Homework 2: Logistic Regression and k-Nearest Neighbors Classification

STAT 665

due Thurs. Feb. 2nd 5PM

Background

The dataset consists of a collection of 57 features relating to about 4600 emails and a label of whether or not the email is considered spam. You have a training set containing about 70% of the data and a test set containing about 30% of the data. Your job is to build effective spam classification rules using the predictors.

A Note about Features

The column names (in the first row of each .csv file) are fairly self-explanatory.

- Some variables are named `word_freq_(word)`, which suggests a calculation of the frequency of how many times a specific word appears in the email, expressed as a percentage of total words in the email multiplied by 100.
- Some variables are named `char_freq_(..)`, which suggests a count of the frequency of the specific ensuing character, expressed as a percentage of total characters in the email multiplied by 100. Note, these characters are not valid column names in R, but you can view them in the raw .csv file.
- Some variables are named `capital_run_length_()` which suggests some information about the average (or maximum length of, or total) consecutive capital letters in the email.
- `spam`: This is the response variable, 0 = not spam, 1 = spam.

Missing Values

Unfortunately, the `capital_run_length_average` variable is corrupted and as a result, contains a fair number of missing values. These show up as `NA` (the default way of representing missing values in R.)

Your Task

Part 1 (20%)

Use k-nearest neighbors regression to **impute** the missing values in the `capital_run_length_average` column with $k = 15$ using the other predictors after standardizing (i.e. rescaling) them. You may use your previous implementation of `myknn()` from the last homework, or use a package that has k-nearest neighbors regression as a built-in function. There is no penalty for using a built-in function.

When you are done with this part, you should have no more NA's in the `capital_run_length_average` column in either the training or the test set. Make sure you show all of your work.

Part 2 (20%)

Write a function named `knnclass()` that performs k-nearest neighbors classification. This function will be more sophisticated than your previous `myknn()` function from the previous assignment in the following way:

- The function should automatically do a split of the training data into a sub-training set (80%) and a validation set (20%) for selecting the optimal k. (More sophisticated cross-validation is not necessary.)
- The function should standardize each column: for a particular variable, say x_1, compute the mean and standard deviation of x_1 **using the training set only**, say $\bar{x}_1$ and s_1; then for each observed x_1 in the training set and test set, subtract $\bar{x}_1$, then divide by s_1.

Function skeletons:

In R, start with:

```
knnclass <- function(xtrain, xtest, ytrain)
```

Or in Python, start with:

```
def knnclass(xtrain, xtest, ytrain):
```

If you choose to use Python, please ensure that your code is compatible with versions 2.7 or 3+. You are allowed to use the `numpy`, `math`, and `pandas` libraries.

You are **not allowed** to use a built-in distance or norm function.

Save this completed function in a file called either `HW2_knnclass.py` or `HW2_knnclass.R`.

Note: You can assume that all columns will be numeric and that Euclidean distance is again to be used as the distance measure.

Part 3 (60%)

In this part, you will need to use a k-NN classifier to fit models on the actual dataset. If you weren't able to successfully write a k-NN classifier in Part 2, you're permitted to use a built-in package for it. If you take this route, you may need to write some code to accomplish the standardizing and selection of k, which is already accomplished by `knnclass()` in Part 2.

If you are using Python, you are, of course, permitted to use the `scikit` implementation of logistic regression.

Now fit 4 models and produce 4 sets of predictions of `spam` on the test set:

- `knnclass()` using all predictors except for `capital_run_length_average` (say, if we were distrustful of our imputation approach). Call these predictions `knn_pred1`.
- `knnclass()` using all predictors including `capital_run_length_average` with the imputed values. Call these predictions `knn_pred2`.
- logistic regression using all predictors except for `capital_run_length_average`. Call these predictions `logm_pred1`.
- logistic regression using all predictors including `capital_run_length_average` with the imputed values. Call these predictions `logm_pred2`.

In 3-4 sentences, provide a quick summary of your second logistic regression model. Which predictors appeared to be most significant? Are there any surprises in the predictors that ended up being significant or not significant?

Put together a .csv file called `HW2_NETID_results.csv` that contains 5 columns:

- `capital_run_length_average`: the predictor in your test set that now contains the imputed values (so that we can check your work on imputation).
- `knn_pred1`
- `knn_pred2`
- `logm_pred1`

- `logm_pred2`

Make sure that row 1 here corresponds to row 1 of the test set, row 2 corresponds to row 2 of the test set, and so on.

What to Submit

- `HW2_NETID_knnclass.py` or `HW2_NETID_knnclass.R` pertaining to Part 2
- `HW2_NETID_results.csv` from Part 3
- A compiled .Rmd or .ipynb file in either Word or PDF format documenting your work from Parts 1 through 3 (including loading the data).