.colab import files
uploaded = files.upload()
```

```python
import numpy as np
import pandas as pd
df= pd.read_csv('simple_loan.csv')
```

- View data in DataFrame df

```python
df
```

| | age | employed | own_house | credit | target |
|---|---|---|---|---|---|
| 0 | young | False | n | fair | no |
| 1 | young | False | n | good | no |
| 2 | young | True | n | good | yes |
| 3 | young | True | y | fair | yes |
| 4 | young | False | n | fair | no |
| 5 | middle | False | n | fair | no |
| 6 | middle | False | n | good | no |
| 7 | middle | True | y | good | yes |
| 8 | middle | False | y | excellent | yes |
| 9 | middle | False | y | excellent | yes |
| 10 | old | False | y | excellent | yes |
| 11 | old | False | y | good | yes |
| 12 | old | True | n | good | yes |
| 13 | old | True | n | excellent | yes |
| 14 | old | False | n | fair | no |
| 15 | old | False | n | excellent | yes |
| 16 | young | True | y | fair | yes |

# Upload and Read Data File

- Separate data between dependent and independent variables

```
X=df.drop(['target'], axis=1)
y=df.target
```

```
X
```

| | age | employed | own_house | credit |
|---|---|---|---|---|
| 0 | young | False | n | fair |
| 1 | young | False | n | good |
| 2 | young | True | n | good |
| 3 | young | True | y | fair |
| 4 | young | False | n | fair |
| 5 | middle | False | n | fair |
| 6 | middle | False | n | good |
| 7 | middle | True | y | good |
| 8 | middle | False | y | excellent |
| 9 | middle | False | y | excellent |
| 10 | old | False | y | excellent |
| 11 | old | False | y | good |
| 12 | old | True | n | good |
| 13 | old | True | n | excellent |
| 14 | old | False | n | fair |
| 15 | old | False | n | excellent |
| 16 | young | True | y | fair |

```
y
```

```
0     no
1     no
2    yes
3    yes
4     no
5     no
6     no
7    yes
8    yes
9    yes
10   yes
11   yes
12   yes
13   yes
14    no
15   yes
16   yes
Name: target, dtype: object
```

# Label Encoding

```
from sklearn.preprocessing import LabelEncoder
def labelEncode(data,columns):
    for i in columns:
        lb=LabelEncoder().fit_transform(data[i])
        data[i+'_'] = lb
```

```
f_columns=['age', 'employed', 'own_house', 'credit']
labelEncode(X,f_columns)
```

```
y_le=LabelEncoder()
y1=y_le.fit_transform(y)
```

```
y1
```

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

No= 0 Yes = 1

# Label Encoding

```
X1=X[['age_', 'employed_', 'own_house_', 'credit_']]
```

X

| | age | employed | own_house | credit | age_ | employed_ | own_house_ | credit_ |
|---|---|---|---|---|---|---|---|---|
| 0 | young | False | n | fair | 2 | 0 | 0 | 1 |
| 1 | young | False | n | good | 2 | 0 | 0 | 2 |
| 2 | young | True | n | good | 2 | 1 | 0 | 2 |
| 3 | young | True | y | fair | 2 | 1 | 1 | 1 |
| 4 | young | False | n | fair | 2 | 0 | 0 | 1 |
| 5 | middle | False | n | fair | 0 | 0 | 0 | 1 |
| 6 | middle | False | n | good | 0 | 0 | 0 | 2 |
| 7 | middle | True | y | good | 0 | 1 | 1 | 2 |
| 8 | middle | False | y | excellent | 0 | 0 | 1 | 0 |
| 9 | middle | False | y | excellent | 0 | 0 | 1 | 0 |
| 10 | old | False | y | excellent | 1 | 0 | 1 | 0 |
| 11 | old | False | y | good | 1 | 0 | 1 | 2 |
| 12 | old | True | n | good | 1 | 1 | 0 | 2 |
| 13 | old | True | n | excellent | 1 | 1 | 0 | 0 |
| 14 | old | False | n | fair | 1 | 0 | 0 | 1 |
| 15 | old | False | n | excellent | 1 | 0 | 0 | 0 |
| 16 | young | True | y | fair | 2 | 1 | 1 | 1 |

X1

| | age_ | employed_ | own_house_ | credit_ |
|---|---|---|---|---|
| 0 | 2 | 0 | 0 | 1 |
| 1 | 2 | 0 | 0 | 2 |
| 2 | 2 | 1 | 0 | 2 |
| 3 | 2 | 1 | 1 | 1 |
| 4 | 2 | 0 | 0 | 1 |
| 5 | 0 | 0 | 0 | 1 |
| 6 | 0 | 0 | 0 | 2 |
| 7 | 0 | 1 | 1 | 2 |
| 8 | 0 | 0 | 1 | 0 |
| 9 | 0 | 0 | 1 | 0 |
| 10 | 1 | 0 | 1 | 0 |
| 11 | 1 | 0 | 1 | 2 |
| 12 | 1 | 1 | 0 | 2 |
| 13 | 1 | 1 | 0 | 0 |
| 14 | 1 | 0 | 0 | 1 |
| 15 | 1 | 0 | 0 | 0 |
| 16 | 2 | 1 | 1 | 1 |

# Model Construction

```python
from sklearn.naive_bayes import CategoricalNB
model=CategoricalNB()
model.fit(X1,y1)
```

```python
print(model.feature_log_prob_)
```

**feature_log_prob_ : *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 empirical log probability of categories given the respective feature and class, `P(x_i|y)`.

```
print(model.feature_log_prob_)

[array([[-1.09861229, -1.5040774 , -0.81093022],
       [-1.25276297, -0.84729786, -1.25276297]]), array([[-0.13353139, -2.07944154],
       [-0.77318989, -0.61903921]]), array([[-0.13353139, -2.07944154],
       [-0.95551145, -0.48550782]]), array([[-2.19722458, -0.58778666, -1.09861229],
       [-0.84729786, -1.54044504, -1.02961942]])]
```

# Model Construction

## 1.9.5. Categorical Naive Bayes

`CategoricalNB` implements the categorical naive Bayes algorithm for categorically distributed data. It assumes that each feature, which is described by the index $i$, has its own categorical distribution.

For each feature $i$ in the training set $X$, `CategoricalNB` estimates a categorical distribution for each feature i of X conditioned on the class y. The index set of the samples is defined as $J = \{1, \dots, m\}$, with $m$ as the number of samples.

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

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

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

`CategoricalNB` assumes that the sample matrix $X$ is encoded (for instance with the help of `OrdinalEncoder`) such that all categories for each feature $i$ are represented with numbers $0, ..., n_i - 1$ where $n_i$ is the number of available categories of feature $i$.

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

# Model Interpretation

```
print(model.category_count_)
```

```
print(model.category_count_)

[array([[2., 1., 3.],
       [3., 5., 3.]]), array([[6., 0.],
       [5., 6.]]), array([[6., 0.],
       [4., 7.]]), array([[0., 4., 2.],
       [5., 2., 4.]])]
```

Count(credit=excellent && target=no) = 0
Count(credit=fair && target=no) = 4
Count(credit=good && target=no) = 2

Count(credit=excellent && target=yes) = 5
Count(credit=fair && target=yes) = 2
Count(credit=good && target=yes) = 4

Count(age=middle && target=no) = 2
Count(age=old && target=no) = 1
Count(age=young && target=no) = 3

Count(age=middle && target=yes) = 3
Count(age=old && target=yes) = 5
Count(age=young && target=yes) = 3

Count(employed=false && target=no) = 6
Count(employed=true && target=no) = 0

Count(employed=false && target=yes) = 5
Count(employed=true && target=yes) = 6

Count(own_house=n && target=no) = 6
Count(own_house=y && target=no) = 0

Count(own_house=n && target=yes) = 4
Count(own_house=y && target=yes) = 7

# Model Interpretation

```
print(model.feature_log_prob_)

[array([[-1.09861229, -1.5040774 , -0.81093022],
       [-1.25276297, -0.84729786, -1.25276297]]), array([[-0.13353139, -2.07944154],
       [-0.77318989, -0.61903921]]), array([[-0.13353139, -2.07944154],
       [-0.95551145, -0.48550782]]), array([[-2.19722458, -0.58778666, -1.09861229],
       [-0.84729786, -1.54044504, -1.02961942]])]
```

$\text{Log}(x) = \text{Log}_e(x)$ or $\ln(x)$

Log(P(age=middle|target=no)) = -1.09861229   Log(P(age=old|target=no)) = -1.5040774   Log(P(age=young|target=no)) = -0.81093022

Log(P(age=middle|target=yes)) = -1.25276297   Log(P(age=old|target=yes)) = -0.84729786   Log(P(age=young|target=yes)) = -1.25276297

Log(P(employed=false|target=no)) = -0.13353139   Log(P(employed=true|target=no)) = -2.07944154

Log(P(employed=false|target=yes)) = -0.77318989   Log(P(employed=true|target=yes)) = -0.61903921

Log(P(own_house=n|target=no)) = -0.13353139   Log(P(own_house=y|target=no)) = -2.07944154

Log(P(own_house=n|target=yes)) = -0.95551145   Log(P(own_house=y|target=yes)) = -0.48550782

Log(P(credit=excellent|target=no)) = -2.19722458   Log(P(credit=fair|target=no)) = -0.58778666   Log(P(credit=good|target=no)) = -1.09861229

Log(P(credit=excellent|target=yes)) = -0.84729786   Log(P(credit=fair|target=yes)) = -1.54044504   Log(P(credit=good|target=yes)) = -1.02961942

# Model Prediction

1. age ="middle", employed = "true", own_house = "y", credit= "fair"
2. age ="old", employed = "false", own_house = "n", credit= "good"

```
new_input=[[0,1,1,1],[1,0,0,2]]
y_prob_pred = model.predict_proba(new_input)
```

```
y_prob_pred
```

```
array([[0.0721808 , 0.9278192 ],
       [0.53238717, 0.46761283]])
```

```
y_new_predict=model.predict(new_input)
n=1
for i in y_new_predict:
    print( 'No' ,n, '=>: ',y_le.classes_[i])
    n=n+1
```

```
y_new_predict=model.predict(new_input)
class_names=list(y_le.classes_)

n=1
for i in y_new_predict:
  print( 'No' ,n, '=>: ',class_names[i])
  n=n+1
```

```
No 1 =>:  yes
No 2 =>:  no
```

# Prediction a New Customer

```
y_prob_pred
array([[0.0721808 , 0.9278192 ],
       [0.53238717, 0.46761283]])
```

- a new customer X
- X = (age ="middle", employed = "true", own_house = "y", credit= "fair")

  0 1 1 1

**P(target = "no") = 6/17 = 0.3529** **P(target = "yes") = 11/17= 0.6471**

P(age = "middle" | target = "no") = (2+1)/(6+3)

P(age = "middle" | target = "yes")= (3+1)/(11+3)

P(employed="true" | target="no") = (0+1)/(6+2)

P(employed="true" | target="yes") = (6+1)/(11+2)

P(own_house = "y" | target="no") = (0+1)/(6+2)

P(own_house = "y" | target="yes") = (7+1)/(11+2)

P(credit= "fair" | target="no") = (4+1)/(6+3)

P(credit= "fair" | target="yes") = (2+1)/(11+3)

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

$\hat{P}(v_j)\prod_{i=1}^{n} P(a_i|v_j)$ When $v_j$= target="no"

= (6/17) x (3/9) x (1/8) x (1/8) x (5/9) = 0.001021

= 0.001021 /(0.001021 + 0.011137) = 0.0721808

$\hat{P}(v_j)\prod_{i=1}^{n} P(a_i|v_j)$ When $v_j$= target="yes"

= (11/17) x (4/14) x (7/13) x (8/13) x (3/14) = 0.011137

= 0.011137 /(0.001021 + 0.011137) = 0.9278192

**Therefore, X belongs to class ("target= yes")**

# Prediction a New Customer

```
y_prob_pred
array([[0.0721808 , 0.9278192 ],
       [0.53238717, 0.46761283]])
```

- a new customer X
- X = (age ="old", employed = "false", own_house = "n", credit= "good")

1 0 0 2

**P(target = "no") = 6/17 = 0.3529** **P(target = "yes") = 11/17= 0.6471**

P(age = "old" | target = "no") = (1+1)/(6+3)

P(age = "old" | target = "yes") = (5+1)/(11+3)

P(employed="false" | target="no") = (6+1)/(6+2)

P(employed="false" | target="yes") = (5+1)/(11+2)

P(own_house = "n" | target="no") = (6+1)/(6+2)

P(own_house = "n" | target= "yes") = (4+1)/(11+2)

P(credit= "good" | target="no") = (2+1)/(6+3)

P(credit= "good" | target="yes") = (4+1)/(11+3)

$$\hat{P}(v_j)\prod_{i=1}^{n} P(a_i|v_j)$$ When $v_j$= target="no"

= (6/17) x (2/9) x (7/8) x (7/8) x (3/9) = 0.020016

= 0.020016/(0.020016+ 0.017581) = 0.53238717

$$\hat{P}(v_j)\prod_{i=1}^{n} P(a_i|v_j)$$ When $v_j$= target="yes"

= (11/17) x (6/14) x (6/13) x (5/13) x (5/14) = 0.017581

= 0.017581 /(0.020016+ 0.017581) = 0.46761283

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

**Therefore, X belongs to class ("target= no")**