CIS/STA 3920, 7-26-21 Prof. L. Tatum

**Lecture Notes 6: Naïve Bayes Classification**

Section Topic

1 Bayes Rule

2 Bayesian Statistics

**3 Anderson’s Iris Data**

**4 Introduction to the Naive Bayes Classification Algorithm**

**5 NB Implementation using R Package e1071**

**6 Cross-Validation using R Packages “klaR” and “caret”**

7 Two Kinds of Independence

8 Sources

Exercises

**Section 1: Bayes Rule**

Suppose the rate of fetal spina bifida is 1/1000. A blood test (on the mother) can detect this disease early in pregnancy. The test has a false positive rate of 3% and a false negative rate of 1%. A woman decides to have the test and it comes back positive. What is the probability that spina bifida is present?

Consider a population of 100,000 pregnancies. We can say that “on average” there will be 100 cases of spina bifida in that population. Of those, on average, given a correct positive rate is 99% (100%-1%) and 100 actual positives, we get 99 correct positive test results because

0.99 x 100 = 99 correct positive test results

will result, so that only one of the 100 disease cases will be missed on, average.

There remain the

99,900 = 100,000 - 100

cases without spina bifida. Of those, on average 3% of those will test positive even though the disease is not present, so we will get on average 2997 incorrect positive test results:

0.03 x 99,900 = 2997 incorrect positive test results

so, on average there will be 2997 incorrect positive tests.

Thus, the total number of positive tests will be on average

99 + 2997 = 3096.

Of those, 99 are actual cases of spina bifida. Therefore, the probability that the disease is present given a positive test result is

99 / 3096 = 0.03198 = 3.198%.

This is approximately 1/33, a surprising and disturbing result.[[1]](#footnote-1)

What we have done, above, is an example of Bayes Rule. This rule is named for the Reverend Bayes (1701-1761), a skilled logician and mathematician.

We started with the conditional probability of a negative test given the disease was present – the false negative rate – and the conditional probability of a positive test given the disease was not present – the false positive rate. Along with the postulated disease rate of 1/1000, we were then able to use a very careful application of common sense to compute the conditional probability that the disease was present given a positive test result.

If we do not wish to use common sense, we can use Bayes Rule instead. This formalism states:

P(A|B) = P(B|A)P(A) / [P(B|A)P(A) + P(B|A’)P(A’)],

where A’ is “not A”, or the complement of A.

In this case, A denotes the event that the disease is present, and B denotes a positive test. We have,

P(A) = 1/1000 = 0.001

P(A’) = 0.999

P(B|A) = probability of a positive test given disease is present = 0.99 = 1 – 0.01

P(B|A’) = probability of a positive test given disease is not present = 0.03,

so,

P(A|B) = 0.99\*0.001 / [0.99\*0.001 + 0.03\* 0.999]

= 0.00099 / [0.00099 + 0.02997]

= 0.00099 / 0.03096

= 0.03198

In my experience, if I find agreement when using both methods, then the chances I have done the computation correctly are very high!

It is not obvious, but Bayes Rule can also be understood in this way:

P(A|B) = P(B and A) / P(B),

where

P(B and A) = P(B|A)P(A)

and

P(B) = P(A and B) + P(A’ and B)

= P(B|A)P(A) + P(B|A’)P(A’),

which is not obvious!

It is vital to understand that Bayes Rule is not limited to such simple cases. We could have far more possible outcomes to deal with, and we can even consider cases in which the random variables are continuous.

**Section 2: Bayesian Statistics**

The problem that was actually studied by Thomas Bayes is called “inverse probability,” which we would today call “statistical inference.” What Bayes realized was that it was possible to use conditional probability arguments to *estimate parameters*. Recall that in the Normal distribution, μ and σ are parameters. Likewise, for a binomial distribution, the parameters are n and π, where π is the probability of a success.

However, this method requires some courage on the part of the user. It requires that we view the parameter to be estimated as a random variable! For example, consider a real coin. We do not know if it is a fair coin or not. That is, letting π denote the probability a toss will result in heads, we do not know the value of π. To use Bayes rule, here, we will need to put forward a probability distribution for π. This is called the *prior distribution*. Then, we toss the coin *n* times and count the number of heads. Making the reasonable assumption that the number of heads follows a binomial distribution conditional on π, we can then employ Bayes Rule to get an updated distribution on π, called the *posterior distribution*.

Let us propose a prior distribution for the probability of heads as follows:

Prior Probability Distribution for π

Outcome Probability

4/10 ¼

5/10 ½

6/10 ¼

To make this situation concrete, let us translate the problem from a coin to a question of the number of black marbles out of 10 marbles in a bag. That is, you are given a bag containing 10 marbles, some black and some white. In fact, you are told that there are either 4, 5, or 6 black marbles, but you don’t know which it is. As a prior distribution, suppose you personally feel the chances are ¼ that there are 4 black marbles, the chances are ½ that there are 5 black marbles, and that the chances are ¼ that there are 6 black marbles.

Now, note that if there are 4 black marbles, then the probability of drawing a black marble is 4/10 = 0.4. Recall that a few paragraphs back, we let π denote the prior probability of getting heads. Here, let π denote the prior probability of getting a black marble. According to our prior probabilities, the probability is ¼ that π = 4/10, or

P(π=4/10) = ¼.

Study the above sentences carefully – they contain probability at two different levels: **one is the probability of drawing a black marble; the other is the probability that the bag contains 4 black marbles!** [The key to making progress in a field like that this is to come back again and again to wrestle with the above paragraphs until you get the meaning of that cryptic statement “P(π=4/10) = ¼.” Do not sit back and figure that understanding this will be done by somebody else!]

In summary, our prior distribution for π is:

P(π=0.4) = ¼, P(π=0.5) = ½, and P(π=0.6)=1/4.

Next, we want to get some data with which to *update* our prior distribution. Let us do this by drawing five marbles (with replacement). That means that each time we draw a marble we put it back and shake the bag before drawing, again. From a practical point of view, sampling with replacement is inefficient, but it results in much simpler probability problem than sampling without replacement. Based on the number of black and white marbles in the sample, we will *update* your prior distribution using Bayes Rule! Sampling with replacement in this setting, the number of black marbles in the sample “conditional on π” follows a *binomial probability distribution*, one of the simplest of useful distributions.

Suppose we find a total of 3 black and 2 white marbles. How do we digest that information? The key step is to evaluate the probability of getting 3 black marbles out of 5 *for each of our three potential values* of π. If π=0.4, what is the probability of getting 3 black marbles? We can use the binomial probability distribution to find out. (See any online source of information on the Binomial distribution or any intro stat text.)

P(k=3|π=0.4) = 5C3 0.43 0.62 = 0.2304.

That term is the P(B|A) that occurs in the numerator of the right-hand side of Bayes Rule. We need to multiply it by P(π=.4), which is given by the prior distribution as ¼. The probability of getting 3 black marbles in 5 and having π=0.4 is computed as

P(k=3 & π=0.4) = P(k=3|π=0.4) P(π=.4) = (0.2304) ( ¼ ) = 0.0576.

We now need the denominator. That is a little more complicated than in the spina bifida case because there are three possible values of π that were postulated by the prior distribution.

Here, our denominator will look like this:

P(k=3|π=0.4) P(π=.4) + P(k=3|π=0.5) P(π=.5) + P(k=3|π=0.6) P(π=.6)

= (5C3 0.43 0.62) ( ¼ ) + 5C3 0.53 0.52 ( ½ ) + 5C3 0.63 0.42 (¼)

= 0.0576 + 0.15625 + 0.0864.

Putting all this together, we have an updated, posterior probability of

P(π=0.4|k=3)

= P(k=3|π=0.4)P(π=.4) / [P(k=3|π=0.4)P(π=.4)+P(k=3|π=0.5)P(π=.5)+P(k=3|π=0.6)P(π=.6)]

= 0.0576 / [0.0576 + 0.15625 + 0.0864]

= 0.0576 / 0.30025

= .1918.

Note that this posterior probability that π=0.4 has been reduced from the prior probability of ¼. In turn, this means that greater weight will be given to other values of π by the posterior distribution.

We will now compute the posterior probability for π=0.5. This is relatively simple because the denominator remains the same. We have,

P(π=0.5|k=3)\*P(k=3) = P(π=0.5 and k=3)

Or,

P(π=0.5|k=3) = P(π=0.5 and k=3) / P(k=3)

P(k=3) = P(π=0.4 and k=3) + P(π=0.5 and k=3) + P(π=0.6 and k=3).

**Section 3: Anderson’s Iris Data**

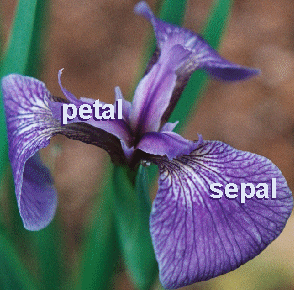
We will be working with a famous data set, “Anderson’s Iris data,” which is found in R under the name of *iris*. The Iris data contains lengths and widths of the petals and sepals of 150 specimens of iris flowers. In the R data set, those four measurements are labeled Sepal.Length, Sepal.Width, Petal.Length, and Petal.Width. There are three species present: Versicolor, Virginica, and Setosa.

The Iris data seems to be used in every multivariate statistics book, perhaps because all reasonable classification methods correctly classify almost all of the 150 flowers. Despite the small size of the iris data, it is also used in almost all texts that contain the phrase “data mining” in the title.

When I studied multivariate statistics, I was not particularly excited by the Iris data set – what did I care? In the back of my mind, I think I believed I could easily tell which specimens belonged to which species by visual inspection. With the aid of photos on the Internet, I now see I was presumptuous! Here are pictures of specimens of each species: this is not a trivial classification problem if one relied only on photographs. However, as we will see, the species are fairly easy to distinguish by the relative *sizes* of the flower sepals and flower petals.



The sepal is the larger portion of the flower, while the petal is smaller, whitish and lies on top of the sepal.



Running the command

>dim(iris)

we find the iris data set contains 150 rows and 5 columns. As it happens, there are 50 of each type of iris flower. I wonder how that was arranged and why.

The following matrix scatterplot shows the pair-wise plots of the four numeric variables and the species type in color. This showcases the remarkable graphical power of R.

> pairs(iris[,1:4], main = "Iris Data (red=setosa,green=versicolor,blue=virginica)",

pch = 21, bg = c("red", "green3", "blue")[unclass(iris$Species)])



There is a lot to ponder in that plot and in the commands that produced it! In particular, notice how in all the pair-wise plots there is a positive slope in the blue points and in green points. This means there is strong, positive correlation between each pair of the four x-variables. Thus, the assumption by NB that the x-variables are “independent” is violated in this application.

Do not be dismayed, though. In practice, we often “violate” the assumptions of the theoretical model we are employing. The question is whether or not the violation is “substantive” in a given setting. And is there some other model that will work better? **Section 4: An Introduction to the Naive Bayes Classification Algorithm**

The naïve Bayes method (NB) is “naïve” in that it assumes the x-variables are independent of each other. For example, NB assumes that whisker length and weight were not correlated in the dog/cat data. That is surprising assumption because in real life it is commonly found that the x-variables are not independent. Note, for example, the high level of correlation between the Lag1 and Lag2 variables for our stock **range** data.

It is surprising, then, to read that the naïve Bayes method often performs well even when the x-variables are clearly correlated. How the method overcomes that drawback is unclear – at least I have not yet found an explanation that has been demonstrated to be valid. It would be interesting to try to create examples where the ill-effects of correlation (non-independence) were revealed; and it would be interesting to try to create examples in which correlation was not a problem and the reasons why could be understood!

Suppose that our y-variable contains K different classes. For the Iris data, K=3. A Bayesian classification method begins with a set of *prior probabilities* that the specimen in any row is of species #1, #2, or #3. In the naïve Bayes method, the prior probability of belonging to species #1 is set equal to the sample proportion of specimens that belonged to species #1. For example, since 50 out of 150 specimens were of species Setosa, each row is assigned a prior probability of 50/150 = 1/3 of belonging to that species.

In the larger world of Bayesian methods, it is our intention that the prior distribution should reflect the opinions of experts. That is, Bayesian statistics is a method designed to incorporate expert opinion *prior* to the collection of *the data.* For example, if we know we would be taking a sample of iris flowers from a particular region of the Appalachian Mountains, we could ask Anderson, as a biologist that specialized in iris species identification in that region, for his guess at the probability that any randomly selected iris flower in that particular region would be of the Setosa species.

To the contrary, in the *naive Bayes* method, no experts are consulted!

The distribution of π is a now a little more complicated because we have 3 classes, not 2. So, we would need to be more specific in our notation. We can let πi(j) denote the probability that the specimen in the ith row belongs to the jth species.

You might think that this is crazy because we can look at the data set and see the correct species for each row. Where is the probability? The answer is that by “prior” we mean *prior to seeing the individual row data!*

I have no quarrel with that method of assigning prior probabilities, but I suspect that more than a few truly *Bayesian* statisticians are displeased that the term “Bayes” is applied to this method. It is similar in spirit, though, to the way in which Bayes rule was used for purposes of “inverse probability” before 1920. The modern version of Bayesian statistics is quite different.

The central step in modern Bayesian methods is to find a *joint probability* by multiplying a conditional probability by an unconditional probability, such as this example:

P(k=3|π=0.4) \* P(π=.4).

Now, in the current setting, the right-hand term above has been *assigned a value* by the prior distribution. The left-hand term is a conditional probability, but what is its subject? Consider the ith row. It contains four numeric values, one for each of the x-variables. Let us consider x1, which was Sepal Length. While NB can be implemented in many ways, in this case we are assuming that the underlying probability distribution of sepal length is Normally distributed with a population average, μ, which is specialized according to its species classification. In addition, the algorithm is assuming that the σ2 is equal for all four x-variables.

Looking back at the matrix scatterplot at the Sepal.Length column, on the left side, we see that the Setosa cases are generally lower in value than the other species. For orientation, remember that in that column, Sepal.Length is on the x-axis.

Thus, if we knew that a flower was Setosa, then we would predict that its sepal length would tend to be smaller than for other species. This is an inference that μ for sepal length in the Setosa case is relatively low. That means when we imagine a Normal distribution of sepal length for iris flowers of the Setosa species, we picture that the bell curve as lying to the left of the bell curves associated with sepal length for the other two species!

Time permitting and depending on class interest, I will expand on this topic.

**Section 5: NB Implementation using R Package e1071**

For implementation naïve Bayes classification in R, there are two primary NB packages: “e1071” and “klaR”. I was able to download and install both from the **France, Paris** mirror site. More recently, I have favored Austria.

To install *e1071* from within R, I used this command:

> install.packages('e1071', dependencies = TRUE)

After downloading, open the package in R by typing,

> library(e1071)

This program also requires us to open up the *class* package we have used previously:

>library(class)

The e1071 NB program turns out to be fairly easy to use. As with knn, it takes two steps. First, we run the program to get classification decisions; then we compare them against the actual values to get the “confusion matrix.”

Step One: In e1071, the NB function is called “naiveBayes”. We will name its output “classifier”. It is similar to a yhat-equation in regression, except it will be used to produce classification predictions rather than yhat numeric values.

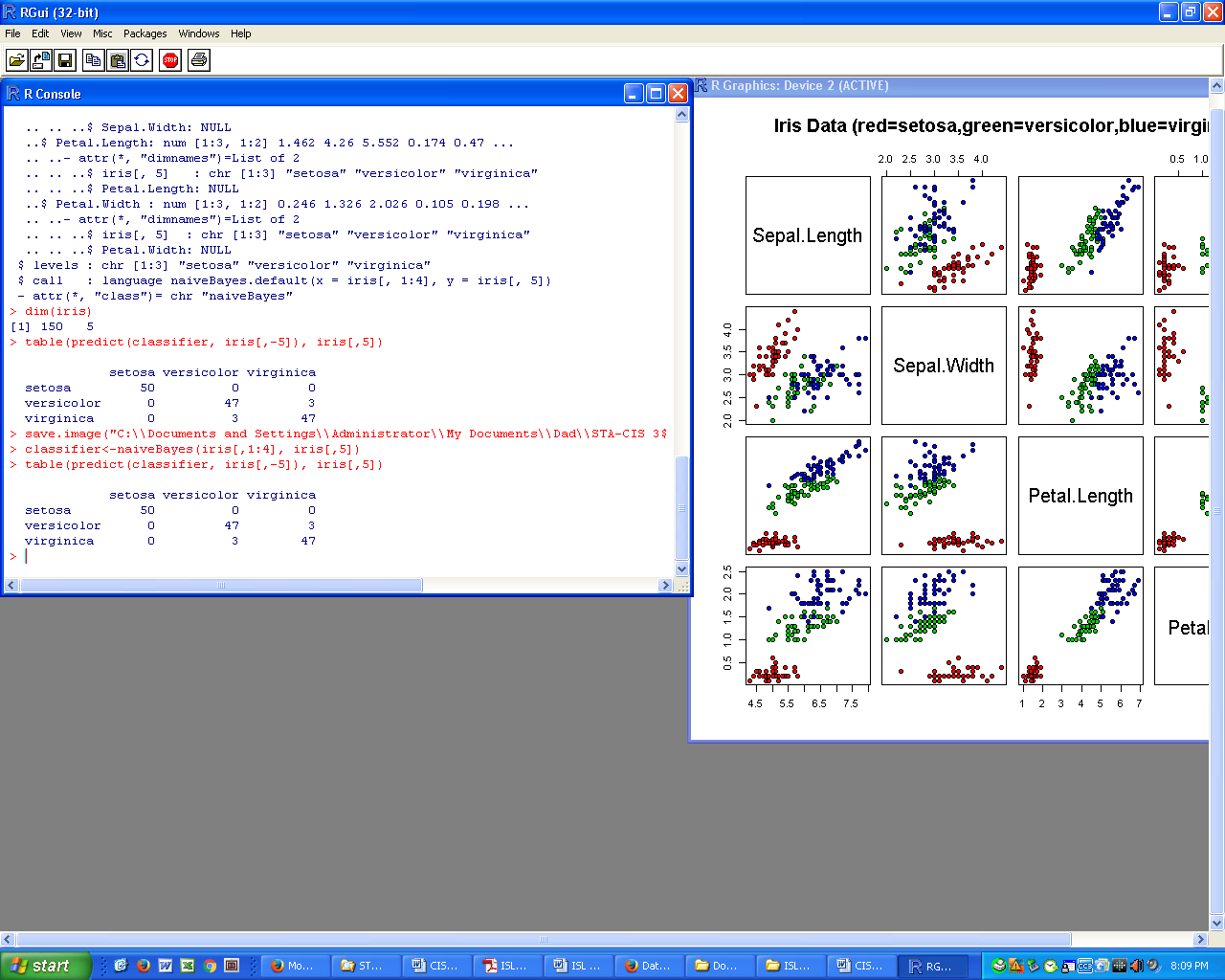
> *classifier* <- **naiveBayes**(iris[,1:4], iris[,5])

Notice the simplicity of the iris inputs to the NB function. What is hidden is the fact that iris[,5] is not a “numeric” column nor it is it a “data frame,” rather, its data type is “factor.” And, of course, “iris” itself is a data frame.

Next, we use the convenient **table** function to check how many of the classifications were correct:

> **table**(**predict**(*classifier*, iris[,-5]), iris[,5])

The output is shown below.



So far, the naïve Bayes classifier has only misclassified 6 specimens out of 150. This is a surprising accomplishment because petal length and petal width have a strong positive correlation (non-independence), which is the property that the naïve Bayes method does not consider! The word “surprising” is often encountered when reading about this method.

**Section 6: Cross-Validation using R Packages “klaR” and “caret”**

In order to employ cross-validation methods, I found it better to switch to the NB program in the R package klaR. The klarR package very well with the package “caret,” which in turn contains a widely admired suite of data mining tools. In particular, it provides easy access to a cross-validation program.

install.packages("caret")

install.packages("klaR")

library(caret)

library(klaR)

In the klaR package, the NB program is called “Naïve.Bayes” rather than “naïveBayes” as in e1071. Fortunately, we are not going to get confused because we will not be calling the klaR program directly. We will only call it within a cross-validation program called “train,” a part of the *caret* package. There, we reference NB as “nb”.

I am now going to show a reasonably sophisticated cross-validated data mining routine. The first step is to set aside a portion of the original data for testing purposes. I am going to set aside one-third of the Iris data for that purpose, and that third will be randomly selected. This step may look familiar from the steps we used from the ISL text with your stock data. The two-thirds used as training data will be set aside by selecting 100 row indices without replacement from the original 150 rows.

> *train*=**sample**(150,100)

Now, prepare the data in four sections as follows – I recommend you take the time to explore what is happening.

> x.train = iris[ train, -5]

> y.train = iris[ train, 5]

> x.test=iris[ -train, -5]

> y.test=iris[ -train, 5]

A lot happens very quickly in the next command. First, we are imputing the “training data”, x.train and y.train. The cross-validation program, given a parameter value of 10, breaks the imputed data training data into 10 sections. Here, we are working with n=100 rows, so 10 rows are assigned to each section. (We have only 100 rows of data because the other 50 have been set aside for a second stage.) The NB program is run without the first section of 10 rows and the output is used to predict the species for each of those left-out 10 rows. That process is repeated for each section of 10 rows – this is a rather weak form of cross-validation – and called using the argument “cv” in the ***train*** function. There are several “tuning parameters” (the “k” in k-nearest neighbors is a tuning parameter) that can be used to adjust the method based on the nature of the setting in which it is used. My understanding is that the **train** function uses cross-validation to check many different combinations of tuning constants and then selects what we hope is the “best” set. This is similar to what we have done in knn with a search across k, including cross-validation to reduce the impact of sampling variation on the choice of k. The output of the **train** function will be termed *model*.

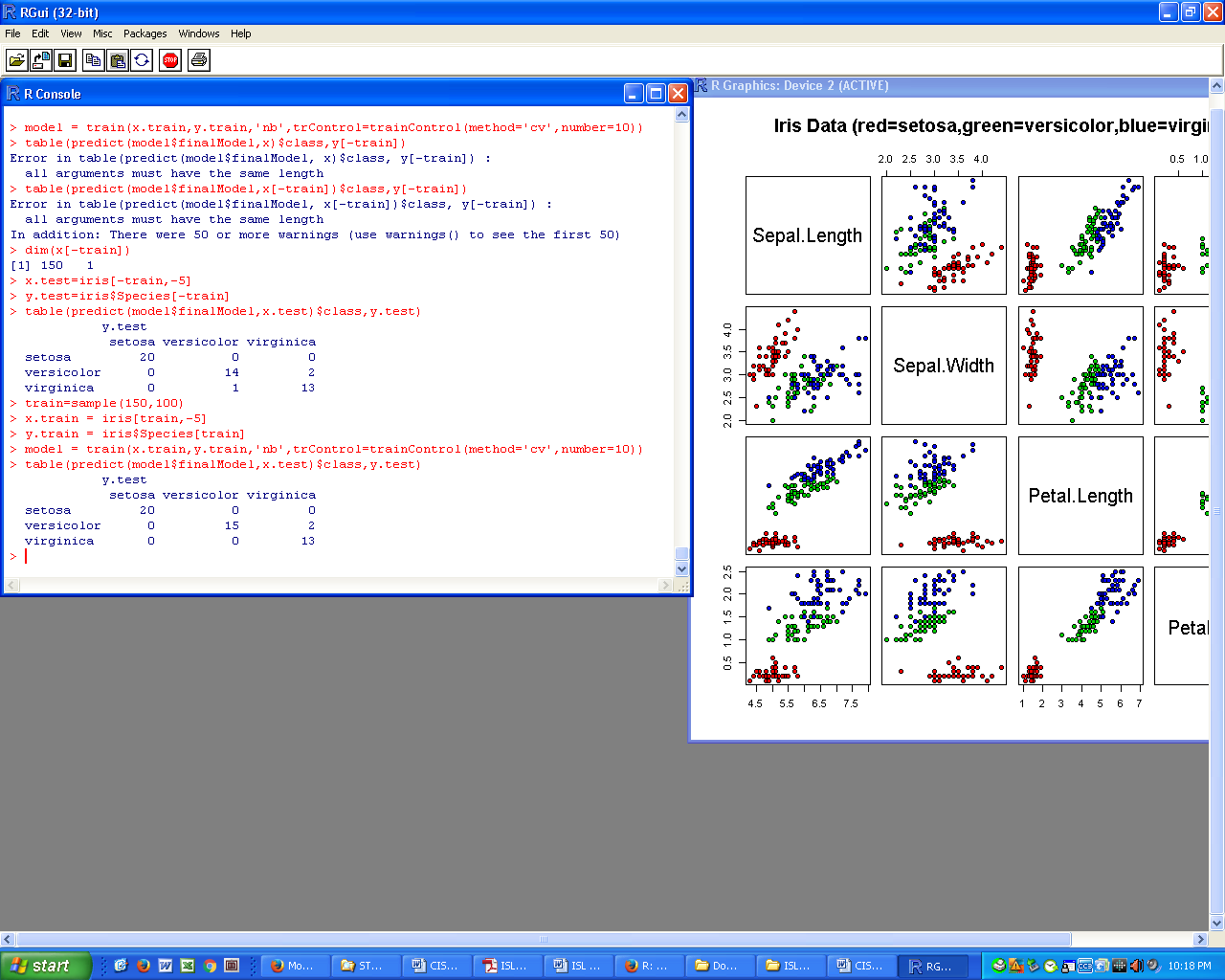
> *model* = **train**(x.train,y.train,'nb',trControl=trainControl(method='cv',number=10))

This *model* output gives instructions as to how to map a particular input of x-values to a classification forecast. The term “model” here is being used in the same way when we use the term “fitted model” to describe the yhat equation in regression.

Next, we input our 50 *test* x-values and the fitted *model* as arguments to the **predict** function. In this implementation, that yields species-predictions for the 50 rows that were *not used* in the development of the classification rule imbedded in the fitted *model*. So, the 100 rows of training data were used to make classification forecasts for the fresh 50 rows of test data. Those 50 classification forecasts are compared against the actual species membership to get the “confusion matrix” using the **table** function, an old friend from the k-nearest neighbors studies.

> **table**(**predict**(*model$finalModel*, x.test)$class, y.test)

Here, this yields,



I return now to my IBM data and use that data. My train\_x is called TrainStLnX.IBM and my train\_y is called TrainY.IBM. I copied and posted those names into the "model = train(....."

command I used on the previous page with the iris data, as below. My IBM variables are contained in the LN5 R file, which I "loaded" into the R window in which I had downloaded the above packages.

Next, I added a "table" name to save the output of the table command used above, so I could compute the proportion of correct forecasts. You will see, below, that I got 0.696, or 69.6%, correct when using naive Bayes with cross-validation.

Interestingly, when I ran the same program again (the same sequence of 4 commands, below), I got a slightly different value of 0.708 (with rounding). A third run gave me again 0.696. I believe the variation is due to the random splitting of the 150 rows into 100 train and 50 test rows; possibly compounded by what I think is a small (10) number of iterations. The statistical standard is 600, at least for the bootstrap, which is a closely related to cross-validation. However, I have not yet checked to see if the low number of replications used in the data mining literature is based on necessity or performance.

> model =train(TrainStLnX.IBM,TrainY.IBM,'nb',trControl=trainControl(method='cv',number=10))

> table=table(predict(model$finalModel, TestStLnX.IBM)$class, TestY.IBM)

> table

TestY.IBM

HighRisk LowRisk

HighRisk 460 168

LowRisk 239 472

> (table[1,1]+table[2,2])/sum(table)

[1] 0.6960418

**Section 7: Two Kinds of Independence**

There is a very real point of confusion lurking nearby. Rather than saying that the x-variables are correlated, I might well have used the broader term and said they were not *independent*. However, in an application like these, we find a completely different usage of the term “independent.” Namely, we often describe the x-variables as the *independent* variables and the y-variable as the dependent variable.

Thus, in Statistics, we have two different meanings of “independence” which might well show up in the same sentence! We could “properly” say: “When using this methodology, the independent variables may or may not be independent of each other!”

Let us take a minute to distinguish those two meanings of “independence.” First, consider a classification method which is going to use variables x1 and x2 to decide whether to guess an animal is a cat or is a dog. We commonly refer to x1 and x2 as “independent variables.”

Also, in simple regression, we often call the y-variable the “dependent” variable and the x-variable the “independent” variable. Likewise, in multiple regression we call the collection of x-variables the “independent” variables.

In all of those cases, the term independent is “nominal” - in name only. It does not literally mean that the y-variable is dependent on the x-variable, although people often succumb to wishful thinking in that direction. But we don’t really mean that we think the animal type depends directly on whisker length. For example, if I use scissors to trim down my cat’s whiskers and then knn classifies it as a dog, have I changed my cat into a dog? No, of course not because we don’t seriously mean that the type of animal itself changes if we were to change the length of its whiskers!

If we are looking at the unemployment rate versus the inflation rate, then in one setting we will call the unemployment rate the dependent or y-variable, and in another setting the unemployment rate is listed as an “independent” or x-variable. The wishful thinking comes in when people forget that the names assigned and the direction of the regression was arbitrary. That often happens with *observational studies*.

Things are quite different when we are doing clinical trials or actual scientific experiments! Then, the terms dependent and independent variable are not arbitrary. In those studies, the independent variable is manipulated by a researcher, such as a study of the effect on heat of changing pressure, or of the volume of sales when Amazon manipulates the price of an item. In that case, the manipulated variable is the independent variable.

**A second usage of “independent” comes from conditional probability**. What is the probability that a randomly chosen card is a Queen? That is 4/52. What is the probability that it is a Queen given that we know it is a Diamond? That is 1/13, which equals 4/52. Thus, knowing the card is a Diamond does not change the probability it is a Queen. Therefore, we say that event of drawing a Queen and the event of drawing a Diamond are *independent*. Thus, A and B are independent if and only if P(A|B) = P(A).

On the other hand, the probability the card is a Queen given that it is also a Face card is 4/12, which is not equal to 4/52. Thus, the event “Queen” and the event “Face” are not independent.

In probability theory, then, the usage of the term “independence” is not merely notational or nominal, it is a deep concept and it has profound implications.

**Section 8. Sources**

[1]

https://en.wikibooks.org/wiki/Data\_Mining\_Algorithms\_In\_R/Classification/Na%C3%AFve\_Bayes

[2]

https://www.quora.com/Classification-machine-learning-When-should-I-use-a-K-NN-classifier-over-a-Naive-Bayes-classifier

[3]

http://joshwalters.com/2012/11/27/naive-bayes-classification-in-r.html

[4]

Introduction to Statistical Learning, pages 138-150.

**Lecture Notes 6 Exercises**

Reading: I am assigning Sections 3, 4, 5, and 6 for reading. If you have time and interest, you should read and work through the other sections, as Bayesian methods have become very popular but are seldom used with understanding in my opinion.

In an Appendix, show you worked through the R commands in Sections 3, 5, and 6. If you cannot get e1071 in Section 5 to install, don't worry about it - just show you tried - and read the section carefully. The e1071 program is commonly discussed in the literature, and it would be useful to understand its format. Fortunately, klaR (in Section 6) is easier to install and seems to me to be a more powerful tool when combined with the caret package. Hopefully, klaR and caret will be installable!

6.1. Apply the naiveBayes klarR program with *cross-validation* to the classification of your stock **range** (HighRisk or LowRisk), using lagged ranges as x-variables. How well does NB do compared to knn using the kcvSearch to select k from the train data? To make this a fair comparison, you need to run knn on the same train-test split that you gave to the NB program.

6.2 Add 8 more lags to your stock range data - the Hi-risk vs Lo-risk data. Do that in Excel using our best practices method. Try running kNN and klaR with that data and with the same train-test split.

(a) What changes do you observe in the performance?

(b) Did adding more regressors (more lags) effect the choice of k in kNN?

(c) Did that slow down the kcvSearch program? Why would that matter?

1. A pamphlet provided at a midwife clinic my wife summarized the issue this way: “For every 33 positive tests, there will be on average only one actual case of spina bifida. The decision of whether or not to have the test is up to you, but we do not recommend it. We recommend you wait and have an amniocentesis test, later, which is far more reliable.” [↑](#footnote-ref-1)