Want help with machine learning? Take the FREE Crash-Course.

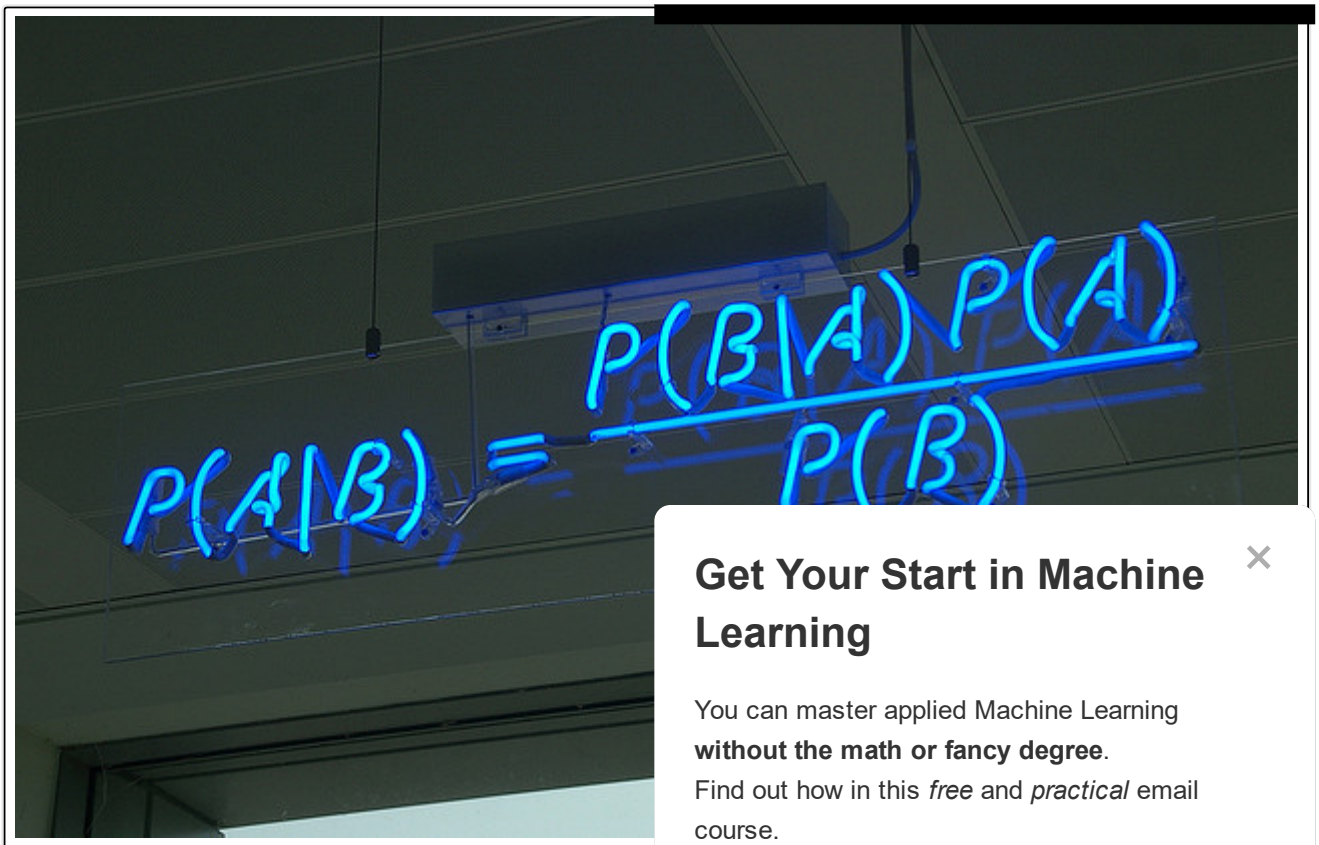
How To Implement Naive Bayes From Scratch in Python

by **Jason Brownlee** on December 8, 2014 in **Algorithms From Scratch**

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"



Naive Baye
Photo by Matt Buck, s

About Naive Bayes

The Naive Bayes algorithm is an intuitive method that uses the probabilities of each attribute belonging to each class to make a prediction. It is the supervised learning approach you would come up with if you wanted to model a predictive modeling problem probabilistically.

Naive bayes simplifies the calculation of probabilities by assuming that the probability of each attribute belonging to a given class 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 I

This problem is comprised of 768 observations of medi
describe instantaneous measurements taken from the p ant
and blood workup. All patients are women aged 21 or o
from attribute to attribute.

Each record has a class value that indicates whether the patient suffered an onset of diabetes within 5 years of when the measurements were taken (1) or not (0).

This is a standard dataset that has been studied a lot in machine learning literature. A good prediction accuracy is 70%-76%.

Below is a sample from the pima-indians.data.csv file to get a sense of the data we will be working with.

NOTE: Download this file and save it with a .csv extension (e.g. **pima-indians-diabetes.data.csv**). See this file for a description of all the attributes.

```
6,148,72,35,0,33.6,0.627,50,1
1,85,66,29,0,26.6,0.351,31,0
8,183,64,0,0,23.3,0.672,32,1
1,89,66,23,94,28.1,0.167,21,0
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 da
quotes. We can open the file with the open function and
csv module.

We also need to convert the attributes that were loaded
m.
Below is the **loadCsv()** function for loading the Pima in

```
import csv
def loadCsv(filename):
    lines = csv.reader(open(filename, "rb"))
    dataset = list(lines)
    for i in range(len(dataset)):
        dataset[i] = [float(x) for x in datase
    return dataset
```

We can test this function by loading the pima indians da
t
were loaded.

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

Running this test, you should see something like:

```
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.

```
import random
def splitDataset(dataset, splitRatio):
    trainSize = int(len(dataset) * splitRatio)
    trainSet = []
    copy = list(dataset)
    while len(trainSet) < trainSize:
        index = random.randrange(len(copy))
        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 training and testing datasets and print them out to see 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), train, test)
```

Running this test, you should see something like:

```
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 t n used when making predictions.

The summary of the training data collected involves the , by class value. For example, if there are two class values nd standard deviation for each attribute (7) and class value

These are required when making predictions to calcula ng to each class value.

We can break the preparation of this summary data dow

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 we can calculate statistics for each class. We can do that by creating a map of each class value to a list of instances that belong to that class and sort the entire dataset of instances into the appropriate lists.

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

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

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

Running this test, you should see something like:

```
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 cl... tendency of the data, and we will use it as the middle of... probabilities.

We also need to calculate the standard deviation of eac... describes the variation of spread of the data, and we w... attribute in our Gaussian distribution when calculating p...

The standard deviation is calculated as the square root... average of the squared differences for each attribute va... which subtracts 1 from the number of attribute values w...


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

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

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

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

Running this test, you should see something like:

```
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 into their own lists so that we can compute the mean and standard deviation values fr...

```
def summarize(dataset):
    summaries = [(mean(attribute), stdev(attribute)) for attribute in zip(*dataset)]
    del summaries[-1]
    return summaries
```

We can test this **summarize()** function with some test data that shows markedly different mean and standard deviation values for the first and second data attributes.

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

Running this test, you should see something like:

```
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
calculate the summaries for each attribute.

```
def summarizeByClass(dataset):
    separated = separateByClass(dataset)
    summaries = {}
    for classValue, instances in separated.ite
        summaries[classValue] = summarize(inst
    return summaries
```

We can test this **summarizeByClass()** function with a

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

Running this test, you should see something like:

```
Summary by class value:
{0: [(3.0, 1.4142135623730951), (21.5, 0.7071067811865476)],
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. Making predictions involves calculating the probability that a given data instance belongs to each class, then selecting the class with the largest probability as the prediction.

We can divide this part into the following tasks:

1. Calculate Gaussian Probability Density Function
2. Calculate Class Probabilities
3. Make a Prediction
4. Estimate Accuracy

Calculate Gaussian Probability Density Functic

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 where 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)/(
4     return (1 / (math.sqrt(2*math.pi) * stdev)
```

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, st
5 print('Probability of belonging to this class:
```

Running this test, you should see something like:

```
1 Probability of belonging to this class: 0.0624
```


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(su[redacted]
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: 6.298736258150442e-05}
```

Make a Prediction

Now that can calculate the probability of a data instance belonging to each class value, we can look for the largest probability and return the associated class.

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

```
1 def predict(summaries, inputVector):
2     probabilities = calculateClassProbabilitie
3     bestLabel, bestProb = None, -1
4     for classValue, probability in probabiliti
5         if bestLabel is None or probability >
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 data instance in our test dataset. 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)
```

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.

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

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

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

Running this test, you should see something like:

```
Accuracy: 66.6666666667
```

6. Tie it Together

Finally, we need to tie it all together.

Below provides the full code listing for Naive Bayes imp

```
# Example of Naive Bayes implemented from Sc
import csv
import random
import math

def loadCsv(filename):
    lines = csv.reader(open(filename, "rb"))
    dataset = list(lines)
    for i in range(len(dataset)):
        dataset[i] = [float(x) for x in dataset[i]]
    return dataset

def splitDataset(dataset, splitRatio):
    trainSize = int(len(dataset) * splitRatio)
    trainSet = []
    copy = list(dataset)
    while len(trainSet) < trainSize:
        index = random.randrange(len(copy))
        trainSet.append(copy.pop(index))
    return [trainSet, copy]

def separateByClass(dataset):
    separated = {}
    for i in range(len(dataset)):
        vector = dataset[i]
        if (vector[-1] not in separated):
            separated[vector[-1]] = []
        separated[vector[-1]].append(vector)
    return separated

```

```
def mean(numbers):
    return sum(numbers)/float(len(numbers))

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

def summarize(dataset):
    summaries = [(mean(attribute), stdev(attribute)) for attribute in zip(*dataset)]
    del summaries[-1]
    return summaries

def summarizeByClass(dataset):
    separated = separateByClass(dataset)
    summaries = {}
    for classValue, instances in separated.iteritems():
        summaries[classValue] = summarize(in
    return summaries

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

def calculateClassProbabilities(summaries, i
    probabilities = {}
    for classValue, classSummaries in summar
        probabilities[classValue] = 1
        for i in range(len(classSummaries)):
            mean, stdev = classSummaries[i]
            x = inputVector[i]
            probabilities[classValue] *= cal
    return probabilities

def predict(summaries, inputVector):
    probabilities = calculateClassProbabilit
    bestLabel, bestProb = None, -1
    for classValue, probability in probabilities.iteritems():
        if bestLabel is None or probability > bestProb:
            bestProb = probability
            bestLabel = classValue
    return bestLabel

def getPredictions(summaries, testSet):
    predictions = []
    for i in range(len(testSet)):
        result = predict(summaries, testSet[i])
        predictions.append(result)
    return predictions

def getAccuracy(testSet, predictions):
    correct = 0
    for i in range(len(testSet)):
        if testSet[i][-1] == predictions[i]:
            correct += 1
    return (correct/float(len(testSet))) * 100.0

def main():
    filename = 'pima-indians-diabetes.data.csv'
    splitRatio = 0.67
    dataset = loadCsv(filename)
    trainingSet, testSet = splitDataset(dataset, splitRatio)
    print('Split {0} rows into train={1} and
    # prepare model
    summaries = summarizeByClass(trainingSet
```

```
    # test model
    predictions = getPredictions(summaries, testSet)
    accuracy = getAccuracy(testSet, predictions)
    print('Accuracy: {0}%').format(accuracy)

main()
```

Running the example provides output like the following:

```
Split 768 rows into train=514 and test=254 rows
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 N

You can extend the implementation further.

- **Calculate Class Probabilities**: Update the examp belonging to each class as a ratio. This can be cal to one class, divided by the sum of the probabilities example an instance had the probability of 0.02 for instance belonging to class A is (0.02/(0.02+0.001
- **Log Probabilities**: The conditional probabilities for they are multiplied together they result in very sma (numbers too small to represent in Python). A common fix for this is to combine the log of the probabilities together. Research and implement this improvement.
- **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 this dataset.

Code

This section links to open source implementations of Naive Bayes in popular machine learning libraries. Review these if you are 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 docume
- Simple Naive Bayes in Weka: Weka implementatic

Books

You may have one or more books on applied machine l
chapters in common applied books on machine learning

- Applied Predictive Modeling, page 353
- Data Mining: Practical Machine Learning Tools and
- Machine Learning for Hackers, page 78
- An Introduction to Statistical Learning: with Applica
- Machine Learning: An Algorithmic Perspective, pa
- Machine Learning in Action, page 61 (Chapter 4)
- Machine Learning, page 177 (chapter 6)

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"

About Jason Brownlee

Dr. Jason Brownlee is a husband, proud fa
and a machine learning practitioner. He is
at applied machine learning. Learn more.

View all posts by Jason Brownlee →

‹ Lessons Learned from Building Machine Learning Systems

Better Naive Baye hm ›

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

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

REPLY

Statistical methods should be developed from scratch because of misunderstandings. Thank you.

Anurag December 14, 2014 at 1:11 pm #

REPLY

This is a wonderful article. Your blog is one of those blogs that I visit everyday. Thanks for sharing this stuff. I had a question about the programming language that should be used for building these algorithms from scratch. I know that Python is widely used because it's easy to write code by importing useful libraries that are already available. Nevertheless, I am a C++ guy. Although I am a beginner in practical ML, I have tried to write efficient codes before I started learning and implementing ML. Now I am aware of the complexities involved in coding if you're using C++: more coding is to be done than what is required in Python. Considering that, what language i

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 #

Hi Jason, found your website and read it in c
ML and what to do.
I did the 2 examples here and I think I will take a look
I have a personal project that I want to use ML, and I

One small note on this post, is on the "1. Handle data

Thank you so much, example is really good to show h

Jason Brownlee January 15, 2015 at 7:43 am

Thanks for the kind words Alcides.

Fixed the reference to the iris dataset.

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

Get Your Start in Machine Learning

Great Article. Learned a Lot. Thanks. Thanks.

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

REPLY

thank u

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

REPLY

You're welcome!

Roy March 7, 2015 at 2:53 pm #

Thanks for your nice article. I really appreci

malini March 17, 2015 at 7:19 pm #

hello sir, plz tell me how to compare the data

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 #

REPLY

As Splitting of dataset into testdata and traindata is done using a random function accuracy varies.

Sheepsy90 March 25, 2015 at 8:12 pm #

REPLY

Hey nice article – one question – why do you

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

REPLY

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 me

Thanks!

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

Sir

When I am running the same code in IDLE (python 2.
same code in eclipse. the error 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 #

REPLY

Great post, Jason.
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

- Ruud - November 26, 2016 at 2:23 am #

REPLY

Good point, thanks!

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 use the P(y) value in your calculateClassProbabilities() ?
If I understood the model correctly, everything is base
P(y|x1….xn) = P(x1…..xn|y) * P(y) / P(x1……xn)
P(x1……xn) will be a constant so we can get rid of it.
Your post explain very well how to calculate P(x1……
independent we then have
P(x1……xn|y) = P(x1|y) * …. P(xn|y))
How about p(y) ? I assume that we should calculate t
and then multiply it to probabilities[classValue] so that
P(y|x1…..xn) = frequency(classValue) * probabilities[c

Otherwise let' s assume that in a training set of 500 li
times 0 et 400 times 1. If we do not compute the freq
I misunderstand something ? Hopefully my post is cle
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 #

Get Your Start in Machine Learning

yeah，I have the same question too, maybe the P(y) is nessary ,but why the accuracy is not so low when P(y) is missing? is it proving that bayes model is powerful?

gd April 7, 2016 at 2:27 am #

hi,

I believe this is because P(y) = 1 as classes are already segregated before calculating P(x1…xn|Y).

Can experts comment on this please?

Babu May 23, 2016 at 7:32 am #

There is huge bug in this imp

First of all the implementation using G
Why is no one is replying even after 2

My concern is, there are so many mo
My lead read this article and now he t

At least the parameters are correct –

```
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)
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
```

```
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, 3.0147673799630748)
( 109.12 699.16 26.44 )==> (109.11976047904191, 26.481293163857107)
( 68.71 286.46 16.93 )==> (68.71257
( 19.74 228.74 15.12 )==> (19.74251
( 68.64 10763.69 103.75 )==> (68.64
( 30.71 58.05 7.62 )==> (30.7107784
( 0.42 0.09 0.29 )==> (0.4228592814
( 30.66 118.36 10.88 )==> (30.65868
Class: 1
( 4.76 12.44 3.53 )==> (4.7611111111
( 139.17 1064.54 32.63 )==> (139.17
( 69.27 525.24 22.92 )==> (69.27222
( 22.64 309.59 17.60 )==> (22.63888
( 101.13 20409.91 142.86 )==> (101.
( 34.99 57.18 7.56 )==> (34.9938888
( 0.54 0.14 0.37 )==> (0.5354444444
( 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 probabilty that you are calculating which is P(x|Ck) and then you make predictions, the result will be biased since you don't multiply by P(Ck) , 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 usage when `vector` is an `int` type object.

If I try the same commands one by one outside the function, the line of code with `vector[-1]` obviously throws a `TypeError: 'int' object has no attribute '__getitem__'`.

Then how is it working inside the function?

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

Sarah August 26, 2015 at 5:50 pm #

REPLY

Hello Jason! I just wanted to leave a messag
for a job in this field and it has helped me so much. K

Jason Brownlee August 26, 2015 at 6:56 pm

You're welcome! Thanks for leaving such

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

Hi Jason,

Very easy to follow your classifier. I try it and works well on your data, but is important to note that it works just on numerical databases, so maybe one have to transform your data from categorical to numerical format.

Another thing, when I transformed one database, sometimes the algorithm find division by zero error, although I avoided to use that number on features and classes.

Any suggestion Jason?

Thanks, Jaime

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 , pleas

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

REPLY

Thanks eduardo!

Thibroll September 29, 2015 at 9:11 pm #

REPLY

Hello.

This is all well explained, and depicts well the steps of machine learning. But the way you calculate your P(y|X) here is false, and may lead to unwanted error.

Here, in theory, using the Bayes law, we know that : P
P(y|X) with a given X, we can ignore P(X) and pick th

2 points remain inconsistent :
– First, you pick a gaussian distribution to estimate P(
the DENSITY of the function to the specific points X, y
actual probability.
– The second point is that you don't take into conside
model (with the correct probability calculation) may w
value of y (considering y is discret), or if you are lucky

Anyway, despite those mathematical issue, this is a g
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 #

REPLY

Hi Jason,
I have trying to get started with machine learning and

push towards that. Thank you for your efforts! 🙂

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 a
where I can head to get ideas for how to add this? Or
recommend? I've brought in all the attributes and spli
categorical so that I can work on them separately bef
the categorical in the same dictionary where the key i
each instance. I'm not sure how to go through the val
back up so that I have the attribute values along with
dictionary? Should I be going in another direction anc

Emmanuel Nuakoh November 19, 2015 at 6:36 ar

Thank you Jason, this tutorial is helping me with my implementation of NB algorithm for my PHD Dissertation. Very elaborate.

Anna January 14, 2016 at 2:32 am #

REPLY

Hi! thank you! Have you tried to do the same for the textual datasets, for example 20Newsgroups http://qwone.com/~jason/20Newsgroups/ ? Would appreciate some hints or ideas)

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}

Anyway, thanks for this tutorial, it was really useful, ch

Kumar Ramanathan February 12, 2016 at 12:20 p

Sincere gratitude for this most excellent site.
It is such an important exercise, to get concepts emb

Syed February 18, 2016 at 8:15 am #

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(X1|Ck)*p(X2|Ck)*…*p(Xm|Ck) and choosing Ck such that this value would be maximum.
However, when I looked in the Wikipedia, you are supposed to calculate p(X1|Ck)*p(X2|Ck)*…*p(Xm|Ck)*p(Ck).
I don't understand when you calculated p(Ck).
Would you tell me about it?

jessie November 24, 2016 at 1:21 am #

REPLY

Had the same thought, where's the prior

Babu May 23, 2016 at 7:36 am #

REPLY

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 doing things I think

Swapnil June 10, 2016 at 1:21 am #

Really nice tutorial. Can you post a detailed i
very helpful for us if you do so.

Thanks!!

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

Great idea Swapnil.

You may find this post useful:
http://machinelearningmastery.com/bagging-and-
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 the code to process it.

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

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

Thanks

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 #

REPLY

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

SAFA August 28, 2016 at 1:39 am #

REPLY

Hello,
there is some errors in "def splitDataset"
in machine learning algorithm , split a dataset into trainning and testing must be done without repetition (duplication) , so the index = random.randrange(len(copy)) generate duplicate data
for example " index = 0 192 1 2 0 14 34 56 1 ……
the spliting method must be done without duplication of data.

Krati Jain September 12, 2016 at 2:35 pm #

REPLY

This is a highly informative and detailed explained article. Although I think that this is suitable for Python 2.x versions for 3.x, we don't have 'iteritems' f

dict object. Secondly, format function is called on lot of print functions, which should have been on string in the print funciton but it has been somehow called on print function, which throws an error, can you please look into it.

upen September 16, 2016 at 5:01 pm #

REPLY

hey Jason
thanks for such a great tutorial im newbie to the concept and want to try naive bayes approach on movie-review on the review of a single movie that i have collected in a text file
can you please provide some hint on the topic how to load my file and perform positve or negative review on it

Jason Brownlee September 17, 2016 at 9:28 a

You might find some benefit in this tutoria
http://machinelearningmastery.com/machine-lear

Abhis September 20, 2016 at 3:00 am #

Would you please help me how i can implem
using their marks and feedback

Jason Brownlee September 20, 2016 at 8:35 am #

REPLY

I'm sorry, I am happy to answer your questions, but I cannot help you with your project. I just don't have the capacity.

Vinay October 13, 2016 at 2:18 pm #

REPLY

Hey Jason,

Thanks a lot for such a nice article, helped a lot in understanding the implementations,

i have a problem while running the script.
I get the below error

"

if (vector[-1] not in separated):

IndexError: list index out of range
”

can you please help me in getting it right?

Jason Brownlee October 14, 2016 at 8:58 am #

REPLY

Thanks Vinay.

Check that the data was loaded successfully. Perhaps there are empty lines or columns in your loaded data?

Viji October 20, 2016 at 8:57 pm #

Hi Jason,

Thank you for the wonderful article. U have used the
u please tell me what it specifies

jeni November 15, 2016 at 9:11 pm #

please send me a code in text classification u
classifies +ve,-ve or neutral

Jason Brownlee November 16, 2016 at 9:28 am #

REPLY

Hi jeni, sorry I don’t have such an example prepared.

MLNewbie November 28, 2016 at 1:21 pm #

REPLY

I am a newbie to ML and I found your website today. It is one of the greatest ML resources available on the Internet. I bookmarked it and thanks for everything Jason and I will visit your website everyday going forward.

Jason Brownlee November 29, 2016 at 8:47 am #

REPLY

Thanks, I’m glad you like it.

Anne January 7, 2017 at 6:58 pm #

REPLY

def predict(summaries, inputVector):
probabilities = calculateClassProbabilities(summaries, inputVector)
bestLabel, bestProb = None, -1
for classValue, probability in probabilities.iteritems():
if bestLabel is None or probability > bestProb:
bestProb = probability
bestLabel = classValue
return bestLabel

why is the prediction different for these
summaries = {'A' : [(1, 0.5)], 'B': [(20, 5.0)]} –p
summaries = {'0' : [(1, 0.5)], '1': [(20, 5.0)]} —
summaries = {0 : [(1, 0.5)], 1: [(20, 5.0)]} — p

ML704 January 18, 2017 at 6:16 pm #

Hi, can someone please explain the code sn

def separateByClass(dataset):
separated = {}
for i in range(len(dataset)):
vector = dataset[i]
if (vector[-1] not in separated):
separated[vector[-1]] = []
separated[vector[-1]].append(vector)
return separated

What do curly brackets mean in separated = {}?
vector[-1] ?

Massive thanks!

Jason Brownlee January 19, 2017 at 7:30 am #

REPLY

Curly brackets are a set or dictionary in Python, you can learn more about sets in Python here:
https://docs.python.org/3/tutorial/datastructures.html#sets

S February 27, 2017 at 8:47 am #

Get Your Start in Machine Learning

I am trying to create an Android app which works as follows:

1) On opening the App, the user types a data in a textbox & clicks on search

2) The app then searches about the entered data via internet and returns some answer (Using machine learning algorithms)

I have a dataset of around 17000 things.

Can you suggest the approach? Python/Java/etc…? Which technology to use for implementing machine learning algorithm & for connecting to dataset? How to include the dataset so that android app size is not increased?

Basically, I am trying to implement an app described in a research paper.

I can implement machine learning(ML) algorithms in I
want to develop an Android app in which the data ent
from a “data set (using ML)” and result is displayed ir
is that the data is of 40 MB & how to reflect the ML re
dataset is also available online. Shall I need a server?

Which python server should I use? I would also need
connect my Android app to live server and localhost s
app? What do you suggest? Is Anaconda software su

Jason Brownlee February 28, 2017 at 8:08 ar

Sorry I cannot make any good suggestions, I think you need to talk to some app engineers, not ML people.

Roy March 1, 2017 at 4:20 am #

REPLY

Hello Jason,

lines = csv.reader(open(filename, “rb”))
IOError: [Errno 2] No such file or directory: ‘pima-indians-diabetes.data.csv’

I have the csv file downloaded and its in the same folder as my code.

What should I do about this?

Jason Brownlee March 1, 2017 at 8:44 am #

REPLY

Hi Roy,

Confirm that the file name in your directory exactl

Confirm you are running from the command line and both the script and the data file are in the same directory and you are running from that directory.

If using Python 3, consider changing the 'rb' to 'rt' (text instead of binary).

Does that help?

Jordan March 1, 2017 at 8:40 pm #

REPLY

Hi Jason. Great Tutorial! But, why did you leave the P(Y) in calculateClassProbability() ? The prediction produces in my machine is fine… But some people up there have mentioned it too that what you actually calculate is probability density function. A

Ali March 7, 2017 at 6:21 am #

Hi Jason,
Can you please help me fixing below error, The split i

Split 769 rows into train=515 and test=254 rows
Traceback (most recent call last):
File "indian.py", line 100, in
main()
File "indian.py", line 95, in main
summaries = summarizeByClass(trainingSet)
File "indian.py", line 45, in summarizeByClass
separated = separateByClass(dataset)
File "indian.py", line 26, in separateByClass
if (vector[-1] not in separated):
IndexError: list index out of range

shankru Guggari March 13, 2017 at 9:52 pm #

REPLY

Class wise selection of training and testing data
For Example
In Iris Dataset : Species Column we have classes called Setosa, versicolor and virginica

I want to select 80% of data from each class values.

Advance thanks
Shankru
shankar286@gmail.Com

Jason Brownlee March 14, 2017 at 8:17 am #

REPLY

You can take a random sample or use a stratified sample to ensure the same mixture of classes in train and test sets.

Namrata March 19, 2017 at 5:33 pm #

REPLY

error in Naive Bayes code
IndexError:list index out of range

Leave a Reply

Name (required)

Email (will not be published) (required)

Website

SUBMIT COMMENT

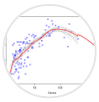
POPULAR


Time Series Prediction with LSTM Recurrent Neura
JULY 21, 2016

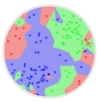

Develop Your First Neural Network in Python With K
MAY 24, 2016


Your First Machine Learning Project in Python Step
JUNE 10, 2016

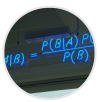

How to Run Your First Classifier in Weka
FEBRUARY 17, 2014


A Tour of Machine Learning Algorithms
NOVEMBER 25, 2013


Sequence Classification with LSTM Recurrent Neural Networks in Python with Keras
JULY 26, 2016


Tutorial To Implement k-Nearest Neighbors in Python From Scratch
SEPTEMBER 12, 2014


Regression Tutorial with the Keras Deep Learning Library in Python
JUNE 9, 2016


How To Implement Naive Bayes From Scratch in Pytho
DECEMBER 8, 2014

Machine Learning for Programmers
AUGUST 17, 2015