Machine Learning with Python

Supervised Learning (Naïve Bayes)

Dr. Aeshah Alsughayyir

Collage of Computer Science and Engineering

Taibah University

2021-2022

Outline:

- Naïve Bayes Classifiers
  - What is Naive Bayes algorithms?
    - Bayes' theorem
  - How Naive Bayes Algorithms works?
  - What are the Pros and Cons of using Naive Bayes?
  - Applications of Naive Bayes Algorithm
  - Steps to build a basic Naive Bayes Model in Python

Naïve Bayes Classifier

Motivation

- Understand one of the most popular and simple Machine Learning classification algorithms, the **Naive Bayes algorithm**
- It is based on the ***Bayes Theorem*** for calculating probabilities and conditional probabilities.

Naïve Bayes Classifier

Bayes' theorem is named for Thomas Bayes (1701–1761), who first used conditional probability to provide an algorithm that uses evidence to calculate an unknown parameter.

What is Naive Bayes algorithms?

- It is a classification technique based on Bayes' Theorem with an assumption of **independence** among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.
- For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as 'Naive'.
- 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.

Naive Bayes classifiers are a collection of classification algorithms based on **Bayes' Theorem**.

Bayes' Theorem

- Bayes theorem provides a way of calculating posterior probability P(A|B) from P(A), P(B) and P(B|A). Look at the equation below:

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

Basically, we are trying to find probability of event A, given the event **B is true**.

- P(A|B) is the posterior probability of class (A, target) given *predictor* (B, attributes).
- P(A) is the prior probability of *class*.
- P(B|A) is the likelihood which is the probability of *predictor* given *class*.
- P(B) is the prior probability of *predictor*. Event B is also termed as **evidence**

Bayes' Theorem (Cont.)

- Bayes' Theorem finds the probability of an event occurring given the probability of another event that has already occurred.

- **Conditional Probability**

Note : $\boldsymbol{P(A,B)}$ = P(A)* P(B)

Example : What is the probability that a fair coin will come up with **heads twice** in two tosses?

A: Head , B:Tial A and B are independent , P(A,A) = P(A) * P(A)

- Two events must occur: a head on the first toss and a head on the second toss.
- Since the probability of each event is 1/2, the probability of both events is: 1/2 * 1/2 = 1/4.

Bayes' Theorem (Cont.)

- Naive Bayes classifiers

- **Assumption #1**
- No pair of features are dependent.
- For example:
  - the temperature being 'Hot' has nothing to do with the humidity
  - the outlook being 'Rainy' has no effect on the winds.
- The features are assumed to be **independent**.

independent features

prediction

PlayTennis: training examples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

f1 f2 f3 f4 class

Bayes' Theorem (Cont.)

- Naive Bayes classifiers

independent features

prediction

PlayTennis: training examples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

- **Assumption #2**

- None of the attributes is irrelevant and assumed to be contributing **equally** to the outcome.

- For example, knowing only temperature and humidity alone can't predict the outcome accurately.

Note: The assumptions made by Naive Bayes are not generally correct in real-world situations. In-fact, the independence assumption is never correct but often works well in practice.

Naive Bayes classifiers

Now, with regards to our dataset, we can apply Bayes' theorem in following way:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

$$X = (x_1, x_2, x_3, ..., x_n)$$

where, y is class variable and X is a dependent feature vector (of size *n*) where:

Just to clear, an example of a feature vector and corresponding class variable can be: (refer 1st row of dataset)

X = (Rainy, Hot, High, weak, False)
y = No

PlayTennis: training examples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

How Naive Bayes Algorithms works?

- Let's understand it using our example. We have a training dataset of weather and corresponding target variable 'Play=yes' or 'Play=No' (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.

How Naive Bayes Algorithms works? (Cont.)

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

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		

- We can solve it using the 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.

Note:
This algorithm is mostly used in text classification and with problems having multiple classes.

How Naive Bayes Algorithms works? (Cont.)

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

$$X = (x_1, x_2, x_3, ..., x_n)$$

$$P(y|x_1, \ldots, x_n) = \frac{P(x_1|y)P(x_2|y)\ldots P(x_n|y)P(y)}{P(x_1)P(x_2)\ldots P(x_n)}$$

Suppose that y ={Yes, No}

$$P(yes|x_1, \ldots, x_n) = \frac{P(x_1|yes) \ldots . . P(x_n|yes)P(yes)}{P(x_1)P(x_2) \ldots . . P(x_n)}$$

$$P(No|x_1, \ldots, x_n) = \frac{P(x_1|No) \ldots . . P(x_n|No)P(No)}{P(x_1)P(x_2) \ldots . . P(x_n)}$$

Max

$P(yes|x_1, \ldots, x_n)$,

$P(No|x_1, \ldots, x_n)$

Predicted value of y {Yes or No}

How Naive Bayes Algorithms works? (Cont.)

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

$$X = (x_1, x_2, x_3, ..., x_n)$$

$$P(y|x_1, ..., x_n) = \frac{P(x_1|y)P(x_2|y)...P(x_n|y)P(y)}{P(x_1)P(x_2)...P(x_n)}$$

which can be expressed as:

$$P(y|x_1, ..., x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1)P(x_2)...P(x_n)}$$

Now, as the denominator remains constant for a given input, we can remove that term:

$$P(y|x_1, ..., x_n) \propto P(y)\prod_{i=1}^{n} P(x_i|y)$$

$$y = argmax_y P(y)\prod_{i=1}^{n} P(x_i|y)$$

How Naive Bayes Algorithms works? (Cont.)

$$y = argmax_y P(y) \prod_{i=1}^{n} P(x_i|y)$$

Note that :

P(y) is also called **class probability**

P(xi | y) is called **conditional probability.**

How Naive Bayes Algorithms works? (Cont.)

Let us try to apply the above formula manually on our weather dataset We need to find $P(x_i \mid y_i)$ for each $x_i \in X$ and for each $y_i \in y$.

$$P(y|x_1, \ldots, x_n) = \frac{P(x_1|y)P(x_2|y)\ldots P(x_n|y)P(y)}{P(x_1)P(x_2)\ldots P(x_n)}$$

x_1 **Outlook**

	Yes	No	P(yes)	P(no)
Sunny	2	3	2/9	3/5
Overcast	4	0	4/9	0/5
Rainy	3	2	3/9	2/5
Total	9	5	100%	100%

x_2 **Temperature**

	Yes	No	P(yes)	P(no)
Hot	2	2	2/9	2/5
Mild	4	2	4/9	2/5
Cool	3	1	3/9	1/5
Total	9	5	100%	100%

x_3 **Humidity**

	Yes	No	P(yes)	P(no)
High	3	4	3/9	4/5
Normal	6	1	6/9	1/5
Total	9	5	100%	100%

x_4 **Wind**

	Yes	No	P(yes)	P(no)
False	6	2	6/9	2/5
True	3	3	3/9	3/5
Total	9	5	100%	100%

y

Play		P(Yes)/P(No)
Yes	9	9/14
No	5	5/14
Total	14	100%

$$P(yes|x_1, \ldots, x_n) = \frac{P(x_1|yes) \ldots . P(x_n|yes)P(yes)}{P(x_1)P(x_2) \ldots . P(x_n)}$$

$$P(No|x_1, \ldots, x_n) = \frac{P(x_1|No) \ldots . P(x_n|No)P(No)}{P(x_1)P(x_2) \ldots . P(x_n)}$$

How Naive Bayes Algorithms works? (Cont.)

Now suppose we want to predict **the probability of playing tennis today** where today's features are as follows:

today = (Sunny, Hot, Normal, weak, NO)

$$P(yes|x_1,\dots,x_n) = \frac{P(x_1|yes)\dots..P(x_n|yes)P(yes)}{P(x_1)P(x_2)\dots..P(x_n)}$$

$$P(Yes|today) = \frac{P(SunnyOutlook|Yes)P(HotTemperature|Yes)P(NormalHumidity|Yes)P(NoWind|Yes)P(Yes)}{P(today)}$$

$$P(No|x_1,\dots,x_n) = \frac{P(x_1|No)\dots..P(x_n|No)P(No)}{P(x_1)P(x_2)\dots..P(x_n)}$$

$$P(No|today) = \frac{P(SunnyOutlook|No)P(HotTemperature|No)P(NormalHumidity|No)P(NoWind|No)P(No)}{P(today)}$$

Since, P(today) is common in both probabilities, we can ignore P(today) and find proportional probabilities as:

$$P(Yes|today) \propto \frac{2}{9}.\frac{2}{9}.\frac{6}{9}.\frac{6}{9}.\frac{9}{14} \approx 0.0141$$

and

$$P(No|today) \propto \frac{3}{5}.\frac{2}{5}.\frac{1}{5}.\frac{2}{5}.\frac{5}{14} \approx 0.0068$$

How Naive Bayes Algorithms works? (Cont.)

Now, since

$$P(Yes|today) + P(No|today) = 1$$

These numbers can be converted into a probability by making the sum equal to 1 (normalization):

$$P(Yes|today) = \frac{0.0141}{0.0141+0.0068} = 0.67$$

and

$$P(No|today) = \frac{0.0068}{0.0141+0.0068} = 0.33$$

Since

$$P(Yes|today) > P(No|today)$$

So, prediction that Tennis would be played is ‘Yes’.

Note:

- The Naive Bayes is applicable for **discrete data**.
- In case of continuous data, we need to make some assumptions regarding the distribution of values of each feature.
- The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of $P(x_i \mid y)$.

- In Gaussian Naive Bayes, continuous values associated with each feature are assumed to be distributed according to a **Gaussian distribution**.

What are the Pros and Cons of Naive Bayes?

Pros:

- It is easy and fast to predict class of test data set.
- It also performs well in multiclass prediction
- When assumption of independence holds, a Naive Bayes classifier performs better compared to other models like logistic regression and you need less training data.
- It performs 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).

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 non practical choice to classify a huge dataset (especially images).
- 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.

Applications of Naive Bayes Algorithms

•**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.

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

Scikit learn (python library) will help here to build a Naive Bayes model in Python. There are different types of Naive Bayes model under the scikit-learn library:

Gaussian, Multinomial, Complement, and Bernoulli.

```
>>> from sklearn.datasets import load_iris
>>> from sklearn.model_selection import train_test_split
>>> from sklearn.naive_bayes import GaussianNB
>>> X, y = load_iris(return_X_y=True)
>>> X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=0)
>>> gnb = GaussianNB()
>>> y_pred = gnb.fit(X_train, y_train).predict(X_test)
>>> print("Number of mislabeled points out of a total %d points : %d"
...       % (X_test.shape[0], (y_test != y_pred).sum()))
Number of mislabeled points out of a total 75 points : 4
```

Load the required libraries

Load the dataset

Split the dataset into 50% testing and 50% training

Choose the model for training

Start the training process to fit the model

Use the testing dataset (x) to predict the output (y)

Print out how many time the model missed predicting the correct output

Note:
Refer to '<u>scikit_learn_tutorial</u>' **Page 106**, for an implementation example on using *GaussianNB* method to construct Naive Bayes classifier from a different dataset

Any questions?