Assignment 2

Statistics and Data Science 365/565

Due: Wednesday February 12th (11:59pm)

This homework treats classification and cross validation, and gives you more experience using R.

Problem 1: Spam, wonderful spam! (30 points)

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_(number)`, 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_(number)` 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 with $k = 15$ to **impute** the missing values in the `capital_run_length_average` column using the other predictors after standardizing (i.e. rescaling) them. You may use a package such as `FNN` that has k-nearest neighbors regression as a built-in function (i.e. `knn.reg`). 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 (30%)

Write a function named `knnclass()` that performs k-nearest neighbors classification, without resorting to a package. Essentially, we are asking you to recreate the `knn` function in `FNN`; though, we do not expect you to implement a fancy nearest neighbor search algorithm like what `knn` uses, just the naive search will suffice. Additionally, this function will be more sophisticated than `knn` 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)
```

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

Part 3 (50%)

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 standardize the variables and select k, which `knnclass()` from part 2 already does.

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

1. `knnclass()` using all predictors except for `capital_run_length_average` (say, if we were distrustful of our imputation approach). Call these predictions `knn_pred1`.
2. `knnclass()` using all predictors including `capital_run_length_average` with the imputed values. Call these predictions `knn_pred2`.
3. logistic regression using all predictors except for `capital_run_length_average`. Call these predictions `logm_pred1`.
4. 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 (model 4). Which predictors appeared to be most significant? Are there any surprises in the predictors that ended up being significant or not significant?

Submit a .csv file called `assn2_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.

Problem 2: Cross validation (20 points)

(a) Generate a simulated data set as follows:

```
set.seed(1)
x = rnorm(100)
y = x - 2*x^2 + rnorm(100)
```

In this data set, what is n and what is p? Write out the model used to generate the data in equation form.

(b) Create a scatterplot of X against Y. Comment on what you find.

(c) Set a random seed, and then compute the Leave-One-Out Cross-Validation (LOOCV) errors that result from fitting the following four models using least squares:

i. $Y = \beta_0 + \beta_1 X + \epsilon$
ii. $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \epsilon$
iii. $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \epsilon$
iv. $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \beta_4 X^4 + \epsilon$

Note you may find it helpful to use the data.frame() function to create a single data set containing both X and Y. For linear regression, the LOOCV error can be computed via the following short-cut formula:

$$\text{LOOCV Error} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{Y_i - \widehat{Y}_i}{1 - H_{ii}}\right)^2$$

where H_{ii} is the i^{th} diagonal entry of the projection matrix $H = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$, and $\mathbf{X}$ is a matrix of predictors (the design matrix). This formula is an alternative to actually carrying out the $n = 100$ regressions you would otherwise need for LOOCV.

(d) Repeat (c) using another random seed to generate data, and report your results. Are your results the same as what you got in (c)? Why?

(e) Which of the models in (c) had the smallest LOOCV error? Is this what you expected? Explain your answer.

(f) Comment on the statistical significance of the coefficient estimates that results from fitting each of the models in (c) using least squares. Do these results agree with the conclusions drawn based on the cross-validation results?