

# Naive Bayes algorithm

## Assignment 1a

Calculate, using the Naive Bayes classifier, the two most probable scenarios – playing tennis or not – in the following situation:

Outlook = Sunny

Temperature = Cool

Humidity = High

Wind = Strong

The included program uses the Naive Bayes classifier to determine if the given situation results in playing tennis or not.

A screenshot of the execution of this program is displayed to the right. The program displays the calculation results on-screen, which are summarized below.

First, the program searches for similar training examples. It will try to determine the probability of each possible outcome based on prior knowledge. In this case, there are two possible outcomes:

- PlayTennis = **Yes**;
- PlayTennis = **No**.

For each outcome, the algorithm ‘compares’ the number of features that match the given situation.

The outcome of the given scenario is **Yes**, where:

- 9 of the 14 examples give the outcome **Yes**;
  - o Of which 2 of those 9 return **Sunny**;
  - o Of which 3 of those 9 return **Cool**;
  - o Of which 3 of those 9 return **High**;
  - o Of which 3 of those 9 return **Strong**.

Calculating the sum over all ratios gives the probability for outcome **Yes**, denoted  $P(y)$ :

$$P(y) = \frac{9}{14} \times \frac{2}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{3}{9} = 0.0053$$

Likewise, we calculate the probability for outcome **No**, denoted  $P(n)$ :

$$P(n) = \frac{5}{14} \times \frac{3}{5} \times \frac{1}{5} \times \frac{4}{5} \times \frac{3}{5} = 0.0206$$

Because the probability for outcome  $P(n)$  (PlayTennis = **No**) is greater than  $P(y)$  (PlayTennis = **Yes**), we may conclude that – according to Naive Bayes – we most likely will not want to play tennis in the given scenario, and thus that the algorithm returns **No**.

Days	Outlook	Temperature	Humidity	Wind	PlayTennis
Day 1	Sunny	Hot	High	Weak	No
Day 2	Sunny	Hot	High	Strong	No
Day 3	Overcast	Hot	High	Weak	Yes
Day 4	Rain	Mild	High	Weak	Yes
Day 5	Rain	Cool	Normal	Weak	Yes
Day 6	Rain	Cool	Normal	Strong	No
Day 7	Overcast	Cool	Normal	Strong	Yes
Day 8	Sunny	Mild	High	Weak	No
Day 9	Sunny	Cool	Normal	Weak	Yes
Day 10	Rain	Mild	Normal	Weak	Yes
Day 11	Sunny	Mild	Normal	Strong	Yes
Day 12	Overcast	Mild	High	Strong	Yes
Day 13	Overcast	Hot	Normal	Weak	Yes
Day 14	Rain	Mild	High	Strong	No

Computing results for:  
Sunny, Cool, High, Strong

Compute for P(y):  
 $P(y) = 9 / 14$   
 $P(\text{Sunny}|y) = 2 / 9$   
 $P(\text{Cool}|y) = 3 / 9$   
 $P(\text{High}|y) = 3 / 9$   
 $P(\text{Strong}|y) = 3 / 9$   
 Result  
 $= (9 / 14) * (2 / 9) * (3 / 9) * (3 / 9) * (3 / 9)$   
 $= 0,0053$

Compute for P(n):  
 $P(n) = 5 / 14$   
 $P(\text{Sunny}|n) = 3 / 5$   
 $P(\text{Cool}|n) = 1 / 5$   
 $P(\text{High}|n) = 4 / 5$   
 $P(\text{Strong}|n) = 3 / 5$   
 Result  
 $= (5 / 14) * (3 / 5) * (1 / 5) * (4 / 5) * (3 / 5)$   
 $= 0,0206$

Winning result is:  
PlayTennis = No

# Assignment 1b

Calculate, using the Naive Bayes classifier, the two most probable scenarios – playing tennis or not – in the following situation:

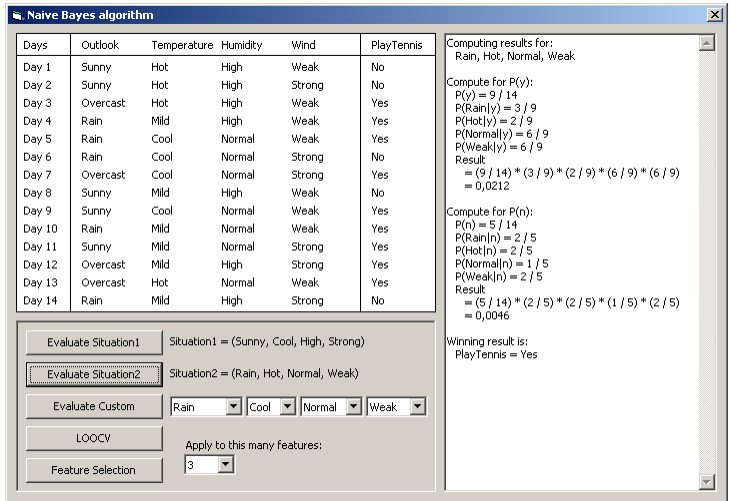
- Outlook = Rain
- Temperature = Hot
- Humidity = Normal
- Wind = Weak

Using the same procedure as described earlier, Naive Bayes calculates the probability of each outcome. In this scenario, the results are:

$$P(y) = \frac{9}{14} \times \frac{3}{9} \times \frac{2}{9} \times \frac{6}{9} \times \frac{6}{9} = 0.0212$$

$$P(n) = \frac{5}{14} \times \frac{2}{5} \times \frac{2}{5} \times \frac{1}{5} \times \frac{2}{5} = 0.0046$$

For this scenario, the outcome would therefore be PlayTennis = No.



# Assignment 2a

Are all features required to come to an accurate classification? Verify this using the Forward Selection (FS) algorithm and the Leave-one-out Cross-validation (LOOCV).

Ideally, the classification of the training set would be 100% accurate<sup>1</sup>. The accuracy can be calculated by comparing the expected sum of all outputs<sup>2</sup> and the computed sum of all outputs.

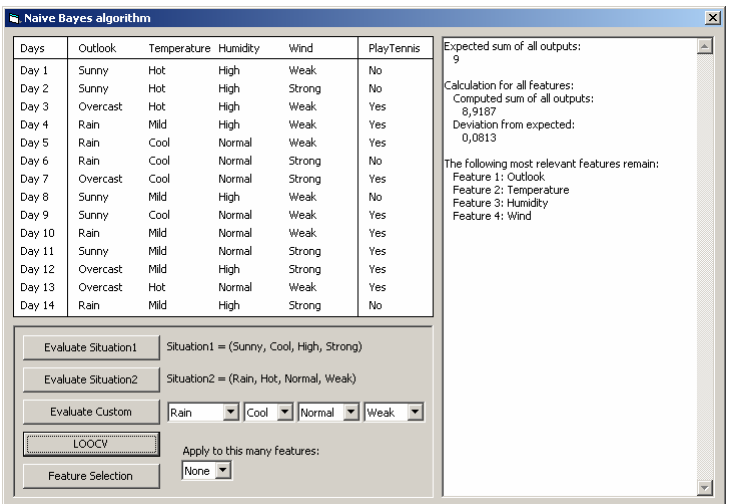
Sometimes a feature isn't related to the outcome and is only 'confusing' the classifier. We could probably improve the accuracy by removing such features. There are two methods of removing irrelevant features:

- Leave-one-out-cross-validation (LOOCV)
- Forward Selection (FS)

The relevancy of a feature can be determined by calculating the difference between the expected result and the computed result, which is known as *feature deviation*.

We can calculate the *total deviation* by subtracting the sum of all feature deviations from the expected sum.

In the PlayTennis example, the expected sum of all outputs is:  
 $9 \times 1 + 5 \times 0 = 9$



<sup>1</sup> In practise, 100% accuracy is usually a sign that your test set is either too small, or you are overfitting your classifier.  
<sup>2</sup> Assuming that your output is a real number. In the PlayTennis example, No corresponds with 0, Yes with 1.

Running the algorithm delivers the deviation from this expected sum. Initially, without removing any features, this is 0.0813.

Both methods determine the best feature(s) using the feature relevancy described above.

- LOOCV removes a feature that is responsible for the largest total deviation;
- FS selects one feature that gives the smallest deviation, disregarding all others.

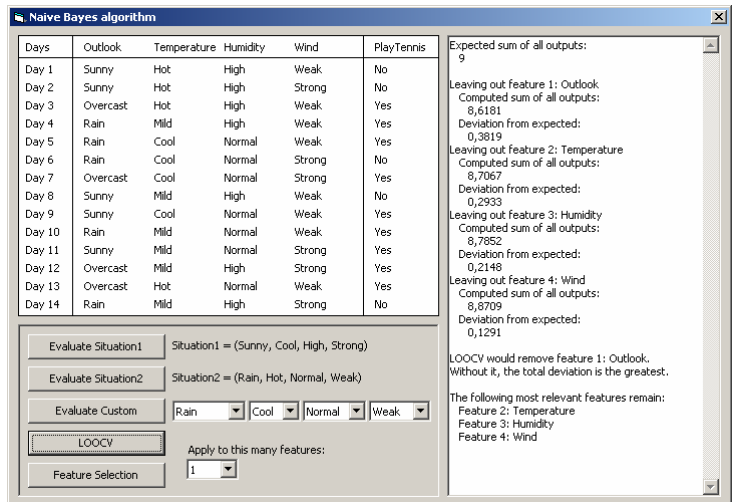
The included program provides the possibility to select how many features you want the function to run on. The methods do this as follows:

- LOOCV removes the ‘worst’  $n$  features;
- FS selects the  $n$  ‘best’ features.

Essentially, by selecting  $n$  features for LOOCV one can acquire the same results as specifying  $N - n$  for FS, where  $N$  is the total number of features.

Is leaving out features useful in the PlayTennis example? Using all four features, our classification error is represented by a total deviation of 0.0813. We can experiment by removing features to see if this number can be reduced.

Removing any single variable using LOOCV gives us only worse results; the deviation is in all four cases larger than 0.0813. Leaving out two or three features only makes matters worse, from which we can conclude that we should stick with all four features for this example.



Note that FS gives the same results but in reverse order.

## Assignment 2b

*What is the predicted class in the two above situations after selecting variables?*

When selecting variables, one must omit all other features from the calculation, as these are no longer participating.

After removing the features **Humidity** and **Wind** from the scenarios from assignment 1a, we get a different result:

$$P(y) = \frac{9}{14} \times \frac{2}{9} \times \frac{3}{9} = 0.0476$$

$$P(n) = \frac{5}{14} \times \frac{3}{5} \times \frac{1}{5} = 0.0429$$

By omitting two features, the chances have shifted in the favor of playing tennis, and thus the algorithm returns PlayTennis = **Yes**.

## Assignment 3

*How do the results compare to the K-nearest-neighbor algorithm?*

Running the K-nearest-neighbor (KNN) algorithm on the scenarios from assignment 1a and 1b indeed give different results, respectively PlayTennis = **No** and PlayTennis = **Yes**.

As KNN simply takes a handful of similar scenarios ('nearest neighbors'), it's performance is rather sensitive to outliers. For that reason, Naive Bayes is a more accurate classifier for predicting outcomes according to a consistent pattern based upon the training set, while KNN can exclusively guarantee that very similar situations will give a similar result.