Naive Bayes Classifier

A probabilistic classification algorithm

Marcel Meyer, Maria Reimer, Martin Schilling, Sebastian Thomas
@ neue fische Bootcamp Data Science

Table of Contents

Bayes Theorem

$$P(A|B) = \frac{P(A)\,P(B|A)}{P(B)}$$

- way of finding the probability, when we know certain other probabilities

Practical example

Can I go kitesurfing when ...?

Wind condition	Go kitesurfing
Windy	Yes
No wind	No
Windy	Yes
Stormy	Yes
Stormy	No
Stormy	No
Windy	Yes

Wind cond	Kiting: Yes	Kiting: No	Total	Likelihood
No wind	0	1	1	1/7
Windy	3	0	3	3/7
Stormy	1	2	3	3/7
Total	4	3		
Likelihood	4/7	3/7		

P(Yes|Windy) = (P(Yes)*P(Windy|Yes)) / P(Windy)

P(Yes|Windy) = ((4/7) * (3/4)) / (3/7)

P(Yes|Windy) = 1

Principle of Naive Bayes

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots x_n \mid y)}{P(x_1, \ldots, x_n)}$$

Assumption: features are independent

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \ldots, x_n)}$$

Since denominator is constant:

$$\hat{y} = \arg\max_{y} P(y) \prod_{i=1}^{n} P(x_i \mid y),$$

Principle of Naive Bayes

This assumption of independence is not fulfilled in practice!

However, we use the formula anyway:

NAIVE Bayes

Types of Naive Bayes Classifier

dependent on the assumption about the distribution of the features

1) Gaussian Naive Bayes:

   → features are normally distributed

2) Multinomial Naive Bayes:

   → features follow a multinomial distribution

3) Bernoulli Naive Bayes:

   → features are independent, binary variables (e.g. True/False)

...

Scikit-learn classes

```
# no scaling needed
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import BernoulliNB, CategoricalNB, ComplementNB, GaussianNB, MultinomialNB

(X, y) = load_iris(return_X_y=True)
(X_train, X_test, y_train, y_test) = train_test_split(X, y, random_state=0)

bernoulli_nb = BernoulliNB().fit(X_train, y_train) # binary features
categorical_nb = CategoricalNB().fit(X_train, y_train) # categorical features
complement_nb = ComplementNB().fit(X_train, y_train) # imbalanced data sets, designed to correct some assumpts
gaussian_nb = GaussianNB().fit(X_train, y_train) # gaussian distributed features
multinomial_nb = MultinomialNB().fit(X_train, y_train) # multinomially distributed discrete features, frequencs

print('accuracy Bernoulli NB:   ', bernoulli_nb.score(X_test, y_test))
print('accuracy Categorical NB:', categorical_nb.score(X_test, y_test))
print('accuracy Complement NB: ', complement_nb.score(X_test, y_test))
print('accuracy Gaussian NB:    ', gaussian_nb.score(X_test, y_test))
print('accuracy Multinomial NB:', multinomial_nb.score(X_test, y_test))
```

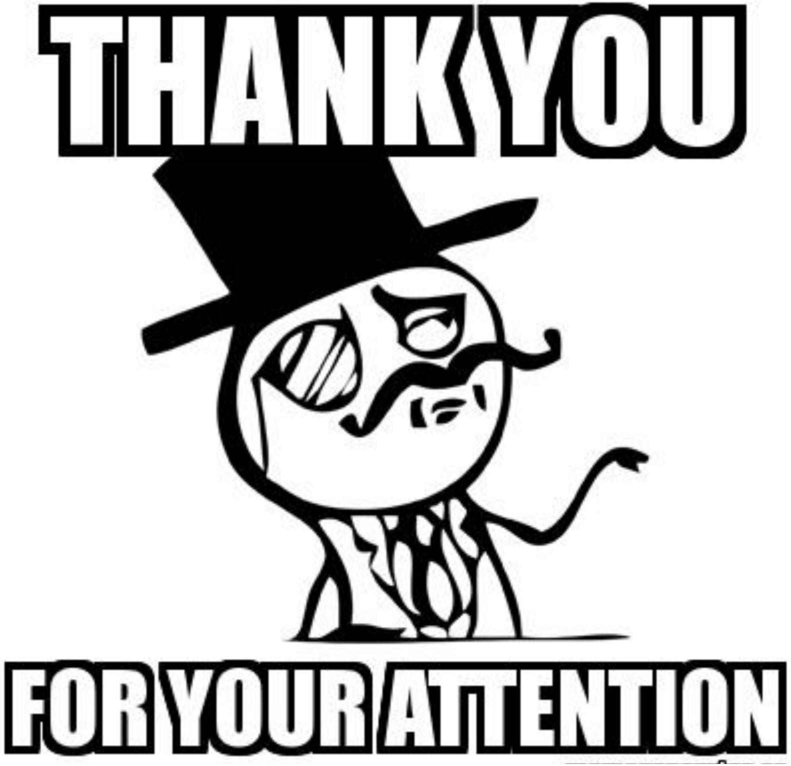
```
accuracy Bernoulli NB:   0.23684210526315788
accuracy Categorical NB: 0.8947368421052632
accuracy Complement NB:  0.5789473684210527
accuracy Gaussian NB:    1.0
accuracy Multinomial NB: 0.5789473684210527
```

Conclusion

- often used in:
  - Spam filtering
  - Recommendation systems

- easy to implement

- Disadvantage: predictors should be independent, in real life cases predictors are often dependent → sometimes performance issues


THANK YOU
FOR YOUR ATTENTION


Resources

- https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c
- https://databraineo.de/ask-the-doc/was-ist-der-naive-bayes-algorithmus/
- https://scikit-learn.org/stable/modules/naive_bayes.html