Naive Bayes Classifier Introductory Overview

Saturday, September 8, 2018 10:16 PM

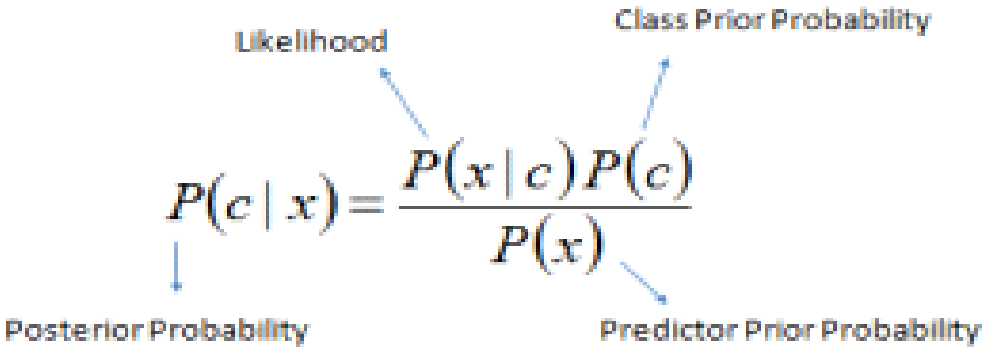
In machine learning, **naive Bayes classifiers** are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naive) independence assumptions between the features

The Naive Bayes Classifier technique is based on the so-called Bayesian theorem and is particularly suited when the dimensionality of the inputs is high. Despite its simplicity, Naive Bayes can often outperform more sophisticated classification methods.

Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

Bayes theorem provides a way of calculating posterior probability P(c|x) from P(c), P(x) and P(x|c).

Look at the equation below:



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

Above,

- $P(c|x)$ is the posterior probability of *class* (c, *target*) given *predictor* (x, *attributes*).
- $P(c)$ is the prior probability of *class*.
- $P(x|c)$ is the likelihood which is the probability of *predictor* given *class*.
- $P(x)$ is the prior probability of *predictor*.

To demonstrate the concept of Naïve Bayes Classification, consider the example displayed in the illustration above. As indicated, the objects can be classified as either GREEN or RED. Our task is to classify new cases as they arrive, i.e., decide to which class label they belong, based on the currently exiting objects.

Since there are twice as many GREEN objects as RED, it is reasonable to believe that a new case (which hasn't been observed yet) is twice as likely to have membership GREEN rather than RED. In the Bayesian analysis, this belief is known as the prior probability. Prior probabilities are based on previous experience, in this case the percentage of GREEN and RED objects, and often used to predict outcomes before they actually happen.

Thus, we can write:

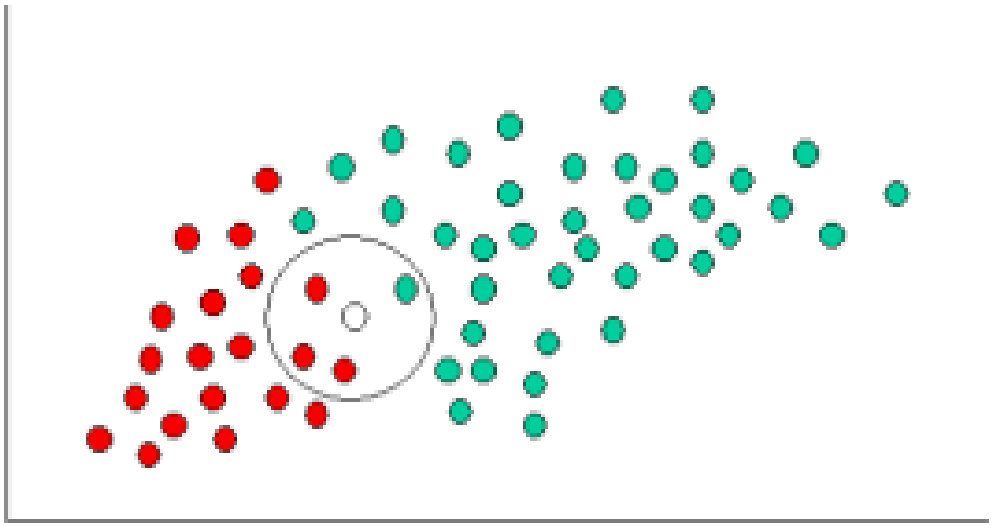
$$Prior\ probability\ for\ GREEN \propto \frac{Number\ of\ GREEN\ objects}{Total\ number\ of\ objects}$$

$$Prior\ probability\ for\ RED \propto \frac{Number\ of\ RED\ objects}{Total\ number\ of\ objects}$$

Since there is a total of 60 objects, 40 of which are GREEN and 20 RED, our prior probabilities for class membership are:

$$Prior\ probability\ for\ GREEN \propto \frac{40}{60}$$

$$Prior\ probability\ for\ RED \propto \frac{20}{60}$$



Having formulated our prior probability, we are now ready to classify a new object (WHITE circle). Since the objects are well clustered, it is reasonable to assume that the more GREEN (or RED) objects in the vicinity of X, the more likely that the new cases belong to that particular color. To measure this likelihood, we draw a circle around X which encompasses a number (to be chosen a priori) of points irrespective of their class labels. Then we calculate the number of points in the circle belonging to each class label. From this we calculate the likelihood:

$$Likelihood\ of\ X\ given\ GREEN \propto \frac{Number\ of\ GREEN\ in\ the\ vicinity\ of\ X}{Total\ number\ of\ GREEN\ cases}$$

$$Likelihood\ of\ X\ given\ RED \propto \frac{Number\ of\ RED\ in\ the\ vicinity\ of\ X}{Total\ number\ of\ RED\ cases}$$

From the illustration above, it is clear that Likelihood of X given GREEN is smaller than Likelihood of X given RED, since the circle encompasses 1 GREEN object and 3 RED ones. Thus:

$$Probability\ of\ X\ given\ GREEN \propto \frac{1}{40}$$

$$Probability\ of\ X\ given\ RED \propto \frac{3}{20}$$

Although the prior probabilities indicate that X may belong to GREEN (given that there are twice as many GREEN compared to RED) the likelihood indicates otherwise; that the class membership of X is RED (given that there are more RED objects in the vicinity of X than GREEN). In the Bayesian analysis, the final classification is produced by combining both sources of information, i.e., the prior and the likelihood, to form a posterior probability using the so-called Bayes' rule (named after Rev. Thomas Bayes 1702-1761).

$$\text{Posterior probability of } X \text{ being } GREEN \propto$$

$$\text{Prior probability of } GREEN \times \text{Likelihood of } X \text{ given } GREEN$$

$$= \frac{4}{6} \times \frac{1}{40} = \frac{1}{60}$$

$$\text{Posterior probability of } X \text{ being } RED \propto$$

$$\text{Prior probability of } RED \times \text{Likelihood of } X \text{ given } RED$$

$$= \frac{2}{6} \times \frac{3}{20} = \frac{1}{20}$$

Finally, we classify X as RED since its class membership achieves the largest posterior probability.

Note. The above probabilities are not normalized. However, this does not affect the classification outcome since their normalizing constants are the same.

How Naive Bayes algorithm works?

Saturday, September 8, 2018 10:29 PM

Let's understand it using an example. Below I have a training data set of weather and corresponding target variable 'Play' (suggesting possibilities of playing). Now, we need to classify whether players will play or not based on weather condition. Let's follow the below steps to perform it.

Step 1: Convert the data set into a frequency table

Step 2: Create Likelihood table by finding the probabilities like Overcast probability = 0.29 and probability of playing is 0.64.

Weather	Play
Sunny	No
Overcast	Yes
Rainy	Yes
Sunny	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Rainy	No
Sunny	Yes
Rainy	Yes
Sunny	No
Overcast	Yes
Overcast	Yes
Rainy	No

Frequency Table		
Weather	No	Yes
Overcast		4
Rainy	3	2
Sunny	2	3
Grand Total	5	9

Likelihood table				
Weather	No	Yes		
Overcast		4	=4/14	0.29
Rainy	3	2	=5/14	0.36
Sunny	2	3	=5/14	0.36
All	5	9		
	=5/14	=9/14		
	0.36	0.64		

Step 3: Now, use Naive Bayesian equation to calculate the posterior probability for each class. The class with the highest posterior probability is the outcome of prediction.

Problem: Players will play if weather is sunny. Is this statement is correct?

We can solve it using above discussed method of posterior probability.

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

Here we have P (Sunny |Yes) = 3/9 = 0.33, P(Sunny) = 5/14 = 0.36, P(Yes)= 9/14 = 0.64

Now, P (Yes | Sunny) = 0.33 * 0.64 / 0.36 = 0.60, which has higher probability.

Naive Bayes uses a similar method to predict the probability of different class based on various attributes. This algorithm is mostly used in text classification and with problems having multiple classes.

##

So far, we learned what the Naive Bayes algorithm is, how the Bayes theorem is related to it, and what the expression of the Bayes' theorem for this algorithm is. Let us take a simple example to understand the functionality of the algorithm. Suppose, we have a training data set of 1200 fruits. The features in the data set are these: is the fruit yellow or not, is the fruit long or not, and is the fruit sweet or not. There are three different classes: mango, banana, and others.

Step 1: Create a frequency table for all the features against the different classes.

Name	Yellow	Sweet	Long	Total

Mango	350	450	0	650
Banana	400	300	350	400
Others	50	100	50	150
Total	800	850	400	1200

What can we conclude from the above table?

- Out of 1200 fruits, 650 are mangoes, 400 are bananas, and 150 are others.
- 350 of the total 650 mangoes are yellow and the rest are not and so on.
- 800 fruits are yellow, 850 are sweet and 400 are long from a total of 1200 fruits.

Let's say you are given with a fruit which is yellow, sweet, and long and you have to check the class to which it belongs.

Step 2: Draw the likelihood table for the features against the classes.

Name	Yellow	Sweet	Long	Total
Mango	350/800=P(Mango|Yellow)	450/850	0/400	650/1200=P(Mango)
Banana	400/800	300/850	350/400	400/1200
Others	50/800	100/850	50/400	150/1200
Total	800=P(Yellow)	850	400	1200

Step 3: Calculate the conditional probabilities for all the classes, i.e., the following in our example:

$$P(\text{Mango}|\text{Yellow, Sweet, Long}) = \frac{P(\text{Yellow}|\text{Mango}).P(\text{Sweet}|\text{Mango}).P(\text{Long}|\text{Mango}).P(\text{Mango})}{P(\text{Yellow, Sweet, Long})}$$

$$= 0$$

$$P(\text{Banana}|\text{Yellow, Sweet, Long}) = \frac{P(\text{Yellow}|\text{Banana}).P(\text{Sweet}|\text{Banana}).P(\text{Long}|\text{Banana}).P(\text{Banana})}{P(\text{Yellow, Sweet, Long})}$$

$$= \frac{400 * 300 * 350 * 400}{400 * 400 * 400 * 1200 * P(\text{Evidence})}$$

$$= \frac{0.21875}{P(\text{Evidence})}$$

$$P(\text{Others}|\text{Yellow, Sweet, Long}) = \frac{P(\text{Yellow}|\text{Others}).P(\text{Sweet}|\text{Others}).P(\text{Long}|\text{Others}).P(\text{Others})}{P(\text{Yellow, Sweet, Long})}$$

$$= \frac{50 * 100 * 50 * 150}{150 * 150 * 150 * 1200 * P(\text{Evidence})}$$

$$= \frac{0.00926}{P(\text{Evidence})}$$

Step 4: Calculate maxiP(Ci|x1,x2,…,xn). In our example, the maximum probability is for the class banana, therefore, the fruit which is long, sweet and yellow is a banana by Naive Bayes Algorithm.

In a nutshell, we say that a new element will belong to the class which will have the maximum conditional probability described above.

How to build a basic model using Naive Bayes in Python?

Saturday, September 8, 2018 10:43 PM

Again, scikit learn (python library) will help here to build a Naive Bayes model in Python. There are three types of Naive Bayes model under scikit learn library:

- **Gaussian:** It is used in classification and it assumes that features follow a normal distribution.
- **Multinomial:** It is used for discrete counts. For example, let's say, we have a text classification problem. Here we can consider bernoulli trials which is one step further and instead of "word occurring in the document", we have "count how often word occurs in the document", you can think of it as "number of times outcome number x_i is observed over the n trials".
- **Bernoulli:** The binomial model is useful if your feature vectors are binary (i.e. zeros and ones). One application would be text classification with 'bag of words' model where the 1s & 0s are "word occurs in the document" and "word does not occur in the document" respectively.

Based on your data set, you can choose any of above discussed model. Below is the example of Gaussian model.

##

Variations of the Naive Bayes algorithm

There are multiple variations of the Naive Bayes algorithm depending on the distribution of P(xj|Ci). Three of the commonly used variations are

1. **Gaussian**: The Gaussian Naive Bayes algorithm assumes distribution of features to be Gaussian or normal, i.e.,

$$P(x_j|C_i) = \frac{1}{\sqrt{2\pi\sigma_{C_i}^2}}\exp\left(-\frac{(x_j-\mu_{C_j})^2}{2\sigma_{C_i}^2}\right)$$

2. Read more about it here.
3. **Multinomial**: The Multinomial Naive Bayes algorithm is used when the data is distributed multinomially, i.e., multiple occurrences matter a lot. You can read more here.
4. **Bernoulli**: The Bernoulli algorithm is used when the features in the data set are binary-valued. It is helpful in spam filtration and adult content detection techniques. For more details, click here.

From <https://www.hackerearth.com/blog/machine-learning/introduction-naive-bayes-algorithm-codes-python-r/>

Coding

Saturday, September 8, 2018 10:45 PM

Check the Jupiter NoteBook

Tips to improve the power of Naive Bayes Model

Saturday, September 8, 2018 10:48 PM

Here are some tips for improving power of Naive Bayes Model:

- If continuous features do not have normal distribution, we should use transformation or different methods to convert it in normal distribution.
- If test data set has zero frequency issue, apply smoothing techniques “Laplace Correction” to predict the class of test data set.
- Remove correlated features, as the highly correlated features are voted twice in the model and it can lead to over inflating importance.
- Naive Bayes classifiers has limited options for parameter tuning like alpha=1 for smoothing, fit_prior=[True|False] to learn class prior probabilities or not and some other options (look at detail here). I would recommend to focus on your pre-processing of data and the feature selection.
- You might think to apply some *classifier combination technique like* ensembling, bagging and boosting but these methods would not help. Actually, “ensembling, boosting, bagging” won’t help since their purpose is to reduce variance. Naive Bayes has no variance to minimize.

What are the Pros and Cons of Naive Bayes?

Saturday, September 8, 2018 10:41 PM

Pros:

- It is easy and fast to predict class of test data set. It also perform well in multi class prediction
- When assumption of independence holds, a Naive Bayes classifier performs better compare to other models like logistic regression and you need less training data.
- It perform well in case of categorical input variables compared to numerical variable(s). For numerical variable, normal distribution is assumed (bell curve, which is a strong assumption).
- It is a relatively easy algorithm to build and understand.
- It is faster to predict classes using this algorithm than many other classification algorithms.
- It can be easily trained using a small data set.

Cons:

- If categorical variable has a category (in test data set), which was not observed in training data set, then model will assign a 0 (zero) probability and will be unable to make a prediction. This is often known as “Zero Frequency”. To solve this, we can use the smoothing technique. One of the simplest smoothing techniques is called Laplace estimation.
- On the other side naive Bayes is also known as a bad estimator, so the probability outputs from predict_proba are not to be taken too seriously.
- Another limitation of Naive Bayes is the assumption of independent predictors. In real life, it is almost impossible that we get a set of predictors which are completely independent.
- If a given class and a feature have 0 frequency, then the conditional probability estimate for that category will come out as 0. This problem is known as the “Zero Conditional Probability Problem.” This is a problem because it wipes out all the information in other probabilities too. There are several sample correction techniques to fix this problem such as “Laplacian Correction.”
- Another disadvantage is the very strong assumption of independence class features that it makes. It is near to impossible to find such data sets in real life.

Applications of Naive Bayes Algorithms

Saturday, September 8, 2018 10:42 PM

- **Real time Prediction:** Naive Bayes is an eager learning classifier and it is sure fast. Thus, it could be used for making predictions in real time.
- **Multi class Prediction:** This algorithm is also well known for multi class prediction feature. Here we can predict the probability of multiple classes of target variable.
- **Text classification/ Spam Filtering/ Sentiment Analysis:** Naive Bayes classifiers mostly used in text classification (due to better result in multi class problems and independence rule) have higher success rate as compared to other algorithms. As a result, it is widely used in Spam filtering (identify spam e-mail) and Sentiment Analysis (in social media analysis, to identify positive and negative customer sentiments)
- **Recommendation System:** Naive Bayes Classifier and Collaborative Filtering together builds a Recommendation System that uses machine learning and data mining techniques to filter unseen information and predict whether a user would like a given resource or not

Interview Stuff

Saturday, September 8, 2018 11:29 PM

https://www.geeksforgeeks.org/naive-bayes-classifiers/