Training, Cross-Validation, and Testing Mechanism

Cheng Peng

STA551 Foundations of Data Science

Contents

1 Introduction

In classical statistical modeling, we select candidate models based on exploratory data analysis and visual analysis. The candidate models are then fit to the analytic data set. Since all models have some sort of assumptions (think about linear regression and logistic and Poisson regression models), and then carry out model diagnostic analyses to identify potential violations of the model assumption. Of cause, we assume that the data represent the population. This modeling process is depicted in the following diagram.

Recall the process of the following two representative statistical models

1.1 Linear Regression Model

First, we look at the process of how to build a linear regression model for a data set.

- **Explicit and Implicit Assumptions**. The following are some of the assumptions about a normal-based linear regression model:

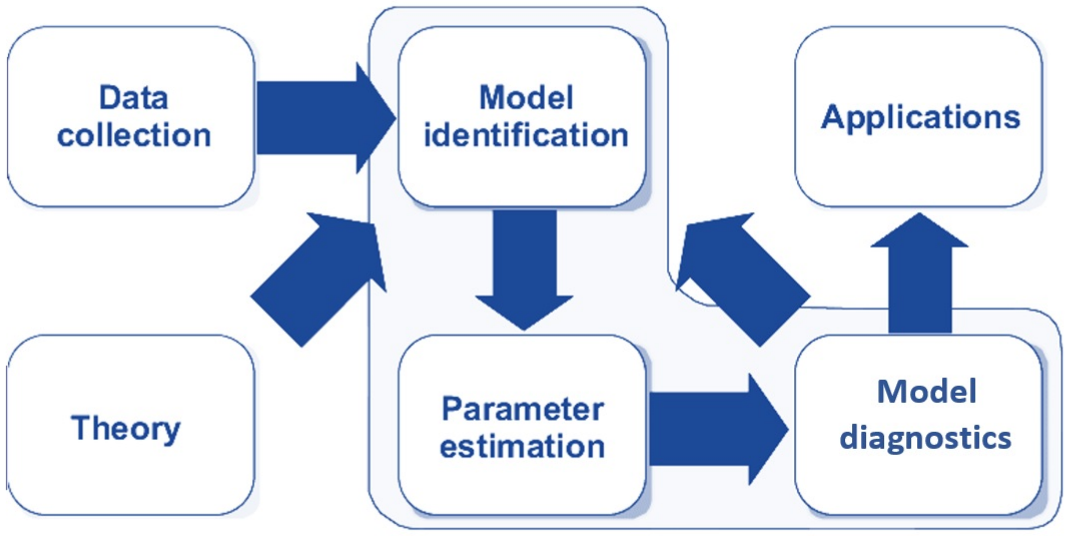


Figure 1: Figure 1. Statistical modeling process.

- Predictor (feature) variables are (linearly or curve-linearly) correlated to the response variable (also called label in machine learning terms);
- Predictor variables themselves are NOT highly linearly correlated (i.e., no serious collinearity);
- The response variable is normally distributed;
- The variance of the response variable is constant;
- There are no outlier (influential) observations;
- information on all relevant predictor variables is available in the data (i.e., no important predictor is missing);

- **Model Fitting (Parameter Estimating)**. The least-square estimator (LSE), which is equivalent to the maximum likelihood estimator (MLE) when the response variable is assumed to be a normal distribution, can be used to estimate the regression coefficients.

- **Model Selection and Diagnostics**. Since several implicit and explicit assumptions have been assumed underlying the linear regression, different sets of diagnostic measures were developed to detect different potential violations of the model.
  - *global goodness-of-fit*: R2, AIC and SBC (requires normality assumption of the response variable), MSE, etc.
  - *local goodness-of-fit/model selection*: R, F-test (need normality and equal variance of the response variable), t-test, likelihood ratio test, Mallow's Cp, etc.
  - *normality*: QQ-plot and probability plot for a visual check, goodness-of-fit tests such as Anderson-Darling, Cramer-von Miss, Kolmogorov-Smirnov, Shapiro-Wilks, and Pearson's chi-square tests, etc.
  - *detecting outliers/influential observations*: leverage point-hat matrix, DFITT - defined based on leave-one-out resampling method, cook's distance, (scaled) residual plots, etc.
  - *verifying constant variance*: F-test (requires normality), Brown-Forsythe test (nonparametric), Breusch-Pagan Test (also nonparametric), Bartlett's Test (requires normality), etc.
  - *detecting collinearity*: Variance inflation factor (VIF) for data-based and structural collinearity.
  - *detecting mission of determinant variable*:

1.2 Logistic Regression Model

For the binary logistic model, we also follow the same steps to identify the final model. For example, the well-known binary logistic regression modeling follows similar steps:

- **Assumptions**: Unlike the linear regression model which has a strong assumption of normality, the logistic regression model assumes the following
  - *binomial distribution*: The dependent variable is binary.
  - *independence*: The logistic regression requires the observations to be independent of each other.
  - *collinearity*: The logistic regression requires there to be little or no multicollinearity among the independent variables.
  - *linearity*: The logistic regression assumes linearity of independent variables and the log odds of the event of interest.
  - *large sample size*: The logistic regression typically requires a large sample size. A general guideline is that you need a minimum of 10 observations with the least frequent outcome for each independent variable in your model.
  - *mis-specification*: No important variables are omitted. No extraneous variables are included.
  - *measurement error*: The independent variables are measured without error.
- **Model Fitting**: The coefficients of the logistic regression model are estimated using a maximum likelihood estimator. Note LSE cannot be estimated for the logistic regression model.
- **Model Selection and Diagnostics**: Unlike the normal linear regression model, there are a few diagnostic methods one can use in logistic regression models.
  - *misspecification*: link test (large sample test);
  - *goodness-of-fit*: log-likelihood chi-square and pseudo-R-square; Hosmer-Lemeshow's lack of fit test AIC and SBC.
  - *multi-collinearity*: VIF
  - *influential points*: Cook's distance, DBETA, deviance residuals

1.3 Inference about Association and Prediction

In classical statistics, most problems are related to the significance of the regression coefficients. The p-values associated with corresponding regression coefficients are calculated based on certain assumptions. Prediction intervals are also constructed based on certain assumptions.

We can also use resampling methods to relax the assumptions about the distribution of the response variable to make inferences about the regression coefficients and construct prediction intervals.

In practice, the prediction problems usually involve both classical statistical models and machine learning algorithms. The model selection process and performance evaluation in classical statistical modeling is dependent on model assumptions, while in machine learning algorithms, model selection and evaluation use data-driven methods that are also valid for modern statistical modeling.

In the next few sections, we will introduce the data-driven methods for machine learning algorithms and statistical models.

2 Data Splitting Methods

When training and testing machine learning and statistical models without assuming strong assumptions (particularly the distributional information), we break down the data into three separate and distinct data sets: training data, validation data, and testing data. The basic idea is depicted in the following chart.

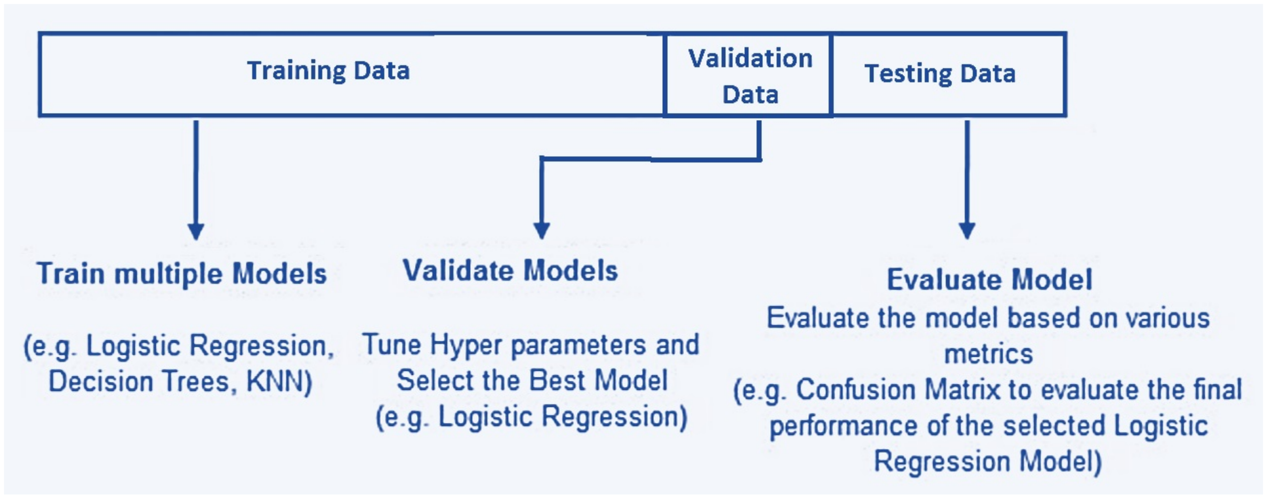

Training Data
Validation Data
Testing Data
Train multiple Models
(e.g. Logistic Regression, Decision Trees, KNN)
Validate Models
Tune Hyper parameters and Select the Best Model
(e.g. Logistic Regression)
Evaluate Model
Evaluate the model based on various metrics
(e.g. Confusion Matrix to evaluate the final performance of the selected Logistic Regression Model)

Figure 2: Figure 2. The-way data splitting method.

2.1 Training Data

The training data set is the sample of data used to fit the model. In other words, the training data teaches the model how it's supposed to learn and think. To do this, training data is often presented in pairs: predictor variables (feature variables) and a response variable (also called a label).

Training data is the first set of data the model/algorithm is exposed to. During each stage of training, the model will be fit to the training data and estimate the parameters (also called weight in some machine learning algorithms such as neural networks).

Because the training data set is used to estimate the parameters (i.e., teaching the algorithm), it requires a certain amount of information to make the algorithms and models reliable. It makes up between 60% and 80% of the total data.

2.2 Validation Data

The validation data set is a sample of data held back from training the model. This data set provides an unbiased evaluation of a model fit on the training data set while tuning model hyperparameters. In more basic terms, validation data is an unused portion of your training data and helps determine if the initial model is accurate.

A model **hyperparameter** is a configuration that is external to the model and whose value cannot be estimated from data.

- They are often used in processes to help estimate model parameters.
- They are often specified by the practitioner.
- They can often be set using heuristics.
- They are often tuned for a given predictive modeling problem.

We cannot know the best value for a model hyperparameter on a given problem. We may use rules of thumb, copy values used on other problems, or search for the best value by trial and error.

When a machine learning algorithm is tuned for a specific problem, such as the cut-off probability determination in the logistic prediction, then we are tuning the hyperparameters of the model or order to discover the parameters of the model that result in the most skillful predictions. The validation data helps tune a machine-learning model while checking for overfitting.

Overfitting happens when the model/algorithm is too closely fitted to the training data — producing results tied to the specifics of that first data set. After validation, the team will often return to the training data and run it again, making adjustments to values and parameters to improve the model.

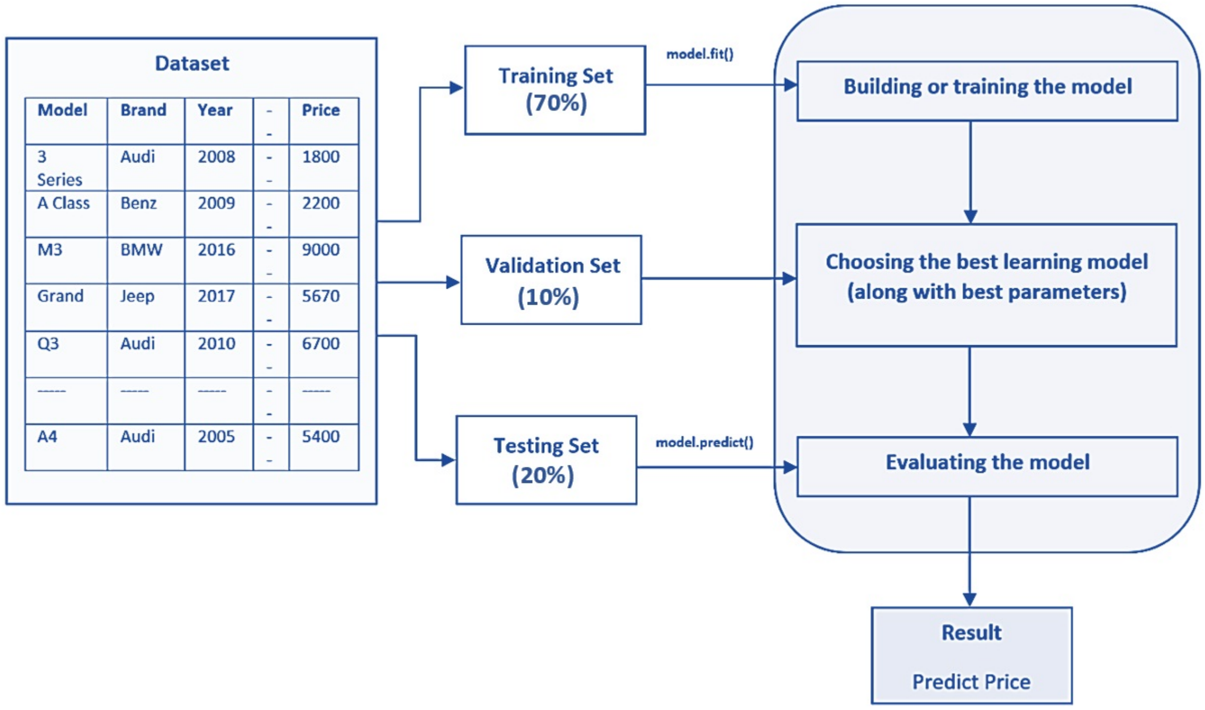
2.3 Test data

The test data set is a sample of data used to provide an unbiased evaluation of a final model fit on the training data set or to test the model. Put more simply, test data is a set of **unlabeled inputs** (i.e., the response value is removed from the data) that test whether the model is producing the correct outputs in the real world.

The key difference between a validation data set and a test data set is that the validation data set is used during model configuration, while the test data set is reserved to evaluate the final model.

Test data is about 20% of the total data and should be completely separate from the training data — which our model should know very well by this point.

In summary, the following chart gives an example of three-way splitting data with a sample configuration of the sub-set sizes.



Model	Brand	Year	--	Price
3 Series	Audi	2008	--	1800
A Class	Benz	2009	--	2200
M3	BMW	2016	--	9000
Grand	Jeep	2017	--	5670
Q3	Audi	2010	--	6700
-----	-----	-----	--	-----
A4	Audi	2005	--	5400

Figure 3: Figure 3. The-way data splitting method with an example of sub-sample sizes.

3 Cross-validation

Partitioning the available data into three sets drastically reduces the number of samples that can be used for training the model, and the results can depend on a particular random choice for the pair of (train, validation) sets. To address this reliability and stability, a solution to this problem is a procedure called cross-validation (CV for short). A test set should still be held out for final evaluation, but the validation set is no longer needed when doing a CV.

3.1 K-fold Cross-validation

Many different cross-validation methods are defined based on the following basic k-fold CV, the **training set** is split into k smaller sets (other approaches are described below, but generally follow the same principles). The following is the procedure that uses 5 “folds”:

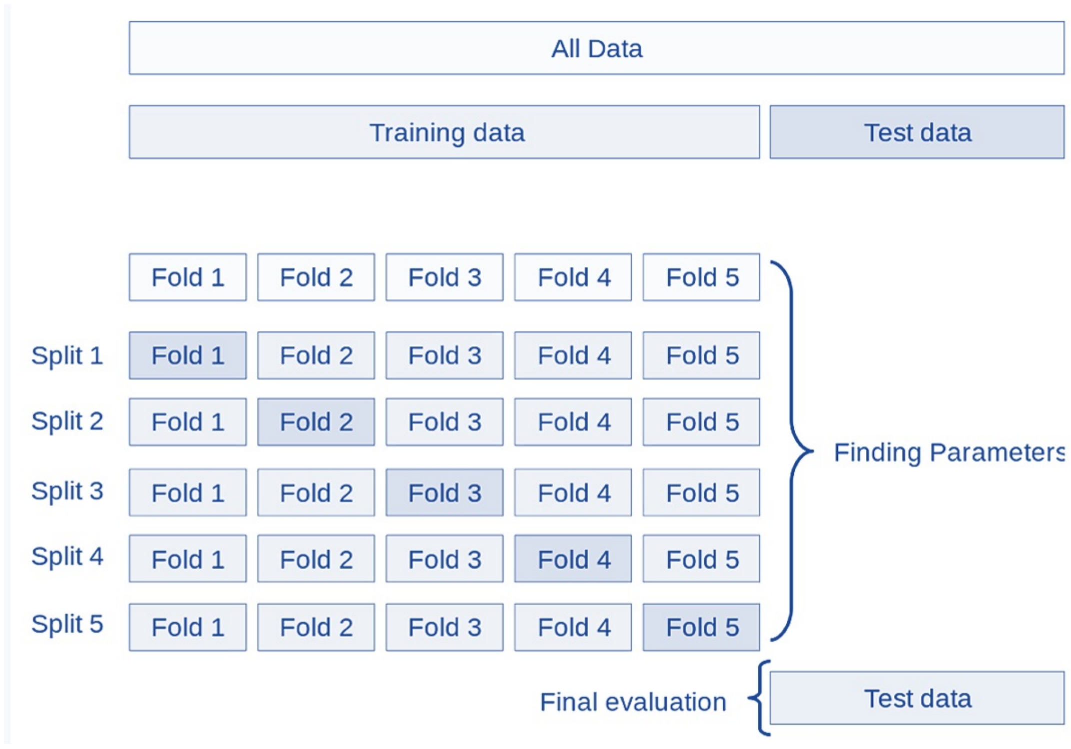


Figure 4: Figure 4. Demonstration of a five-fold cross-validation.

As an example, we explain the five-fold cross-validation for determining the optional cut-off probability in logistic prediction problems. Without loss of generality, we consider possible cut-off probabilities. For each cut-off probability, we calculate the prediction accuracy of the model based on the confusion matrix using the one-fold of validation data (see the following figure).

With the above process of calculation of the confusion matrix and the accuracy metrics based on the first iteration of the 5-fold CV with a given cut-off probability, we next will explain the logic of finding the optimal cut-off probability (also called hyperparameter) based on a set of given cut-off probabilities.

From the above flow chart, we see that the performance measure reported by 5-fold cross-validation is the average of the values of accuracy computed in the loop. This approach can be computationally expensive but does not waste too much data (as is the case when fixing an arbitrary validation set), which is a major advantage in problems such as inverse inference where the number of samples is very small.

3.2 Other Cross-Validation Methods

There are many other cross-validation methods defined using the same logic in k-fold cross-validation. We will not detail these CV methods. Instead, we describe some of them that are occasionally used in practice.

- **Leave-one-out (LOO)** cross-validation is a simple cross-validation. Each training set is created by taking all the records (also called **samples** in machine learning) except one, the validating set being the record (also called **sample** in machine learning) left out. Therefore, this cross-validation procedure does not waste much data as only one sample is removed from the training set.

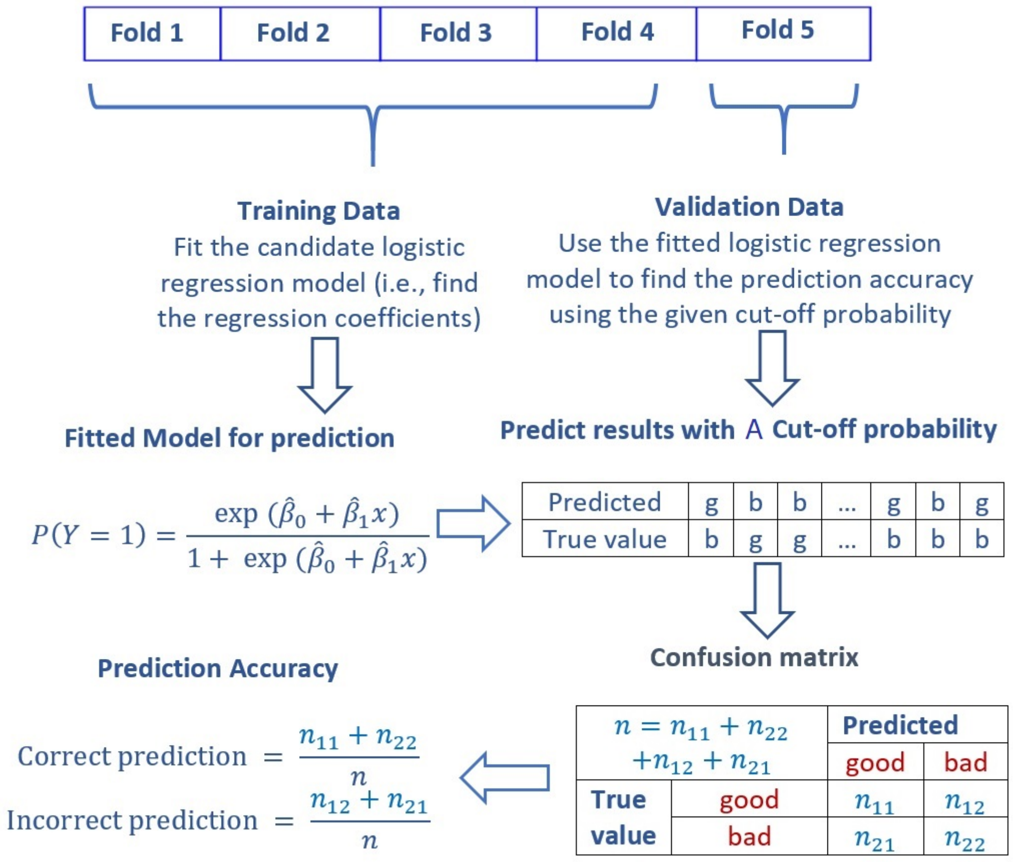

Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Training Data
Fit the candidate logistic regression model (i.e., find the regression coefficients)
Validation Data
Use the fitted logistic regression model to find the prediction accuracy using the given cut-off probability
Fitted Model for prediction
Predict results with A Cut-off probability
$P(Y=1) = \frac{\exp(\hat{\beta}_0 + \hat{\beta}_1 x)}{1 + \exp(\hat{\beta}_0 + \hat{\beta}_1 x)}$
Predicted g b b ... g b g
True value b g g ... b b b
Confusion matrix
Prediction Accuracy
Correct prediction $= \frac{n_{11}+n_{22}}{n}$
Incorrect prediction $= \frac{n_{12}+n_{21}}{n}$
$n = n_{11}+n_{22} + n_{12}+n_{21}$
Predicted
good
bad
True value
good
bad
n_{11}
n_{12}
n_{21}
n_{22}

Figure 5: Figure 5. Confusion matrix and accuracy metrics.

- **Leave-P-Out** is very similar to Leave-One-Out as it creates all the possible training/test sets by removing samples (i.e., records) from the complete set. Unlike Leave-One-Out and K-Fold, the test sets will overlap for $p > 1$.
- **Stratified K-Fold** is a variation of k-fold that returns stratified folds: each set contains approximately the same percentage of samples of each target class as the complete set.
- **Leave-One-Group-Out** is a cross-validation scheme that holds out the samples according to a third-party-provided array of integer groups. This group information can be used to encode arbitrary domain-specific pre-defined cross-validation folds. Each training set is thus constituted by all the samples except the ones related to a specific group.

4 Case Study Using Fraud Data

In this section, we use the fraud data at https://pengdsci.github.io/datasets/FraudIndex/fraudidx.csv. The data set has only two variables. The response variable is a binary fraud status variable with character values “good” and “bad”. The predictor variable is an index variable. The sample size is 33236. The fraud proportion is about 23.34%. The objective is to build a logistic regression model to predict fraud for future transactions.

4.1 Data Partition

Since the same size is large, we split the sample by 70%:30% with 70% data for training and validating models and 30% for testing purposes. The labels (value of the fraud status) of testing and validation data will be

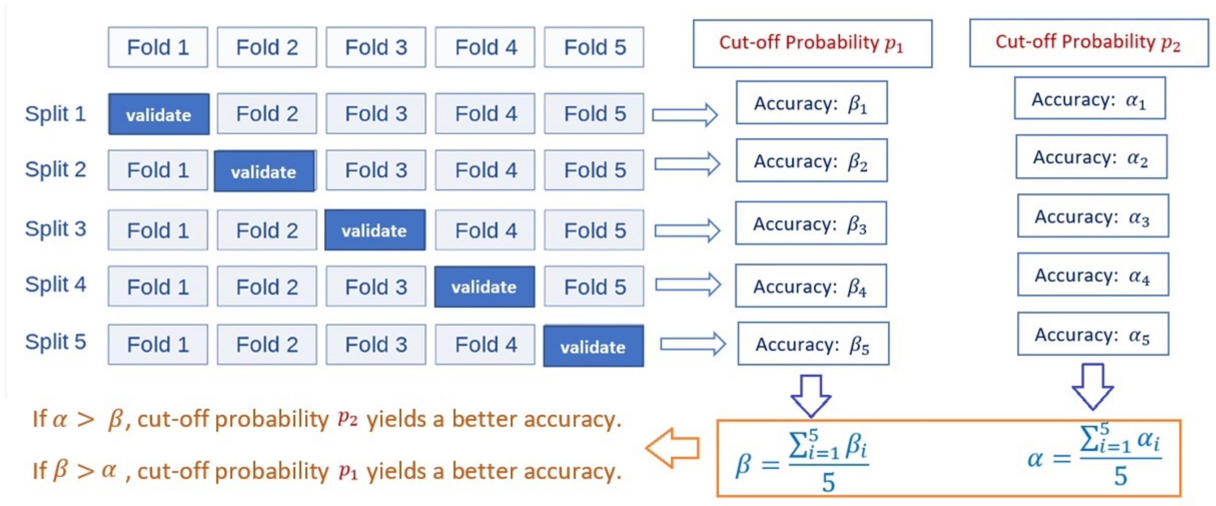

Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Cut-off Probability p_1
Cut-off Probability p_2
Split 1
Split 2
Split 3
Split 4
Split 5
validate
Accuracy: β_1
Accuracy: β_2
Accuracy: β_3
Accuracy: β_4
Accuracy: β_5
Accuracy: α_1
Accuracy: α_2
Accuracy: α_3
Accuracy: α_4
Accuracy: α_5
If $\alpha > \beta$, cut-off probability p_2 yields a better accuracy.
If $\beta > \alpha$, cut-off probability p_1 yields a better accuracy.
$\beta = \frac{\sum_{i=1}^5 \beta_i}{5}$
$\alpha = \frac{\sum_{i=1}^5 \alpha_i}{5}$

Figure 6: Figure 6. The pseudo-program of 5-fold CV for tuning hyperparameter - cut-off probability.

removed when calculating the accuracy measures.

```
fraud.data = read.csv("https://pengdsci.github.io/datasets/FraudIndex/fraudidx.csv")[,-1]
## recode status variable: bad = 1 and good = 0
good.id = which(fraud.data$status == " good")
bad.id = which(fraud.data$status == "fraud")
##
fraud.data$fraud.status = 0
fraud.data$fraud.status[bad.id] = 1
nn = dim(fraud.data)[1]
train.id = sample(1:nn, round(nn*0.7), replace = FALSE)
training = fraud.data[train.id,]
testing = fraud.data[-train.id,]
```

4.2 Finding Optimal Cut-off Probability

We define a sequence of 20 candidate cut-off probabilities and then use 5-fold cross-validation to identify the optimal cut-off probability for the final detection model.

```
n0 = dim(training)[1]/5
cut.0ff.prob = seq(0,1, length = 22)[-c(1,22)]        # candidate cut off prob
pred.accuracy = matrix(0,ncol=20, nrow=5, byrow = T)  # null vector for storing prediction accuracy
## 5-fold CV
for (i in 1:5){
  valid.id = ((i-1)*n0 + 1):(i*n0)
  valid.data = training[valid.id,]
  train.data = training[-valid.id,]
  train.model = glm(fraud.status ~ index, family = binomial(link = logit), data = train.data)
  newdata = data.frame(index= valid.data$index)
  pred.prob = predict.glm(train.model, newdata, type = "response")
  # define confusion matrix and accuracy
  for(j in 1:20){
    pred.status = rep(0,length(pred.prob))
    valid.data$pred.status = as.numeric(pred.prob >cut.0ff.prob[j])
    a11 = sum(valid.data$pred.status == valid.data$fraud.status)
    pred.accuracy[i,j] = a11/length(pred.prob)
  }
}
###
avg.accuracy = apply(pred.accuracy, 2, mean)
max.id = which(avg.accuracy ==max(avg.accuracy ))
### visual representation
tick.label = as.character(round(cut.Off.prob,2))
plot(1:20, avg.accuracy, type = "b",
     xlim=c(1,20),
     ylim=c(0.5,1),
     axes = FALSE,
     xlab = "Cut-off Probability",
     ylab = "Accuracy",
     main = "5-fold CV performance"
     )
axis(1, at=1:20, label = tick.label, las = 2)
axis(2)
segments(max.id, 0.5, max.id, avg.accuracy[max.id], col = "red")
text(max.id, avg.accuracy[max.id]+0.03, as.character(round(avg.accuracy[max.id],4)), col = "red", cex =
```

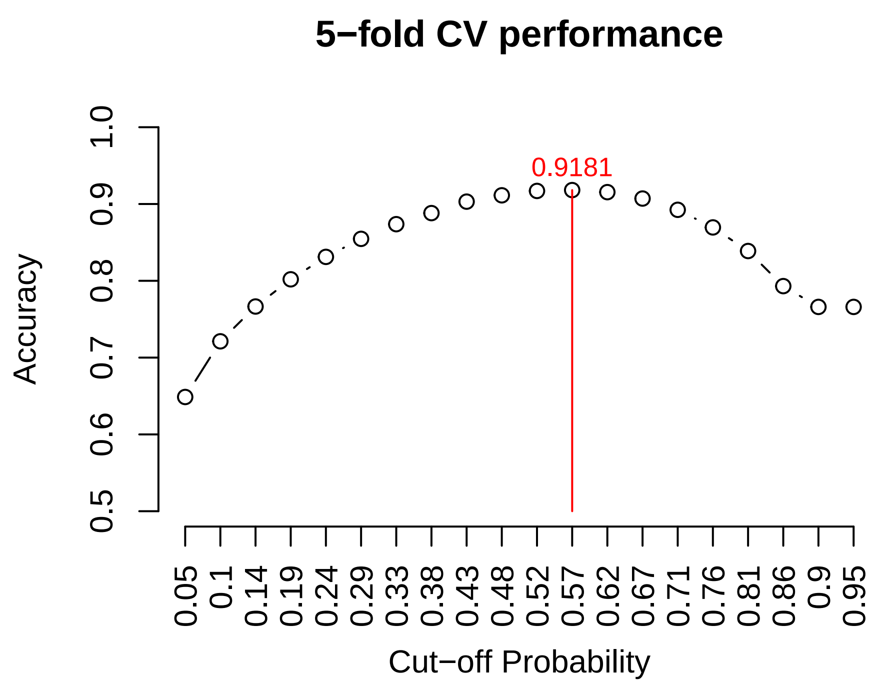


Figure 7: Figure 7. 5-fold CV performance plot

The above figure indicates that the optimal cut-off probability that yields the best accuracy is 0.57.

4.3 Reporting Test

This subsection reports the performance of the model using the test data set. Note that the model needs to be fit to the original training data to find the regression coefficients and then use the holdout testing sample to find the accuracy.

```r
test.model = glm(fraud.status ~ index, family = binomial(link = logit), data = training)
newdata = data.frame(index= testing$index)
pred.prob.test = predict.glm(test.model, newdata, type = "response")
testing$test.status = as.numeric(pred.prob.test > 0.57)
a11 = sum(testing$test.status == testing$fraud.status)
test.accuracy = a11/length(pred.prob.test)
kable(as.data.frame(test.accuracy), align='c')
```

test.accuracy
0.9221743

The accuracy is 92.18%. This accuracy indicates that there is no under-fitting. In models and algorithms with multiple feature variables, CV can help detect (and, hence, avoid)over and underfitting.

5 Concluding Remarks

We have used 5-fold cross-validation to find the hyperparameter (optimal cut-off probability). This is only one application for estimating a hyperparameter. In practice, we can use cross-validation to select the best models in different model families. For example, we have candidate models such as decision tree-based models, support vector machines, neural networks, and logistic regression models. Cross-validation is a powerful tool for selecting the best model.