From:
https://onlinelibrary.wiley.com/doi/pdf/10.1111/ina.12513

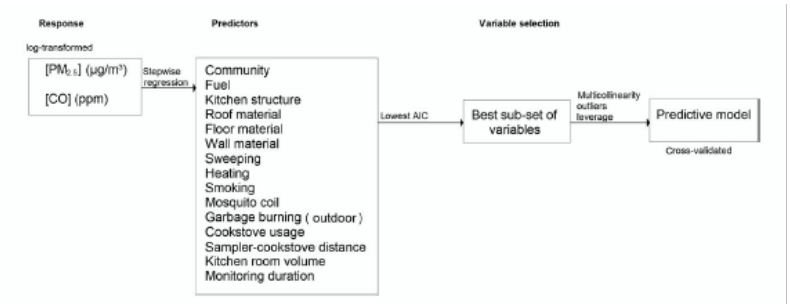

FIGURE 3 Plan of data analysis used to generate the predictive models for indoor $PM_{2.5}$ and CO concentrations

Cross-Validation Essentials in R

kassambara | 11/03/2018 | 65616 | Comments (6) | Regression Model Validation

Cross-validation refers to a set of methods for measuring the performance of a given predictive model on new test data sets.

The basic idea, behind cross-validation techniques, consists of dividing the data into two sets:

1. The training set, used to train (i.e. build) the model;
2. and the testing set (or validation set), used to test (i.e. validate) the model by estimating the prediction error.

Cross-validation is also known as a *resampling method* because it involves fitting the same statistical method multiple times using different subsets of the data.

In this chapter, you'll learn:

1. the most commonly used statistical metrics (Chapter @ref(regression-model-accuracy-metrics)) for measuring the performance of a regression model in predicting the outcome of new test data.
2. The different cross-validation methods for assessing model performance. We cover the following approaches:
   - Validation set approach (or data split)
   - Leave One Out Cross Validation
   - k-fold Cross Validation
   - Repeated k-fold Cross Validation

Each of these methods has their advantages and drawbacks. Use the method that best suits your problem. Generally, the (repeated) k-fold cross validation is recommended.

3. Practical examples of R codes for computing cross-validation methods.

Contents:

Practical Guide in R

Loading required R packages

- `tidyverse` for easy data manipulation and visualization
- `caret` for easily computing cross-validation methods

```
library(tidyverse)
library(caret)
```

Example of data

We'll use the built-in R `swiss` data, introduced in the Chapter @ref(regression-analysis), for predicting fertility score on the basis of socio-economic indicators.

```
# Load the data
data("swiss")
# Inspect the data
sample_n(swiss, 3)
```

Model performance metrics

After building a model, we are interested in determining the accuracy of this model on predicting the outcome for new unseen observations not used to build the model. Put in other words, we want to estimate the prediction error.

To do so, the basic strategy is to:

1. Build the model on a training data set
2. Apply the model on a new test data set to make predictions
3. Compute the prediction errors

In Chapter @ref(regression-model-accuracy-metrics), we described several statistical metrics for quantifying the overall quality of regression models. These include:

- **R-squared** (R2), representing the squared correlation between the observed outcome values and the predicted values by the model. The higher the adjusted R2, the better the model.
- **Root Mean Squared Error** (RMSE), which measures the average prediction error made by the model in predicting the outcome for an observation. That is, the average difference between the observed known outcome values and the values predicted by the model. The lower the RMSE, the better the model.
- **Mean Absolute Error** (MAE), an alternative to the RMSE that is less sensitive to outliers. It corresponds to the average absolute difference between observed and predicted outcomes. The lower the MAE, the better the model

In classification setting, the prediction error rate is estimated as the proportion of misclassified observations.

R2, RMSE and MAE are used to measure the regression model performance during **cross-validation**.

In the following section, we'll explain the basics of cross-validation, and we'll provide practical example using mainly the `caret` R package.

Cross-validation methods

Briefly, cross-validation algorithms can be summarized as follow:

1. Reserve a small sample of the data set
2. Build (or train) the model using the remaining part of the data set
3. Test the effectiveness of the model on the the reserved sample of the data set. If the model works well on the test data set, then it's good.

The following sections describe the different cross-validation techniques.

The Validation set Approach

The validation set approach consists of randomly splitting the data into two sets: one set is used to train the model and the remaining other set sis used to test the model.

The process works as follow:

1. Build (train) the model on the training data set
2. Apply the model to the test data set to predict the outcome of new unseen observations
3. Quantify the prediction error as the mean squared difference between the observed and the predicted outcome values.

The example below splits the `swiss` data set so that 80% is used for training a linear regression model and 20% is used to evaluate the model performance.

```
# Split the data into training and test set
set.seed(123)
training.samples <- swiss$Fertility %>%
  createDataPartition(p = 0.8, list = FALSE)
train.data  <- swiss[training.samples, ]
test.data <- swiss[-training.samples, ]
# Build the model
model <- lm(Fertility ~., data = train.data)
# Make predictions and compute the R2, RMSE and MAE
predictions <- model %>% predict(test.data)
data.frame( R2 = R2(predictions, test.data$Fertility),
            RMSE = RMSE(predictions, test.data$Fertility),
            MAE = MAE(predictions, test.data$Fertility))
```

```
##     R2 RMSE  MAE
## 1 0.39 9.11 7.48
```

When comparing two models, the one that produces the lowest test sample RMSE is the preferred model.

the RMSE and the MAE are measured in the same scale as the outcome variable. Dividing the RMSE by the average value of the outcome variable will give you the prediction error rate, which should be as small as possible:

```
RMSE(predictions, test.data$Fertility)/mean(test.data$Fertility)
```

```
## [1] 0.128
```

Note that, the validation set method is only useful when you have a large data set that can be partitioned. A disadvantage is that we build a model on a fraction of the data set only, possibly leaving out some interesting information about data, leading to higher bias. Therefore, the test error rate can be highly variable, depending on which observations are included in the training set and which observations are included in the validation set.

Leave one out cross validation - LOOCV

This method works as follow:

1. Leave out one data point and build the model on the rest of the data set
2. Test the model against the data point that is left out at step 1 and record the test error associated with the prediction
3. Repeat the process for all data points
4. Compute the overall prediction error by taking the average of all these test error estimates recorded at step 2.

Practical example in R using the `caret` package:

```
# Define training control
train.control <- trainControl(method = "LOOCV")
# Train the model
model <- train(Fertility ~., data = swiss, method = "lm",
               trControl = train.control)
# Summarize the results
print(model)
```

```
## Linear Regression
##
## 47 samples
##  5 predictor
##
## No pre-processing
## Resampling: Leave-One-Out Cross-Validation
## Summary of sample sizes: 46, 46, 46, 46, 46, 46, ...
## Resampling results:
##
##   RMSE  Rsquared  MAE
##   7.74  0.613     6.12
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

The advantage of the LOOCV method is that we make use all data points reducing potential bias.

However, the process is repeated as many times as there are data points, resulting to a higher execution time when n is extremely large.

Additionally, we test the model performance against one data point at each iteration. This might result to higher variation in the prediction error, if some data points are outliers. So, we need a good ratio of testing data points, a solution provided by the **k-fold cross-validation method**.

K-fold cross-validation

The k-fold cross-validation method evaluates the model performance on different subset of the training data and then calculate the average prediction error rate. The algorithm is as follow:

1. Randomly split the data set into k-subsets (or k-fold) (for example 5 subsets)
2. Reserve one subset and train the model on all other subsets
3. Test the model on the reserved subset and record the prediction error
4. Repeat this process until each of the k subsets has served as the test set.
5. Compute the average of the k recorded errors. This is called the cross-validation error serving as the performance metric for the model.

K-fold cross-validation (CV) is a robust method for estimating the accuracy of a model.

The most obvious advantage of k-fold CV compared to LOOCV is computational. A less obvious but potentially more important advantage of k-fold CV is that it often gives more accurate estimates of the test error rate than does LOOCV (James et al. 2014).

Typical question, is how to choose right value of k?

Lower value of K is more biased and hence undesirable. On the other hand, higher value of K is less biased, but can suffer from large variability. It is not hard to see that a smaller value of k (say k = 2) always takes us towards validation set approach, whereas a higher value of k (say k = number of data points) leads us to LOOCV approach.

> In practice, one typically performs k-fold cross-validation using k = 5 or k = 10, as these values have been shown empirically to yield test error rate estimates that suffer neither from excessively high bias nor from very high variance.

The following example uses 10-fold cross validation to estimate the prediction error. Make sure to set seed for reproducibility.

```
# Define training control
set.seed(123)
train.control <- trainControl(method = "cv", number = 10)
# Train the model
model <- train(Fertility ~., data = swiss, method = "lm",
               trControl = train.control)
# Summarize the results
print(model)
```

```
## Linear Regression
##
## 47 samples
##  5 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 43, 42, 42, 41, 43, 41, ...
## Resampling results:
##
##   RMSE  Rsquared  MAE
##   7.38  0.751     6.03
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

Repeated K-fold cross-validation

The process of splitting the data into k-folds can be repeated a number of times, this is called repeated k-fold cross validation.

The final model error is taken as the mean error from the number of repeats.

The following example uses 10-fold cross validation with 3 repeats:

```
# Define training control
set.seed(123)
train.control <- trainControl(method = "repeatedcv",
                              number = 10, repeats = 3)
# Train the model
model <- train(Fertility ~., data = swiss, method = "lm",
               trControl = train.control)
# Summarize the results
print(model)
```

Discussion

In this chapter, we described 4 different methods for assessing the performance of a model on unseen test data.

These methods include: validation set approach, leave-one-out cross-validation, k-fold cross-validation and repeated k-fold cross-validation.

We generally recommend the (repeated) k-fold cross-validation to estimate the prediction error rate. It can be used in regression and classification settings.

Another alternative to cross-validation is the bootstrap resampling methods (Chapter @ref(bootstrap-resampling)), which consists of repeatedly and randomly selecting a sample of n observations from the original data set, and to evaluate the model performance on each copy.

References

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2014. *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated.

Last update : 01/04/2018