Homework 5

2025-09-25

Question 8.1

> Describe a situation or problem from your job, everyday life, current events, etc., for which a linear regression model would be appropriate. List some (up to 5) predictors that you might use.

An example of regression that I used in my Ph.D. research was using approximations and transformations to convert highly non-linear equations into simple linear data, then fitting this to experimental data to extract coefficients. One method that I proposed, was in the calculation of the work function of a material from thermionic emission data. The standard practice for this method is to perform two linear fits

1. From the current at multiple voltage measurements extract the zero voltage current, then
2. From multiple zero voltage currents at different temperatures, the work function and "Richardson's coefficient" (a measure of emission efficiency) can be extracted.

However, this method requires maintaining a constant temperature and my collaborators and I were having difficulty doing this (thermionic emission is very non-linear and small swings in temperature can lead to large changes in measured current). So instead I proposed to decompose the governing equation as fitting a plane and extract the coefficients from these parameters instead. The predictors were the inverse of the temperature ($1/T$) and the square root of the voltage over the temperature ($\sqrt{V}/T$) and the response was $\ln(i/T^2)$. Therefore we were able to fit

$$\ln\left(\frac{i}{T^2}\right) = \ln(AS) - \frac{\phi}{k}\left(\frac{1}{T}\right) + \sqrt{\frac{q^3}{4\pi\varepsilon_0 d}}\left(\frac{\sqrt{V}}{T}\right)$$

Question 8.2

Using crime_stats data from http://www.statsci.org/data/general/uscrime.txt (file uscrime.txt, description at http://www.statsci.org/data/general/uscrime.html), use regression (a useful R function is lm or glm) to predict the observed crime_stats rate in a city with the following data:

M = 14.0	So = 0	Ed = 10.0
Po1 = 12.0	Po2 = 15.5	LF = 0.640
M.F = 94.0	Pop = 150	NW = 1.1
U1 = 0.120	U2 = 3.6	Wealth = 3200
Ineq = 20.1	Prob = 0.04	Time = 39.0

Show your model (factors used and their coefficients), the software output, and the quality of fit.

Note that because there are only 47 data points and 15 predictors, you'll probably notice some overfitting. We'll see ways of dealing with this sort of problem later in the course.

First, we load the necessary libraries:

```
library(printr)     # pretty print for Rmd
library(lubridate)  # dates
library(ggplot2)    # plots
library(dplyr)      # dataframes
library(tidyr)
library(tidyverse)
library(recipes)
library(caret)
library(MASS)      # step models

# set seed for reproducibility
set.seed(42)
```

Now let's load the data:

```
df <- read.csv("./data 8.2/uscrime.txt", sep = "\t")
```

Note: As was stated in the question, our data set is extremely limited, therefore I will do 10-fold cross validation on the final model, but am not going to be creating a final test set.

Before doing any analysis, I want to get a baseline of whether these analyses are improving model quality. So we're going to look at the performance of the raw data. I am going to use a log transform on the predictor because crime_stats rates are limited to the non-negative domain which is not guaranteed by an untransformed linear model.

```
model <- lm(formula = Crime ~ ., data = df)
summary(model)$adj.r.squared
```

```
## [1] 0.7078062
```

Feature engineering

Our data has 15 predictors and one response. Before starting I looked at the data and determined what each variable is and what the distribution looked like using `pairs`. From this there were some derived features that seemed obvious to me:

```
crime_stats <- df
```

- Labor force participation and percent of non-white population appears to have a quadratic-like relationship

```
crime_stats$LF2 = crime_stats$LF^2
crime_stats$NW2 = crime_stats$NW^2
```

- Crime probability appears to have an inverse relationship

```
crime_stats$InvProb = 1 / crime_stats$Prob
```

So, let's see what the model quality is with these added terms

```
model <- lm(formula = log(Crime) ~ ., data = crime_stats)
summary(model)$adj.r.squared
```

```
## [1] 0.8652832
```

The added features made a noticeable improvement; however, because the coefficients are not normalized we cannot get an understanding of their significance based on coefficient size. So, let's normalize all the non-boolean values (`So`). Since we are going to want to do inference later on we need to retain our scaling parameters, it looks like the `recipes` package allows pipelines similar to sklearn and so that is what we'll use to standardize this.

```
rec <- recipe(Crime ~ ., data = crime_stats)

step <- rec |>
  step_normalize(all_numeric_predictors(), -all_of("So"))

scaler <- prep(step, training = crime_stats)

crime_scaled <- bake(scaler, new_data = crime_stats)
```

So, let's see what the model coefficients using the scaled data

```
full_model <- lm(formula = log(Crime) ~ ., data = crime_scaled)
summary(full_model)
```

```
##
## Call:
## lm(formula = log(Crime) ~ ., data = crime_scaled)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.28164 -0.08080  0.00515  0.07149  0.33802
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.72085    0.04696 143.133  < 2e-16 ***
## M            0.11459    0.04047   2.831 0.008490 **
## So           0.01200    0.12184   0.098 0.922237
## Ed           0.26370    0.05065   5.206 1.58e-05 ***
## Po1          0.28118    0.24394   1.153 0.258794
## Po2         -0.03679    0.25115  -0.146 0.884600
## LF           1.97092    0.79321   2.485 0.019212 *
## M.F         -0.07462    0.04486  -1.663 0.107406
## Pop         -0.09925    0.04065  -2.442 0.021193 *
## NW           0.57562    0.13643   4.219 0.000233 ***
## U1           0.04202    0.05841   0.719 0.477830
## U2           0.08549    0.05351   1.598 0.121329
## Wealth       0.09904    0.07898   1.254 0.220235
## Ineq         0.32131    0.06938   4.631 7.60e-05 ***
## Prob        -0.04133    0.04184  -0.988 0.331705
## Time        -0.02940    0.04438  -0.662 0.513082
## LF2         -1.88373    0.78220  -2.408 0.022860 *
## NW2         -0.48576    0.12139  -4.002 0.000418 ***
## InvProb      0.12239    0.04644   2.635 0.013547 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1509 on 28 degrees of freedom
## Multiple R-squared:  0.918,  Adjusted R-squared:  0.8653
## F-statistic: 17.41 on 18 and 28 DF,  p-value: 1.052e-10
```

First notice that the R^2 is the exact same as the unscaled model, indicating that scaling did not influence our fitting significantly. Further we can see that some coefficients such as educational attainment, unemployment, percentage of non-whites in the population, and absolute number of police in 1959 and 1960 were good predictors along with the derived characteristics of the ratio of adult to youth unemployment, and the squares of non-white percentage and labor force participation.

Step AIC

It was recommended on Piaza by Emily Chia-Huei Ko by to look at "An Introduction to Statistical Learning with Applications in R". The part that stood out to me was step selection of variables, I had used the technique in in a previous job but had completely forgotten about it. So let's use `stepAIC`[1] to remove some of the unimportant factors and make our model more parsimonious.

```
constant_model <- lm(log(Crime) ~ 1, data = crime_scaled)

# pages of output
invisible(
  capture.output(
    step_model <- stepAIC(
      constant_model,
      scope = list(lower = constant_model, upper = full_model),
      direction = "both"
    )
  )
)

summary(step_model)
```

```
##
## Call:
## lm(formula = log(Crime) ~ Po1 + Ineq + Ed + InvProb + So + Wealth +
##     M + U2 + NW + NW2 + Pop, data = crime_scaled)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.31386 -0.09556 -0.02279  0.11575  0.28259
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.77604    0.04204 161.197  < 2e-16 ***
## Po1          0.20830    0.05040   4.133 0.000212 ***
## Ineq         0.33043    0.06323   5.226 8.10e-06 ***
## Ed           0.29016    0.04674   6.208 4.11e-07 ***
## InvProb      0.13925    0.03105   4.484 7.54e-05 ***
## So          -0.15012    0.10313  -1.456 0.154409
## Wealth       0.11277    0.07456   1.512 0.139398
## M            0.09651    0.03924   2.459 0.019008 *
## U2           0.06632    0.02761   2.402 0.021755 *
## NW           0.69949    0.12629   5.539 3.13e-06 ***
## NW2         -0.58229    0.10484  -5.554 2.99e-06 ***
## Pop         -0.06979    0.03187  -2.190 0.035306 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1585 on 35 degrees of freedom
## Multiple R-squared:  0.8869, Adjusted R-squared:  0.8514
## F-statistic: 24.95 on 11 and 35 DF,  p-value: 2.349e-13
```

This has significantly cut down our number of parameters reducing our risk of overfitting with minimal impact to our adjusted R-squared.

[1] https://www.rdocumentation.org/packages/MASS/versions/7.3-65/topics/stepAIC

Cross-validation

As stated above we are using the entire dataset for training; however, I want to get an idea of the validity of the fit metrics we've been outputting. Let's do a cross validation of the data to see if there's any problems hidden in our final model

```
train_control <- trainControl(method = "cv", number = 10)
cross_val_stats <- train(
  formula(step_model),
  data = crime_scaled,
  method = "lm",
  trControl = train_control
)

cross_val_stats
```

```
## Linear Regression
##
## 47 samples
## 11 predictors
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 44, 42, 42, 42, 42, 43, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.1812351  0.8239829  0.1589623
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

We do see a significant amount of degradation in the cross validation indicating that we are overfitting the data, unfortunately there is not much that we can do about that with our current toolset and the overfitting would have been worse if we had not trimmed coefficients from the model.

Datapoint prediction

Let's see what the predicted crime_stats for the specified parameters is

```
crime_test <- data.frame(
  M      = 14.0, So     = 0, Ed     = 10.0, Po1    = 12.0,
  Po2    = 15.5, LF     = 0.640, M.F    = 94.0, Pop    = 150,
  NW     = 1.1, U1     = 0.120, U2     = 3.6, Wealth = 3200,
  Ineq   = 20.1, Prob   = 0.04, Time   = 39.0
) %>% mutate(
  InvProb = 1 / Prob,
  LF2 = LF^2,
  NW2 = NW^2
)

crime_test_scaled <- bake(scaler, new_data = crime_test)

# reverse the log scaling because for some reason that's not done automatically
exp(predict(step_model, crime_test_scaled))
```

```
##        1
## 490.4742
```

This result is in the lower end of the training data. Because this is a linear model, it has high explainibility. I copied the coefficients and the scaled crime_stats data and multiplied each term out.

Variable	Estimate	Scaled Value	Impact
NW	0.70	-0.88	-0.61
NW2	-0.58	-0.54	0.32
Po1	0.21	1.18	0.25
Wealth	0.11	-2.13	-0.24
Pop	-0.07	2.98	-0.21
Ed	0.29	-0.50	-0.15
Ineq	0.33	0.18	0.06
InvProb	0.14	-0.15	-0.02
U2	0.07	0.24	0.02
M	0.10	0.11	0.01
So	-0.15	0.00	0.00

Interestingly it turns out that NW and NW^2 partially end up cancelling each other out; however combined they are still the most important factor with increasing non-white fraction correlated with decreasing the crime_stats per hundred thousand. Other important factors are: police per capita correlated with to an increase in crime_stats, and wealth, population, and education all correlated with a decrease in crime_stats. The fact that non-white population is the strongest predictor points to systematic issues in my opinion, possibly crime stats reporting being lower in the non-white community? The police per capita being high seems like an auto-correlation effect, if crime_stats was previously high there will be more police and current crime stats will be higher. The impact of wealth, education, and population all conform to my priors.