Stroke risk prediction

Guy Maskall

23 March, 2020

Introduction

Having previously explored the stroke dataset, we can now start to do some predictive modelling.

```
stroke <- read_csv("train_2v.csv") %>%
    select(-id, -smoking_status, -work_type) %>%
    filter(complete.cases(.))
```

```
## Parsed with column specification:
## cols(
##   id = col_double(),
##   gender = col_character(),
##   age = col_double(),
##   hypertension = col_double(),
##   heart_disease = col_double(),
##   ever_married = col_character(),
##   work_type = col_character(),
##   Residence_type = col_character(),
##   avg_glucose_level = col_double(),
##   bmi = col_double(),
##   smoking_status = col_character(),
##   stroke = col_double()
## )
```

```
stroke %>% glimpse
```

```
## Observations: 41,938
## Variables: 9
## $ gender            <chr> "Male", "Male", "Female", "Female", "Male", "Fema...
## $ age               <dbl> 3, 58, 8, 70, 14, 47, 52, 75, 32, 74, 79, 79, 37,...
## $ hypertension      <dbl> 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0...
## $ heart_disease     <dbl> 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0...
## $ ever_married      <chr> "No", "Yes", "No", "Yes", "No", "Yes", "Yes", "Ye...
## $ Residence_type    <chr> "Rural", "Urban", "Urban", "Rural", "Rural", "Urb...
## $ avg_glucose_level <dbl> 95.12, 87.96, 110.89, 69.04, 161.28, 210.95, 77.5...
## $ bmi               <dbl> 18.0, 39.2, 17.6, 35.9, 19.1, 50.1, 17.7, 27.0, 3...
## $ stroke            <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
```

Partition

```
set.seed(47)
stroke_train <- stroke %>%
```

```
    group_by(stroke) %>%
    sample_frac(.7)
stroke_test <- stroke %>%
    anti_join(stroke_train)
```

```
## Joining, by = c("gender", "age", "hypertension", "heart_disease",
## "ever_married", "Residence_type", "avg_glucose_level", "bmi", "stroke")
```

```
set.seed(470)
stroke_tr_bal <- stroke_train %>%
    group_by(stroke) %>%
    sample_n(min(group_size(.)))
```

Models

Age alone

```
age_alone_model <- glm(stroke ~ age, stroke_tr_bal, family=gaussian)
age_alone_model %>% summary
```

```
##
## Call:
## glm(formula = stroke ~ age, family = gaussian, data = stroke_tr_bal)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.8884  -0.2712   0.1356   0.2519   1.0654
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.2618022  0.0323193   -8.10 1.78e-15 ***
## age          0.0140265  0.0005474   25.63  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.144723)
##
##     Null deviance: 225.00  on 899  degrees of freedom
## Residual deviance: 129.96  on 898  degrees of freedom
## AIC: 818.45
##
## Number of Fisher Scoring iterations: 2
```

All features

Logistic regression

```
all_feat_model <- glm(stroke ~ ., stroke_tr_bal, family=gaussian)
all_feat_model %>% summary
```

```
##
## Call:
```

```
## glm(formula = stroke ~ ., family = gaussian, data = stroke_tr_bal)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.11781  -0.22073   0.07204   0.24321   1.11106
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)         -0.2100994  0.0618464  -3.397 0.000711 ***
## genderMale           0.0394324  0.0251797   1.566 0.117695
## age                  0.0136984  0.0007119  19.242  < 2e-16 ***
## hypertension         0.1093741  0.0343531   3.184 0.001504 **
## heart_disease        0.1259794  0.0393349   3.203 0.001410 **
## ever_marriedYes     -0.0781468  0.0348850  -2.240 0.025329 *
## Residence_typeUrban  0.0303482  0.0248597   1.221 0.222491
## avg_glucose_level    0.0006963  0.0002492   2.794 0.005314 **
## bmi                 -0.0043066  0.0019195  -2.244 0.025103 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.1379915)
##
##     Null deviance: 225.00  on 899  degrees of freedom
## Residual deviance: 122.95  on 891  degrees of freedom
## AIC: 782.54
##
## Number of Fisher Scoring iterations: 2
```

L1 regularized logistic regression

```
x_tr <- model.matrix(stroke~., stroke_tr_bal)[, -1]
y_tr <- stroke_tr_bal$stroke
all_feat_l1_mod <- cv.glmnet(x_tr, y_tr)
coef(all_feat_l1_mod)
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                                    1
## (Intercept)         -0.1723980089
## genderMale           .
## age                  0.0118124260
## hypertension         0.0409010413
## heart_disease        0.0676618866
## ever_marriedYes      .
## Residence_typeUrban  .
## avg_glucose_level    0.0001269647
## bmi                  .
```

Using lasso for feature selection, we are left with

- age
- hypertension
- heart_disease
- avg_glucose_level

as predictors.

```
selected_feats_mod <- glm(stroke ~ age +
                          hypertension +
                          heart_disease +
                          avg_glucose_level,
                        stroke_tr_bal, family=gaussian)
selected_feats_mod %>% summary
```

```
##
## Call:
## glm(formula = stroke ~ age + hypertension + heart_disease + avg_glucose_level,
##     family = gaussian, data = stroke_tr_bal)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.11277  -0.24646   0.08826   0.24680   1.07794
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -0.2873597  0.0387850  -7.409 2.94e-13 ***
## age                0.0125062  0.0005965  20.964  < 2e-16 ***
## hypertension       0.1163264  0.0344194   3.380 0.000757 ***
## heart_disease      0.1425465  0.0391213   3.644 0.000284 ***
## avg_glucose_level  0.0005926  0.0002466   2.403 0.016464 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.139768)
##
##     Null deviance: 225.00  on 899  degrees of freedom
## Residual deviance: 125.09  on 895  degrees of freedom
## AIC: 790.08
##
## Number of Fisher Scoring iterations: 2
```

Model comparison

We have two main models for stroke risk prediction. In initial EDA, it was clear that age was a dominant factor, and so the first model simply uses that. Age is a feature that requires no diagnostic measurement. A second model adds a feature denoting whether the patient is suffering from hypertension, a feature denoting whether they are suffering from heart disease, and their average glucose level measured after a meal. The first two of these can be expected to already be in a patient's records. The glucose level measurement is probably the most awkward additional feature. It also appears to be the least significant in the joint model. We may subsequently wish to review how difficult it is to acquire this for patients and investigate further how useful it is in predicting stroke risk.

It is encouraging that body mass index fell out. This had a small number of missing values, suggesting it may be harder to acquire for some reason. We could arguably add those previously omitted records back into the dataset. It is also appealing that residence type is not included; this would likely be difficult for doctors to acquire reliably. That is to say, the classification of residence type may be somewhat subjective.

So now, how do these models perform?

```
stroke_test %>%
    count(stroke) %>%
```

```
    pivot_wider(names_from=stroke, values_from=n) %>%
    mutate(stroke_pc = 100*`1`/`0`)
```

```
## # A tibble: 1 x 3
##     `0`   `1` stroke_pc
##   <int> <int>     <dbl>
## 1 12389   193      1.56
```

The prevalence of stroke in the test set is the same as in the overall dataset, specifically 1.56%.

```
age_preds <- predict(age_alone_model,
                     newdata=stroke_test,
                     type="response") > 0.5
age_tp <- sum((age_preds == 1) & (stroke_test$stroke == 1))
age_fp <- sum((age_preds == 1) & (stroke_test$stroke == 0))
age_fn <- sum((age_preds == 0) & (stroke_test$stroke == 1))
age_precision <- 100 * age_tp / (age_tp + age_fp)
age_recall <- 100 * age_tp / (age_tp + age_fn)
print(c(age_precision, age_recall))
```

```
## [1]  3.759584 78.756477
```

Using the age-only model, we flag 4043 individuals, or 32% of the population for stroke. Of these, 3.8% actually suffered a stroke. This is over twice the "hit" rate compared to a random sampling of the population, remembering the natural prevalence is 1.56%. Of the population who did suffer a stroke, the model picked up just over 78% of them. In other words, to pick up 78% of stroke victims via random sampling, we'd expect to have to sample 78% of the population.

This is not bad for a simple model with a single feature. We've more than halved the number of people to follow up on whilst more than doubling the hit rate.

How does the model with more features perform?

```
selected_feats_preds <- predict(selected_feats_mod,
                     newdata=stroke_test,
                     type="response") > 0.5
selected_feats_tp <- sum((selected_feats_preds == 1) &
                         (stroke_test$stroke == 1))
selected_feats_fp <- sum((selected_feats_preds == 1) &
                         (stroke_test$stroke == 0))
selected_feats_fn <- sum((selected_feats_preds == 0) &
                         (stroke_test$stroke == 1))
selected_feats_precision <- 100 * selected_feats_tp /
    (selected_feats_tp + selected_feats_fp)
selected_feats_recall <- 100 * selected_feats_tp /
    (selected_feats_tp + selected_feats_fn)
print(c(selected_feats_precision, selected_feats_recall))
```

```
## [1]  4.066143 77.720207
```

Using the larger model, we flag 3689 individuals, or 29% of the population. Of these, 4.1% actually suffered a stroke. This is even better than the previous model, whilst maintaining a similar recall rate. To put it another way, we would target 354 fewer people whilst catching about the same number of stroke victims.

Final note

There are many unknowns here, from the origin and sampling strategy of the dataset, through to the interpretation of some of the features, and the use to which any stroke prediction result might be put. With any health intervention, there are costs to consider. Are we sending out leaflets here or contacting patients to arrange for them to attend their GP practice for tests, or even some preventative treatment? These options have very different costs and therefore different optimum models.

Health services are inevitably resource constrained, meaning a model that flags more alarms than the capacity of the health service can handle is of little use. This is frequently a limitation of binary classification models in applications where data are noisy and uncertain. By adopting a logistic regression model here, we naturally deal with probabilities and don't need to adopt the threshold of 0.5. Indeed, we don't even need to apply a threshold at all. In a resource-constrained health service, we could rank patients by predicted risk and pursue those deemed most at risk first.

Session info

```
sessionInfo()
```

```
## R version 3.6.3 (2020-02-29)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Linux Mint 19.3
##
## Matrix products: default
## BLAS:   /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.7.1
## LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.7.1
##
## locale:
##  [1] LC_CTYPE=en_GB.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_GB.UTF-8        LC_COLLATE=en_GB.UTF-8
##  [5] LC_MONETARY=en_GB.UTF-8    LC_MESSAGES=en_GB.UTF-8
##  [7] LC_PAPER=en_GB.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_GB.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] glmnet_3.0-2    Matrix_1.2-18   knitr_1.26      forcats_0.4.0
##  [5] stringr_1.4.0   dplyr_0.8.3     purrr_0.3.3     readr_1.3.1
##  [9] tidyr_1.0.0     tibble_2.1.3    ggplot2_3.2.1   tidyverse_1.3.0
## [13] rmarkdown_2.0   nvimcom_0.9-78
##
## loaded via a namespace (and not attached):
##  [1] shape_1.4.4      tidyselect_0.2.5 xfun_0.11        haven_2.2.0
##  [5] lattice_0.20-40  colorspace_1.4-1 vctrs_0.2.1      generics_0.0.2
##  [9] htmltools_0.4.0  yaml_2.2.0       utf8_1.1.4       rlang_0.4.2
## [13] pillar_1.4.3     withr_2.1.2      glue_1.3.1       DBI_1.1.0
## [17] dbplyr_1.4.2     modelr_0.1.5     readxl_1.3.1     foreach_1.4.7
## [21] lifecycle_0.1.0  munsell_0.5.0    gtable_0.3.0     cellranger_1.1.0
## [25] rvest_0.3.5      codetools_0.2-16 evaluate_0.14    fansi_0.4.0
## [29] broom_0.5.3      Rcpp_1.0.3       scales_1.1.0     backports_1.1.5
```

```
## [33] jsonlite_1.6     fs_1.3.1         hms_0.5.2        digest_0.6.23
## [37] stringi_1.4.3    grid_3.6.3       cli_2.0.0        tools_3.6.3
## [41] magrittr_1.5     lazyeval_0.2.2   crayon_1.3.4     pkgconfig_2.0.3
## [45] zeallot_0.1.0    xml2_1.2.2       reprex_0.3.0     lubridate_1.7.4
## [49] iterators_1.0.12 assertthat_0.2.1 httr_1.4.1       rstudioapi_0.10
## [53] R6_2.4.1         nlme_3.1-144     compiler_3.6.3
```