

**Navigation**[Start Here](#) [Blog](#) [Products](#) [About](#) [Contact](#)

Search...



# How To Implement Naive Bayes From Scratch in Python

by **Jason Brownlee** on December 8, 2014 in **Machine Learning Algorithms**, **Python Machine Learning**

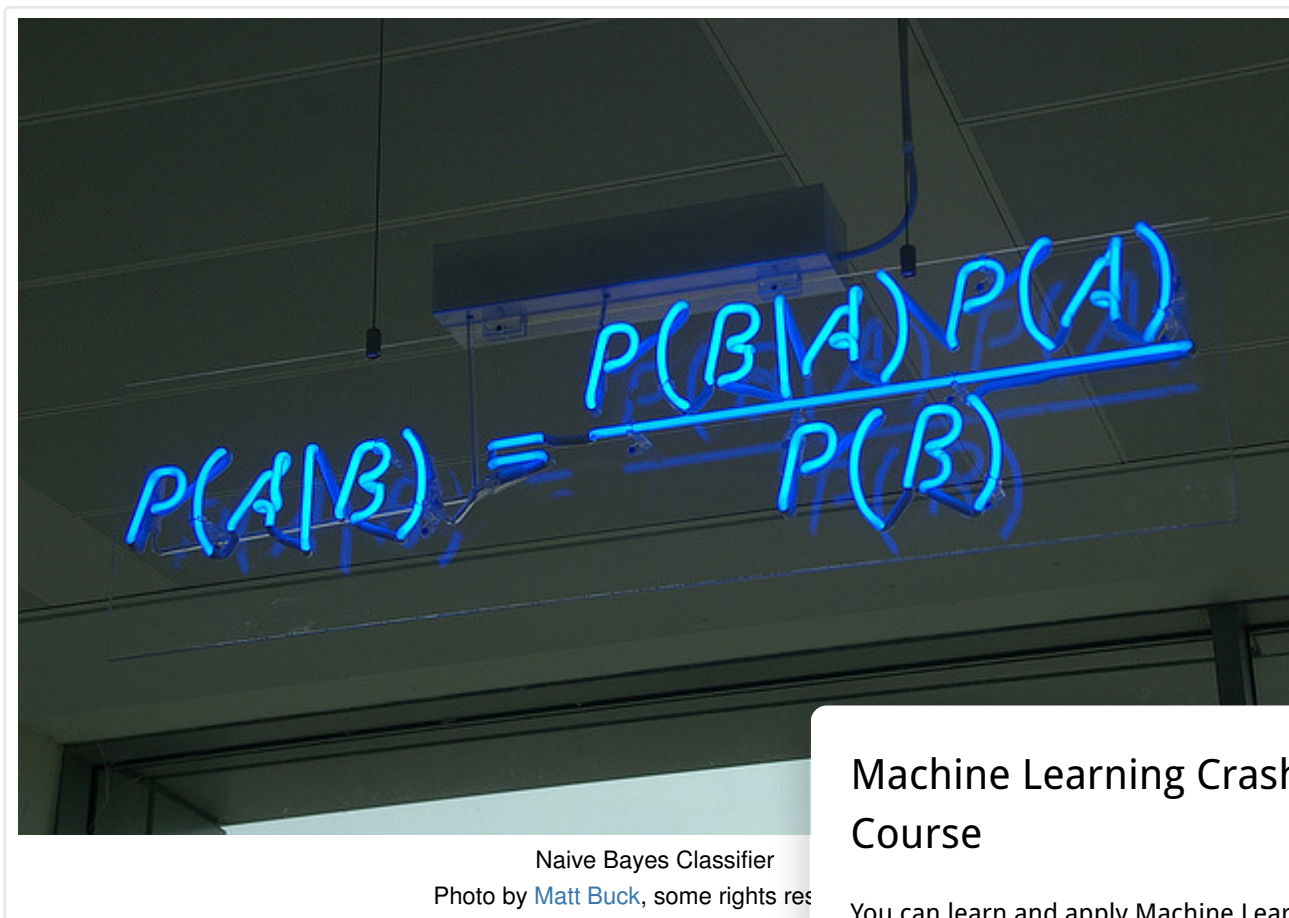
29

87

The Naive Bayes algorithm is simple and effective and should be one of the first methods you try on a classification problem.

In this tutorial you are going to learn about the Naive Bayes algorithm including how it works and how to implement it from scratch in Python.

**Update:** Check out the follow-up on tips for using the naive bayes algorithm titled: “[Better Naive Bayes: 12 Tips To Get The Most From The Naive Bayes Algorithm](#)”



## About Naive Bayes

The Naive Bayes algorithm is an intuitive method that uses the probabilities of events to make a prediction. It is the supervised learning approach you would come up with if you were to think probabilistically.

Naive bayes simplifies the calculation of probabilities by assuming that the prob

### Machine Learning Crash Course

You can learn and apply Machine Learning even without a background in maths or a fancy degree. Find out how in this ridiculously practical (and totally free) email course.

SIGN ME UP

value is independent of all other attributes. This is a strong assumption but results in a fast and effective method.

The probability of a class value given a value of an attribute is called the conditional probability. By multiplying the conditional probabilities together for each attribute for a given class value, we have a probability of a data instance belonging to that class.

To make a prediction we can calculate probabilities of the instance belonging to each class and select the class value with the highest probability.

Naive bases is often described using categorical data because it is easy to describe and calculate using ratios. A more useful version of the algorithm for our purposes supports numeric attributes and assumes the values of each numerical attribute are normally distributed (fall somewhere on a bell curve). Again, this is a strong assumption, but still gives robust results.

## Predict the Onset of Diabetes

The test problem we will use in this tutorial is the [Pima Indians Diabetes problem](#).

This problem is comprised of 768 observations of medical details for Pima indian patients. The records describe instantaneous measurements taken from the patient such as their age, the number of times pregnant, body mass index, blood sugar level, and whether the patient is aged 21 or older. All attributes are numeric, and their units vary from attribute to attribute.

Each record has a class value that indicates whether the patient suffered an onset of diabetes within 10 years after the measurements were taken (1) or not (0).

This is a standard dataset that has been studied a lot in machine learning literature.

Below is a sample from the [pima-indians.data.csv](#) file to get a sense of the data.

**NOTE:** Download [this file](#) and save it with a .csv extension (e.g. **pima-indians-attributes.csv**).

### Machine Learning Crash Course

You can learn and apply Machine Learning even without a background in maths or a fancy degree. Find out how in this ridiculously practical (and totally free) email course.

SIGN ME UP

```
1 6,148,72,35,0,33.6,0.627,50,1
2 1,85,66,29,0,26.6,0.351,31,0
3 8,183,64,0,0,23.3,0.672,32,1
4 1,89,66,23,94,28.1,0.167,21,0
5 0,137,40,35,168,43.1,2.288,33,1
```

## Naive Bayes Algorithm Tutorial

This tutorial is broken down into the following steps:

1. **Handle Data:** Load the data from CSV file and split it into training and test datasets.
2. **Summarize Data:** summarize the properties in the training dataset so that we can calculate probabilities and make predictions.
3. **Make a Prediction:** Use the summaries of the dataset to generate a single prediction.
4. **Make Predictions:** Generate predictions given a test dataset and a summarized training dataset.
5. **Evaluate Accuracy:** Evaluate the accuracy of predictions made for a test dataset as the percentage correct out of all predictions made.
6. **Tie it Together:** Use all of the code elements to present a complete and standalone implementation of the Naive Bayes algorithm.

### 1. Handle Data

The first thing we need to do is load our data file. The data is in CSV format with the open function and read the data lines using the reader function in the csv module.

We also need to convert the attributes that were loaded as strings into numbers using the float function for loading the Pima indians dataset.

```
1 import csv
2 def loadCsv(filename):
3     lines = csv.reader(open(filename, "rb"))
4     dataset = list(lines)
5     for i in range(len(dataset)):
6         dataset[i] = [float(x) for x in dataset[i]]
```

### Machine Learning Crash Course

You can learn and apply Machine Learning even without a background in maths or a fancy degree. Find out how in this ridiculously practical (and totally free) email course.

SIGN ME UP

```
7 return dataset
```

We can test this function by loading the pima indians dataset and printing the number of data instances that were loaded.

```
1 filename = 'pima-indians-diabetes.data.csv'
2 dataset = loadCsv(filename)
3 print('Loaded data file {0} with {1} rows').format(filename, len(dataset))
```

Running this test, you should see something like:

```
1 Loaded data file pima-indians-diabetes.data.csv rows
```

Next we need to split the data into a training dataset that Naive Bayes can use to make predictions and a test dataset that we can use to evaluate the accuracy of the model. We need to split the data set randomly into train and datasets with a ratio of 67% train and 33% test (this is a common ratio for testing an algorithm on a dataset).

Below is the **splitDataset()** function that will split a given dataset into a given split ratio.

```
1 import random
2 def splitDataset(dataset, splitRatio):
3     trainSize = int(len(dataset) * splitRatio)
4     trainSet = []
5     copy = list(dataset)
6     while len(trainSet) < trainSize:
7         index = random.randrange(len(copy))
8         trainSet.append(copy.pop(index))
9     return [trainSet, copy]
```

We can test this out by defining a mock dataset with 5 instances, split it into train and test sets, and print out which data instances ended up where.

```
1 dataset = [[1], [2], [3], [4], [5]]
2 splitRatio = 0.67
3 train, test = splitDataset(dataset, splitRatio)
4 print('Split {0} rows into train with {1} and test with {2}').format(len(dataset), len(train), len(test))
```

Running this test, you should see something like:

## Machine Learning Crash Course

You can learn and apply Machine Learning even without a background in maths or a fancy degree. Find out how in this ridiculously practical (and totally free) email course.

SIGN ME UP

```
1 Split 5 rows into train with [[4], [3], [5]] and test with [[1], [2]]
```

## 2. Summarize Data

The naive bayes model is comprised of a summary of the data in the training dataset. This summary is then used when making predictions.

The summary of the training data collected involves the mean and the standard deviation for each attribute, by class value. For example, if there are two class values and 7 numerical attributes, then we need a mean and standard deviation for each attribute (7) and class value (2) combination, that is 14 attribute summaries.

These are required when making predictions to calculate the probability of specific attribute values belonging to each class value.

We can break the preparation of this summary data down into the following sub-tasks:

1. Separate Data By Class
2. Calculate Mean
3. Calculate Standard Deviation
4. Summarize Dataset
5. Summarize Attributes By Class

### Separate Data By Class

The first task is to separate the training dataset instances by class value so that by creating a map of each class value to a list of instances that belong to the appropriate lists.

The **separateByClass()** function below does just this.

```
1 def separateByClass(dataset):  
2     separated = {}  
3     for i in range(len(dataset)):
```

### Machine Learning Crash Course

You can learn and apply Machine Learning even without a background in maths or a fancy degree. Find out how in this ridiculously practical (and totally free) email course.

SIGN ME UP

```
4     vector = dataset[i]
5     if (vector[-1] not in separated):
6         separated[vector[-1]] = []
7         separated[vector[-1]].append(vector)
8     return separated
```

You can see that the function assumes that the last attribute (-1) is the class value. The function returns a map of class values to lists of data instances.

We can test this function with some sample data, as follows:

```
1 dataset = [[1,20,1], [2,21,0], [3,22,1]]
2 separated = separateByClass(dataset)
3 print('Separated instances: {0}'.format(separated))
```

Running this test, you should see something like:

```
1 Separated instances: {0: [[2, 21, 0]], 1: [[1, 20, 1], [3, 22, 1]]}
```

## Calculate Mean

We need to calculate the mean of each attribute for a class value. The mean is the average of the values, and we will use it as the middle of our gaussian distribution when calculating probabilities.

We also need to calculate the standard deviation of each attribute for a class value. The standard deviation is a measure of the spread of the data, and we will use it to characterize the expected spread of each attribute when calculating probabilities.

The standard deviation is calculated as the square root of the variance. The variance is the average of the squared differences for each attribute value from the mean. Note we are using the N-1 degrees of freedom when calculating the variance.

```
1 import math
2 def mean(numbers):
3     return sum(numbers)/float(len(numbers))
```

## Machine Learning Crash Course

You can learn and apply Machine Learning even without a background in maths or a fancy degree. Find out how in this ridiculously practical (and totally free) email course.

[SIGN ME UP](#)

```
4
5 def stdev(numbers):
6     avg = mean(numbers)
7     variance = sum([pow(x-avg,2) for x in numbers])/float(len(numbers)-1)
8     return math.sqrt(variance)
```

We can test this by taking the mean of the numbers from 1 to 5.

```
1 numbers = [1,2,3,4,5]
2 print('Summary of {0}: mean={1}, stdev={2}').format(numbers, mean(numbers), stdev(numbers))
```

Running this test, you should see something like:

```
1 Summary of [1, 2, 3, 4, 5]: mean=3.0, stdev=1.58113883008
```

## Summarize Dataset

Now we have the tools to summarize a dataset. For a given list of instances (for a class value) we can calculate the mean and the standard deviation for each attribute.

The zip function groups the values for each attribute across our data instances and standard deviation values for the attribute.

```
1 def summarize(dataset):
2     summaries = [(mean(attribute), stdev(attribute)) for attribute in dataset[0]]
3     del summaries[-1]
4     return summaries
```

We can test this **summarize()** function with some test data that shows marked first and second data attributes.

```
1 dataset = [[1,20,0], [2,21,1], [3,22,0]]
2 summary = summarize(dataset)
3 print('Attribute summaries: {0}').format(summary)
```

Running this test, you should see something like:

## Machine Learning Crash Course

You can learn and apply Machine Learning even without a background in maths or a fancy degree. Find out how in this ridiculously practical (and totally free) email course.

SIGN ME UP



```
1 Attribute summaries: [(2.0, 1.0), (21.0, 1.0)]
```

## Summarize Attributes By Class

We can pull it all together by first separating our training dataset into instances grouped by class. Then calculate the summaries for each attribute.

```
1 def summarizeByClass(dataset):
2     separated = separateByClass(dataset)
3     summaries = {}
4     for classValue, instances in separated.iteritems():
5         summaries[classValue] = summarize(instances)
6     return summaries
```

We can test this **summarizeByClass()** function with a small test dataset.

```
1 dataset = [[1,20,1], [2,21,0], [3,22,1], [4,22,0]]
2 summary = summarizeByClass(dataset)
3 print('Summary by class value: {0}').format(summary)
```

Running this test, you should see something like:

```
1 Summary by class value:
2 {0: [(3.0, 1.4142135623730951), (21.5, 0.7071067811865476)],
3 1: [(2.0, 1.4142135623730951), (21.0, 1.4142135623730951)]}
```

## 3. Make Prediction

We are now ready to make predictions using the summaries prepared from our training data. To make a prediction, we calculate the probability that a given data instance belongs to each class, then selecting the class with the highest probability.

We can divide this part into the following tasks:

1. Calculate Gaussian Probability Density Function
2. Calculate Class Probabilities

### Machine Learning Crash Course

You can learn and apply Machine Learning even without a background in maths or a fancy degree. Find out how in this ridiculously practical (and totally free) email course.

SIGN ME UP

3. Make a Prediction
4. Estimate Accuracy

### Calculate Gaussian Probability Density Function

We can use a Gaussian function to estimate the probability of a given attribute value, given the known mean and standard deviation for the attribute estimated from the training data.

Given that the attribute summaries were prepared for each attribute and class value, the result is the conditional probability of a given attribute value given a class value.

See the references for the details of this equation for the Gaussian probability density function. In summary we are plugging our known details into the Gaussian (attribute value, mean and standard deviation) and reading off the likelihood that our attribute value belongs to the class.

In the **calculateProbability()** function we calculate the exponent first, then calculate the main division. This lets us fit the equation nicely on two lines.

```
1 import math
2 def calculateProbability(x, mean, stdev):
3     exponent = math.exp(-(math.pow(x-mean,2)/(2*math.pow(stdev,2))))
4     return (1 / (math.sqrt(2*math.pi) * stdev)) * exponent
```

We can test this with some sample data, as follows.

```
1 x = 71.5
2 mean = 73
3 stdev = 6.2
4 probability = calculateProbability(x, mean, stdev)
5 print('Probability of belonging to this class: {}'.format(probability))
```

Running this test, you should see something like:

```
1 Probability of belonging to this class: 0.0624896575937
```

### Machine Learning Crash Course

You can learn and apply Machine Learning even without a background in maths or a fancy degree. Find out how in this ridiculously practical (and totally free) email course.

**SIGN ME UP**

## Calculate Class Probabilities

Now that we can calculate the probability of an attribute belonging to a class, we can combine the probabilities of all of the attribute values for a data instance and come up with a probability of the entire data instance belonging to the class.

We combine probabilities together by multiplying them. In the **calculateClassProbabilities()** below, the probability of a given data instance is calculated by multiplying together the attribute probabilities for each class. the result is a map of class values to probabilities.

```
1 def calculateClassProbabilities(summaries, inputVector):
2     probabilities = {}
3     for classValue, classSummaries in summaries.iteritems():
4         probabilities[classValue] = 1
5         for i in range(len(classSummaries)):
6             mean, stdev = classSummaries[i]
7             x = inputVector[i]
8             probabilities[classValue] *= calculateProbability(x, mean, stdev)
9     return probabilities
```

We can test the **calculateClassProbabilities()** function.

```
1 summaries = {0:[(1, 0.5)], 1:[(20, 5.0)]}
2 inputVector = [1.1, '?']
3 probabilities = calculateClassProbabilities(summaries, inputVector)
4 print('Probabilities for each class: {0}'.format(probabilities))
```

Running this test, you should see something like:

```
1 Probabilities for each class: {0: 0.7820853879509118, 1: 6.2987362581e-05}
```

## Make a Prediction

Now that can calculate the probability of a data instance belonging to each class return the associated class.

The **predict()** function belong does just that.

### Machine Learning Crash Course

You can learn and apply Machine Learning even without a background in maths or a fancy degree. Find out how in this ridiculously practical (and totally free) email course.

SIGN ME UP

```
1 def predict(summaries, inputVector):
2     probabilities = calculateClassProbabilities(summaries, inputVector)
3     bestLabel, bestProb = None, -1
4     for classValue, probability in probabilities.items():
5         if bestLabel is None or probability > bestProb:
6             bestProb = probability
7             bestLabel = classValue
8     return bestLabel
```

We can test the **predict()** function as follows:

```
1 summaries = {'A':[(1, 0.5)], 'B':[(20, 5.0)]}
2 inputVector = [1.1, '?']
3 result = predict(summaries, inputVector)
4 print('Prediction: {0}').format(result)
```

Running this test, you should see something like:

```
1 Prediction: A
```

## 4. Make Predictions

Finally, we can estimate the accuracy of the model by making predictions for each test instance. The **getPredictions()** will do this and return a list of predictions for each test instance.

```
1 def getPredictions(summaries, testSet):
2     predictions = []
3     for i in range(len(testSet)):
4         result = predict(summaries, testSet[i])
5         predictions.append(result)
6     return predictions
```

We can test the **getPredictions()** function.

```
1 summaries = {'A':[(1, 0.5)], 'B':[(20, 5.0)]}
2 testSet = [[1.1, '?'], [19.1, '?']]
3 predictions = getPredictions(summaries, testSet)
4 print('Predictions: {0}').format(predictions)
```

### Machine Learning Crash Course

You can learn and apply Machine Learning even without a background in maths or a fancy degree. Find out how in this ridiculously practical (and totally free) email course.

SIGN ME UP

Running this test, you should see something like:

```
1 Predictions: ['A', 'B']
```

## 5. Get Accuracy

The predictions can be compared to the class values in the test dataset and a classification accuracy can be calculated as an accuracy ratio between 0% and 100%. The **getAccuracy()** will calculate this accuracy ratio.

```
1 def getAccuracy(testSet, predictions):
2     correct = 0
3     for x in range(len(testSet)):
4         if testSet[x][-1] == predictions[x]:
5             correct += 1
6     return (correct/float(len(testSet))) * 100.0
```

We can test the **getAccuracy()** function using the sample code below.

```
1 testSet = [[1,1,1,'a'], [2,2,2,'a'], [3,3,3,'b']]
2 predictions = ['a', 'a', 'a']
3 accuracy = getAccuracy(testSet, predictions)
4 print('Accuracy: {0}'.format(accuracy))
```

Running this test, you should see something like:

```
1 Accuracy: 66.666666667
```

## 6. Tie it Together

Finally, we need to tie it all together.

Below provides the full code listing for Naive Bayes implemented from scratch in Python.

```
1 # Example of Naive Bayes implemented from Scratch in Python
2 import csv
3 import random
```

### Machine Learning Crash Course

You can learn and apply Machine Learning even without a background in maths or a fancy degree. Find out how in this ridiculously practical (and totally free) email course.

SIGN ME UP

```
4 import math
5
6 def loadCsv(filename):
7     lines = csv.reader(open(filename, "rb"))
8     dataset = list(lines)
9     for i in range(len(dataset)):
10         dataset[i] = [float(x) for x in dataset[i]]
11     return dataset
12
13 def splitDataset(dataset, splitRatio):
14     trainSize = int(len(dataset) * splitRatio)
15     trainSet = []
16     copy = list(dataset)
17     while len(trainSet) < trainSize:
18         index = random.randrange(len(copy))
19         trainSet.append(copy.pop(index))
20     return [trainSet, copy]
21
22 def separateByClass(dataset):
23     separated = {}
24     for i in range(len(dataset)):
25         vector = dataset[i]
26         if (vector[-1] not in separated):
27             separated[vector[-1]] = []
28         separated[vector[-1]].append(vector)
29     return separated
30
31 def mean(numbers):
32     return sum(numbers)/float(len(numbers))
33
34 def stdev(numbers):
35     avg = mean(numbers)
36     variance = sum([pow(x-avg,2) for x in numbers])/float(len(numbers))
37     return math.sqrt(variance)
38
39 def summarize(dataset):
40     summaries = [(mean(attribute), stdev(attribute)) for attribute in dataset[0:-1]]
41     del summaries[-1]
42     return summaries
43
44 def summarizeByClass(dataset):
45     separated = separateByClass(dataset)
46     summaries = {}
```

## Machine Learning Crash Course

You can learn and apply Machine Learning even without a background in maths or a fancy degree. Find out how in this ridiculously practical (and totally free) email course.

**SIGN ME UP**

```
47     for classValue, instances in separated.iteritems():
48         summaries[classValue] = summarize(instances)
49     return summaries
50
51 def calculateProbability(x, mean, stdev):
52     exponent = math.exp(-(math.pow(x-mean,2)/(2*math.pow(stdev,2))))
53     return (1 / (math.sqrt(2*math.pi) * stdev)) * exponent
54
55 def calculateClassProbabilities(summaries, inputVector):
56     probabilities = {}
57     for classValue, classSummaries in summaries.iteritems():
58         probabilities[classValue] = 1
59         for i in range(len(classSummaries)):
60             mean, stdev = classSummaries[i]
61             x = inputVector[i]
62             probabilities[classValue] *= calculateProbability(x, mean, stdev)
63     return probabilities
64
65 def predict(summaries, inputVector):
66     probabilities = calculateClassProbabilities(summaries, inputVector)
67     bestLabel, bestProb = None, -1
68     for classValue, probability in probabilities.iteritems():
69         if bestLabel is None or probability > bestProb:
70             bestProb = probability
71             bestLabel = classValue
72     return bestLabel
73
74 def getPredictions(summaries, testSet):
75     predictions = []
76     for i in range(len(testSet)):
77         result = predict(summaries, testSet[i])
78         predictions.append(result)
79     return predictions
80
81 def getAccuracy(testSet, predictions):
82     correct = 0
83     for i in range(len(testSet)):
84         if testSet[i][-1] == predictions[i]:
85             correct += 1
86     return (correct/float(len(testSet))) * 100.0
87
88 def main():
89     filename = 'pima-indians-diabetes.data.csv'
```

## Machine Learning Crash Course

You can learn and apply Machine Learning even without a background in maths or a fancy degree. Find out how in this ridiculously practical (and totally free) email course.

**SIGN ME UP**

```

90     splitRatio = 0.67
91     dataset = loadCsv(filename)
92     trainingSet, testSet = splitDataset(dataset, splitRatio)
93     print('Split {0} rows into train={1} and test={2} rows').format(len(dataset), len(trainingSet), len(testSet))
94     # prepare model
95     summaries = summarizeByClass(trainingSet)
96     # test model
97     predictions = getPredictions(summaries, testSet)
98     accuracy = getAccuracy(testSet, predictions)
99     print('Accuracy: {0}%').format(accuracy)
100
101 main()

```

Running the example provides output like the following:

```

1 Split 768 rows into train=514 and test=254 rows
2 Accuracy: 76.3779527559%

```

## Implementation Extensions

This section provides you with ideas for extensions that you could apply and investigate with the Python code you have implemented as part of this tutorial.

You have implemented your own version of Gaussian Naive Bayes in python from scratch.

You can extend the implementation further.

- **Calculate Class Probabilities:** Update the example to summarize the probabilities for each class as a ratio. This can be calculated as the probability of a data instance belonging to each class. For example an instance had features A and B, the likelihood of the instance belonging to class A is  $(0.02 / (0.02 + 0.001))$ .
- **Log Probabilities:** The conditional probabilities for each class given an attribute value. Together they result in very small values, which can lead to floating point underflow. A common fix for this is to combine the log of the probabilities together. Research this further.

### Machine Learning Crash Course

You can learn and apply Machine Learning even without a background in maths or a fancy degree. Find out how in this ridiculously practical (and totally free) email course.

**SIGN ME UP**



- **Nominal Attributes:** Update the implementation to support nominal attributes. This is much similar and the summary information you can collect for each attribute is the ratio of category values for each class. Dive into the references for more information.
- **Different Density Function** (*bernoulli* or *multinomial*): We have looked at Gaussian Naive Bayes, but you can also look at other distributions. Implement a different distribution such as multinomial, bernoulli or kernel naive bayes that make different assumptions about the distribution of attribute values and/or their relationship with the class value.

## Resources and Further Reading

This section will provide some resources that you can use to learn more about the Naive Bayes algorithm in terms of both theory of how and why it works and practical concerns for implementing it in code.

### Problem

More resources for learning about the problem of predicting the onset of diabetes.

- [Pima Indians Diabetes Data Set](#): This page provides access to the dataset files, describes the attributes and lists papers that use the dataset.
- [Dataset File](#): The dataset file.
- [Dataset Summary](#): Description of the dataset attributes.
- [Diabetes Dataset Results](#): The accuracy of many standard algorithms on the dataset.

### Code

This section links to open source implementations of Naive Bayes in popular machine learning libraries, as well as considering implementing your own version of the method for operational use.

- [Naive Bayes in Scikit-Learn](#): Implementation of naive bayes in the scikit-learn library.
- [Naive Bayes documentation](#): Scikit-Learn documentation and sample code.
- [Simple Naive Bayes in Weka](#): Weka implementation of naive bayes

### Machine Learning Crash Course

You can learn and apply Machine Learning even without a background in maths or a fancy degree. Find out how in this ridiculously practical (and totally free) email course.

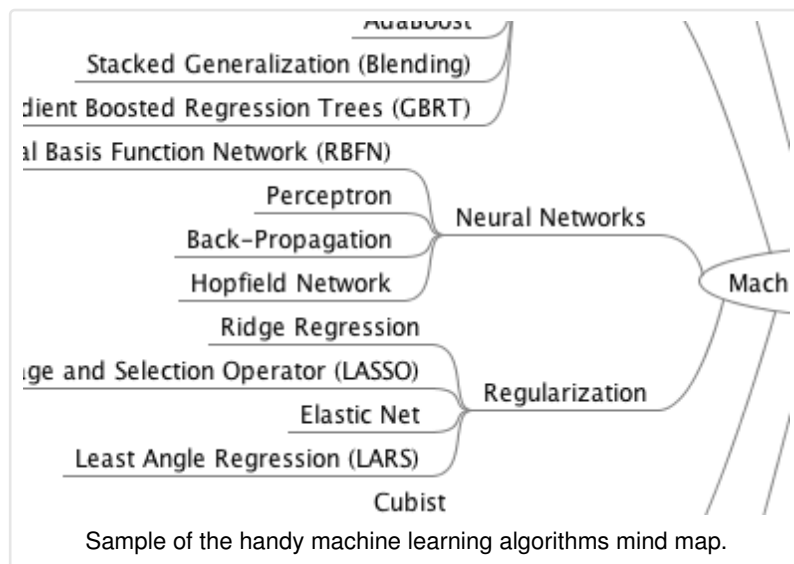
SIGN ME UP

## Books

You may have one or more books on applied machine learning. This section highlights the sections or chapters in common applied books on machine learning that refer to Naive Bayes.

- [Applied Predictive Modeling](#), page 353
- [Data Mining: Practical Machine Learning Tools and Techniques](#), page 94
- [Machine Learning for Hackers](#), page 78
- [An Introduction to Statistical Learning: with Applications in R](#), page 138
- [Machine Learning: An Algorithmic Perspective](#), page 171
- [Machine Learning in Action](#), page 61 (Chapter 4)
- [Machine Learning](#), page 177 (chapter 6)

## Get your FREE Algorithms Mind Map



I've created a handy mind map of 60+ algorithms organized by type.

Download it, print it, and use it.

Also get exclusive access to the

### Machine Learning Crash Course

You can learn and apply Machine Learning even without a background in maths or a fancy degree. Find out how in this ridiculously practical (and totally free) email course.

**SIGN ME UP**

## Next Step

Take action.

Follow the tutorial and implement Naive Bayes from scratch. Adapt the example to another problem. Follow the extensions and improve upon the implementation.

Leave a comment and share your experiences.

**Update:** Check out the follow-up on tips for using the naive bayes algorithm titled: “[Better Naive Bayes: 12 Tips To Get The Most From The Naive Bayes Algorithm](#)”

---

## Need Help Getting Past The Math?

Finally understand how machine learning algorithms work, step-by-step in the new Ebook:

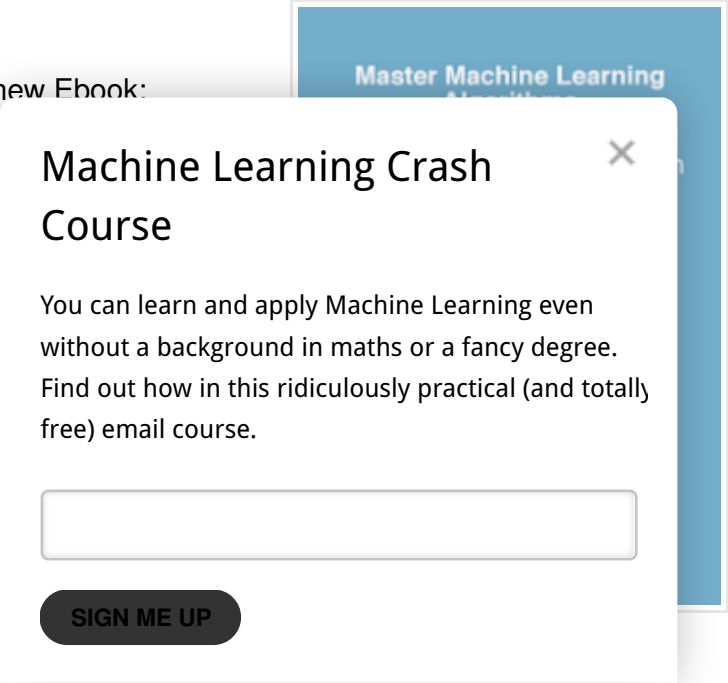
[Master Machine Learning Algorithms](#)

Take the next step with **12 self-study tutorials** across  
**10 top machine learning algorithms**.

Includes spreadsheets that show exactly how everything is calculated.

Ideal for beginners with no math background.

[Pull Back the Curtain on Machine Learning Algorithms](#)

A sign-up modal for a 'Machine Learning Crash Course'. It features a title, a description, a text input field, and a 'SIGN ME UP' button. The modal is overlaid on a background image of a book titled 'Master Machine Learning Algorithms'.

### Machine Learning Crash Course

You can learn and apply Machine Learning even without a background in maths or a fancy degree. Find out how in this ridiculously practical (and totally free) email course.

SIGN ME UP



About Jason Brownlee

Jason is the editor-in-chief at MachineLearningMastery.com. He is a husband, proud father, academic researcher, author, professional developer and a machine learning practitioner. He has a Masters and PhD in Artificial Intelligence, has published books on Machine Learning and has written operational code that is running in production. [Learn more](#).

[View all posts by Jason Brownlee](#) →

< Lessons Learned from Building Machine Learning Systems

Better Naive Bayes: 12 Tips To Get The Most From The Naive Bayes Algorithm >

56 Responses to *How To Implement Naive Bayes From Scratch in Python*



**david jensen** December 12, 2014 at 3:28 am #

Statistical methods should be developed from scratch because of misund



**Anurag** December 14, 2014 at 1:11 pm #

This is a wonderful article. Your blog is one of those blogs that I visit eve about the programming language that should be used for building these algorithm because it's easy to write code by importing useful libraries that are already availa beginner in practical ML, I have tried to write efficient codes before I started learni complexities involved in coding if you're using C++: more coding is to be done tha

Machine Learning Crash Course

You can learn and apply Machine Learning even without a background in maths or a fancy degree. Find out how in this ridiculously practical (and free) email course.

SIGN ME UP

language is your preference and under what situations? I know that it's lame to ask about preferences of programming language as it is essentially a personal choice. But still I'd like you to share your take on this. Also try to share the trade-offs while choosing these programming languages.

Thank you.



**Jason Brownlee** December 15, 2014 at 7:53 am #

REPLY ↩

Thanks Anurag



**Alcides Schulz** January 15, 2015 at 12:32 am #

REPLY ↩

Hi Jason, found your website and read it in one day. Thank you, it really helped me to understand ML and what to do. I did the 2 examples here and I think I will take a look at scikit-learn now.

I have a personal project that I want to use ML, and I'll keep you posted on the progress.

One small note on this post, is on the "1. Handle data" you refer to iris.data from p

Thank you so much, example is really good to show how to do it. Please keep it c



**Jason Brownlee** January 15, 2015 at 7:43 am #

Thanks for the kind words Alcides.

Fixed the reference to the iris dataset.

## Machine Learning Crash Course

You can learn and apply Machine Learning even without a background in maths or a fancy degree. Find out how in this ridiculously practical (and totally free) email course.

SIGN ME UP



**toolate** January 22, 2015 at 2:16 am #

REPLY ↩

Hi Jason, still one more note on your post, is on the “1. Handle data” the flower measures that you refer to iris.data.



**Jason Brownlee** January 22, 2015 at 5:43 am #

REPLY ↩

Thanks, fixed!



**Tamilselvan** February 4, 2015 at 11:37 pm #

REPLY ↩

Great Article. Learned a Lot. Thanks. Thanks.



**Abhinav kumar** February 23, 2015 at 8:13 pm #

thank u



**Jason Brownlee** February 24, 2015 at 7:40 am #

You're welcome!



**Roy** March 7, 2015 at 2:53 pm #

REPLY ↩

## Machine Learning Crash Course

You can learn and apply Machine Learning even without a background in maths or a fancy degree. Find out how in this ridiculously practical (and totally free) email course.

SIGN ME UP



Thanks for your nice article. I really appreciate the step by step instructions.



**malini** March 17, 2015 at 7:19 pm #

REPLY ↩

hello sir, plz tell me how to compare the data set using naive Bayes algorithm.



**Isha** March 21, 2015 at 5:40 pm #

REPLY ↩

Why does the accuracy change every time you run this code?

when i tried running this code every time it gave me different accuracy percentage in the range from 70-78%

Why is it so?

Why is it not giving a constant accuracy percent?



**Harry** April 9, 2015 at 8:37 am #

As Splitting of dataset into testdata and traindata is done using a ran



**Sheepsy90** March 25, 2015 at 8:12 pm #

Hey nice article – one question – why do you use the N-1 in the STD Dev

## Machine Learning Crash Course



You can learn and apply Machine Learning even without a background in maths or a fancy degree. Find out how in this ridiculously practical (and totally free) email course.

SIGN ME UP

REPLY ↩



**Jason Brownlee** March 26, 2015 at 5:33 am #

Thanks!

N-1 is the sample (corrected or unbiased) standard deviation. [See this wiki page](#).



**Vaishali** April 8, 2015 at 6:01 pm #

REPLY ↩

Hey! Thanks a ton! This was very useful.

It would be great if you give an idea on how other metrics like precision and recall can be calculated.

Thanks!



**Ashwin Perti** April 24, 2015 at 5:28 pm #

REPLY ↩

Sir

When I am running the same code in IDLE (python 2.7) – the code is working fine coming is:

- 1) warning – unused variable dataset
- 2) undefined variable dataset in for loop

Why this difference.



**Melvin Tjon Akon** May 21, 2015 at 1:46 am #

Great post, Jason.

## Machine Learning Crash Course

You can learn and apply Machine Learning even without a background in maths or a fancy degree. Find out how in this ridiculously practical (and totally free) email course.

REPLY ↩

SIGN ME UP



For a MBA/LLM, it makes naive bayes very easy to understand and to implement in legal coding. Looking forward to read more. Best,  
Melvin



**Igor Balabine** June 10, 2015 at 11:44 am #

REPLY ↩

Jason,

Great example. Thanks! One nit: "calculateProbability" is not a good name for a function which actually calculates Gaussian probability density – pdf value may be greater than 1.

Cheers,

-Igor



**Alex Ubot** July 2, 2015 at 10:06 pm #

REPLY ↩

Hi Jason,

Fantastic post. I really learnt a lot. However I do have a question? Why don't you

?

If I understood the model correctly, everything is based on the bayes theorem :

$$P(y|x_1.....x_n) = P(x_1.....x_n|y) * P(y) / P(x_1.....x_n)$$

$P(x_1.....x_n)$  will be a constant so we can get rid of it.

Your post explain very well how to calculate  $P(x_1.....x_n|y)$  (assumption made that

$$P(x_1.....x_n|y) = P(x_1|y) * .... P(x_n|y)$$

How about  $p(y)$  ? I assume that we should calculate the frequency of the observed probabilities[classValue] so that we have :

$$P(y|x_1.....x_n) = \text{frequency}(\text{classValue}) * \text{probabilities}[\text{classValue}]$$

## Machine Learning Crash Course

You can learn and apply Machine Learning even without a background in maths or a fancy degree. Find out how in this ridiculously practical (and totally free) email course.

SIGN ME UP

Otherwise let's assume that in a training set of 500 lines, we have two class 0 and 1 but observed 100 times 0 et 400 times 1. If we do not compute the frequency, then the probability may be biased, right ? Did I misunderstand something ? Hopefully my post is clear. I really hope that you will reply because I am a bit confused.

Thanks  
Alex



**Babu** February 28, 2016 at 7:43 am #

REPLY ↩

I have the same question – why is multiplying by  $p(y)$  is omitted?



**Babu** March 10, 2016 at 2:09 pm #

REPLY ↩

No Answer yet – no one on internet has answer to this.

Just don't want to accept answers without understanding it.



**frong** April 3, 2016 at 3:15 pm #

yeah , I have the same question too, maybe the  $P(y)$  is ness missing? is it proving that bayes model is powerful?



**gd** April 7, 2016 at 2:27 am #

## Machine Learning Crash Course

You can learn and apply Machine Learning even without a background in maths or a fancy degree. Find out how in this ridiculously practical (and totally free) email course.

SIGN ME UP

hi,

I believe this is because  $P(y) = 1$  as classes are already segregated before calculating  $P(x_1 \dots x_n | Y)$ .

Can experts comment on this please?



**Babu** May 23, 2016 at 7:32 am #

There is huge bug in this implementation;

First of all the implementation using GaussianNB gives totally a different answer.

Why is no one is replying even after 2 months of this.

My concern is, there are so many more bad bayesians in a wrong concept.

My lead read this article and now he thinks I am wrong

At least the parameters are correct – something wrong with calculating probs.

```
def SplitXy(Xy):
    Xy10=Xy[0:8]
    Xy10 = Xy;
    #print Xy10
    #print "======"
    zXy10=list(zip(*Xy10))
    y= zXy10[-1]
    del zXy10[-1]
    z1=zip(*zXy10)
    X=[list(t) for t in z1]
    return X,y

from sklearn.naive_bayes import GaussianNB
X,y = SplitXy(trainingSet)
```

## Machine Learning Crash Course

You can learn and apply Machine Learning even without a background in maths or a fancy degree. Find out how in this ridiculously practical (and totally free) email course.

**SIGN ME UP**

```

Xt,yt = SplitXy(testSet)

model = GaussianNB()
model.fit(X, y)

### Compare the models built by Python

print ("Class: 0")
for i,j in enumerate(model.theta_[0]):
    print ("({:8.2f} {:9.2f} {:7.2f} )".format(j, model.sigma_[0][i], sqrt(model.sigma_[0][i])) , end="")
    print ("==> ", summaries[0][i])

print ("Class: 1")
for i,j in enumerate(model.theta_[1]):
    print ("({:8.2f} {:9.2f} {:7.2f} )".format(j, model.sigma_[1][i], sqrt(model.sigma_[1][i])) , end="")
    print ("==> ", summaries[1][i])

"""
Class: 0
( 3.18 9.06 3.01 )==> (3.1766467065868262, 3.014767379963071)
( 109.12 699.16 26.44 )==> (109.11976047904191, 26.48129316)
( 68.71 286.46 16.93 )==> (68.712574850299404, 16.950414098)
( 19.74 228.74 15.12 )==> (19.742514970059879, 15.146913806)
( 68.64 10763.69 103.75 )==> (68.640718562874255, 103.90387)
( 30.71 58.05 7.62 )==> (30.710778443113771, 7.630215185470)
( 0.42 0.09 0.29 )==> (0.42285928143712581, 0.2940929986424)
( 30.66 118.36 10.88 )==> (30.658682634730539, 10.895778423)
Class: 1
( 4.76 12.44 3.53 )==> (4.761111111111111, 3.5365037952376)
( 139.17 1064.54 32.63 )==> (139.17222222222222, 32.7183393)
( 69.27 525.24 22.92 )==> (69.27222222222226, 22.982099071)
( 22.64 309.59 17.60 )==> (22.638888888888889, 17.644143437)
( 101.13 20409.91 142.86 )==> (101.12777777777778, 143.2617)

```

## Machine Learning Crash Course

You can learn and apply Machine Learning even without a background in maths or a fancy degree. Find out how in this ridiculously practical (and totally free) email course.

**SIGN ME UP**

```
( 34.99 57.18 7.56 )==> (34.99388888888889, 7.5825893182809425)
( 0.54 0.14 0.37 )==> (0.53544444444444439, 0.3702077209795522)
( 36.73 112.86 10.62 )==> (36.727777777777774, 10.653417924304598)
'''
```



**EL YAMANI** May 22, 2016 at 8:57 am #

REPLY ↩

Hello,

Thanks for this article , it is very helpful . I just have a remark about the probability that you are calculating which is  $P(x|C_k)$  and then you make predictions, the result will be biased since you don't multiply by  $P(C_k)$  ,  $P(x)$  can be omitted since it's only a normalisation constant.



**Anand** July 20, 2015 at 9:12 pm #

REPLY ↩

Thanks a lot for this tutorial, Jason.

I have a quick question if you can help.

In the `separateByClass()` definition, I could not understand how `vector[-1]` is a right

If I try the same commands one by one outside the function, the line of code with no attribute `'__getitem__'`.

Then how is it working inside the function?

I am sorry for my ignorance. I am new to python. Thank you.

## Machine Learning Crash Course

You can learn and apply Machine Learning even without a background in maths or a fancy degree. Find out how in this ridiculously practical (and totally free) email course.

SIGN ME UP



**Sarah** August 26, 2015 at 5:50 pm #

REPLY ↩

Hello Jason! I just wanted to leave a message to say thank you for the website. I am preparing for a job in this field and it has helped me so much. Keep up the amazing work!! 😊



**Jason Brownlee** August 26, 2015 at 6:56 pm #

REPLY ↩

You're welcome! Thanks for leaving such a kind comment, you really made my day 😊



**Jaime Lopez** September 7, 2015 at 8:52 am #

REPLY ↩

Hi Jason,

Very easy to follow your classifier. I try it and works well on your data, but is impossible to follow on my data, so maybe one have to transform your data from categorical to numerical format.

Another thing, when I transformed one database, sometimes the algorithm find different number on features and classes.

Any suggestion Jason?

Thanks, Jaime

## Machine Learning Crash Course

You can learn and apply Machine Learning even without a background in maths or a fancy degree. Find out how in this ridiculously practical (and totally free) email course.

SIGN ME UP



**syed belgam** April 11, 2016 at 2:05 pm #

REPLY ↩

Thanks



**eduardo** September 28, 2015 at 1:32 pm #

REPLY ↩

It is by far the best material I've found , please continue helping the community!



**Jason Brownlee** September 29, 2015 at 5:25 am #

REPLY ↩

Thanks eduardo!



**Thibroll** September 29, 2015 at 9:11 pm #

Hello.

This is all well explained, and depicts well the steps of machine learning. But the v lead to unwanted error.

Here, in theory, using the Bayes law, we know that :  $P(y|X) = P(y).P(X|y)/P(X)$ . As ignore  $P(X)$  and pick the result for the maximized value of  $P(y).P(X|y)$

2 points remain inconsistent :

- First, you pick a gaussian distribution to estimate  $P(X|y)$ . But here, you calculate the specific points  $X, y$ , with associated mean and deviation, and not the actual pr
- The second point is that you don't take into consideration the calculation of  $P(y)$

## Machine Learning Crash Course

REPLY ↩

You can learn and apply Machine Learning even without a background in maths or a fancy degree. Find out how in this ridiculously practical (and totally free) email course.

SIGN ME UP

probability calculation) may work only if all samples have same amount in every value of  $y$  (considering  $y$  is discret), or if you are lucky enough.

Anyway, despite those mathematical issue, this is a good work, and a god introduction to machine learning.



**mondet** October 6, 2015 at 10:08 am #

REPLY ↩

Thanks Jason for all this great material. One thing that i adore from you is the intellectual honesty, the spirit of collaboration and the parsimony.

In my opinion you are one of the best didactics exponents in the ML.

Thanks to Thibroll too. But i would like to have a real example of the problem in R, python or any other language.

Regards,

Emmanuel:.



**Erika** October 15, 2015 at 10:03 am #

Hi Jason,

I have trying to get started with machine learning and your article has given me th  
your efforts! 😊

## Machine Learning Crash Course



You can learn and apply Machine Learning even without a background in maths or a fancy degree. Find out how in this ridiculously practical (and totally free) email course.

SIGN ME UP





**Swagath** November 9, 2015 at 5:35 pm #

REPLY ↩

i need this code in java.. please help me//



**Sarah** November 16, 2015 at 11:54 pm #

REPLY ↩

I am working with this code – tweaking it here or there – have found it very helpful as I implement a NB from scratch. I am trying to take the next step and add in categorical data. Any suggestions on where I can head to get ideas for how to add this? Or any particular functions/methods in Python you can recommend? I've brought in all the attributes and split them into two datasets for continuous vs. categorical so that I can work on them separately before bringing their probabilities back together. I've got the categorical in the same dictionary where the key is the class and the values are lists of attributes for each instance. I'm not sure how to go through the values to count frequencies and then how to store this back up so that I have the attribute values along with their frequencies/probabilities. A dictionary within a dictionary? Should I be going in another direction and not using a similar format?



**Emmanuel Nuakoh** November 19, 2015 at 6:36 am #

Thank you Jason, this tutorial is helping me with my implementation of N



**Anna** January 14, 2016 at 2:32 am #

Hi! thank you! Have you tried to do the same for the textual datasets, for [/20Newsgroups/](#) ? Would appreciate some hints or ideas )

## Machine Learning Crash Course



You can learn and apply Machine Learning even without a background in maths or a fancy degree. Find out how in this ridiculously practical (and totally free) email course.

SIGN ME UP



**Randy** January 16, 2016 at 4:15 pm #

REPLY ↩

Great article, but as others pointed out there are some mathematical mistakes like using the probability density function for single value probabilities.



**Meghna** February 7, 2016 at 7:45 pm #

REPLY ↩

Thank you for this amazing article!! I implemented the same for wine and MNIST data set and these tutorials helped me so much!! 😊



**David** February 7, 2016 at 11:17 pm #

REPLY ↩

I got an error with the first print statement, because your parenthesis are closing the call to print (which returns None) before you're calling format, so instead of

```
print('Split {0} rows into train with {1} and test with {2}').format(len(dataset), train, test)
```

it should be

```
print('Split {0} rows into train with {1} and test with {2}'.format(len(dataset), train, test))
```

Anyway, thanks for this tutorial, it was really useful, cheers!

## Machine Learning Crash Course

You can learn and apply Machine Learning even without a background in maths or a fancy degree. Find out how in this ridiculously practical (and totally free) email course.

REPLY ↩

SIGN ME UP



**Kumar Ramanathan** February 12, 2016 at 12:20 pm #

Sincere gratitude for this most excellent site. Yes, I never learn until I write

exercise, to get concepts embedded into one's brain. Brilliant effort, truly !



**Syed** February 18, 2016 at 8:15 am #

REPLY ↩

Just to test the algorithm, i change the class of few of the data to something else i.e 3 or 4, (last digit in a line) and i get divide by zero error while calculating the variance. I am not sure why. does it mean that this particular program works only for 2 classess? cant see anything which restricts it to that.



**Takuma Udagawa** March 20, 2016 at 1:19 pm #

REPLY ↩

Hi, I'm a student in Japan.

It seems to me that you are calculating  $p(X_1|C_k) * p(X_2|C_k) * \dots * p(X_m|C_k)$  and choosing  $C_k$  such that this value would be maximum.

However, when I looked in the Wikipedia, you are supposed to calculate  $p(X_1|C_k) * p(X_2|C_k) * \dots * p(X_m|C_k) * p(C_k)$ .

I don't understand when you calculated  $p(C_k)$ .

Would you tell me about it?



**Babu** May 23, 2016 at 7:36 am #

This is the same question as Alex Ubot above.

Calculating the parameters are correct.  
but prediction implementation is incorrect.

Unfortunately this article comes up high and everyone is learning incorrect way of

## Machine Learning Crash Course

You can learn and apply Machine Learning even without a background in maths or a fancy degree. Find out how in this ridiculously practical (and totally free) email course.

SIGN ME UP



**Swapnil** June 10, 2016 at 1:21 am #

REPLY ↩

Really nice tutorial. Can you post a detailed implementation of RandomForest as well ? It will be very helpful for us if you do so.

Thanks!!



**Jason Brownlee** June 14, 2016 at 8:20 am #

REPLY ↩

Great idea Swapnil.

You may find this post useful:

<http://machinelearningmastery.com/bagging-and-random-forest-ensemble-algorithms-for-machine-learning/>



**sourena maroofi** July 22, 2016 at 12:24 am #

REPLY ↩

thanks Jason...very nice tutorial.



**Gary** July 27, 2016 at 5:44 pm #

REPLY ↩

Hi,

I was interested in this Naive Bayes example and downloaded the .csv data and t

However, when I try to run it in Pycharm IDE using Python 3.5 I get no end of run.

Has anyone else run the code successfully? And if so, what IDE/environment did

Thanks

## Machine Learning Crash Course

You can learn and apply Machine Learning even without a background in maths or a fancy deg. Find out how in this ridiculously practical (and totally free) email course.

SIGN ME UP

Gary



**Sudarshan** August 10, 2016 at 5:05 pm #

REPLY ↩

Hi Gary,

You might want to run it using Python 2.7.



**Sudarshan** August 10, 2016 at 5:02 pm #

REPLY ↩

Hi,

Thanks for the excellent tutorial. I've attempted to implement the same in Go.

Here is a link for anyone that's interested interested.

<https://github.com/sudsred/gBay>



**Atlas** August 13, 2016 at 6:40 am #

This is AWESOME!!! Thank you Jason.

Where can I find more of this?



**Alex** August 20, 2016 at 4:34 pm #

## Machine Learning Crash Course



You can learn and apply Machine Learning even without a background in maths or a fancy degree. Find out how in this ridiculously practical (and totally free) email course.

SIGN ME UP

REPLY ↩



That can be implemented in any language because there're no special libraries involved.

---

## Leave a Reply

Name (required)

Email (will not be published) (required)

Website

SUBMIT COMMENT

## Welcome!



Hi, my name is Jason and welcome to Machine Learning Mastery.  
We make developers awesome at machine learning.

## Machine Learning Crash Course



You can learn and apply Machine Learning even without a background in maths or a fancy degree. Find out how in this ridiculously practical (and totally free) email course.

SIGN ME UP

## Free Guide to the Best Resources

---

Discover the best videos, books, courses and MUCH MORE in my Machine Learning Resource Guide.

**Download For Free**

## Algorithm that is Winning Competitions

---

Do you want to discover the machine learning algorithm that is winning competitions?

### XGBoost With Python

Gradient Boosted Trees With  
XGBoost and scikit-learn

[Get Started With XGBoost in Python Today!](#)

POPULAR













**How to Run Your First Classifier in Weka**

FEBRUARY 17, 2014

## Machine Learning Crash Course

You can learn and apply Machine Learning even without a background in maths or a fancy degree. Find out how in this ridiculously practical (and totally free) email course.

**SIGN ME UP**

	
	<b>A Tour of Machine Learning Algorithms</b> NOVEMBER 25, 2013
	<b>Machine Learning for Programmers: Leap from developer to machine learning practitioner</b> AUGUST 17, 2015
	<b>Tutorial To Implement k-Nearest Neighbors in Python From Scratch</b> SEPTEMBER 12, 2014
	<b>How To Implement Naive Bayes From Scratch in Python</b> DECEMBER 8, 2014
	<b>Develop Your First Neural Network in Python With Keras Step-By-Step</b> MAY 24, 2016
	<b>4 Self-Study Machine Learning Projects</b> JANUARY 3, 2014
	<b>8 Tactics to Combat Imbalanced Classes in Your Machine Learning Dataset</b> AUGUST 19, 2015
	<b>Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras</b> JULY 21, 2016
	<b>Feature Selection with the Caret R Package</b> SEPTEMBER 22, 2014

## Machine Learning Crash Course

You can learn and apply Machine Learning even without a background in maths or a fancy degree. Find out how in this ridiculously practical (and totally free) email course.

SIGN ME UP





## Buy Some Machine Learning Training

Take the next step. What topic do need help with?

- [Machine Learning Algorithms](#)
- [Applied Machine Learning with Weka](#)
- [Python Machine Learning](#)
- [R Machine Learning](#)
- [Deep Learning With Python](#)

[VIEW ALL PRODUCTS](#)

## Blog Post Categories

- [Deep Learning](#) (36)
- [Machine Learning Algorithms](#) (50)
- [Machine Learning Process](#) (52)
- [Machine Learning Resources](#) (37)
- [Python Machine Learning](#) (65)
- [R Machine Learning](#) (38)
- [Start Machine Learning](#) (65)
- [Weka Machine Learning](#) (37)
- [XGBoost](#) (4)

## Looking for something?

### Machine Learning Crash Course



You can learn and apply Machine Learning even without a background in maths or a fancy degree. Find out how in this ridiculously practical (and totally free) email course.

[SIGN ME UP](#)



---

© 2016 Machine Learning Mastery. All Rights Reserved.

[Privacy](#) | [Contact](#) | [About](#)

## Machine Learning Crash Course ×

You can learn and apply Machine Learning even without a background in maths or a fancy degree. Find out how in this ridiculously practical (and totally free) email course.

**SIGN ME UP**