DATA 221

Feature Selection in Machine Learning

2023-02-02

Model Selection

We might also want to compare multiple models to each other. In this case, we can use various measures of model fit.

- r^2
- Adjusted r^2
- Akaike Information Criterion (AIC)
- Bayesian Information Criterion (BIC)

r^2

- You may have seen r^2 in another class.
- The formula for r^2 is:

$$r^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} = 1 - \frac{SS_{error}}{SS_{total}}$$

- r^2 has the benefit of having a nice interpretation–it is the proportion of variation in the response variable that can be explained by the model.
- r^2 **always falls between 0 and 1, with values closer to 1 being more desirable.**

Adjusted r^2

- A disadvantage of r^2 (and in statistics, we would consider it a fairly large disadvantage) is that it always increases when you add a predictor to your model, even if the predictor doesn't make any sense in the context of your data.
- To counter this disadvantage, we can use something called adjusted r^2–the adjustment is tied to the number of parameters in your model.
- The formula for adjusted r^2 is:

$$r^2_{adj.} = 1 - \frac{SS_{error}/(n-p)}{SS_{total}/(n-1)}$$

- **Adjusted r^2 always falls between 0 and 1, with values closer to 1 being more desirable.**

AIC

- Unlike r^2 and adjusted r^2, the Akaike Information Criterion is related to the likelihood of the data (so you do have to assume a likelihood function).
- The equation for AIC is given by

$$\text{AIC} = 2k - 2\ln(\hat{L}).$$

- The $2k$ penalty discourages overfitting relative to maximizing "goodness of fit" as measured by $\hat{L}$, the likelihood function of the data.
- This formula is derived from estimated Kullback-Leibler divergence (!!)–we can think of it as measuring information lost from a perfect model.
- **There is no "scale" for AIC! You have to compare it to the AIC for another model fit from the same data. Smaller values are more desirable.**

BIC

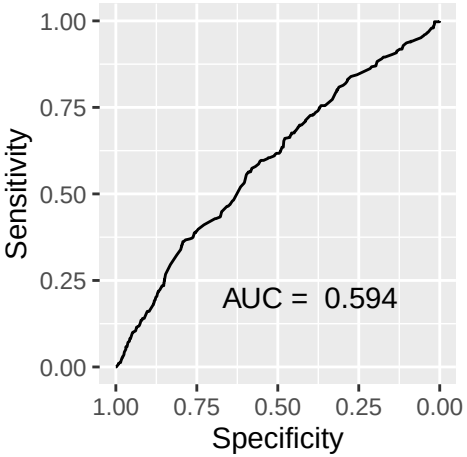
- The Bayesian Information Criterion (BIC) is related to AIC (a.k.a., also has a likelihood).
- The equation for BIC is given by

$$\text{BIC} = k \ln(n) - 2 \ln(\widehat{L}).$$

- Once again, there is a penalty that discourages overfitting relative to maximizing the likelihood.
- This formula is derived from the Bayesian belief that $p(\theta|\mathbf{x}) \propto L(\mathbf{x}|\theta)p(\theta)$.
- **There is no scale for BIC! You have to compare it to the BIC for another model fit from the same data. Smaller values are more desirable.**

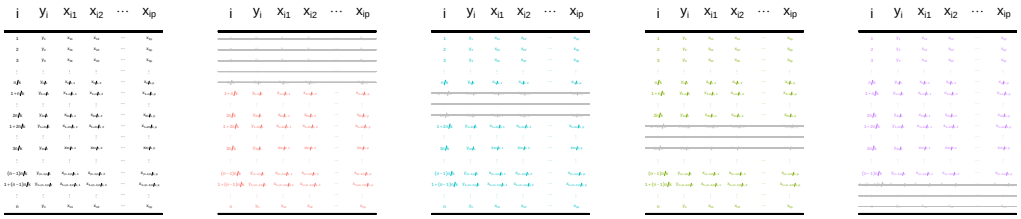
AUC

- AUC stands for **A**rea **U**nder the (receiver operating characteristic) **C**urve.
- The ROC curve plots sensitivity (true positive rate) against 1 - specificity (true negative rate) at various thresholds.
- **AUC always falls between 0 and 1, with values closer to 1 being more desirable.**
- “



Cross Validation

- Most of these model fit statistics are calculated using the training data.
- We are often more concerned with how the model might perform on a test dataset–in that case, we could use CV to estimate the testing error.



- **There is no scale for CV (of any kind). You have to compare it to the testing error for another model fit from the same data. Smaller values imply the predictions were close to the actual values, which is more desirable.**

Hypothesis Testing 1

Let's take a second to review hypothesis testing for a mean (using a frequentist perspective).

- We think that some value for a population parameter exists.
- We don't have any way to calculate the parameter, but we do have a nice randomly selected sample of the data.
- Our best guess for the parameter is some estimate/sample statistic.
- If we assume a particular value of the parameter, is our sample statistic far enough away to say that the true value of the parameter is significantly different from our assumption?

Hypothesis Testing 2

In the context of linear regression:

$$H_0 : \beta_p = 0$$
$$H_A : \beta_p \neq 0$$

Saying that the slope is equal to zero is equivalent to saying that there is no relationship between the predictor and response. Saying that the slope is equal to literally anything else implies that there is a linear relationship.

Hypothesis Testing 3

- Remember that b_p is a statistic that we use to estimate β_p–b_p would vary every time we took a new sample .
- The quantity that describes this variation is called the standard error.
- If we borrow some assumptions from the Central Limit Theorem, we know the sampling distribution for the slope:

$$b_p \sim N(\beta_p,\ \sigma_{b_p})$$

Hypothesis Testing 4

- We typically don't know the value of σ_{b_p}, so we estimate it instead.
- Then, the sampling distribution is

$$\frac{b_p - \beta_p}{SE_{b_p}} \sim t_{n-p}$$

- The quantity $t = \frac{b_p - \beta_p}{SE_{b_p}}$ is known as the test statistic.
- You can also find the probability of the test statistic occurring under the null hypothesis, also known as the p-value.

Hypothesis Testing 5

- If we observe a p-value of greater than 0.05 (or some other significance level), we would say that the slope is not significantly different from 0 (a.k.a., there is not enough evidence to say that there is a linear relationship between X and Y).
- If we observe a p-value of less than 0.05, we would say that the slope is significantly different from 0 (a.k.a., there is statistically significant evidence that there is a relationship between X and Y).

Hypothesis Testing 6

- You can use the same tools in a logistic regression!
- This time, the sampling distribution is:

$$\frac{b_p - \beta_p}{SE_{b_p}} \sim N(0, 1)$$

- Even though the distribution is slightly different, the same process (calculating the test statistic, p-value, comparing to the significance level) applies.

Resumes Dataset

Data from 4,870 fictitious resumes, including the following:

- `name`: factor indicating applicant's first name.
- `gender`: factor indicating gender.
- `ethnicity`: factor indicating ethnicity (i.e., Caucasian-sounding vs. African-American sounding first name).
- `quality`: factor indicating quality of resume.
- `jobs`: number of jobs listed on resume.
- `experience`: number of years of work experience on the resume.
- `call`: factor. Was the applicant called back?

Implementation in R


```
Call:
glm(formula = as.numeric(call) ~ gender + ethnicity + jobs +
    experience, family = binomial(link = "logit"), data = resumes)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-0.7207  -0.4326  -0.3962  -0.3473   2.4867

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  -2.833688   0.181450 -15.617  < 2e-16 ***
gendermale   -0.100533   0.129883  -0.774    0.439
ethnicitycauc  0.440383   0.107560   4.094 4.23e-05 ***
jobs         -0.045879   0.045481  -1.009    0.313
experience    0.041992   0.009685   4.336 1.45e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2726.9  on 4869  degrees of freedom
Residual deviance: 2691.4  on 4865  degrees of freedom
AIC: 2701.4

Number of Fisher Scoring iterations: 5
```

Implementation in Python 1


```
import matplotlib.pyplot as plt
import numpy as np
import sklearn
import statsmodels.api as sm
```

```
X1 = r.resumes_num[["gender", "ethnicity", "jobs", "experience"]]
X1 = sm.add_constant(X1)
Y = r.resumes_num["call"]

model = sm.Logit(Y, X1)
```

Implementation in Python 2


```
m_resumes = model.fit(method='newton')
```

```
Optimization terminated successfully.
         Current function value: 0.276327
         Iterations 7
```

```
m_resumes.params
```

```
const         -3.173539
gender        -0.100533
ethnicity      0.440383
jobs          -0.045879
experience     0.041992
dtype: float64
```

Implementation in Python 3


```
m_resumes.summary2()
```

```
<class 'statsmodels.iolib.summary2.Summary'>
"""
                         Results: Logit
================================================================
Model:              Logit            Pseudo R-squared: 0.013
Dependent Variable: call             AIC:              2701.4250
Date:               2023-02-01 19:15 BIC:              2733.8792
No. Observations:   4870             Log-Likelihood:   -1345.7
Df Model:           4                LL-Null:          -1363.5
Df Residuals:       4865             LLR p-value:      3.6738e-07
Converged:          1.0000           Scale:            1.0000
No. Iterations:     7.0000
----------------------------------------------------------------
               Coef.   Std.Err.     z      P>|z|    [0.025   0.975]
----------------------------------------------------------------
const         -3.1735    0.2790  -11.3746  0.0000  -3.7204  -2.6267
gender        -0.1005    0.1299   -0.7740  0.4389  -0.3551   0.1540
ethnicity      0.4404    0.1076    4.0943  0.0000   0.2296   0.6512
jobs          -0.0459    0.0455   -1.0088  0.3131  -0.1350   0.0433
experience     0.0420    0.0097    4.3356  0.0000   0.0230   0.0610
================================================================

"""
```

Warnings! 1

- The values of the coefficients, their standard errors, test statistics, and p-values (and therefore, their significance!) from linear and logistic regression models are all dependent on the other variables that are in the model.
- This means that you cannot necessarily make decisions about multiple variables at a time using these inference procedures.

```
X2 = r.resumes_num[["ethnicity", "jobs", "experience"]]
X2 = sm.add_constant(X2)

model2 = sm.Logit(Y, X2)

m_resumes2 = model2.fit(method='newton')
```

```
Optimization terminated successfully.
         Current function value: 0.276389
         Iterations 7
```

Warnings! In Practice 1


```
m_resumes2.summary2()
```

```
<class 'statsmodels.iolib.summary2.Summary'>
"""
                         Results: Logit
=================================================================
Model:              Logit            Pseudo R-squared: 0.013
Dependent Variable: call             AIC:              2700.0335
Date:               2023-02-01 19:15 BIC:              2725.9969
No. Observations:   4870             Log-Likelihood:   -1346.0
Df Model:           3                LL-Null:          -1363.5
Df Residuals:       4866             LLR p-value:      1.2868e-07
Converged:          1.0000           Scale:            1.0000
No. Iterations:     7.0000
-----------------------------------------------------------------
               Coef.   Std.Err.     z      P>|z|    [0.025   0.975]
-----------------------------------------------------------------
const         -3.2835    0.2404  -13.6573  0.0000  -3.7547  -2.8123
ethnicity      0.4391    0.1075    4.0832  0.0000   0.2283   0.6499
jobs          -0.0498    0.0452   -1.1011  0.2708  -0.1384   0.0388
experience     0.0424    0.0097    4.3885  0.0000   0.0235   0.0613
=================================================================

"""
```

Warnings! 2

- A significance level of 0.05 or 5% implies that you will make Type I (false positive) errors 5% of the time.
- When fitting many multiple regression models, you are making a lot of decisions and therefore are likely to make some error.
- Instead of using such a high significance level, you should use some multiple testing correction.

Warnings! In Practice 2


```
X3 = r.resumes_num.loc[:, r.resumes_num.columns != "call"]
X3 = sm.add_constant(X3)

model3 = sm.Logit(Y, X3)

m_resumes3 = model3.fit(method='newton')
```

```
Optimization terminated successfully.
         Current function value: 0.264192
         Iterations 7
```

Warnings! In Practice 3


```
m_resumes3.summary2()
```

```
<class 'statsmodels.iolib.summary2.Summary'>
"""
                         Results: Logit
=================================================================
Model:              Logit            Pseudo R-squared: 0.056
Dependent Variable: call             AIC:              2627.2256
Date:               2023-02-01 19:15 BIC:              2802.4785
No. Observations:   4870             Log-Likelihood:   -1286.6
Df Model:           26               LL-Null:          -1363.5
Df Residuals:       4843             LLR p-value:      4.4183e-20
Converged:          1.0000           Scale:            1.0000
No. Iterations:     7.0000
-----------------------------------------------------------------
               Coef.   Std.Err.     z     P>|z|    [0.025   0.975]
-----------------------------------------------------------------
const         -3.3033    1.2933  -2.5541  0.0106  -5.8381  -0.7684
name          -0.0014    0.0054  -0.2617  0.7936  -0.0120   0.0092
gender        -0.1305    0.1464  -0.8910  0.3729  -0.4175   0.1565
ethnicity      0.4388    0.1138   3.8567  0.0001   0.2158   0.6619
quality       -0.2023    0.2900  -0.6976  0.4855  -0.7708   0.3661
city          -0.4515    0.1317  -3.4277  0.0006  -0.7096  -0.1933
jobs          -0.0995    0.0513  -1.9379  0.0526  -0.2001   0.0011
experience     0.0350    0.0112   3.1150  0.0018   0.0130   0.0571
honors         0.3666    0.2001   1.8322  0.0669  -0.0256   0.7587
volunteer     -0.1952    0.1745  -1.1184  0.2634  -0.5373   0.1469
military      -0.1081    0.2190  -0.4936  0.6216  -0.5373   0.3211
```

Lasso Introduction 1

- In least squares linear regression with p predictors and n data points, estimates of the parameters in the model are calculated by minimization of the least-squares objective function.
- Typically, all of the least-squares estimates will be nonzero.
  - This will make interpretation of the final model challenging if p is large.
  - In fact, if $p > N$, the least-squares estimates are not unique.
  - There is an infinite set of solutions that make the objective function equal to zero, and these solutions almost surely overfit the data as well.

Lasso Introduction 2

- Thus, there is a need to constrain, or regularize the estimation process ("regularization" can be thought of as constraints on the estimation process).
- In the Least Absolute Shrinkage and Selection Operator (LASSO) or ℓ_1-regularized regression, we estimate the parameters by solving the problem

$$\underset{\beta_0,\beta}{\text{minimize}} \sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 \text{ subject to } ||\beta||_1 \leq t$$

where $||\beta||_1 \sum_{j=1}^{p} |\beta_j|$ is the ℓ_1 norm of β and t is a user specified parameter.

- "We can think of t as a budget on the total ℓ_1 norm of the parameter vector, and the lasso finds the best fit within this budget."

ℓ_1 Norm

- The lasso specifically uses the ℓ_1 norm.
- You could use any other norm, but it turns out that using the ℓ_1 solves a few problems for us:
  - First of all, if the "budget" t is small enough, the lasso "yields sparse solution vectors"–meaning that only a few terms are non-zero. This in turn implies that only the most important features remain in the model!
  - This does not occur for other norms–the ℓ_1 norm is the smallest value that gives a convex problem (makes optimization a lot easier–you can even apply these methods to problems with millions of parameters).

“Bet on Sparsity” Principle

- The third advantage–
  Use a procedure that does well in sparse problems, since no procedure does well in dense problems.

Notation Review

$$\underset{\beta_0,\beta}{\text{minimize}} \sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 \text{ subject to } ||\beta||_1 \leq t$$

- We can "translate" this equation–under this paradigm, we are looking for the parameters (β_0, β) that *minimize* the loss function $(\sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2)$ under the constraint that the sum of the absolute value of the parameters is less than some value t.

Reviewing Regularization

- Again, in this context–regularization refers to the fact that there are constraints placed on the parameters during the estimation (optimization) process.
- Wikipedia says that "regularization is a process that changes the result to be 'simpler' "–and even though I don't agree with everything that Wikipedia says on the topic of regularization, I think I do at least agree with that!
- Why regularize?
  - Can be used to find a unique solution for the parameters
  - Can be used to prevent over-fitting
  - Can be used to create more "parsimonious" models–a.k.a., models with fewer parameters.

Regularization Parameter Selection

- As always–things you do when setting up your model/parameter estimation process impact your results.
  - Choice of loss function
  - Choice of matrix norm
  - Choice of regularization parameter (your "budget" t).
    - If t is too small, you might leave important variables out of your model, resulting in a decrease of accuracy.
    - If t is too large, you might let too many variables into your model, resulting in loss of interpretability.
- How do you choose t? Usually cross validation.

Implementing Lasso 1

1. Randomly select a training and test set.

```
index <- sample(1:dim(resumes_num)[1],
                size = 0.7*dim(resumes_num)[1])

resume_train_preds <- resumes_num[index,-5]
resume_train_response <- resumes_num[index,5]

resume_test_preds <- resumes_num[-index,-5]
resume_test_response <- resumes_num[-index,5]
```

Implementing Lasso 2

2. Transform the data so that the variables are centered at 0 and have variances of 1 (standardized).

```
library(caret)

pre_proc_val <- preProcess(resume_train_preds,
                           method = c("center", "scale"))
resume_train_pp <- predict(pre_proc_val, resume_train_preds)
resume_test_pp <- predict(pre_proc_val, resume_test_preds)
```

Implementing the Lasso 3

3. Choose the regularization parameter. We can't give good advice about how to pick the actual number, but we can give advice about the best way to pick the parameter–cross validation. This is so common it should be automated in most packages.

```
library(glmnet)

reg_parm <- cv.glmnet(x = data.matrix(resume_train_pp), y = resume_tr
                      family = "binomial", type.measure = "auc")

reg_parm$lambda.min
```

```
[1] 0.003387462
```

Implementing the Lasso 4

4. Use the regularization parameter to fit the model.

```
resume_lasso <- glmnet(x = data.matrix(resume_train_pp),
                       y = resume_train_response,
                       alpha = 1, lambda = reg_parm$lambda.min)

resume_lasso$beta
```

```
26 x 1 sparse Matrix of class "dgCMatrix"
                       s0
name           .
gender         .
ethnicity      0.011375654
quality        .
city          -0.006401102
```

Implementing the Lasso 5

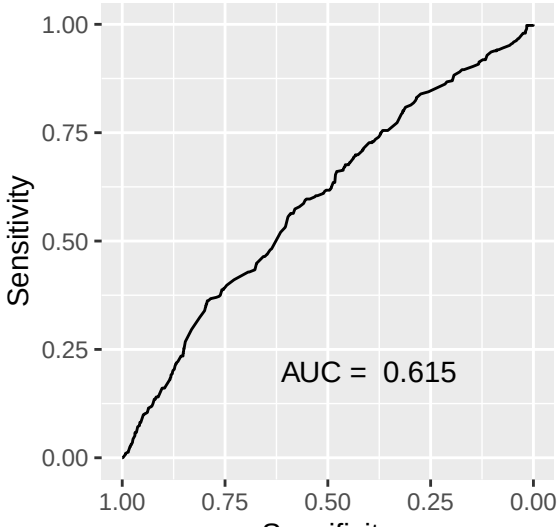
5. Measure model performance!

```
test_predictions <- predict(resume_lasso, s = reg_parm$lambda.min, ne

resumes_lasso_auc <- AUC(test_predictions, resume_test_response)
roc_resumes_lasso <- roc(resume_test_response, test_predictions)

ggroc(roc_resumes) +
  xlab("Specificity") +
  ylab("Sensitivity") +
  annotate("text", x = 0.4, y = 0.2, label = paste("AUC = ", round(re
```

Implementing the Lasso 5



Other Models

- Models with Regularization:
  - Ridge regression (addresses multicollinearity)
- Lasso
  - Elastic Net (combination of ℓ_1 and ℓ_2 norms)
  - Group Lasso–leaves whole groups of terms out of the model. IIRC, common in genetic analysis?

Resources I Used

Text

- Hastie, Trevor, Robert Tibshirani, and Martin Wainwright. "Statistical learning with sparsity." Monographs on statistics and applied probability 143 (2015): 143.

Python Resources

- `glmnet` module maintained by the creators of the LASSO
- StackOverflow proposing various other approaches