homework1_answers_only

Zach Olivier
5/22/2018

Question 2.1

Question:

Describe a situation or problem from your job, everyday life, current events, etc., for which a classification model would be appropriate. List some (up to 5) predictors that you might use.

Answer:

One example I am excited about is an attempt to build a classification model to determine whether a person will purchase a certain product or not, say a new car.

Based on a list of attributes, a classification model could sort customers into groups of 'most likely will purchase' and 'most likely will not purchase' with high accuracy. This would help optimize our marketing budget by only targeting customers who are the most likely to purchase.

Predictors I would use for this model:

- Household Income
- Location
- Have they purchased a car before
- Time since last vehicle purchase
- Type of vehicle last purchased

Question 2.2.1

The files credit_card_data.txt (without headers) and credit_card_data-headers.txt (with headers) contain a data set with 654 data points, 6 continuous and 4 binary predictor variables. It has credit card applications data with a binary response variable (last column) indicating if the application was positive or negative. The data set is the "Credit Approval Data Set" from the UCI Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets/Credit+Approval) without the categorical variables and without data points that have missing values.

Question:

Using the support vector machine function ksvm contained in the R package kernlab, find a good classifier for this data.

Show the equation of your classifier, and how well it classifies the data points in the full data set. (Don't worry about test/validation data yet; we'll cover that topic soon.)

Answer:

First we read in the data and load the needed packages for this analysis. Then, we quickly look at the data and see if it matches the course description.

Our problem is classifying the data based on the 11th column - which I relabeled as response_y for clarity.

I also established a baseline accuracy that our model based classifiers should beat easily. My naive classifier will pick '1' as the answer for every value in the training set.

In this case my naive classifier will accurately classify ~45% of the observations in the

training set.

Our proposed SVM and KNN models should aim to beat this baseline accuracy measure.

Answer:

The following code sets up the Support Vector Machine algorithm using the kernlab package in R.

The ksvm function is supplied with a formula to model the response_y binary variable based on all remaining predictors in the data set. The scaled argument is set to TRUE - the function will automatically scale the predictor data. Type is 'C-svc' to indicate we are using this function for classification. Finally, the kernal is set to 'vanilladot' which initializes a linear decision boundary.

I then set up a for loop to pass my model different regularization (C - Cost) parameters and then extract each model's accuracy. The model with the highest training set accuracy based on values of C will be selected as my 'best' model. **(Note: training set accuracy is not a valid measure of model performance!)**

Here are the results:

The best tuned model with C = 95.45 achieved 95% training set accuracy. This outperforms the naive classifer baseline accuracy greatly

However, this does not indicate we have a good model. We may have modeled our training set real and random effects accurately - but our model will likely not generalize well to new data. There is a high likelihood that we have over fit this model to the training set.

The formula for the best tuned linear decision boundary is:

y =

(-36.5 x A2) + (-8.36 x A3) + (54.2 x A8) + (48.6 x A9) + (-17.5 x A1) + (-22.6 x A10) + (14.5 x A11) + (-22.1 x A12) + (-56.4 x A14) + (50.2 x A15) + (-.778)

Question 2.2.3

Question:

Using the k-nearest-neighbors classification function kknn contained in the R kknn package, suggest a good value of k, and show how well it classifies that data points in the full data set. Don't forget to scale the data (scale=TRUE in kknn).

Answer:

Here we apply Leave One Out Cross Validation (LOOCV) to the credit dataset in order to find the optimal K for our KNN model. To do this, I implement the train.kknn function to cycle through possible values of K and determine the best value based on LOOCV error. Variables are scaled within the algorithm by setting scale = True. The kmax arguement is upper bound of K values we would like to consider for the model.

In this implementation we find that the optimal K is 58.

Using this optimal K value, we fit a KNN model to the entire training set to determine accuracy.

The accuracy of the K = 58 KNN model is ~88%

Note: Using this model to predict back onto the training set will give misleading results. Smaller values for K will actually predict better on the training set. However, this will not generalize to an independent or new dataset. K = 1 may perform perfectly on the training set but K = 58 will likely generalize better to other datasets.

Question 3.1

Question:

Using the same data set (credit_card_data.txt or credit_card_data-headers.txt) as in Question 2.2, use the ksvm or kknn function to find a good classifier:

using cross-validation (do this for the k-nearest-neighbors model; SVM is optional) splitting the data into training, validation, and test data sets (pick either KNN or SVM; the other is optional).

Answer: (a)

The kknn package provides a function 'train.kknn' which will search for the best value of K based on cross validation error. I specified kcv = 10 to run a 10-fold cross validation on the full credit dataset.

The train.kknn function will run the 10-fold cross validation and extract the K (nearest neighbor parameter K, not cross validation fold k) with the minimial error.

Based on the results the optimal **K is 58 and results in around 10% mean squared error**

I also provide the results for K = 30, 20, and 10 to illustrate that the accuracy improvement for using K = 58 is not that much smaller than using less neighbors.

Note: Using this model to predict back onto the training set will give misleading results. Smaller values for K will actually predict better on the training set. However, this will not generalize to an independent or new dataset. K = 1 may perform perfectly on the training set but K = 58 will likely generalize better to other datasets.

Cross validation is a better estimate of test set error and we should trust those results more than accuracy or error on the training set.

Answer: (b)

I will use the caret package to split our data set into training, test, and validation data sets. **I will portion the available data 60% training, 20% test, and 20% validation.**

We train the model on the training data set, measure accuracy and tune parameters using the test set. Once we think we have a good model, we will re-train the model on the entire training and test sets, then calculate accuracy for our best model using the validation data set.

Answer: (b)

Here we determine the best value of K by training our KNN model on the training data set, and measuring accuracy on the test data set. Based on this method the K value that produced the maximum accuracy on the test data set is **K = 40, with accuracy of ~88%**.

Now that we have our optimal value of K based on our test data set - we will apply this model to the final holdout data in the validation set. We do this to get an unbiased view of how our model will perform on new data.

The KNN model with K = 40 achieves ~85% accuracy on the validation dataset. This means we should expect our classifier to accurately predict 85% of new 'unseen' data into the correct category.