Final Project

Statistical Computing

E. Kelley

Fall 2022

Introduction

I will be analyzing a Kaggle data set, Length of Hospital Stay (LHS) (https://www.kaggle.com/datasets/aayushchou/hospital-length-of-stay-dataset-microsoft). My motivation for selecting this data set was driven by a few aspects. This was the first time I have explored Kaggle, and I was surprised by how many of the datasets included relatively few variables. I wanted to find a data set rich in predictors, so I could perform model feature selection. Additionally, I reasoned that a "medical" data set might present challenges similar to what I might encounter in the "wild," and, consequently, would help me gain practical experience. This data set has 29 variables and 100,000 observations. Given the large number of observations, I will take a subsample of 8,000 observations for ease of computation. I had originally proposed a subsample of 1,000 observations, but after I began modeling, realized there were too few samples in some factor categories which were causing errors. The focus of this analysis will be prediction of `lengthofstay` (days spent in hospital) using a mix of categorical (n=12) and numeric (n=10) variables. `lengthofstay` ranges from one to 17 days (Figure 1).

Materials

The categorical variables appear to be a collection of various diseases (eg, asthma) and risk factors (eg, malnutrition) (Table 1). I expect to drop `fibrosisandother` as my sample contains few observations that are positive instances. There is a fairly even split across gender. The numeric variables are largely composed of blood metrics, but BMI, pulse, and respiration are also present. Some dates are included: vdate (visit date) and discharge (discharge date). I may consider month of visit date in the modeling, as well. Reviewing histograms of the numeric predictors, I see most look approximately normally distributed with the exceptions of `bloodureanitro` and `neutrophils`(Figure 2).

There are some weak correlations present in the set of continuous predictors (Figure 3), specifically between `hematocrit` and `respiration`, between `lengthofstay` and both `bloodureanitro` and `respiration`. I anticipate those metrics being important for prediction of `lengthofstay`. I also see `lenghtofstay` and `neutrophils` are negatively correlated. Based on the histograms, summary table, and correlation analysis, I don't see any "red flags" that need to be addressed with further data cleaning.* Additionally, this data set, provided by Microsoft, was assigned a "use-ability" score of 10 on Kaggle. While I'm not sure how much blind faith to put in that score, it does give me a bit more confidence in my assessment that this data set is ready for analysis.

*After starting analysis, I realized that numerical data should be centered and scaled prior to modeling, so I implemented that using the `skimr` package. It has a nice feature for summarizing data in a data frame, including variable types, and even plots little histograms within a column of the data frame. I was going to include it here as a figure but the unicode characters were creating a problem in pdf document knitting.

Methods

caret for cross validation

After a bit of research, it looks as though using the `caret` package for cross-validation of models will be a stream-lined solution and address the problem of model-specific function arguments. Looking into `caretEnsemble`, it appears well-suited for cross validating a list of models across the same splits of training data for direct comparison of models. You must make sure to provide the indices to `trainControl` to ensure the same data is used for each model k-fold. Additionally, `caretEnsemble` provides functionality to create ensemble models, but that is outside the scope of this project. As a starting point, I will use `method = cv` instead of `repeatedcv`, but if possible I will look into the latter. It seems as though `repeatedcv` is a bootstrapping method for getting estimates of spread for cv error across folds. Looking into the available methods listed in the package documentation, all of my selected models are available (Table 2).
`rfe()` is a potential option for feature selection for linear models. I am uncertain what is the benefit over lasso, if any. As I understand it, they both aim to increase model bias. Additionally, I feel some confusion around the proper use of `set.seed()` within R markdown and also between model implementations. I've seen recommendations of calling `set.seed()` prior to running every model. That seems unnecessary, but perhaps there is some esoteric reason behind that line of thinking.

Selection of models

A critical aspect of the project that I overlooked in my project proposal was the approximate distribution of my response variable. Once I started looking at model results and thinking about interpretation, I realized my error. `lengthofdays` is count data, and thus violates the assumption of normally distributed data for a linear model. Given my error, I researched an appropriate model that would be available in `caret` and came up with poisson glm as a good replacement. A negative binomial model would likely also be good to try since the mean and variance are different for `lengthofdays`. Decision trees are still an appropriate model, since they are non-parametric. So, I will move forward with applying poisson regression, negative binomial regression and decision trees for this project. Before I realized the problem with modeling a count data response variable, I had applied linear models and lasso regression, and was surprised by the RMSE for both being relatively low (even across the test data set). Since the lm and lasso seemed to perform better than expected, out of curiosity, I will keep them in the modeling, but will focus my discussion mostly on the glms and decision tree.

Results

RMSE for 10-fold cross validation

For 10-fold cross validation, lasso produced the lowest RMSE, followed closely by negative binomial and linear models (Figure 4 cv rmse). Poisson had the highest RMSE (2.53), and the decision tree was second highest. The 0.95 confidence interval for Poisson RMSE was much larger than all other models (~2.2-2.9).

The negative binomial, poisson, linear and lasso models all performed closely when measured as RMSE on the unseen test data of 4000 observations (Table 2). The rpart decision was less accurate.

Tracing errors in `caretList`

Initially, I ran into a warning coming from the `caretList` function: "There were missing values in resampled performance measures." I found a related question on stackoverflow, and the best answer was that the decision tree in rpart couldn't find a split, so it averages the outcome and uses that as the predictor. One drawback of using `caretList` is that errors can be more challenging to trace since it is unclear from which model method they originate. I ended up just searching the web for the error message and/or commenting out code chunks until I could locate the source. I wondered if the errors are captured in the returned list, and browsed through it but didn't see any list element that looked like an error/warning message.

Negative Binomial GLM model: tuning of the link function

The model results for negative binomial list the 'identity' as the `bestTune` link function for the glm. I think this may be a clue as to why the linear model performed better than expected, in fact, comparably to the negative binomial model. Essentially, what I think is happening is all three link functions are being evaluated (sqrt, identity, log), and identity is the best. I don't have intuition for why this might be the case. The 'link' function is listed as a tuning parameter in the `caret` manual; `caret` wraps the `MASS` `glm.nb` method. The help documentation for `MASS::glm.nb` has a default of `link=log`. This supports the theory that `caret` is optimizing the link function and `identity` just so happens to be the best. Later, I reviewed the `caret` output, and this theory was correct (Table 3).

Rpart

The optimal complexity parameter determined for the tree was 0.028. This resulted in two terminal nodes with the only internal node `facidE`, facility id E. This result has limited utility for inference, aside from the conclusion that facility E generally keeps patients for fewer days (Figure 6). Some things to look into are expanding the grid of complexity parameters to optimize (I used the default) and the other, less likely problem, `minsplit` which is the parameter that defines number of observations required to create a new node (the default is 20). I think `minsplit` is less likely to be the issue since I have a large enough sample size that I would anticipate at least 20 observations falling into new nodes.

Discussion

Model selection via cross validation

The negative binomial model performed the best when applied to the unseen test data, but the linear model, Poisson, and lasso were all very close (Table 4). I was surprised by this result. I would have expected the negative binomial and poisson models to outperform the linear model and lasso since the assumption of normally distributed counts data doesn't hold. My conclusion here is that none of the models have very good predictive power. Looking at the summary of the final linear model as a representative example, I see the adjusted R-squared value is 0.217, which I think supports the idea that none of the models have great predictive ability given they all have a close RMSE. I thought the decision tree would perform better "out-of-the-box," but alas this was not the case, so I think a future direction there would be to optimize the complexity parameter. It could be interesting to try a neural network since there seems to be potential for non-linear relationships. GAMs would be another potential future direction.

Model predictions

In order to understand the usefulness of the final model, it's important to think about the observed error in the context of our observed responses. The first quartile is 2 days spent in the hospital and the third quartile is 6 days. I'm unclear on how the average error applies across our model; are lower counts more impacted by an RMSE of ~2 days vs. higher counts it is relatively less? The context of how the model will be applied is also important. If we are looking for a general idea of factors leading to longer hospital stay (ie, inference), then perhaps this is "good enough," but if we're looking to predict length of stay for an individual and implement some kind of patient-specific mitigation, then I think we have more work to do.

Model inference

Since the negative binomial model performed the best on the test data, I will focus on it for model inference. Generally, there would be a log link function to take into account, but since the best link function identified by cross validation was the identity function, the response will not be transformed. The most important variables contributing to increase in `lengthofstay` are `hemo1`, `facid` (C/D/E), types of psychological problems: `psychologicaldisordermajor1`, `substancedependence1`, `psychother1`; `dialysisrenalendstage1` (Table 5). For our most influential variable, `hemo1`, a one unit increase will result in a 1.4 unit increase in `lengthofstay`. The relative importance of variables can be seen in the importance plot output by `caret` (Figure 7).

Conclusions

Overall, I really liked using `caret` for cross validation since it's quite flexible and efficient. A potential drawback is that it could be easier to misuse a method without realizing it. I spent a fair bit of time just familiarizing myself with the model methods and tuning parameter optimization, along with learning the output structure in order to troubleshoot and extract results, and I still feel like I'm only scratching the surface. Regardless, I think it is worth the time spent learning it since it removes the burden of learning several different packages, especially if you use the `caretList` function from `caretEnsemble`.

Citations

caretEnsemble

Avoid Mistakes in Machine Learning Models with Skewed Count Data

Appendix

Tables and Figures

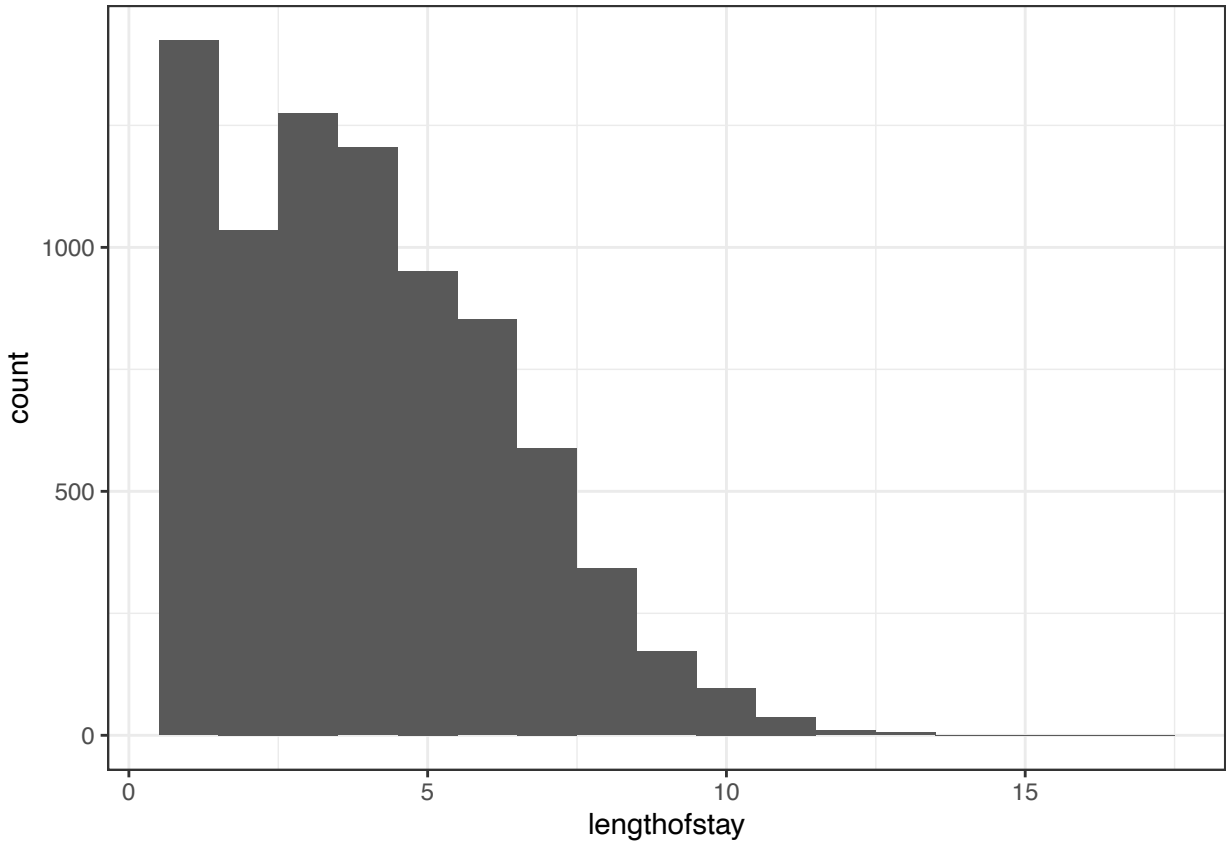

Figure 1: Histogram of length of hospital stay

Table 1: Summary of categorical predictors

name	0	1
asthma	7702	298
depress	7595	405
dialysisrenalendstage	7690	310
fibrosisandother	7950	50
hemo	7346	654
irondef	7290	710
malnutrition	7622	378
pneum	7690	310
psychologicaldisordermajor	5988	2012
psychother	7624	376
substancedependence	7478	522

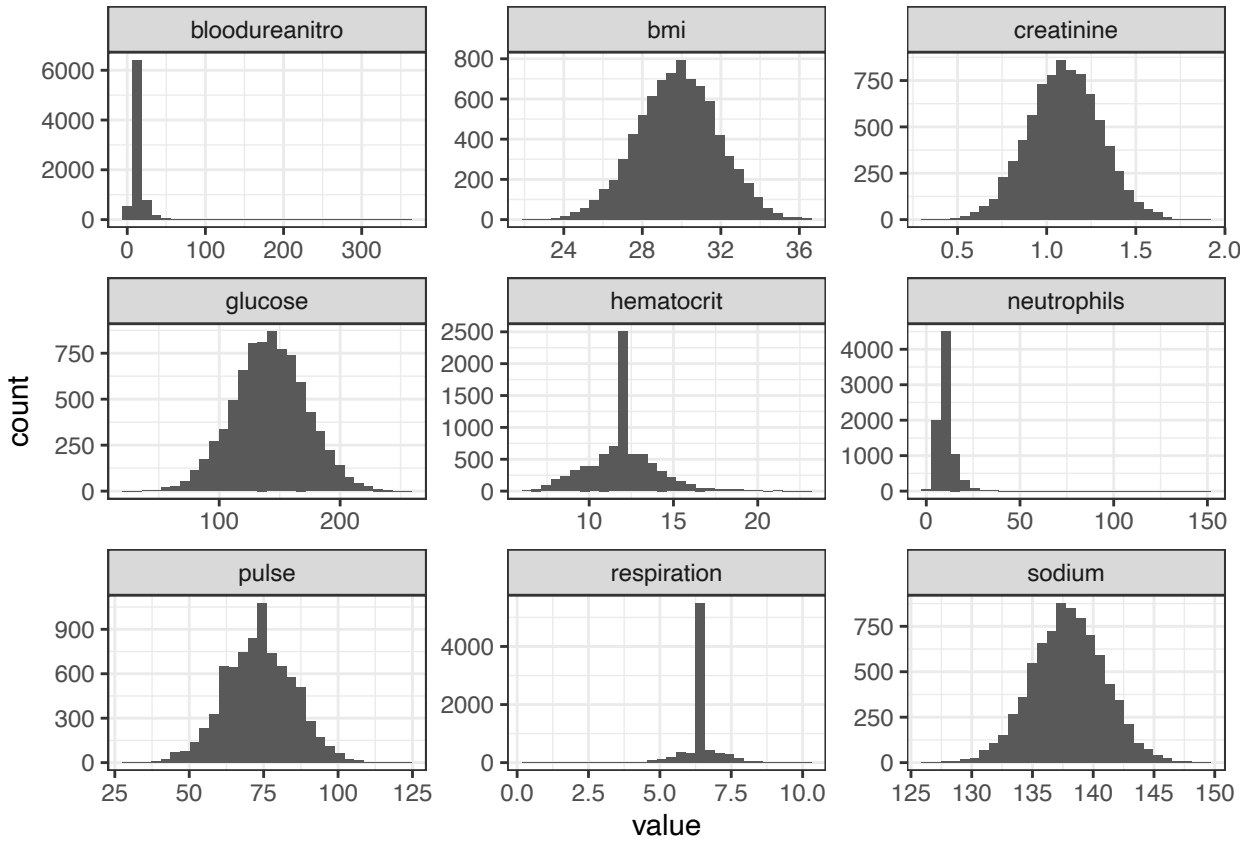

Figure 2: Histograms of continuous predictors

Table 2: List of models in caret

model	parameter	label	forReg	forClass	probModel
rpart	cp	Complexity Parameter	TRUE	TRUE	TRUE
glm	parameter	parameter	TRUE	TRUE	TRUE
lasso	fraction	Fraction of Full Solution	TRUE	FALSE	FALSE
glm.nb	link	Link Function	TRUE	FALSE	FALSE

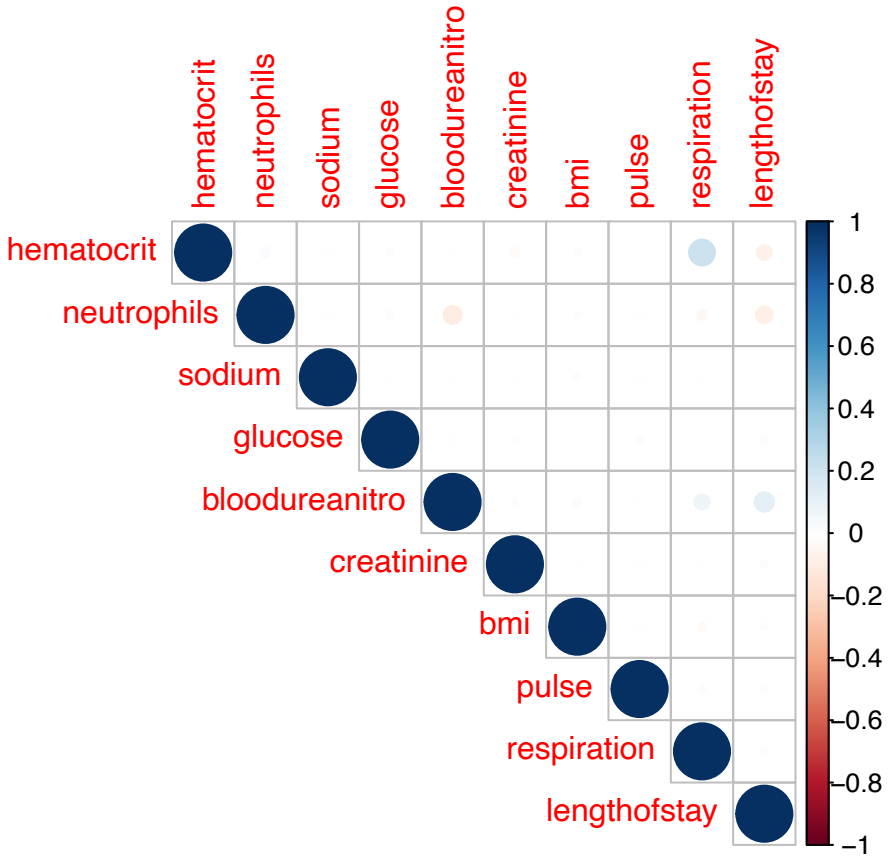

Figure 3: Correlation of continuous predictors

Table 3: Negative binomial CV results for link function

link	RMSE	Rsquared	MAE	RMSESD	RsquaredSD	MAESD
identity	2.172	0.1758	1.780	0.0271	0.0135	0.0193
log	2.543	0.1195	1.828	0.4888	0.0494	0.0468
sqrt	2.210	0.1620	1.795	0.0581	0.0168	0.0229

Table 4: RMSE of final models across the test data

Model	RMSE
NB	2.109
LM	2.111
POISSON	2.122
LASSO	2.152
RPART	2.244

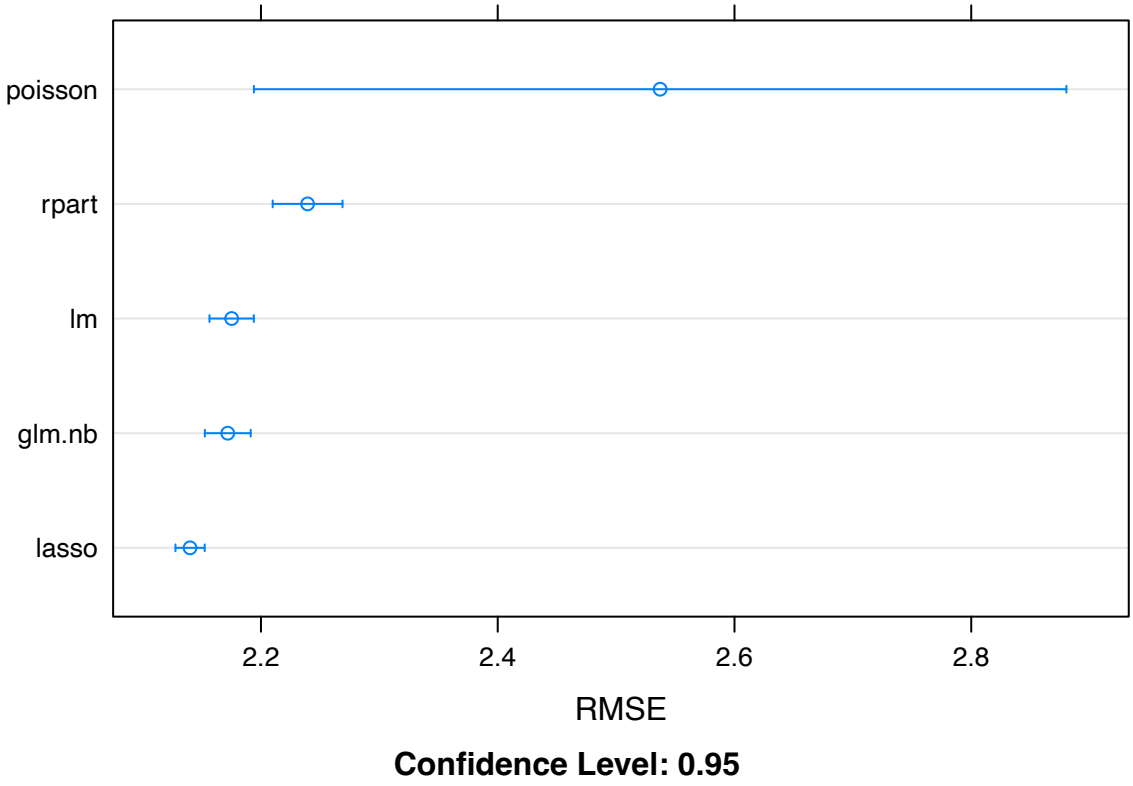

Figure 4: RMSE across 10-fold validation for each model

Table 5: Negative binomial model coefficients

	coefficient
(Intercept)	3.1609
hemo1	1.4084
facidC	1.2446
facidD	1.1660
facidE	0.9063
psychologicaldisordermajor1	0.7097
substancedependence1	0.6306
psychother1	0.6095
dialysisrenalendstage1	0.5556
depress1	0.5119
asthma1	0.4336
malnutrition1	0.4293
pneum1	0.3506
irondef1	0.2817
bloodureanitro	0.1751
facidA	-0.1498
neutrophils	0.1027
hematocrit	0.0839

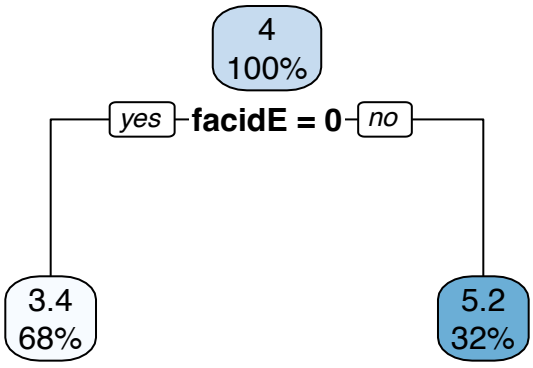

Figure 5: Rpart plot of the final decision tree

	coefficient
genderM	-0.0455
pulse	0.0361
respiration	-0.0357
glucose	-0.0254
creatinine	0.0216
bmi	-0.0060
sodium	-0.0042

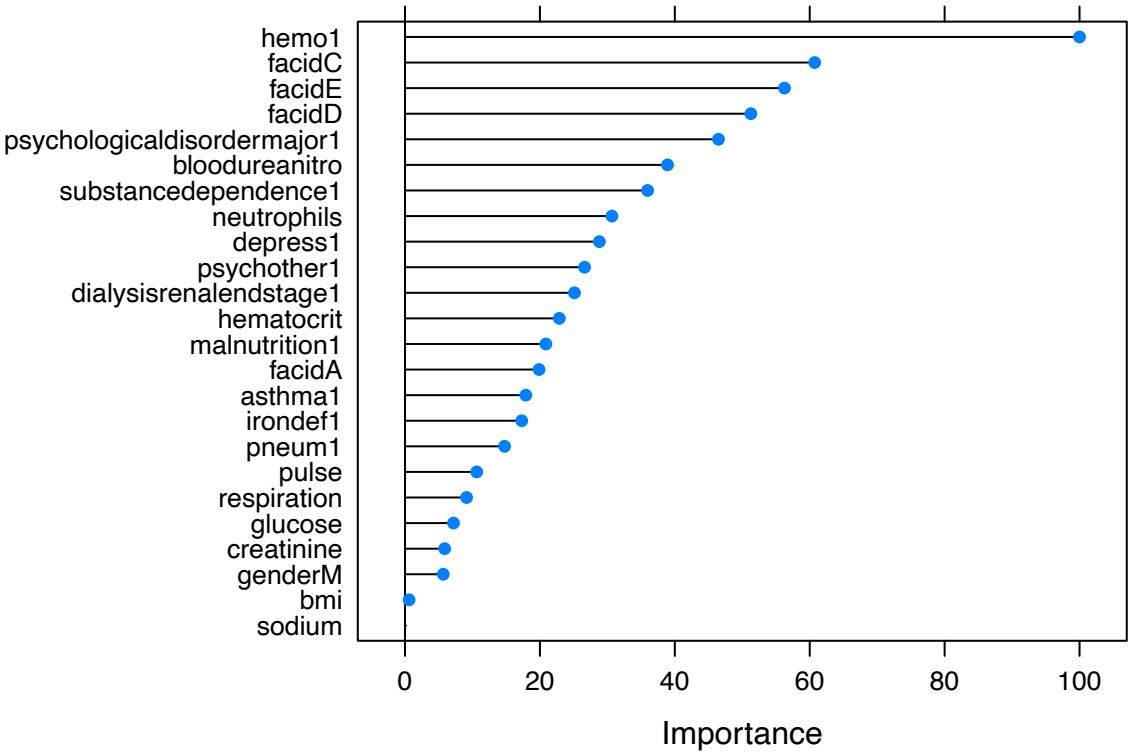

Figure 6: Ranked variable importance for negative binomial glm

Code Appendix

```
library(tidyverse)
library(MASS)
library(caret)
library(caretEnsemble)
library(glmnet)
library(corrplot)
library(lubridate)
```

```
set.seed(578)

raw_data <- read_csv("LengthOfStay.csv", col_types = "ccfffffffffffffnnnnnnnnnnncfn")
# glimpse(raw_data)

data <- raw_data %>%
  mutate(vdate_month = month(parse_date(vdate, "%m/%d/%Y"))) %>%
  sample_n(size = 8000) # actually, take 8k samples so we can save 4k for a
# final test of the CV models
# summary(data)
```

```
ggplot(data, aes(lengthofstay)) +
  geom_histogram(binwidth = 1) +
  theme_bw()
```

```
d1 <- data %>%
  dplyr::select(5:15) %>%
  pivot_longer(cols = everything()) %>%
  group_by(name, value) %>%
  summarize(count = n()) %>%
  pivot_wider(id_cols = name, names_from = value, values_from = count)
knitr::kable(d1, caption = "Summary of categorical predictors")
```

```
numdata <- data %>%
  dplyr::select(16:24) %>%
  pivot_longer(everything())
```

```
ggplot(numdata, aes(value)) +
  geom_histogram() +
  facet_wrap(~name, scales = "free") +
  theme_bw()
```

```
cmat <- cor(data[, c(16:24, 28)], method = "spearman")
corrplot(cmat, type = "upper")
```

```
# 1 - set up a for loop for 10-fold cross validation
# 2 - set up train and test indices from data (df); nested for loop?
# 3 - each model will use a different data structure, so just make sure to use
# the same indices to
```

```r
# subset test and train prior to creating the required data structures.
# 4 - apply models and record MSE.
# lm and lasso can use the same data struc (pg. 120 class notes)
# decision tree
# maybe use gams
# 5 - caretEnsemble will actually do this through model methods

set.seed(578)
# split data equally into train and test sets; cross validate using train.
# Apply final models to the test set.
# ToDo drop extra cols that aren't predictors; address rcount variable. Could
# try one-hot encoding. Need to think carefully about how to handle this. It
# is a strange mix of ordinal and numeric data. I think you can make the
# argument to convert the 5+ values to 5?
# first center and scale numeric predictors
los <- data$lengthofstay
data_cent_scale <- data %>%
  dplyr::select(-lengthofstay) %>%
  dplyr::mutate_if(is.numeric, scale) %>%
  dplyr::mutate_if(is.numeric, scale) %>%
  bind_cols(., lengthofstay = los)
train <- sample(1:nrow(data_cent_scale), nrow(data_cent_scale) / 2)
test <- (-train)
train_set <- data_cent_scale[train, -c(1, 2, 3, 13, 25, 26, 28)] # drop id, vdate, discharged,
# vdate_month, fibrosisandother
test_set <- data_cent_scale[test, -c(1, 2, 3, 13, 25, 26, 28)]

library(skimr)
skimmed <- skim_to_wide(data_cent_scale[, -c(1, 2, 3, 13, 25, 26, 28)])
# skimmed
```

```r
# There's a large list of model algos available in caret.
# To get the details, use
# modelLookup(<algorithm>)

knitr::kable(models <- bind_rows(
  modelLookup("rpart"),
  modelLookup("glm"),
  modelLookup("lasso"),
  modelLookup("glm.nb")
), caption = "List of models in caret")
```

```r
# set up for caretEnsemble to CV all models in one go.
library(caretEnsemble)
grid <- 10^seq(10, -2, length = 100)
trainControl_args <- trainControl(
  method = "cv",
  number = 10,
  savePredictions = "final",
  index = createFolds(
    test_set$lengthofstay,
    10
  )
)
```

```r
# createFolds sets up k folds to use across all models

algorithmList <- c(
  "lm",
  "lasso",
  "rpart",
  "glm.nb"
)
tune_list <- list(poisson = caretModelSpec(method = "glm", family = "poisson"))
# family is required for glm and must be passed via caretModelSpec (instead of algorithmList)

set.seed(578)

models <- caretList(lengthofstay ~ ., # model formula
  data = train_set, # training set
  trControl = trainControl_args, # cv params
  methodList = algorithmList, # which models
  tuneList = tune_list, # glm needs this
  continue_on_fail = FALSE # stop if something fails
)

out <- resamples(models)

options(digits = 4)
model_results <- data.frame(
  LM = min(models$lm$results$RMSE),
  POISSON = min(models$poisson$results$RMSE),
  LASSO = min(models$lasso$results$RMSE),
  RPART = min(models$rpart$results$RMSE),
  NB = min(models$glm.nb$results$RMSE)
) %>%
  pivot_longer(cols = everything(), names_to = "Model", values_to = "RMSE") %>%
  arrange(RMSE)
```

```r
NB_link_tune <- models[["glm.nb"]][["results"]]
knitr::kable(NB_link_tune, caption = "Negative binomial CV results for link function")
```

```r
# ?resamples
resamples <- resamples(models)
dotplot(resamples, metric = "RMSE")
```

```r
plot(varImp(models$rpart))
```

```r
rpart.plot::rpart.plot(models$rpart$finalModel)
```

```r
plot(varImp(models$glm.nb))
```

```r
# evaluate final models on test data
pred_lm <- predict.train(models$lm, newdata = test_set)
pred_poisson <- predict.train(models$poisson, newdata = test_set)
pred_lasso <- predict.train(models$lasso, newdata = test_set)
pred_rpart <- predict.train(models$rpart, newdata = test_set)
pred_glm <- predict.train(models$glm.nb, newdata = test_set)
pred_RMSE <- data.frame(
  LM = RMSE(pred_lm, test_set$lengthofstay),
```

```
  POISSON = RMSE(pred_poisson, test_set$lengthofstay),
  LASSO = RMSE(pred_lasso, test_set$lengthofstay),
  RPART = RMSE(pred_rpart, test_set$lengthofstay),
  NB = RMSE(pred_glm, test_set$lengthofstay)
) %>%
  pivot_longer(cols = everything(), names_to = "Model", values_to = "RMSE") %>%
  arrange(RMSE)
knitr::kable(pred_RMSE, caption="RMSE of final models across the test data")
```

```
nb_coeffs <- as.data.frame(models$glm.nb$finalModel$coefficients) %>%
  set_names("coefficient") %>%
  arrange(desc(abs(coefficient)))
knitr::kable(nb_coeffs, caption = "Negative binomial model coefficients")
```