Lab 6: Logistic Regression and Support Vector Machines

Email service providers, the email clients they develop, and the users of those clients are all interested in decreasing the number of spam emails users receive. Most email programs try to catch spam before they reach users. But how do they do that? In this lab you will explore how to use logistic regression and support vector machines to try to predict whether an email is spam or not.

Getting started

Click the Github Assignment link on Moodle and open the repository in Posit Cloud.

Warm up

Let's warm up with some simple exercises. Open the `lab6.qmd` Quarto file. Update the YAML of your Quarto file with your information, knit, commit, and push your changes. Make sure to commit with a meaningful commit message.

Please make sure to keep committing throughout while working on the assignment!

Packages

You will need the `e1071` package. You will need either the `ROCR` or `pROC` package to make ROC curves and calculate the AUC. We strongly recommend that you use the `tidyverse` suite of packages for data visualization and data wrangling. The `patchwork` package may also come in useful for combining plots.

```
# load packages here
```

Data

For this lab we will be working with the `email` data in the `openintro` package. This dataset contains information about 3921 emails received during the first three months of 2012 by one of the authors of the OpenIntro project. The dataset contains such difference pieces of information about each email, such as the number of characters in the email, whether the email had "Re:" in the subject, the time at which email was sent, number of times the word "inherit" shows up in the email, etc. Type `?email` into the console in order to see the associated documentation.

```
#Import data here
```

Exercises

Data Visualization

Q1

Choose two of the variables that you think could most help predict whether an email will be spam or not. Write a short paragraph explaining why you think these two variables in particular could help predict whether an email will be spam or not, and in which way.[1]

Q2

Visualize the two variables together (one should be on the x-axis, one on the y-axis), and color the points by whether an email is spam or not.

Does it look like the relationships match the ones you expected to see? Do the classes seem to be linearly separable or not?

Two-Variable Models

Q3

Create a 70/30 training and test split with the data. Set the seed to `145` before doing so.

[1] For example, if there were a variable called "'num_this_is_not_spam" that counted up how many times an email contained the phrase (or a similar phrase) "this is not spam," I would expect that to be positively associated with whether an email will be spam or not because spammers try hard to make spam seem not like spam.

Q4

Fit a logistic regression model with the two variables you chose above. Plot the ROC curve and report the AUC *for the test data* (you can use `ROCR`, `pROC`, or `ggplot2` for the visualization).

Based on the coefficients, do the relationships match the ones you expected to see (which you wrote down in Q1)?

Q5

Fit a support vector machine with the two variables you chose above.

Use cross-validation to find the optimal model. Set the seed to `246` before doing so and the search over the following tuning parameter space:

```
list(
  cost = c(0.1, 1, 10),
  gamma = c(0.5, 1, 2, 3),
  kernel = c("linear", "radial")
    )
```

What is the optimal combination of parameters?

Note: Tuning may take some time. We are also searching over a somewhat constrained space – ideally we would search over more of parameter space, but that can take significant amounts of time, especially as the number of observations also increases.

Plot the decision boundary using the built in `plot()` function.

Note: An annoying quirk of `plot.svm()` (the `plot()` method for `svm` objects) is that it wants its `data` argument to only include the variables included in the fitting of the SVM, plus the response.

Plot the ROC curve and report the AUC *for the test data* (you can use `ROCR`, `pROC`, or `ggplot2` for the visualization).

Full Models

Q6

Repeat Q4, but this time use all variables in `emails`.

Compare performance on the test set with the two-variable model.

Q7

Repeat Q5 (you do not have to plot the decision boundary) with all the variables in `email` **except for `viagra`.**[2] This time set the seed to `247` before cross-validation. To save on time, do only 5-fold cross-validation (read the helpfile for `tune()` to figure out which argument you need to use to do this).

Compare performance on the test set with the two-variable model.

Q8

Compare performance on the test set of the two full models.

Final Steps

When you have made your final edits, remember to render, stage the changed files, then commit, and push to the repo.

Don't forget to submit your pdf on Canvas.

Grading

Please put your names on your answer document and add the date. You won't get points for doing this, but you will *lose* points for not doing so.

Remember to comment your code. You don't get points for commenting, but you will *lose* points for not commenting your code.

For clarity's sake, you should label your code chunks (this helps make debugging easier). You will *lose* points for not naming chunks.

If an exercise asks a question, you should answer it in narrative form and not just rely on the code. You will be *penalized* otherwise.

Please suppress warnings and messages. You will *lose* points if you do not suppress warnings and messages.

If you are asked to make a visualization, make sure that it is presentable, professional in appearance, has a title, and has properly labeled axes. You will *lose* points if you do not do this.

[2] The `viagra` variable has too little variation for it to be usable in cross-validation – the likelihood is high that at least one fold will not have an email that contained the word "viagra", which is a problem for model fitting.

Points Breakdown

The different components of this lab are worth the following points:

- Q1: 2 pts
- Q2: 4 pts (2 pts for visualization, 2 pts for narrative response)
- Q3: 1 pt
- Q4: 5 pts
- Q5: 5 pts
- Q6: 6 pts
- Q7: 6 pts
- Q8: 2 pts
- Workflow (includes making sufficient commits): 4 pts

Total: 35 pts