ACTL4305/5305 Actuarial Data Analytic Application

Week 4: Questions

Model Selection and Assessment

Learning Objectives

- Understand how to build models for insurance premiums: a linear model without any penalty, a linear model with the Lasso penalty, and a generalized linear model with log link function.
- Compare the three models using various statistical techniques, including AIC and BIC, validation set, cross-validation, and bootstrap.

1 Introduction: Health Insurance Premium Modelling

Assume you are a junior data analyst in a consulting company. One day, your supervisor sends you a data set about premiums collected from a health insurance company. This data set contains important information about policyholders, such as age, sex, body mass index (BMI), etc. Your job is to understand how this insurance company calculates premiums based on policyholders' characteristics.

Questions for Lab 4:

- Use the code below to build a linear model without any penalty, a linear model with the Lasso penalty, and a generalized linear model with log link function. The code for model building is already provided.

- Compare the three models based on AIC and BIC, validation set approach, cross-validation approach, and bootstrap approach.

- Make a conclusion: which model is the best?

2 Model Details

2.1 Linear Model

$$E(y_i) = \beta x_i, \tag{1}$$

where y_i is the premium of i-th observation, x_i represents the vector of independent variables, and β is the corresponding coefficient. The disadvantage of the linear model is that the predicted premiums may take negative values.

2.2 Linear Model with Lasso Penalty

We need to add Lasso penalty to shrink β in (1).

2.3 Generalized Linear Model with Log Link Function

The advantage of the generalized linear model with a log link function is that the predicted premiums are always positive.

$$E(y_i) = \exp(\beta x_i), \tag{2}$$

3 Model Assessment Methods

3.1 AIC

Akaike information criterion (`AIC`) is a model selection technique based on in-sample fit to estimate the likelihood of a model to predict the future values. The idea of `AIC` is to penalize the objective function if extra variables without strong predictive abilities are included in the model. A good model is the one that has minimum `AIC` among all the other models.

$$\text{AIC} = -2 \times \ln(L) + 2 \times k, \tag{3}$$

where L is the value of the likelihood, N is the number of recorded measurements, and k is the number of estimated parameters. The `AIC` statistic penalizes complex models less, meaning that it may put more emphasis on model performance on the training dataset, and, in turn, select more complex models. We can see that the penalty for `AIC` is less than for `BIC` in Equation (4). This causes `AIC` to pick more complex models.

3.2 BIC

Bayesian information criterion (`BIC`) is another criteria for model selection that measures the trade-off between model fit and complexity of the model. A lower `BIC` value indicates a better fit.

$$\text{BIC} = -2 \times \ln(L) + \ln(N) \times k. \tag{4}$$

The `BIC` penalizes the model more for its complexity, meaning that more complex models will have a worse (larger) score and will, in turn, be less likely to be selected. `AIC` and `BIC` are both statistical approaches for estimating how well a given model fits a dataset and how complex the model is.

Both `AIC` and `BIC` can not be calculated directly from the `glmnet` package. However, we can estimate them using Equations (4) and (3). To be able to implement these formula in `R`, we make an assumption that the residual errors are normally distributed and this gives us Equations (5) and (6) for calculating `AIC` and `BIC`, respectively:

$$\text{AIC} = N \times \ln(\hat{\sigma}^2) + 2 \times k. \tag{5}$$

$$\text{BIC} = N \times \ln(\hat{\sigma}^2) + \ln(N) \times k. \tag{6}$$

where $\hat{\sigma}^2$ is the variance estimate of the error associated with each response measurement. See more details at proof and reference.

3.3 Validation Set Approach

We want to estimate the test error associated with fitting a particular statistical learning method on a set of observations. The data set is randomly split into two sets, called the training set and the testing set or or hold-out set or validation set. The model is fitted using the training set only. Then the model is asked to predict the output values for the unseen data in the testing set. The validation set approach can be summarised in the following steps:

- Fit the model on the training set.
- Use the resulting fitted model to predict the responses for the observations in the validation set.
- The resulting validation set error rate is typically assessed using the mean square error (MSE) in the case of a quantitative response. The mean squared error is given by:

$$\text{MSE} = \frac{1}{N}\sum_{i}^{N}(y_i - \hat{y}_i)^2, \tag{7}$$

where y_i is the observed and $\hat{y}_i$ is the predicted values from the validation set.

However, its evaluation can have a high variance. The evaluation may depend heavily on which data points end up in the training set and which end up in the test set, and thus the evaluation may be significantly different depending on how the division is made. Furthermore, only a subset of the observations are used to fit the model. In this tutorial, the training data set is randomly split in 70% and 30% ratios for model fitting and testing respectively.

3.4 Cross-validation (CV) Approach

CV can be used to estimate the test error associated with a statistical learning method to evaluate its performance.

- This approach involves randomly dividing the set of observations into k groups, or folds, of approximately equal size.
- The mean squared error, MSE_1, is then computed on the observations in the held-out fold using Equation (7).
- This procedure is repeated k times. Each time, a different group of observations is treated as a validation set. This process results in k estimates of the test error $\text{MSE}_1, \text{MSE}_2, \ldots, \text{MSE}_k$.
- The k-fold cross-validation (CV) estimate is computed by averaging these values:

$$\text{CV}_k = \frac{1}{k}\sum_{i=1}^{k}\text{MSE}_i. \tag{8}$$

The advantage of this method is that it matters less how the data gets divided. Every data point gets to be in a test set exactly once and gets to be in a training set $k-1$ times. The variance of the resulting estimate increases as k increase, but bias decreases.

The disadvantage of this method is that the training algorithm has to be rerun from scratch k times, which means it takes k times as much computation to make an evaluation. A variant of this method is to randomly divide the data into a test and training set k different times. The advantage of doing this is that you can independently choose how large each test set is and how many trials you average over.

3.5 Bootstrap Approach

For each observation, we only keep track of predictions from bootstrap samples not containing that observation. The *leave-one-out bootstrap* estimation of prediction error is

$$\hat{\text{Err}}^{(1)} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{|C^{-i}|}\sum_{b\in C^{-i}} L(y_i, \hat{f}^{*b}(x_i)).$$

Here C^{-i} is the set of indices of the bootstrap samples b that do not contain observation i, and $|C^{-i}|$ is the number of such samples.

In summary, the models are fit using the training data sets in Figure 1 and predict out-of-bag samples (test set).

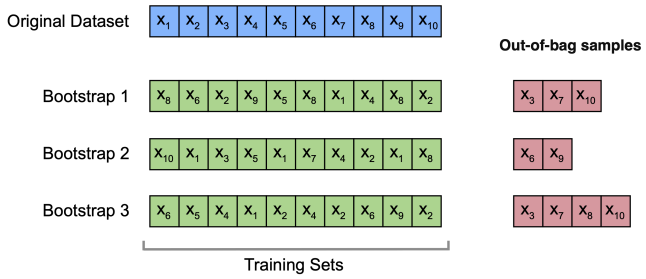

Figure 1: Bootstrap Sampling

4 Emperical Study

We first load the data and build three target models.

```
Data <- read.csv("insurance.csv")
#Divide data into X and Y
Data_X <- Data[,-7]
Data_Y <- Data[,7]

X_matrix <- model.matrix(~., Data_X)
Y_matrix <- as.matrix(Data_Y)

#Linear model
#model_linear<- lm(charges~., data=Data)

#Lasso-linear model
#set.seed(2023)
#CV_lasso<-cv.glmnet(X_matrix, Y_matrix, family="gaussian",
#                    alpha = 1, nfolds = 10)
#model_lasso <- glmnet(X_matrix,  Y_matrix,  lambda = CV_lasso$lambda.min, alpha = 1)

#Generalized linear model with log link
#model_loglinear<- glm(charges~., data=Data, family = gaussian(link="log") )
```