*Cairo University*
*Faculty of Computers and Artificial Intelligence*
*Machine Learning Course*

# Lab 5 – Part 1: Naïve Bayes

## Lab Outline:

- ➢ Summary of Naïve Bayes
- ➢ Solved example
- ➢ Steps of applying Naïve Bayes to a dataset
- ➢ Pseudocode for Categorical Naïve Bayes (training phase)
- ➢ Practice

## Summary of Naïve Bayes:

Naïve Bayes is a ***supervised*** machine learning algorithm that is used for ***classification*** tasks. It is a ***probabilistic classifier*** that applies ***Bayes' Theorem*** to calculate the probability of a class occurring ***(posterior probability)*** given certain features.

Likelihood
Class Prior Probability

$$P(c \mid x) = \frac{P(x \mid c)\,P(c)}{P(x)}$$

Posterior Probability
Predictor Prior Probability

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

As shown in the image above, the likelihoods and the prior probabilities are calculated to yield the posterior probability. Naïve Bayes will return the class which has the ***maximum posterior*** probability (out of a group of classes).

$$\boxed{c_{MAP} \equiv \underset{c \in C}{\operatorname{argmax}}\ P(c \mid X)}$$

Naïve Bayes operates under a couple of key ***"naïve" assumptions*** which are often violated in real-world scenarios:

1) It assumes that predictors (features) are conditionally independent, or unrelated to any of the other features in the model.
2) It also assumes that all features contribute equally to the outcome.

Naïve Bayes is also one of the ***generative*** learning algorithms as it seeks to ***model the distribution*** of inputs of a given class or category. Unlike discriminative classifiers, like logistic regression, it does not learn which features are most important to differentiate between classes.

The types of the Naïve Bayes classifier that exist are based on the distributions of the feature values. For example, there is Gaussian Naïve Bayes, Bernoulli Naïve Bayes, and so on.

## Solved Example:

Given the following dataset containing the genders, heights, weights and T-shirt sizes of some customers, use Naïve Bayes to predict the T-shirt size of a male customer with medium height and whose weight is 62.5 kg.

| Gender | Height | Weight (kg) | T-shirt Size |
|---|---|---|---|
| male | short | 63 | M |
| female | short | 58 | Other |
| male | short | 59 | Other |
| female | short | 60 | Other |
| female | short | 64 | M |
| female | medium | 60 | M |
| male | medium | 61 | M |
| male | medium | 64 | L |
| female | medium | 61 | L |
| male | medium | 65 | L |
| male | tall | 69 | L |
| male | tall | 76 | Other |
| female | tall | 66 | L |
| male | tall | 88 | Other |

We will calculate the following posterior probabilities and choose the maximum:

***p(M | male,medium,62.5)***

***p(L | male,medium,62.5)***

***p(Other | male,medium,62.5)***

Let's start with *p(M | male,medium,62.5):*

***p(M | male,medium,62.5) = p(male | M) * p(medium | M) * p(62.5 | M) * p(M)***

- *p(male | M)* = number of rows with "male" and "M" divided by the number of rows with "M"

  ***p(male | M) = 2/4***

- ***p(medium | M) = 2/4***

- Since the weight is a continuous feature, to calculate *p(62.5 | M)* we will assume the weight follows a ***Gaussian distribution***, so we will use:

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

  *$\mu$(Weight when y=M) = 62*

  *$\sigma^2$(Weight when y=M) = 2.5*

  ***p(62.5 | M) = 0.24***

- *p(M)* = number of rows with "M" divided by the total number of rows

  ***p(M) = 4/14***

Therefore:

***p(M | male,medium,62.5) = (2/4) * (2/4) * (0.24) * (4/14) = 0.017***

Similarly, we will calculate the posterior probabilities for the other two classes*:*

***p(L | male,medium,62.5) = (3/5) * (3/5) * (0.096) * (5/14) = 0.012***

***p(Other | male,medium,62.5) = (3/5) * (0/5) * (0.029) * (5/14) = 0***

Since *p(M | male,medium,62.5)* is the maximum posterior probability, the predicted T-shirt size will be "M".

Notice that *p(Other | male,medium,62.5)* was zero because a "medium" height was found with class "Other" in the train data. This is a ***zero conditional probability problem*** for the test data. The most reliable solution to this problem is to use a ***smoothing technique*** like *m*-estimate or Laplace smoothing.

- ***m*-estimate can be calculated according to the following formula:**

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

where $\boldsymbol{N_{tic}}$ is the number of times category $\boldsymbol{t}$ appears in $\boldsymbol{x_i}$ in the samples which belong to class $\boldsymbol{c}$, $\boldsymbol{m}$ is the number of virtual examples, $\boldsymbol{p}$ is the prior estimate (1 ÷ number of categories of $\boldsymbol{x_i}$), and $\boldsymbol{N_c}$ is the number of samples with class $\boldsymbol{c}$**.**

- **Laplace smoothing can be applied according to the following formula:**

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

where $\boldsymbol{N_{tic}}$ is the number of times category $\boldsymbol{t}$ appears in $\boldsymbol{x_i}$ in the samples which belong to class $\boldsymbol{c}$, $\boldsymbol{\alpha}$ is the smoothing parameter, $\boldsymbol{N_c}$ is the number of samples with class $\boldsymbol{c}$, and $\boldsymbol{n_i}$ is the number of categories of $\boldsymbol{x_i}$

Re-estimating the posterior probabilities using *m*-estimate *(m=1)*:

***p(M | male,medium,62.5) = (5/10) * (7/15) * (0.24) * (4/14) = 0.016***

***p(L | male,medium,62.5) = (7/12) * (10/18) * (0.096) * (5/14) = 0.011***

***p(Other | male,medium,62.5) = (7/12) * (1/18) * (0.029) * (5/14) = 0.00034***

## Steps of Applying Naïve Bayes to a Dataset:

1. **Analyze** the dataset
2. Perform data **preprocessing** *(including **splitting** into train and test sets)*
3. **Fit** the Naïve Bayes model to the training data ***(training phase)***
4. **Assess** the fitted model on the test data
5. Generate **predictions** for new data

## Pseudocode for Categorical Naïve Bayes (Training Phase):

```
Function fit_NB
------------------------------------------------------------
Input: X, y
Output: model_parameters

1:  Set alpha
2:  Let classes be the list of unique values in y
3:  for c in classes
4:      Let c_rows be the rows of X whose corresponding y value = c
5:      Let n_class = c_rows.length
6:      Let c_prior = n_class / y.length
7:      Insert c_prior in model_parameters
8:      for j=0 to X.num_columns
9:          Let xj_categories be the list of unique values of feature X_j
10:         Let n_xj = xj_categories.length
11:         for t in xj_categories
12:             Let n_t = number of rows in c_rows that have X_j = t
13:             Calculate the conditional probability (with Laplace smoothing)::
                    cond_prob = (n_t + alpha) / (n_class + alpha * n_xj)
14:             Insert cond_prob in model_parameters
15:         end
16:     end
17: end
```

*Note: For Gaussian Naïve Bayes, we will replace the code from line 9 to line 14 with code that calculates the mean and variance of $X_j$ in c_rows and inserts them in model_parameters.*

## Practice:

I. Write the code of the function **"fit_NB"** in Python, then use it to train a model on training data from the solved example, and print the parameters of that model.

*Note: You need to implement* ***mixed Naïve Bayes*** *which, depending on* ***each feature's type*** *(continuous and normally distributed or categorical), uses a Gaussian distribution or a categorical distribution for the feature.*

II. Could you provide the manual implementation of the "**fit**" and "**predict**" methods for a Gaussian Naïve Bayes classifier in Python without using libraries that have pre-built Naive Bayes methods?

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

class NaiveBayes:
    def fit(self, X, y):
        // to be implemented

    def predict(self, X):
       // to be implemented

# Load the Iris dataset from a URL
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']
df = pd.read_csv(url, names=names)

# Split dataset into features and target
X = df.iloc[:, :-1].values
y = df.iloc[:, -1].values

# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=1)  # 70% training and 30% test

# Create a Naive Bayes Classifier
nb = NaiveBayes()

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

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

# Model Accuracy
print("Accuracy:", accuracy_score(y_test, y_pred))
```

*Note: In this implementation, **"fit"** calculates the mean, variance, and prior probability for each class, then **"predict"** calculates the Gaussian probability density for each class for each sample, multiplies it by the prior probability, and chooses the class with the highest probability.*