Support Vector Machines – The Accuracy Check

Karissa Maharaj

May 19th, 2021

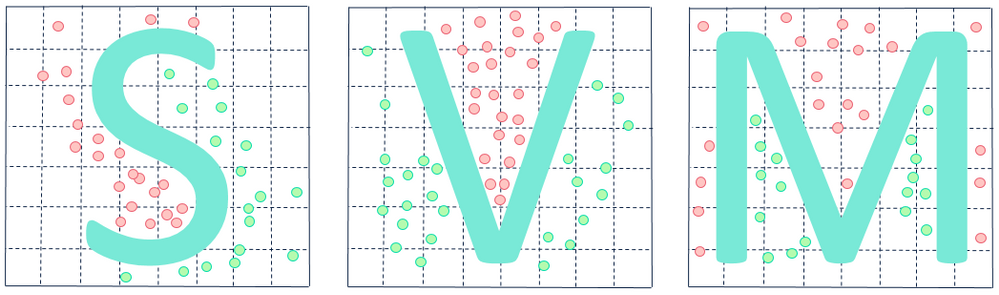


Table of Contents

**Cover Page1**

**Table of Contents2**

**Part 1: K-Nearest Neighbors3**

**Part 2: Naïve Bayes Method5**

**Part 3: Logistic Regression7**

**Part 4: Small Vector Machines9**

**Results11**

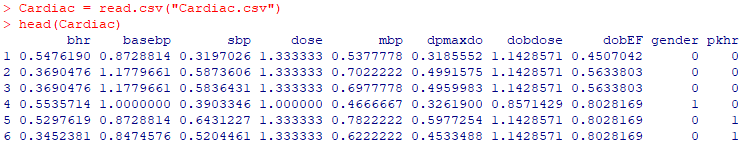
Baruch College, STA-CIS 3920 Maharaj, Karissa

May 19th, 2021

Accuracy Checks Project

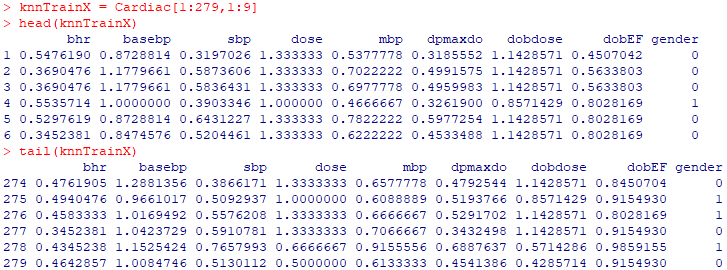
In this project, I will be manipulating the Cardiac.csv dataset. This dataset contains 558 rows of data and 1 row for the titles of each column. I am taking the bhr, basebp, sbp, dose, mbp, dpmaxdo, dobdose, dobEF, and gender columns as my x-variables and I am taking the pkhr column as my y-variable. Within the csv file, I kept all the rows I wanted and deleted any rows that I did not want, then found the median of pkhr and created an IF statement that dictated that if a pkhr value was above the median of the data, a 1 would be put in its place, and if it was less than or equal to the median of the data, a 0 would be put in its place. Next, I loaded the Cardiac csv file into my dataset and started creating variables for my first data forecasting method:

**Part 1: K-Nearest Neighbors (knn)**

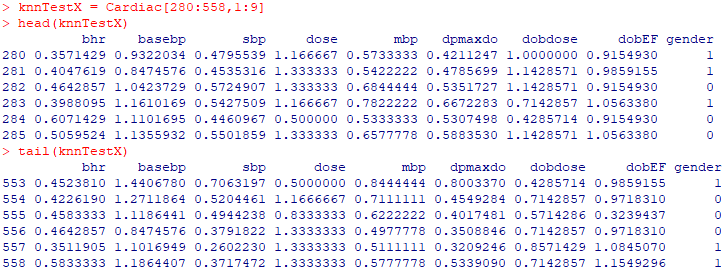


We see all the variables I am using in this image.

Now, I am going to create my variables:



As stated before, my data contains 558 rows. For my train variable, I am taking exactly half of that data.



And I took the other half of my data and placed it in the test variable.

Next, I am creating my y-variable:



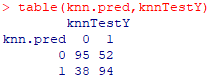
I also create a TestY variable, but this will not be used in the knn function. It will be used as a means of testing the accuracy of the forecast of knn. TestY is the actual data, while knn.pred, the predictions made by knn, are the forecast of that data.

And then I create a k variable and set it equal to 3.

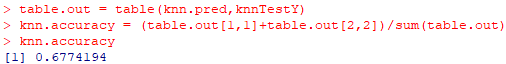
Next, I use the knn function and enter my created variables:



I almost forgot to import the class library! That is needed in order to use the knn function. Now, I can create a table:



This table tells us what is correctly predicted and what is not clearly predicted. We can calculate the accuracy of the knn method using this formula:



I had to put the table created before into a variable in order to properly access the rows and columns of data I needed in order to calculate the accuracy of knn. We can see here that the accuracy of this table is about 67.7%. Let’s make this number easier to see for easier comparison later on:

**K-Nearest Neighbors Accuracy: 67.7%**

**Part 2: Naïve Bayes’ Method**

For this method, we are going to once again begin by creating the variables we need.

First off, in our Excel file, we need to take the natural logs of our data. I did not take the natural logs of my gender data, as it is all 0’s and 1’s.

I upload my altered csv file in:



And then create my x-variable:



I create a few variables in order to shuffle my data around. I want to be able to take a sample of my data at complete random. Therefore, these variables will allow me to shuffle my data around:



Now, I can create a sample of my x-variable:



And I can create my y-variable:

  
Along with the sample of my y-variable:



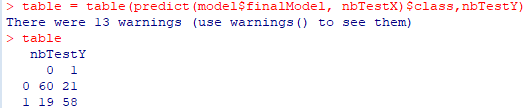
Next, I can put all of these into my model:



And I can now create variables to test my nb model against:



These are would be the values we are trying to forecast. Here is the creation of the table:



I chose to ignore the warnings, as my table was returned without a hitch. I calculated the accuracy of the table next:

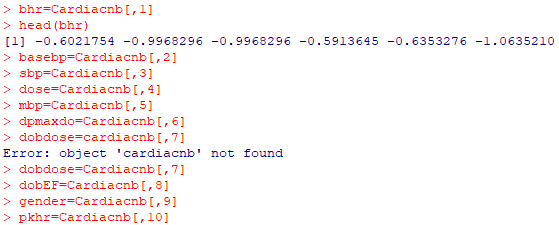


So we can see that the nb method gives us an accuracy of about 74.7%. This is slightly higher than our previous knn method. Let’s make this bolder for better comparison:

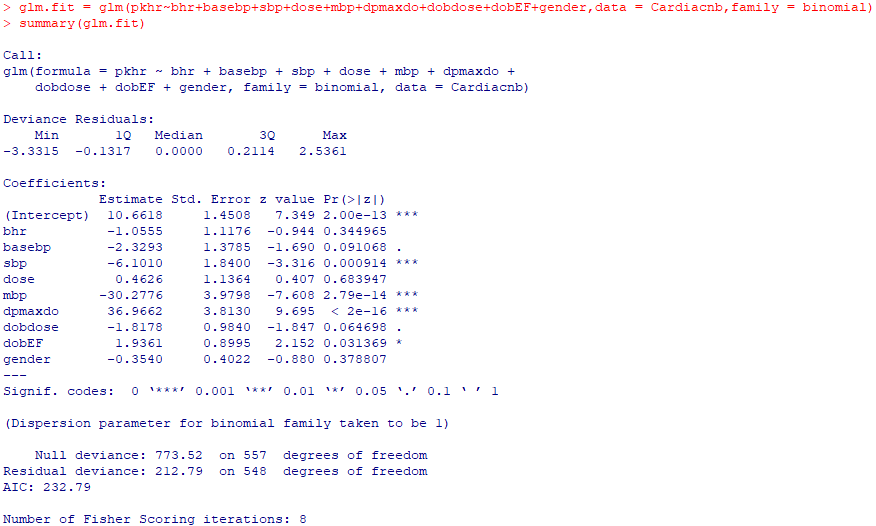
**Naïve Bayes’ Method Accuracy: 74.7%**

**Part 3: Logistic Regression**

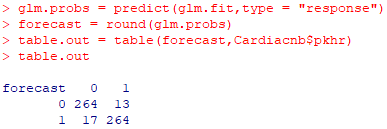
Now, we are going to look at logistic regression. For this, we need to split our data into distinct variables. I do so here:



Here, you can see that each x-variable I am using to forecast my data now has its own variable name in R. Now, I can use the glm function:



I use all of my variables in the correct places. My y-variable is put first, and then all of my x-variables are put next. I then am able to summarize my data. Next, I can create my table:



Now I can calculate the accuracy of my data here:



With a whopping 94.6% accuracy rate! That is going to be super difficult to beat. Let’s make that bold:

**Logistic Regression Accuracy: 94.6%**

**Part 4: Small Vector Machines**

Here, I created my x and y-variables and factored and standardized them, as needed:



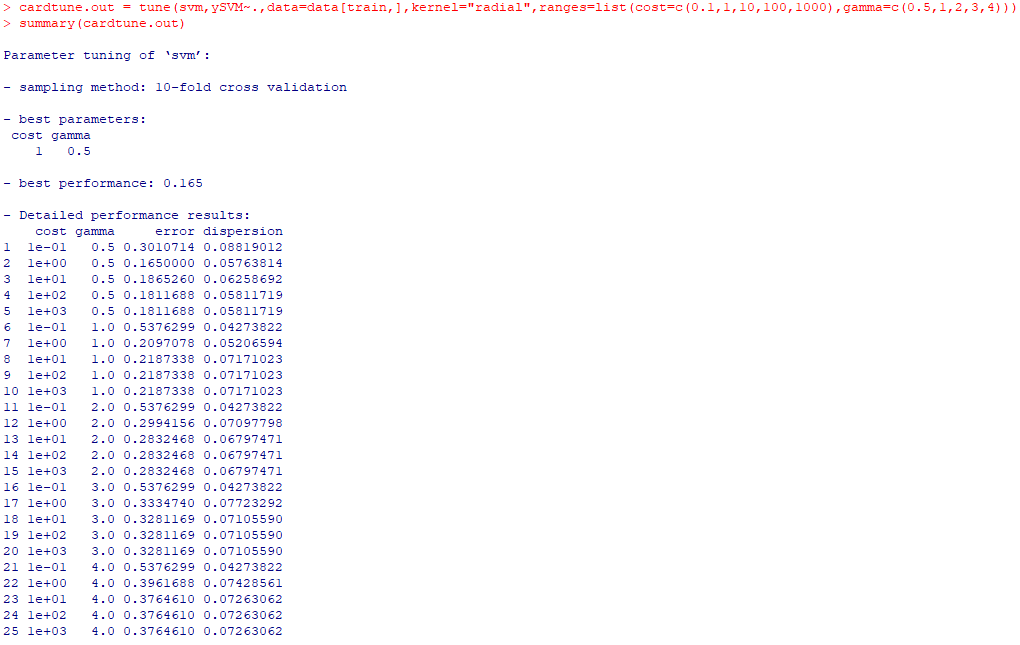
I also created a dataframe called “data” to use in my svm function:



Next, I used the svm function and stored it in the variable cardsvmfit:



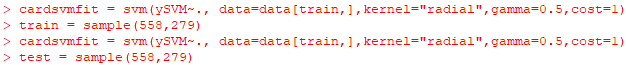
These were my first attempts at using the svm function on my Cardiac data. I created the tune.out variable and summarized it to see my best parameters.



I tried putting the data straight into a table, but it did not work out, as it gave me the same errors it did beforehand when I first tried svm in A1:



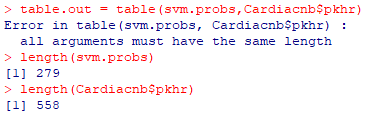
At this point, I thought, why put the data straight into a table when I have the best parameters right in front of me? I recreated the cardsvmfit variable with the best parameters:



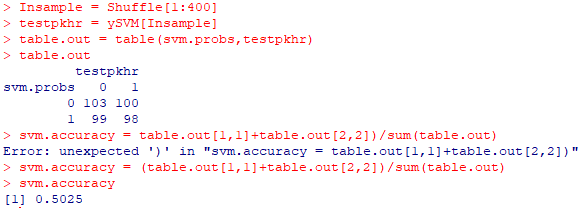
I figured I could take some knowledge from my other forecasting methods and try to make this table work. I created train and test samples at first, but I recreated them later on.



I tried making a ‘.probs’ variable like in the other forecasting methods I used, and it proved to be a success later on.



I tried the table again, but saw that my data was not the same length. One was the full length of the dataset, while the other was only half the dataset. I know how to fix that!



I created the testpkhr variable, which took a sample of the data in the table at the same length as the ‘.probs’ variable. This allowed the table to run. I ensured that cardsvmfit was indeed using ySVM in the svm function. I thought I did the svm function correctly. However, we got a very low accuracy rating of 50.3%. I was expecting a higher forecast accuracy rate for svm, as our forecast accuracies have only been increasing as we looked at each method. However, this one dropped substantially. If I am correct, svm has an accuracy rating of 50.3%. Let’s now put all of our accuracies together:

**K-Nearest Neighbors Accuracy: 67.7%**

**Naïve Bayes’ Method Accuracy: 74.7%**

**Logistic Regression Accuracy: 94.6%**

**Support Vector Machines: 50.3%**

There you have it! Logistic Regression gives us the most accurate forecasts with an accuracy score of 94.6%, and goes down as the most accurate forecasting method, at least for this specific dataset.