Chapter 19

Naive Bayes

Naive Bayes is a simple but surprisingly powerful algorithm for predictive modeling. In this chapter you will discover the Naive Bayes algorithm for classification. After reading this chapter, you will know:

- The representation used by naive Bayes that is actually stored when a model is written to a file.
- How a learned model can be used to make predictions.
- How you can learn a naive Bayes model from training data.
- How to best prepare your data for the naive Bayes algorithm.

Let's get started.

19.1 Quick Introduction to Bayes' Theorem

In machine learning we are often interested in selecting the best hypothesis (h) given data (d). In a classification problem, our hypothesis (h) may be the class to assign for a new data instance (d). One of the easiest ways of selecting the most probable hypothesis given the data that we have that we can use as our prior knowledge about the problem. Bayes' Theorem provides a way that we can calculate the probability of a hypothesis given our prior knowledge. Bayes' Theorem is stated as:

$$P(h|d) = \frac{P(d|h) \times P(h)}{P(d)} \tag{19.1}$$

Where:

- $P(h|d)$ is the probability of hypothesis h given the data d. This is called the posterior probability.
- $P(d|h)$ is the probability of data d given that the hypothesis h was true.
- $P(h)$ is the probability of hypothesis h being true (regardless of the data). This is called the prior probability of h.

- $P(d)$ is the probability of the data (regardless of the hypothesis).

You can see that we are interested in calculating the posterior probability of $P(h|d)$ from the prior probability $p(h)$ with $P(d)$ and $P(d|h)$. After calculating the posterior probability for a number of different hypotheses, you can select the hypothesis with the highest probability. This is the maximum probable hypothesis and may formally be called the maximum a posteriori (MAP) hypothesis. This can be written as:

$$\begin{aligned} MAP(h) &= max(P(h|d)) \\ MAP(h) &= max(\frac{P(d|h) \times P(h)}{P(d)}) \\ MAP(h) &= max(P(d|h) \times P(h)) \end{aligned} \tag{19.2}$$

The $P(d)$ is a normalizing term which allows us to calculate the probability. We can drop it when we are interested in the most probable hypothesis as it is constant and only used to normalize. Back to classification, if we have an even number of instances in each class in our training data, then the probability of each class (e.g. $P(h)$) will be the same value for each class (e.g. 0.5 for a 50-50 split). Again, this would be a constant term in our equation and we could drop it so that we end up with:

$$MAP(h) = max(P(d|h)) \tag{19.3}$$

This is a useful exercise, because when reading up further on Naive Bayes you may see all of these forms of the theorem.

19.2 Naive Bayes Classifier

Naive Bayes is a classification algorithm for binary (two-class) and multiclass classification problems. The technique is easiest to understand when described using binary or categorical input values. It is called naive Bayes or idiot Bayes because the calculation of the probabilities for each hypothesis are simplified to make their calculation tractable. Rather than attempting to calculate the values of each attribute value $P(d1, d2, d3|h)$, they are assumed to be conditionally independent given the target value and calculated as $P(d1|h) \times P(d2|h)$ and so on. This is a very strong assumption that is most unlikely in real data, i.e. that the attributes do not interact. Nevertheless, the approach performs surprisingly well on data where this assumption does not hold.

19.2.1 Representation Used By Naive Bayes Models

The representation for naive Bayes is probabilities. A list of probabilities is stored to file for a learned naive Bayes model. This includes:

- Class Probabilities: The probabilities of each class in the training dataset.
- Conditional Probabilities: The conditional probabilities of each input value given each class value.

19.2.2 Learn a Naive Bayes Model From Data

Learning a naive Bayes model from your training data is fast. Training is fast because only the probability of each class and the probability of each class given different input (x) values need to be calculated. No coefficients need to be fitted by optimization procedures.

Calculating Class Probabilities

The class probabilities are simply the frequency of instances that belong to each class divided by the total number of instances. For example in a binary classification the probability of an instance belonging to class 1 would be calculated as:

$$P(class = 1) = \frac{count(class = 1)}{count(class = 0) + count(class = 1)} \tag{19.4}$$

In the simplest case each class would have the probability of 0.5 or 50% for a binary classification problem with the same number of instances in each class.

Calculating Conditional Probabilities

The conditional probabilities are the frequency of each attribute value for a given class value divided by the frequency of instances with that class value. For example, if a `weather` attribute had the values `sunny` and `rainy` and the class attribute had the class values `go-out` and `stay-home`, then the conditional probabilities of each weather value for each class value could be calculated as:

$$\begin{aligned}
P(weather = sunny|class = \text{go-out}) &= \frac{count(weather = sunny \wedge \text{class=go-out})}{count(\text{class=go-out})}\\
P(weather = sunny|class = \text{stay-home}) &= \frac{count(weather = sunny \wedge class = \text{stay-home})}{count(class = \text{stay-home})}\\
P(weather = rainy|class = \text{go-out}) &= \frac{count(weather = rainy \wedge class = \text{go-out})}{count(class = \text{go-out})}\\
P(weather = rainy|class = \text{stay-home}) &= \frac{count(weather = rainy \wedge class = \text{stay-home})}{count(class = \text{stay-home})}
\end{aligned} \tag{19.5}$$

Where $\wedge$ means conjunction (and).

19.2.3 Make Predictions With a Naive Bayes Model

Given a naive Bayes model, you can make predictions for new data using Bayes theorem.

$$MAP(h) = max(P(d|h) \times P(h)) \tag{19.6}$$

Using our example above, if we had a new instance with the weather of `sunny`, we can calculate:

$$\begin{aligned}
\text{go-out} &= P(weather = sunny|class = \text{go-out}) \times P(class = \text{go-out})\\
\text{stay-home} &= P(weather = sunny|class = \text{stay-home}) \times P(class = \text{stay-home})
\end{aligned} \tag{19.7}$$

We can choose the class that has the largest calculated value. We can turn these values into probabilities by normalizing them as follows:

$$\begin{aligned} P(\text{go-out}|weather = sunny) &= \frac{\text{go-out}}{\text{go-out} + \text{stay-home}} \\ P(\text{stay-home}|weather = sunny) &= \frac{\text{stay-home}}{\text{go-out} + \text{stay-home}} \end{aligned} \tag{19.8}$$

If we had more input variables we could extend the above example. For example, pretend we have a `car` attribute with the values `working` and `broken`. We can multiply this probability into the equation. For example below is the calculation for the `go-out` class label with the addition of the car input variable set to `working`:

$$\begin{aligned} \text{go-out} =& P(weather = sunny|class = \text{go-out})\times \\ & P(car = working|class = \text{go-out})\times \\ & P(class = \text{go-out}) \end{aligned} \tag{19.9}$$

19.3 Gaussian Naive Bayes

Naive Bayes can be extended to real-valued attributes, most commonly by assuming a Gaussian distribution. This extension of naive Bayes is called Gaussian Naive Bayes. Other functions can be used to estimate the distribution of the data, but the Gaussian (or Normal distribution) is the easiest to work with because you only need to estimate the mean and the standard deviation from your training data.

19.3.1 Representation for Gaussian Naive Bayes

Above, we calculated the probabilities for input values for each class using a frequency. With real-valued inputs, we can calculate the mean and standard deviation of input values (x) for each class to summarize the distribution. This means that in addition to the probabilities for each class, we must also store the mean and standard deviations for each input variable for each class.

19.3.2 Learn a Gaussian Naive Bayes Model From Data

This is as simple as calculating the mean and standard deviation values of each input variable (x) for each class value.

$$mean(x) = \frac{1}{n} \times \sum_{i=1}^{n} x_i \tag{19.10}$$

Where n is the number of instances and x are the values for an input variable in your training data. We can calculate the standard deviation using the following equation:

$$StandardDeviation(x) = \sqrt{\frac{1}{n} \times \sum_{i=1}^{n} (x_i - mean(x))^2} \tag{19.11}$$

This is the square root of the average squared difference of each value of x from the mean value of x, where n is the number of instances, x_i is a specific value of the x variable for the i'th instance and $mean(x)$ is described above.

19.3.3 Make Predictions With a Gaussian Naive Bayes Model

Probabilities of new x values are calculated using the Gaussian Probability Density Function (PDF). When making predictions these parameters can be plugged into the Gaussian PDF with a new input for the variable, and in return the Gaussian PDF will provide an estimate of the probability of that new input value for that class.

$$pdf(x, mean, sd) = \frac{1}{\sqrt{2 \times \pi} \times sd} \times e^{-(\frac{(x-mean)^2}{2 \times sd^2})} \tag{19.12}$$

Where $pdf(x)$ is the Gaussian PDF, $mean$ and sd are the mean and standard deviation calculated above, π is the numerical constant PI, e is the numerical constant Euler's number raised to power and x is the input value for the input variable. We can then plug in the probabilities into the equation above to make predictions with real-valued inputs. For example, adapting one of the above calculations with numerical values for weather and car:

$$\begin{aligned} \text{go-out} =& P(pdf(weather)|class = \text{go-out}) \times \\ & P(pdf(car)|class = \text{go-out}) \times \\ & P(class = \text{go-out}) \end{aligned} \tag{19.13}$$

19.4 Preparing Data For Naive Bayes

This section provides some tips for preparing your data for Naive Bayes.

- **Categorical Inputs**: Naive Bayes assumes label attributes such as binary, categorical or nominal.
- **Gaussian Inputs**: If the input variables are real-valued, a Gaussian distribution is assumed. In which case the algorithm will perform better if the univariate distributions of your data are Gaussian or near-Gaussian. This may require removing outliers (e.g. values that are more than 3 or 4 standard deviations from the mean).
- **Classification Problems**: Naive Bayes is a classification algorithm suitable for binary and multiclass classification.
- **Log Probabilities**: The calculation of the likelihood of different class values involves multiplying a lot of small numbers together. This can lead to an underflow of numerical precision. As such it is good practice to use a log transform of the probabilities to avoid this underflow.
- **Kernel Functions**: Rather than assuming a Gaussian distribution for numerical input values, more complex distributions can be used such as a variety of kernel density functions.
- **Update Probabilities**: When new data becomes available, you can simply update the probabilities of your model. This can be helpful if the data changes frequently.

19.5 Summary

In this chapter you discovered the Naive Bayes algorithm for classification. You learned about:

- The Bayes Theorem and how to calculate it in practice.
- Naive Bayes algorithm including representation, making predictions and learning the model.
- The adaptation of Naive Bayes for real-valued input data called Gaussian Naive Bayes.
- How to prepare data for Naive Bayes.

You now know about the Naive Bayes algorithm for classification. In the next chapter you will discover how to implement Naive Bayes from scratch for categorical variables.

Chapter 20

Naive Bayes Tutorial

Naive Bayes is a very simple classification algorithm that makes some strong assumptions about the independence of each input variable. Nevertheless, it has been shown to be effective in a large number of problem domains. In this chapter you will discover the Naive Bayes algorithm for categorical data. After reading this chapter you will know.

- How to work with categorical data for Naive Bayes.
- How to prepare the class and conditional probabilities for a Naive Bayes model.
- How to use a learned Naive Bayes model to make predictions.

Let's get started.

20.1 Tutorial Dataset

The dataset describes two categorical input variables and a class variable that has two outputs.

```
Weather   Car       Class
sunny     working   go-out
rainy     broken    go-out
sunny     working   go-out
sunny     working   go-out
sunny     working   go-out
rainy     broken    stay-home
rainy     broken    stay-home
sunny     working   stay-home
sunny     broken    stay-home
rainy     broken    stay-home
```

Listing 20.1: Naive Bayes Tutorial Data Set.

We can convert this into numbers. Each input has only two values and the output class variable has two values. We can convert each variable to binary as follows:

- Weather: sunny = 1, rainy = 0
- Car: working = 1, broken = 0
- Class: go-out = 1, stay-home = 0

Therefore, we can restate the dataset as:

```
Weather   Car   Class
1         1     1
0         0     1
1         1     1
1         1     1
1         1     1
0         0     0
0         0     0
1         1     0
1         0     0
0         0     0
```

Listing 20.2: Simplified Naive Bayes Tutorial Data Set.

This can make the data easier to work with in a spreadsheet or code if you are following along.

20.2 Learn a Naive Bayes Model

There are two types of quantities that need to be calculated from the dataset for the naive Bayes model:

- Class Probabilities.
- Conditional Probabilities.

Let's start with the class probabilities.

20.2.1 Calculate the Class Probabilities

The dataset is a two class problem and we already know the probability of each class because we contrived the dataset. Nevertheless, we can calculate the class probabilities for classes 0 and 1 as follows:

$$\begin{aligned} P(class = 1) &= \frac{count(class = 1)}{count(class = 0) + count(class = 1)} \\ P(class = 0) &= \frac{count(class = 0)}{count(class = 0) + count(class = 1)} \end{aligned} \tag{20.1}$$

or

$$\begin{aligned} P(class = 1) &= \frac{5}{5 + 5} \\ P(class = 0) &= \frac{5}{5 + 5} \end{aligned} \tag{20.2}$$

This works out to be a probability of 0.5 for any given data instance belonging to class 0 or class 1.

20.2.2 Calculate the Conditional Probabilities

The conditional probabilities are the probability of each input value given each class value. The conditional probabilities for the dataset can be calculated as follows:

Weather Input Variable

$$\begin{aligned}
P(weather = sunny|class = \text{go-out}) &= \frac{count(weather = sunny \wedge class = \text{go-out})}{count(class = \text{go-out})} \\
P(weather = rainy|class = \text{go-out}) &= \frac{count(weather = rainy \wedge class = \text{go-out})}{count(class = \text{go-out})} \\
P(weather = sunny|class = \text{stay-home}) &= \frac{count(weather = sunny \wedge class = \text{stay-home})}{count(class = \text{stay-home})} \\
P(weather = rainy|class = \text{stay-home}) &= \frac{count(weather = rainy \wedge class = \text{stay-home})}{count(class = \text{stay-home})}
\end{aligned} \tag{20.3}$$

Remember that the $\wedge$ symbol is just a shorthand for conjunction (AND). Plugging in the numbers we get:

$$\begin{aligned}
P(weather = sunny|class = \text{go-out}) = 0.8 \\
P(weather = rainy|class = \text{go-out}) = 0.2 \\
P(weather = sunny|class = \text{stay-home}) = 0.4 \\
P(weather = rainy|class = \text{stay-home}) = 0.6
\end{aligned} \tag{20.4}$$

Car Input Variable

$$\begin{aligned}
P(car = working|class = \text{go-out}) &= \frac{count(car = working \wedge class = \text{go-out})}{count(class = \text{go-out})} \\
P(car = broken|class = \text{go-out}) &= \frac{count(car = broken \wedge class = \text{go-out})}{count(class = \text{go-out})} \\
P(car = working|class = \text{stay-home}) &= \frac{count(car = working \wedge class = \text{stay-home})}{count(class = \text{stay-home})} \\
P(car = broken|class = \text{stay-home}) &= \frac{count(car = broken \wedge class = \text{stay-home})}{count(class = \text{stay-home})}
\end{aligned} \tag{20.5}$$

Plugging in the numbers we get:

$$\begin{aligned}
P(car = working|class = \text{go-out}) = 0.8 \\
P(car = broken|class = \text{go-out}) = 0.2 \\
P(car = working|class = \text{stay-home}) = 0.2 \\
P(car = broken|class = \text{stay-home}) = 0.8
\end{aligned} \tag{20.6}$$

We now have every thing we need to make predictions using the Naive Bayes model.

20.3 Make Predictions with Naive Bayes

We can make predictions using Bayes Theorem, defined and explained in the previous chapter.

$$P(h|d) = \frac{P(d|h) \times P(h)}{P(d)} \tag{20.7}$$

In fact, we don't need a probability to predict the most likely class for a new data instance. We only need the numerator and the class that gives the largest response, which will be the predicted output.

$$MAP(h) = max(P(d|h) \times P(h)) \tag{20.8}$$

Let's take the first record from our dataset and use our learned model to predict which class we think it belongs. First instance: $weather = sunny, car = working$.

We plug the probabilities for our model in for both classes and calculate the response. Starting with the response for the output `go-out`. We multiply the conditional probabilities together and multiply it by the probability of any instance belonging to the class.

$$\begin{aligned}\text{go-out} =& P(weather = sunny|class = \text{go-out})\times\\ & P(car = working|class = \text{go-out})\times\\ & P(class = \text{go-out})\end{aligned} \tag{20.9}$$

or

$$\begin{aligned}\text{go-out} &= 0.8 \times 0.8 \times 0.5\\ \text{go-out} &= 0.32\end{aligned} \tag{20.10}$$

We can perform the same calculation for the stay-home case:

$$\begin{aligned}\text{stay-home} =& P(weather = sunny|class = \text{stay-home})\times\\ & P(car = working|class = \text{stay-home})\times\\ & P(class = \text{stay-home})\end{aligned} \tag{20.11}$$

or

$$\begin{aligned}\text{stay-home} &= 0.4 \times 0.2 \times 0.5\\ \text{stay-home} &= 0.04\end{aligned} \tag{20.12}$$

We can see that 0.32 is greater than 0.04, therefore we predict `go-out` for this instance, which is correct. We can repeat this operation for the entire dataset, as follows:

```
Weather   Car       Class      go-out? stay-home? Prediction
sunny     working   go-out     0.32    0.04       go-out
rainy     broken    go-out     0.02    0.24       stay-home
sunny     working   go-out     0.32    0.04       go-out
sunny     working   go-out     0.32    0.04       go-out
sunny     working   go-out     0.32    0.04       go-out
rainy     broken    stay-home  0.02    0.24       stay-home
rainy     broken    stay-home  0.02    0.24       stay-home
sunny     working   stay-home  0.32    0.04       go-out
sunny     broken    stay-home  0.08    0.16       stay-home
rainy     broken    stay-home  0.02    0.24       stay-home
```

Listing 20.3: Naive Bayes Predictions for the Dataset.

If we tally up the predictions compared to the actual class values, we get an accuracy of 80%, which is excellent given that there are conflicting examples in the dataset.

20.4 Summary

In this chapter you discovered exactly how to implement Naive Bayes from scratch. You learned:

- How to work with categorical data with Naive Bayes.
- How to calculate class probabilities from training data.
- How to calculate conditional probabilities from training data.
- How to use a learned Naive Bayes model to make predictions on new data.

You now know how to implement Naive Bayes from scratch for categorical data. In the next chapter you will discover how to can implement Naive Bayes from scratch for real-valued data.

Chapter 21

Gaussian Naive Bayes Tutorial

Naive Bayes is a simple model that uses probabilities calculated from your training data to make predictions on new data. The basic Naive Bayes algorithm assumes categorical data. A simple extension for real-valued data is called Gaussian Naive Bayes. In this chapter you will discover how to implement Gaussian Naive Bayes from scratch. After reading this chapter you will know:

- The Gaussian Probability Density Function and how to calculate the probability of real values.
- How to learn the properties for a Gaussian Naive Bayes model from your training data.
- How to use a learned Gaussian Naive Bayes model to make predictions on new data.

Let's get started.

21.1 Tutorial Dataset

A simple dataset was contrived for our purposes. It is comprised of two input variables $X1$ and $X2$ and one output variable Y. The input variables are drawn from a Gaussian distribution, which is one assumption made by Gaussian Naive Bayes. The class variable has two values, 0 and 1, therefore the problem is a binary classification problem.

Data from class 0 was drawn randomly from a Gaussian distribution with a standard deviation of 1.0 for $X1$ and $X2$. Data from class 1 was drawn randomly from a Gaussian distribution with a mean of 7.5 for $X1$ and 2.5 for $X2$. This means that the classes are nicely separated if we plot the input data on a scatter plot. The raw dataset is listed below:

```
X1            X2            Y
3.393533211   2.331273381   0
3.110073483   1.781539638   0
1.343808831   3.368360954   0
3.582294042   4.67917911    0
2.280362439   2.866990263   0
7.423436942   4.696522875   1
5.745051997   3.533989803   1
9.172168622   2.511101045   1
7.792783481   3.424088941   1
7.939820817   0.791637231   1
```

Listing 21.1: Gaussian Naive Bayes Tutorial Data Set.

You can clearly see the separation of the classes in the plot below. This will make the data relatively easy to work with as we implement and test a Gaussian Naive Bayes model.

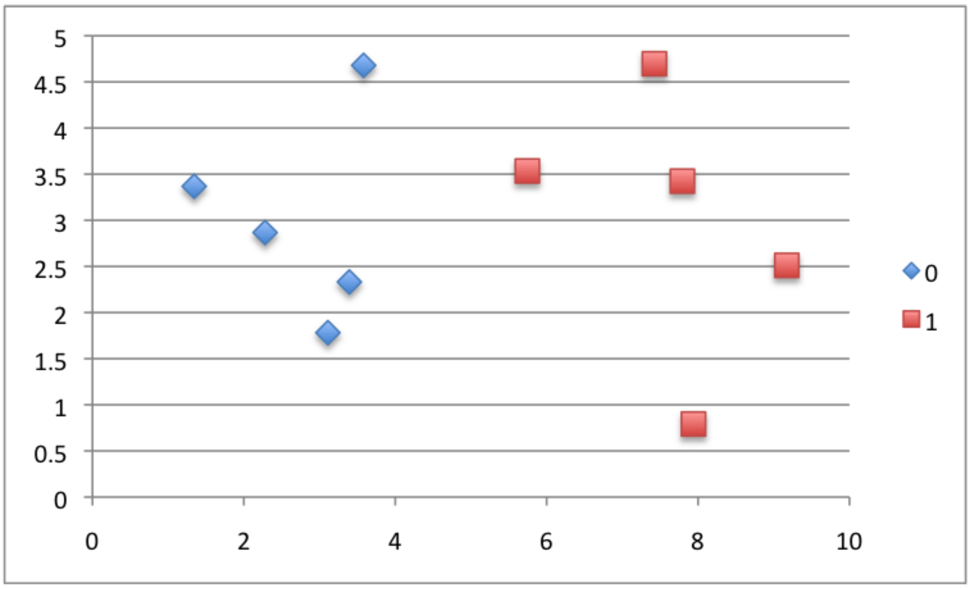

Figure 21.1: Gaussian Naive Bayes Tutorial Dataset.

21.2 Gaussian Probability Density Function

The Gaussian Probability Density Function (PDF) will calculate the probability of a value given the mean and standard deviation of the distribution from which it came. The Gaussian PDF is calculated as follows:

$$pdf(x, mean, sd) = \frac{1}{\sqrt{2 \times \pi} \times sd} \times e^{-(\frac{(x-mean)^2}{2 \times sd^2})} \tag{21.1}$$

Where $pdf(x)$ is the Gaussian PDF, $mean$ and sd are the mean and standard deviation calculated above, π is the numerical constant PI, e is Euler's number raised to a power and x is the input value for the input variable. Let's look at an example. Let's assume we have real values drawn from a population that has a mean of 0 and a standard deviation of 1. Using the Gaussian PDF we can estimate the likelihood of range of values.

```
X       PDF(x)
-5      1.48672E-06
-4      0.00013383
-3      0.004431848
-2      0.053990967
-1      0.241970725
0       0.39894228
1       0.241970725
2       0.053990967
3       0.004431848
```

```
4       0.00013383
5       1.48672E-06
```

Listing 21.2: Test of the Gaussian Probability Density Function.

You can see that the mean has the highest probability of nearly 0.4 (40%). You can also see values that are far away from the mean like -5 and +5 (5 standard deviations from the mean) have a very low probability. Below is a plot of the probabilities values.

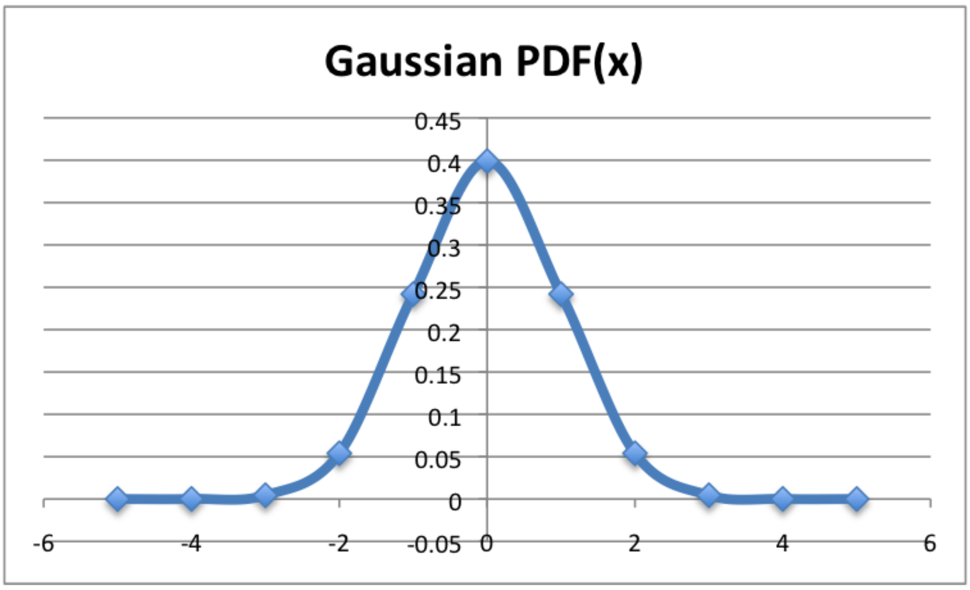

Figure 21.2: Gaussian Probability Density Function.

This function is really useful for Naive Bayes. We can assume that the input variables are each drawn from a Gaussian distribution. By calculating the mean and standard deviation of each input variable from the training data, we can use the Gaussian PDF to estimate the likelihood of each value for each class. We will see how this is used to calculate conditional probabilities in the next section.

21.3 Learn a Gaussian Naive Bayes Model

There are two types of probabilities that we need to summarize from our training data for the naive Bayes model:

- Class Probabilities.
- Conditional Probabilities.

21.3.1 Class Probabilities

The dataset is a two class problem and we already know the probability of each class because we contrived the dataset. Nevertheless, we can calculate the class probabilities for classes 0 and

1 as follows:

$$
\begin{aligned}
P(Y=1) &= \frac{count(Y=1)}{count(Y=0)+count(Y=1)} \\
P(Y=0) &= \frac{count(Y=0)}{count(Y=0)+count(Y=1)}
\end{aligned}
\tag{21.2}
$$

or

$$
\begin{aligned}
P(Y=1) &= \frac{5}{5+5} \\
P(Y=0) &= \frac{5}{5+5}
\end{aligned}
\tag{21.3}
$$

This works out to be a probability of 0.5 (50%) for any given data instance belonging to class 0 or class 1.

21.3.2 Conditional Probabilities

The conditional probabilities are the probabilities of each input value given each class value. The conditional probabilities that need to be collected from the training data are as follows:

$$
\begin{aligned}
&P(X1|Y=0) \\
&P(X1|Y=1) \\
&P(X2|Y=0) \\
&P(X2|Y=1)
\end{aligned}
\tag{21.4}
$$

The $X1$ and $X2$ input variables are real values. As such we will model them as having being drawn from a Gaussian distribution. This will allow us to estimate the probability of a given value using the Gaussian PDF described above. The Gaussian PDF requires two parameters in addition to the value for which the probability is being estimated: the mean and the standard deviation. Therefore we must estimate the mean and the standard deviation for each group of conditional probabilities that we require. We can estimate these directly from the dataset. The results are summarized below.

```
      P(X1|Y=0)    P(X1|Y=1)    P(X2|Y=0)    P(X2|Y=1)
Mean  2.742014401  7.614652372  3.005468669  2.991467979
Stdev 0.926568329  1.234432155  1.107329589  1.454193138
```

Listing 21.3: Summary of population statistics by class.

We now have enough information to make predictions for the training data or even a new dataset.

21.4 Make Prediction with Gaussian Naive Bayes

We can make predictions using Bayes Theorem, introduced and explained in a previous chapter. We don't need a probability to predict the most likely class for a new data instance. We only need the numerator and the class that gives the largest response is the predicted response.

$$MAP(h) = max(P(d|h) \times P(h)) \tag{21.5}$$

Let's take the first record from our dataset and use our learned model to predict which class we think it belongs. Instance: $X1 = 3.393533211$, $X2 = 2.331273381$, $Y = 0$. We can plug the probabilities for our model in for both classes and calculate the response. Starting with the response for the output class 0. We multiply the conditional probabilities together and multiply it by the probability of any instance belonging to the class.

$$\begin{aligned}
\text{class } 0 &= P(\mathit{pdf}(X1)|\mathit{class} = 0) \times P(\mathit{pdf}(X2)|\mathit{class} = 0) \times P(\mathit{class} = 0) \\
\text{class } 0 &= 0.336255919 \times 0.299321284 \times 0.5 \\
\text{class } 0 &= 0.050324277
\end{aligned} \tag{21.6}$$

We can perform the same calculation for class 1:

$$\begin{aligned}
\text{class } 1 &= P(\mathit{pdf}(X1)|\mathit{class} = 1) \times P(\mathit{pdf}(X2)|\mathit{class} = 1) \times P(\mathit{class} = 1) \\
\text{class } 1 &= 0.000934051 \times 0.247475201 \times 0.5 \\
\text{class } 1 &= 0.000115577
\end{aligned} \tag{21.7}$$

We can see that 0.050324277 is greater than 0.000115577, therefore we predict the class as 0 for this instance, which is correct. Repeating this process for all instances in the dataset we get the following outcomes for class 0 and class 1. By selecting the class with the highest output we make accurate predictions for all instances in the training dataset.

```
X1            X2            Output Y=0    Output Y=1    Prediction
3.393533211   2.331273381   0.050324277   0.000115577   0
3.110073483   1.781539638   0.038912299   4.02571E-05   0
1.343808831   3.368360954   0.023541958   1.06771E-07   0
3.582294042   4.67917911    0.016404354   0.000108931   0
2.280362439   2.866990263   0.067972749   3.89292E-06   0
7.423436942   4.696522875   6.91958E-08   0.022027299   1
5.745051997   3.533989803   0.000362402   0.013133374   1
9.172168622   2.511101045   2.4463E-12    0.018937265   1
7.792783481   3.424088941   2.54861E-08   0.04197207    1
7.939820817   0.791637231   1.5426E-09    0.013636753   1
```

Listing 21.4: Predictions using Gaussian Naive Bayes.

The prediction accuracy is 100%, as was expected given the clear separation of the classes.

21.5 Summary

In this chapter you discovered how to implement the Gaussian Naive Bayes classifier from scratch. You learned about:

- The Gaussian Probability Density Function for estimating the probability of any given real value.
- How to estimate the probabilities required by the Naive Bayes model from a training dataset.
- How to use the learned Naive Bayes model to make predictions.

You now know how to implement Gaussian Naive Bayes from scratch for real-valued data. In the next chapter you will discover the k-Nearest Neighbors algorithm for classification and regression.