Naive Bayes Classifier
Assignment 1 of Machine Learning I, 2022/2023

Mura Alessio

DIBRIS - Dipartimento di Informatica, Bioingegneria, Robotica e Ingegneria dei Sistemi
Università Degli Studi di Genova

1 Introduction

Naive Bayes is a statistical classification technique based on Bayes Theorem. It is one of the simplest supervised learning algorithms. Naive Bayes classifier is the fast, accurate and reliable algorithm. Naive Bayes classifiers have high accuracy and speed on large datasets.

2 Theory of NBC

2.1 Mathematical theory

Let's call the input *x (observation)*. A Classification is a rule *y()* that, given an input, produces an output called *t (class)*, so that:

$$t = y(x) \tag{1}$$

In order to output the most appropriate class, the classifier choose the classification that has the minimum error probability. In particular, the NBC is based on a *naive* assumption, which basically is not correct but works just the same:

$$P_r(x_1, ..., x_d|t_i) = P_r(x_1|t_i)P_r(x_2|t_i)...P_r(x_d|t_i) \tag{2}$$

The NBC classifier therefore pretends that input variables are all independent of each other.

To select the best class we use a discriminant function $g_i(x)$. This function can be compute as follows:

$$\begin{aligned} g_i(x) = P_r(t_i)[P_r(x_1|t_i)P_r(x_2|t_i)...P_r(x_d|t_i)] = \\ = P_r(t_i)\prod_{j=1}^{d} P_r(x_j|t_d) \end{aligned} \tag{3}$$

After computing every $g_i(x)$, the classifier takes the one that has the highest value, and classifies *x* with *t*.

2.2 Training session

During the training session the classifier acquires data from the training set. In particular the NBC computes every conditional probability, using this formula:

$$P_r(x_j = v_k|t_i) = \frac{\text{number of } x_j = v_j \text{ in class } t_i}{\text{number of instance of } t_i} \tag{4}$$

where v_j is the possible value for the attribute *j*.

2.3 Improvements with Laplace Smoothing

Using a small data set, some combinations that appear in the test set were not encountered in the training set, so their probability is assumed to be zero. When you multiply many terms, just a zero sets the overall results to zero.

Generally speaking, some values of some attributes could not appear in the training set, but are actually inside the entire set. The classifier should know this information, and the **Laplace Smoothing** comes in hand.

With the Laplace smoothing, a *trust factor* **a** is implemented in the conditioned probability computation as follows:

$$P_r(x_j = v_k|t_i) = \frac{(\text{number of } x_j = v_j \text{ in class } t_i) + \mathbf{a}}{(\text{number of instance of } t_i) + \mathbf{a}\text{v}} \tag{5}$$

where *v* is the number of possible values of the variable *x*. When $a < 1$ the classifier trusts his prior belief less than the data, while with $a > 1$ the classifier trusts his prior belief more than the data. Adding *a* results in a probability that is never equal to zero, even if the value does not appear in the training set.

3 THE ASSIGNMENT

The given assignment consist of three tasks:

- **Task 1**: Data pre-processing
- **Task 2**: Build a naive Bayes classifier
- **Task 3**: Improve the classifier with Laplace (additive) smoothing

3.1 Task 1: Data pre-processing

Our dataset is initially stored as .csv files. To import it in Python, it is required to use the function read-csv(), inside Pandas library, in order to store our data into a dataframe. Usually attributes value should be converted in positive integer values, in order to make them easier to manipulate.

The used set is a Weather data set composed by:

- 14 observations;
- 4 attributes (outlook, temperature, humidity, windy);
- 2 classes (play YES, play NO).

The set is built in such a way that the first four columns are the attributes, while the last column is the target of the observation. In particular, outlook can assume three different values *(overcast, rainy or sunny)*, temperature can also assume three different values *(hot, mild or cool)*, humidity can assume two values *(high or normal)* and windy can assume two values *(true or false)*.

3.2 Task 2: Build a naive Bayes classifier

The naive Bayes classifier is divided in two sections: the training section and the classification section. in order to compute the classification, we created a function: *NaiveBayesClassifier()*.

Inside this function, we call other functions useful for the computation:

- *Likelihood()*
- *FinalProbability()*
- *Predictions()*

3.2.1 Likelihood()

In this function we pass three values: the **train-set**; a vector that counts the number of columns' unique values of the dataset (**v**) and our dataframe **df**.

This function is useful for computing the occurrences of *Yes (value = 2)* and *No (value = 1)* in the target column. In fact, the final result will be two different lists containing the occurrences for each of the four attributes (outlook, temperature, humidity, windy) depending on the value of the target column (yes or no).

Moreover, inside this function it is called another function, *LaplaceSmoothing()*, but this will be topic of discussion in subsection 3.3.

3.2.2 FinalProbability()

In this function, we classified each observation that was inside the test set, according to the inferred rule of maximizing the discriminant function $g_i(x)$ (equation 3).

we also called a function, *PriorProbability()*, in order to compute the prior probability of *Yes* and *No*.

The function's outputs are two lists: **final-yes**, which contains the probability of *yes* and **final-no**, which contains the probability of *no*.

3.2.3 Predictions()

This function is quite simple: we compare the final probability of *yes* and the final probability of *no* in order to compute the prediction.

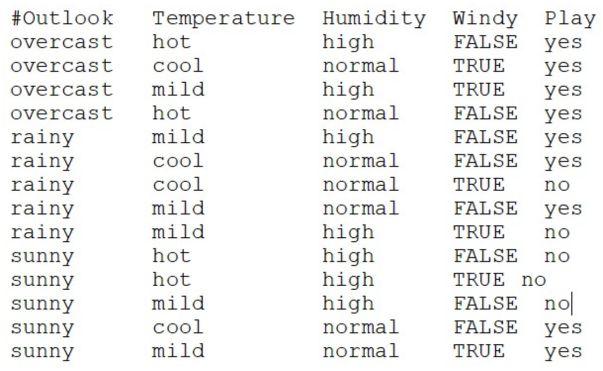
If the probability of *Yes* is greater than the probability of *No* the prediction returns *yes (value: 2)*; whereas if the probability of *No* is greater than the probability of *Yes* the prediction returns *no (value: 1)*.

3.3 Task 3: Improve the classifier with Laplace (additive) smoothing

As seen in the equation 5, the **a** factor is inserted inside the equation 4. The **a** factor is specified inside *LaplaceSmoothing()* function. The classifier is just the same as before, but this time according with the theory explained in subsection 2.3.

We chose to assign it the value of 0.85, in order to make the classifier trusts his prior belief less than the data.

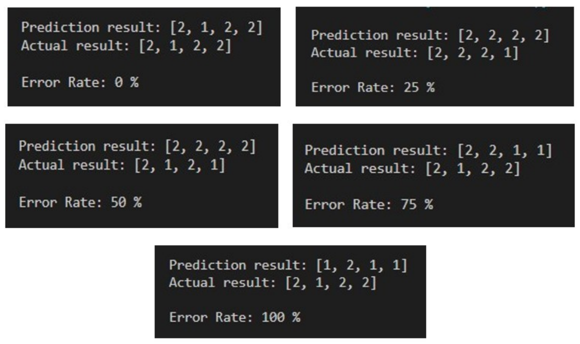
4 RESULTS



#Outlook	Temperature	Humidity	Windy	Play
overcast	hot	high	FALSE	yes
overcast	cool	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
rainy	mild	normal	FALSE	yes
rainy	mild	high	TRUE	no
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
sunny	mild	normal	TRUE	yes

Figure 1. Wheather dataset

The figure 1 shows the dataset in which we computed our classification.



```
Prediction result: [2, 1, 2, 2]
Actual result: [2, 1, 2, 2]

Error Rate: 0 %
```

```
Prediction result: [2, 2, 2, 2]
Actual result: [2, 2, 2, 1]

Error Rate: 25 %
```

```
Prediction result: [2, 2, 2, 2]
Actual result: [2, 1, 2, 1]

Error Rate: 50 %
```

```
Prediction result: [2, 2, 1, 1]
Actual result: [2, 1, 2, 2]

Error Rate: 75 %
```

```
Prediction result: [1, 2, 1, 1]
Actual result: [2, 1, 2, 2]

Error Rate: 100 %
```

Figure 2. Results of classification

Each time we run our program, it will return on the console three parameters: the result of our prediction, the actual values and the error rate of our classification, as we can clearly see in figure 2.

The error rate's value is computed as the number of correct classification divided by the number of observation. Notice that in this test, with only 4 observation inside the test set, the error rate can assume only five fixed values:

- 0 % in case every classification is correct,
- 25 % if there is only one wrong classification,
- 50 % if there are two wrong classifications,
- 75 % if there are three wrong classifications,
- 100 % in case every classification is incorrect.

All the classifications are executed with a training set made of 10 observation and a test set made of 4 observation. With two larger sets the Laplace could be more incisive in the classification process.