Week 10: Model Validation

10/31/2019

Jake Campbell

The Importance of Cross-validation

- Building a model is great, but what is important are the predictions
- Gauging model performance on predictions on data the model was trained on is a no-no
    - Your model has already seen this data; predictions are likely to be overly-optimistic
- A real test of model performance is looking at how well the model predicts data it hasn't seen

Training/Testing Approach

- A simple method of cross-validation is to hold out some data from the model training process
- This will act as a test set
- Generally speaking, you would want the majority of observations to be in your training set (70-80%)
- `createDataPartition` can be used to create a stratified random sample of indices to use as a training set

```
data.split <- createDataPartition(Sonar$Class, p = .75, list = F)
# Index Sonar by data.split to get a training and testing set
training <- Sonar[data.split, ]
testing <- Sonar[-data.split, ]
```

Issues with the Training/Testing Approach

- Depending on which rows are randomly chosen in our dataset, we can have varied results
  - Maybe the model was trained on the easy to predict observations
- We are leaving out data we could be training the model on
- Can be easily influenced by researcher bias
  - If we have just one testing set, we can continue to adjust model parameters to try and make our test set look ideal
  - If we don't like the results we get for our test set, we can just resample post-hoc until we like our test set results

Other Cross-validation Methods

- Leave One Out
  - A model is built on all observations except one; this one observation is used as the test set
  - This process is repeated for each row and the error is averaged across all observations
- Bootstrap
  - The dataset is continuously sampled with replacement creating several training sets
  - Test sets are the unused observations from each sample
- K-fold Cross Validation
  - The dataset is split into k-folds with a model being trained on each combination of folds, leaving one fold out to act as a testing set each time
  - Results are averaged across each of the test sets

caret

- Classification And REgression Training
- Allows us to perform other validation methods using several different types of models
- Full list of models available and what package they are from available on caret's documentation
    - https://topepo.github.io/caret/available-models.html
- Note that the model created using caret is a caret object and not a specific model's object
- Also has several preprocessing functions
    - Splitting, scaling, centering, etc.

Creating Models with `caret`

- Model syntax is the same as other models, but there are additional arguments to be made
- Input model formula into caret's `train()` function
- Specify `method` argument to specify what type of model you want caret to use
- Specify `trControl` using the `trainControl()` function
    - We can input the cross validation method within `trainControl` function
- Specify a grid of tuning parameters using the `tuneGrid` argument
    - Must be input as a dataframe where each parameter is a column

7/13

Common Cross-validation Methods with `caret`

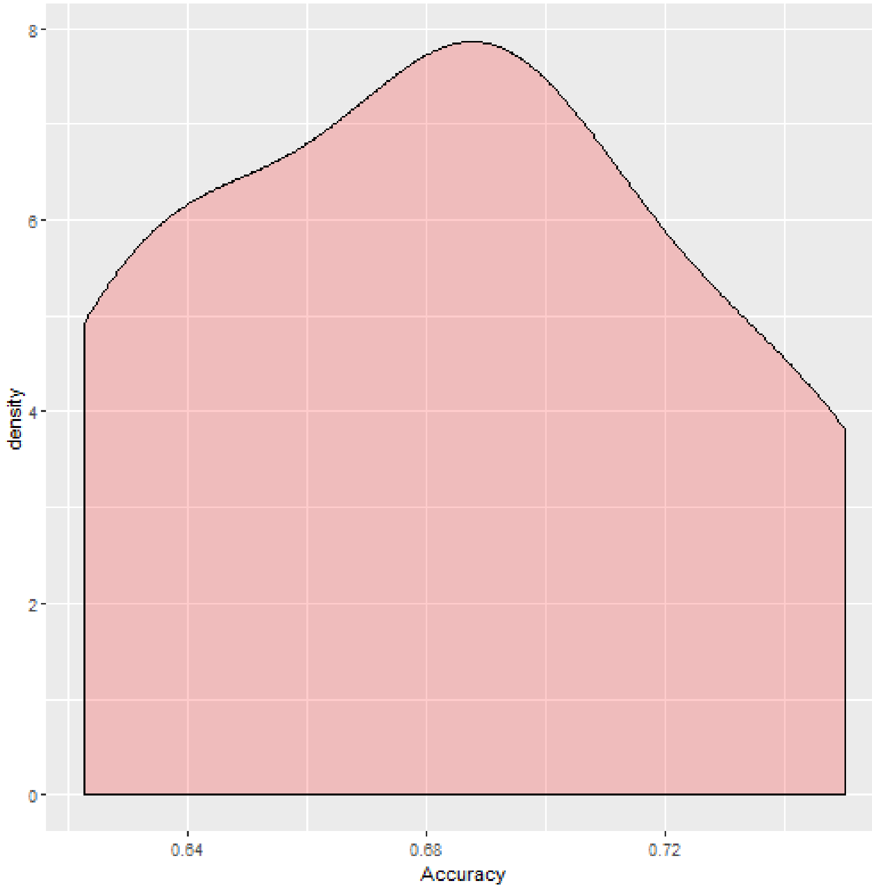
- We need to specify the cross validation method within the `trainControl()` function
  - Specify within `method` argument
- K-Fold CV: `method = "repeatedcv"`
  - Specify number of folds and number of repeats
- Leave One Out: `method = "LOOCV"`
- Bootstrap: `method = "boot"`
  - Specify number of resamples

`caret` Output

- The final model created by caret is a model using all data observations
- By specifying `savePredictions = T` in the `trainControl` function, we save results from each cv fold
  - We can use this to identify an expected distribution of what error metric to expect

caret Output

```
ggplot(data = sonar.glm.cv$resample, aes(x = Accuracy)) +
  geom_density(alpha = .2, fill="red")
```



10/13

`caret` and `confusionMatrix`

- `caret` has a `confusionMatrix` function
  - Creates a confusion matrix as well as gives several different accuracy measurements
  - Specify `data` as your predictions and `reference` as the actual values
  - Set the positive class with the `positive` argument

caret and confusionMatrix

```
confusionMatrix(data = class.predictions, reference = testing$Class,
                positive = "R")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  M  R
##          M 19  5
##          R  8 19
##
##                Accuracy : 0.7451
##                  95% CI : (0.6037, 0.8567)
##     No Information Rate : 0.5294
##     P-Value [Acc > NIR] : 0.001311
##
##                   Kappa : 0.492
##
##  Mcnemar's Test P-Value : 0.579100
##
##             Sensitivity : 0.7917
##             Specificity : 0.7037
##          Pos Pred Value : 0.7037
##          Neg Pred Value : 0.7917
```

`caret` for Future Use

- In addition to being a tool for cross validation, `caret` has importance in model selection
- Not really pertinent to this class, because models we go over have no additional tuning parameters
- Several models can be built based off of different tuning parameters
  - Ex: Boosted trees can be built with a different number of tree based models; caret can build models at different intervals for number of trees (`100`, `500`, `1000`, etc.) and the models can be compared based on different metrics