Notes on Naive Bayes

Paulo S. A. Sousa

2023-04-04

Abstract

The basic ideas underlying the Naive Bayes model are exposed. Its implementation in Python is also presented.

1 Introduction

Naive Bayes is another prediction model for classification problems. It operates by neglecting a fundamental theorem of Statistics: The Bayes theorem. Surprisingly, despite using a wrong version of such a theorem, this model performs extremely well in certain problems and is fast.

A limitation of this model is that the predictors must be all categorical. There are several ways to overcome that limitation.

2 The Bayes theorem

Consider the two events A and B. Bayes' theorem states that:

$$P\left(A|B\right) = \frac{P\left(B|A\right)P\left(A\right)}{P\left(B\right)}.$$

Let's use that theorem to compute a prediction.

Example 2.1 (Bayes theorem)**.** Consider the following tiny dataset, where X_1 and X_2 are the predictors and Y is the outcome variable:

	X_1	X_2	Y
0	B	A	1
1	A	A	0
2	A	A	1
3	A	B	1
4	A	B	0
5	B	B	0

Suppose that we want to predict the outcome variable of the case $(X_1, X_2) = (B, A)$. One way to get a prediction is to calculate the following conditional probabilities:

$$P(Y = 0|X_1 = B, X_2 = A) \text{ and } P(Y = 1|X_1 = B, X_2 = A).$$

If the first probability is greater than the second one, then the prediction will be $Y = 0$; the prediction will be $Y = 1$, otherwise. In sum, we take the most probable class (between class 0 and class 1) as a prediction.

Since

$$P(Y = 0|X_1 = B, X_2 = A) = 1 - P(Y = 1|X_1 = B, X_2 = A),$$

we need to calculate only one of these two probabilities.

By applying Bayes' theorem, we have

$$P(Y = 0|X_1 = B, X_2 = A) = \frac{P(X_1 = B, X_2 = A|Y = 0)\, P(Y = 0)}{P(X_1 = B, X_2 = A)}.$$

From the table, we get the following:

- $P(Y = 0) = 1/2$,
- $P(X_1 = B, X_2 = A) = 1/6$,
- $P(X_1 = B, X_2 = A|Y = 0) = 0/3$.

We can now calculate the wanted probability:

$$P(Y = 0|X_1 = B, X_2 = A) = \frac{\frac{1}{2} \times 0}{\frac{1}{6}} = 0.$$

Thus, the prediction for the case $(X_1, X_2) = (B, A)$ is class 1.

An important fact to remark is that if in the dataset there were no pair $(X_1, X_2) = (B, A)$, we would *not* be able to compute the prediction!

In general, when we have a dataset with predictors $X_1, X_2, \cdots, X_p$, and a categorical outcome variable, Y, which has, as possible values, the classes $C_1, C_2, \cdots, C_m$.

Applying Bayes' theorem, we get:

$$P\left(C_i|x_1, x_2, \cdots, x_p\right) = \frac{P\left(x_1, x_2, \cdots, x_p|C_i\right) P\left(C_i\right)}{P\left(x_1, x_2, \cdots, x_p\right)},$$

where $x_1, x_2, \cdots, x_p$ are the values of the p predictors of the case we want to predict, and $i = 1, 2, \cdots, m$.

3 Naive Bayes

As discussed earlier in Example 2.1, Bayes' theorem may not be applicable if we cannot find a row in the dataset where the predictors have the same values as the respective predictors of the example we want to classify.

A solution to overcome this difficulty is to, naively, assume independence among the predictors $X_1, X_2, \cdots, X_p$:

$$P\left(x_1, x_2, \cdots, x_p|C_i\right) = P\left(x_1|C_i\right) P\left(x_2|C_i\right) \cdots P\left(x_p|C_i\right).$$

That is amazing that Naive Bayes model works well in a great variety of problems, in spite of being based on an assumption that may be not supported by reality!

Example 3.1 (Naive Bayes)**.** We will use the same dataset we used in Example 2.1:

	X_1	X_2	Y
0	B	A	1
1	A	A	0
2	A	A	1
3	A	B	1
4	A	B	0
5	B	B	0

Suppose that we want to predict the outcome variable of the same case $(X_1, X_2) = (B, A)$.

By applying the naive version of Bayes' theorem, we have

$$P\left(Y = 0|X_1 = B, X_2 = A\right) = \frac{P\left(X_1 = B|Y = 0\right) P\left(X_2 = A|Y = 0\right) P\left(Y = 0\right)}{P\left(X_1 = B|Y = 0\right) P\left(X_2 = A|Y = 0\right) P\left(Y = 0\right) + P\left(X_1 = B|Y = 1\right) P\left(X_2 = A|Y = 1\right) P\left(Y = 1\right)}.$$

From the table, we get the following:

- $P(Y=0) = 1/2$,
- $P(X_1 = B|Y=0) = 1/3$,
- $P(X_2 = A|Y=0) = 1/3$,

- $P(Y=1) = 1/2$,
- $P(X_1 = B|Y=1) = 1/3$,
- $P(X_2 = A|Y=1) = 2/3$.

We can now calculate the wanted probability:

$$P(Y=0|X_1=B, X_2=A) = \frac{\frac{1}{2}\times\frac{1}{3}\times\frac{1}{3}}{\frac{1}{2}\times\frac{1}{3}\times\frac{1}{3}+\frac{1}{2}\times\frac{1}{3}\times\frac{2}{3}} = \frac{1}{3}.$$

Thus, the prediction for the case $(X_1, X_2) = (B, A)$ is class 1.

4 Python implementation

We need to first load the needed function:

```
from sklearn.naive_bayes import BernoulliNB
```

Since `BernoulliNB` requires a numerical dataframe, we need to replace classes A and B with, for instance, 1 and 0, respectively.

```
df = pd.DataFrame({
  'X1': ["B", "A", "A", "A", "A", "B"],
  'X2': ["A", "A", "A", "B", "B", "B"],
  'Y': [1, 0, 1, 1, 0, 0],
})

df = df.replace({'A': 1, 'B': 0})
print(df)
```

```
   X1  X2  Y
0   0   1  1
1   1   1  0
2   1   1  1
3   1   0  1
4   1   0  0
5   0   0  0
```

`BernoulliNB` uses a numerical smoothing mechanism, which we are going to deactivate, so that it produces the same results that pure naive bayes does, just to check the correctness of our hand calculations. In general, we must *not* turn off numerical smoothing. To turn off the numerical smoothing mechanism, we use `alpha=0` and `force_alpha=True`.

We can now apply the model:

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

nb_model = BernoulliNB(alpha=0, force_alpha=True)
nb_model.fit(X, y)

X_new = pd.DataFrame({
  'X1': [0],
  'X2': [1]
})

y_proba = nb_model.predict_proba(X_new)
y_proba
```

```
array([[0.33333333, 0.66666667]])
```

5 Naive Bayes generalization

Naive Bayes is based on the assumption that the predictors are categorical. This is a strong limitation! Fortunately, there is a version of Naive Bayes that works with numerical predictors: It is called Gaussian Naive Bayes.

Gaussian Naive Bayes assumes that all predictors follow Gaussian distribution and also that are independent of each other. This may not hold in reality, but, despite that, it can produce good predictions – that is, therefore, worth trying.