Comparing Machine Learning Models for Insurance Ratemaking - Rough Draft

Sam Castillo

21 October 2018

Contents

Introduction

A few months ago, I sat on the first conference call with 43 other actuaries from around the U.S. to talk about how machine learning is changing the actuarial industry. The Casualty Actuarial Society (CAS) had formed a working party to research this area, and the turnout has been bigger than anyone has expected. We have since divided into two sub-groups to focus on 1) producing a worked-example of using ML, which is where I am, and 2) researching how to encorporate ML while following regulatory limitations.

The purpose of this paper is to gain practical experience creating models for auto insurance claims.

How is pricing insurance different than pricing anything else?

Right in front of me is a small English Breakfast cup of tea from Cafe Nero. This delicious hot beverage had a price of $2.49. Why was it $2.49? Why not $249.00? Why not $5.00 or $10.00? The price is $2.49 because Cafe Nero knows exactly how much it costs them to create the tea, how much they pay their hourly employees to serve the tea, the cost of rent to keep the store open, and they know that the average person is willing to pay this price. At $2.49, they believe that their profitability is highest. The price is set based on known, fixed quantities.

In pricing a cup of tea, profit is a function of fixed quantities

$$\text{Profit per Cup of Tea} = \text{Price Customer Pays} - \text{Cost to Cafe Nero}$$

If Cafe Nero was selling me insurance instead of a cup of tea, then finding the right price would be a different exercise. Let's say that Cafe Nero sold me auto insurance for the next 12-months for \$249 per month. Would they make profit off of this policy? This would depend on whether I was involved in an accident that year. In years where I was not involed in accidents, they would be profitable, but if I happened to get into an accident that cost more than the annual premium, then they would lose money.

Their profit equation would involve *random variables* instead of fixed amounts. That is, on years where the claims are higher than the premiums, they are not profitable.

$$\text{Profit per Policy} = \text{Premium} - \text{Claims}$$

Now imagine that Cafe Nero has crystal ball which could predict the future and tell whether or not I would be in an accident and exactly how much this could cost them in claims. Then they could just set their premium to be larger than this by any amount that they choose. For instance, let's say that they want to make \$50 per policy month. Then they would just set their premium to be \$50 higher than the predicted future claims.

$$\text{Profit per Policy} = 50 + \text{Predicted Claims} - \text{Annual Claims} = 50$$

This would only be true so long as Predicted Claims is a good predictiion! If the predictions were bad, then the profitability would be unstable. One of the main goals of pricing insurance is to predict future claims, and this requires building models which can determine the risk of selling coverage to policyholders.

Key Definitions

- **Exposure:** A measure of the amount of risk hazard. This is usually the length of coverage term. This can also be the number of miles driven.
- **Loss Amount:** This is the dollar amount of actual damage that my car takes.
- **Severity:** The dollar amount paid from an insurance company to the insured. In auto insurance, this is the amount the insurance company pays to repair my car.
- **Frequency:** The number of claims that are filed for a given policy period. Longer policy periods have more claims filed on average.
- **Deductible:** The amount paid by the insured when a claim is filed. For example, if my Cafe Nero Auto Insurance plan has a \$1000 deductible, then if I then if my car costs \$2000 to fix, I will pay \$1000 and Cafe Nero will pay \$1000.

Model Objective:

The goal is to estimate the risk, or cost of future claims, for each policyholder. The most common method is to seperate the losses into `Frequency`, how much the claim will cost, and `Severity`, how often claims occur.

$$E[\text{Loss}] = E[\text{Number of Claims}] \cdot E[\text{Cost per claim}] = E[\text{Frequency}] \cdot E[\text{Severity}]$$

Because `frequency` is assumed to be independent of `severity`, the expected value of the product is the product of the expected values. Another method is to model the amount of risk per unit exposure directly, known as Pure Premium $= \frac{\text{Dollars of Loss}}{\text{Exposure}}$ directly. This does not provide as much inference in to the data, so I chose to go with the first approach.

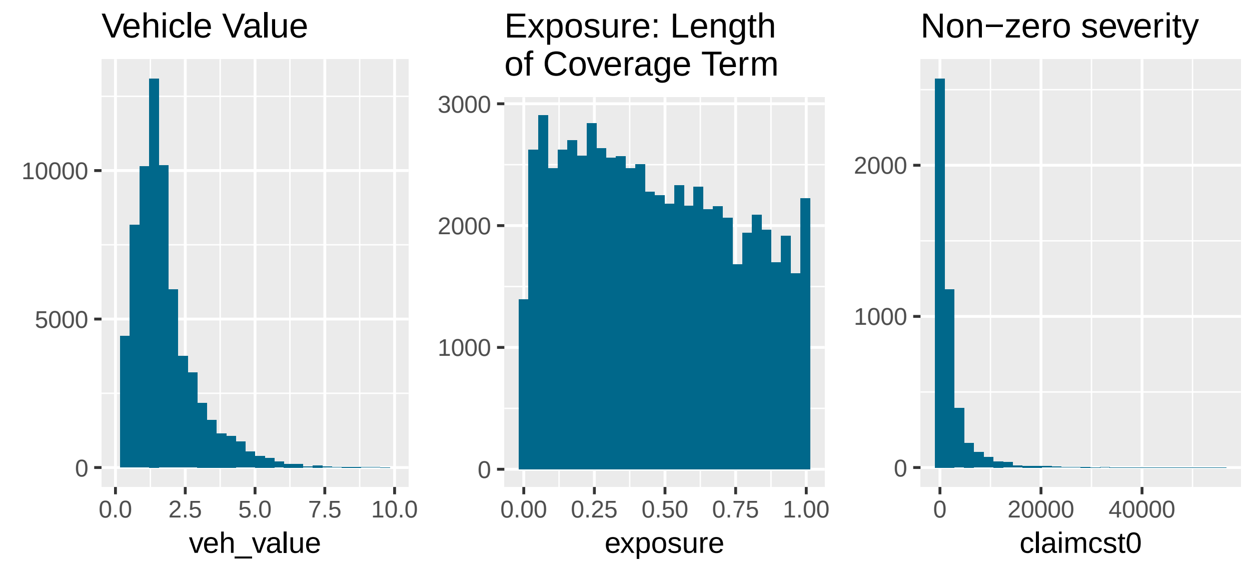
Exploratory Data Analysis (EDA)

This data set is based on one-year vehicle insurance policies taken out in 2004 or 2005. There are 67,856 policies. Each record represents a single policyholder, over a set coverage period, for a given vehicle, along with other policy-level characteristics. Anyone can access this data from the R package `insuranceData`.

Feature	Description
veh_value	The value of the vehicle in $10,000s
exposure	Percentage of year of coverage from 0-1
clm	Whether a claim was filed
numclaims	The number of claims filed
claimcst0	Claim amount (including 0 for no claim)
veh_body	vehicle body type
veh_age	1 (youngest), 2, 3, 4
gender	Gender of policyholder
area	Geographic region
agecat	1 (youngest), 2, 3, 4, 5, 6

Histograms

```r
data(dataCar)
my_theme <- scale_color_brewer(palette="PuPuGn")

car <- dataCar %>%
  mutate_at(c("clm", "agecat", "X_OBSTAT_", "veh_body"), as.factor) %>%
  #because ther are only a few roadsters, these are being grouped together as sports cars
  mutate(veh_body = as.character(veh_body),
         veh_body =  as.factor(ifelse(veh_body %in% c("RDSTR", "CONVT"),
                                      yes = "SPRT", no = veh_body)))

p1 <- car %>%
  ggplot(aes(veh_value)) +
  geom_histogram(fill = 'deepskyblue4') +
  xlim(0, 10) +
  ggtitle("Vehicle Value")  +
  theme(axis.title.y=element_blank())

p2 <- car %>%
  ggplot(aes(exposure)) +
  geom_histogram(fill = 'deepskyblue4') +
  ggtitle("Exposure: Length \nof Coverage Term") +
  theme(axis.title.y=element_blank())

p3 <- car %>%
  dplyr::filter(claimcst0 > 0) %>%
  ggplot(aes(claimcst0)) +
  geom_histogram(fill = 'deepskyblue4') +
  ggtitle("Non-zero severity") +
  theme(axis.title.y=element_blank())

ggarrange(p1, p2, p3, ncol = 3)
```



The histograms above show the shapes of the distributions. `veh_value` (left) is skewed right, as vehicles become less common as they become more expensive. The length of the exopsure period for the policy, `exposure` (center), is almost uniform which indicates that we have about the same number of 1-year policies as shorter term policies. The cost of the claims, `claimcst0` (right), is also highly right-skewed and inflated at zero because most policies do not experience an accident in any given year.

Possible Feature Interactions

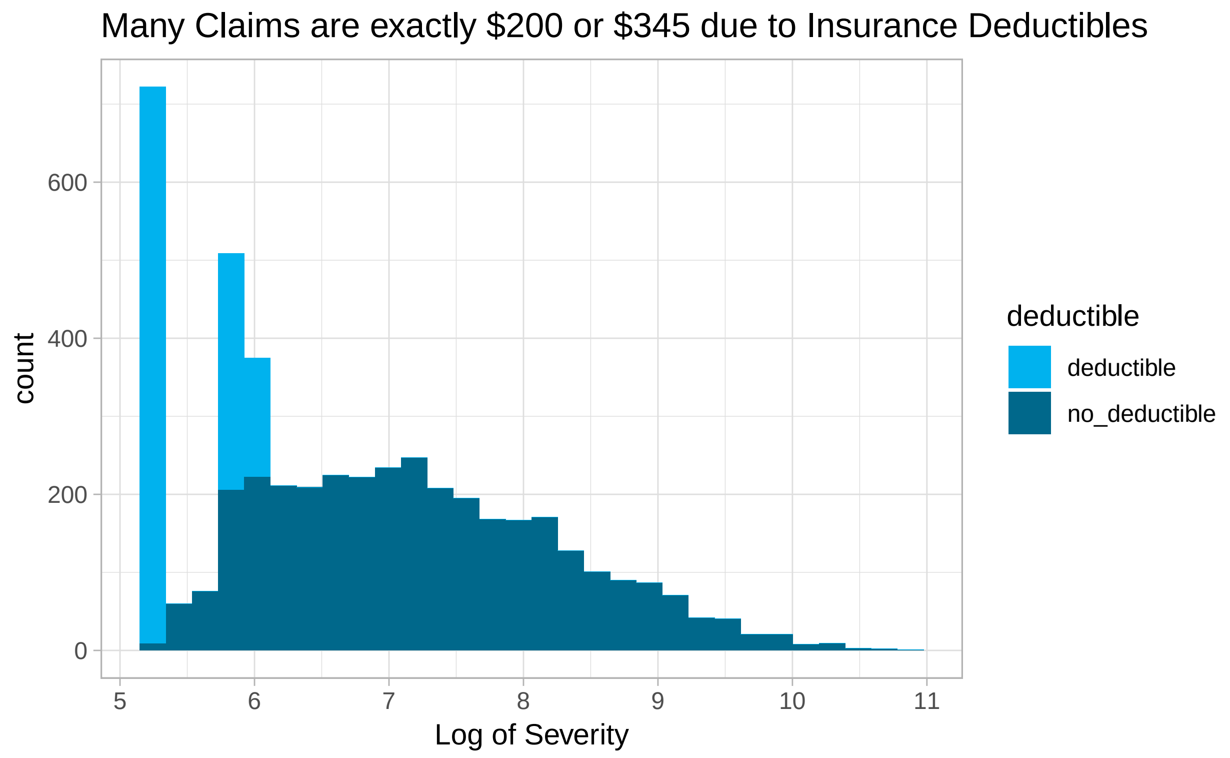
A feature interaction, or an "interaction effect" is when the level of one variable impacts the impact that another variable has on the outcome. In this example, we find that impact of a vehicle's value on the cost of claims changes depending on the vehicle type. The jargon for this is that there is an interaction effect between `veh_value` and `veh_body`. We look at graphs to see if there are any interactions.

```
severity_order <- car %>%
  group_by(veh_body) %>%
  filter(claimcst0 > 0 ) %>%
  summarise(claimcst0 = mean(claimcst0/exposure, na.rm = T)) %>%
  arrange(claimcst0) %>%
  select(veh_body) %>%
  unlist() %>%
  as.character()

severity_plot <- car %>%
  group_by(veh_body) %>%
  filter(claimcst0 > 0 ) %>%
  summarise
  group_by(claim_amount) %>%
  summarise(percent_of_claims = 100*n()/nrow(car)) %>%
  arrange(desc(percent_of_claims)) %>%
  top_n(5) %>%
  kable(., digits = 2)
```

claim_amount	percent_of_claims
0	93.19
200	1.05

claim_amount	percent_of_claims
354	0.38
390	0.22
368	0.06

```r
car <- car %>%
  mutate(deductible = as.factor(ifelse(round(claimcst0, 0) %in% c(200, 354, 390, 345) , "deductible", "

car %>%
  filter(claimcst0 > 0) %>%
  mutate(claimcst0 = log(claimcst0)) %>%
  ggplot(aes(claimcst0, fill = deductible)) +
  geom_histogram() +
  ggtitle("Many Claims are exactly $200 or $345 due to Insurance Deductibles") +
  xlab("Log of Severity") +
  scale_fill_manual(values=c( "deepskyblue2", "deepskyblue4")) +
  theme_light()
```



For a loss amount X, the distribution of the claim amount Y afer the deductible d is

$$Y = \begin{cases} X - d, 0 \leq X \leq d \\ 0, X \leq d \end{cases}$$

This is to say that Y is actually a conditional distribution! We expected the underlying loss distribution to be distributed on $[0, +\infty)$, however, the truncated data is on $[d, +\infty)$. That is to say,

$$P[Y = y] = P[X - d = x | X > d]$$

This is a very common problem in insurance data, and the author cites several methods for fixing this. The simplest approach is to use `deductible` as a new feature. We would also remove the deductibles and fit a distribution to fill in the amounts less than the deductible, after adjusting the maximum likelihood estimate for truncation. The most throrough approach is to remove these from the data and model them seperately.

Correlations

If two features are positively correlated, then one tends to be high then the other tends to be high and vice versa when it is low. When a feature is inversely correlated, an high value in the first usually occurs when there is a low value in the second. The matrix below shows how each of the numeric features are related to each other.

variable_left	veh_value	exposure	numclaims	claimcst0	veh_age
veh_value	1.000	0.042	0.003	0.002	-0.538
exposure	0.042	1.000	0.133	-0.119	-0.027
numclaims	0.003	0.133	1.000	0.108	0.014
claimcst0	0.002	-0.119	0.108	1.000	0.032
veh_age	-0.538	-0.027	0.014	0.032	1.000

This is computed using the weighted Pearson correlation, where the weights are the number of claims. The reason why this is the number of claims instead of the exposure is because we are not given individual line items for each claim. For example, if the number of claims in a policy is 4, then the number of observations for that occurence should be four.

- vehicle age is negatively correlated with value given that cars depreciate in value over time
- As the length of coverage increases, `exposure`, there is an increase in the frequency of claims.
- There is no correlation between the vehicle's age the the number of claims. We would expect that the risk of an accident does not directly depend on the age of the vehicle, but here could be non-linear relationships here. For example, vehicle's which the owner has just purchased could be at a higher risk due to the driver having less experience behind the wheel

Data Preparation

The goal of this step is to put the data into the right shape for modeling. The target of the severity model is the average loss amount per claim. We have already added indicator variables for deductibles. The author recommends taking the log transform of any continuous features if a log-link is used (see GLM section below).

```r
severity <- car %>%
  filter(claimcst0 > 0) %>%
  mutate(
    avg_loss = claimcst0 / numclaims)
```

Even insurance companies have limits to the amount of risk that they will insure. They set limits to the highest claim amount that they will cover, and can then pass the amount above these claims on the reinsurance companies. This is known as ceeding risk. The range of claim values that are covered is known as the *coverage layer.*

Another item which can improve the model outcomes is related to the factor levels. This is related to how R chooses base levels (AKA, reference levels). The author says to set these to the levels which have the largest sample sizes in order to reduce the variance of the parameters in GLMs. Fortunately, the library `forcats` has a function designed specifically to do just this in just two lines of code!

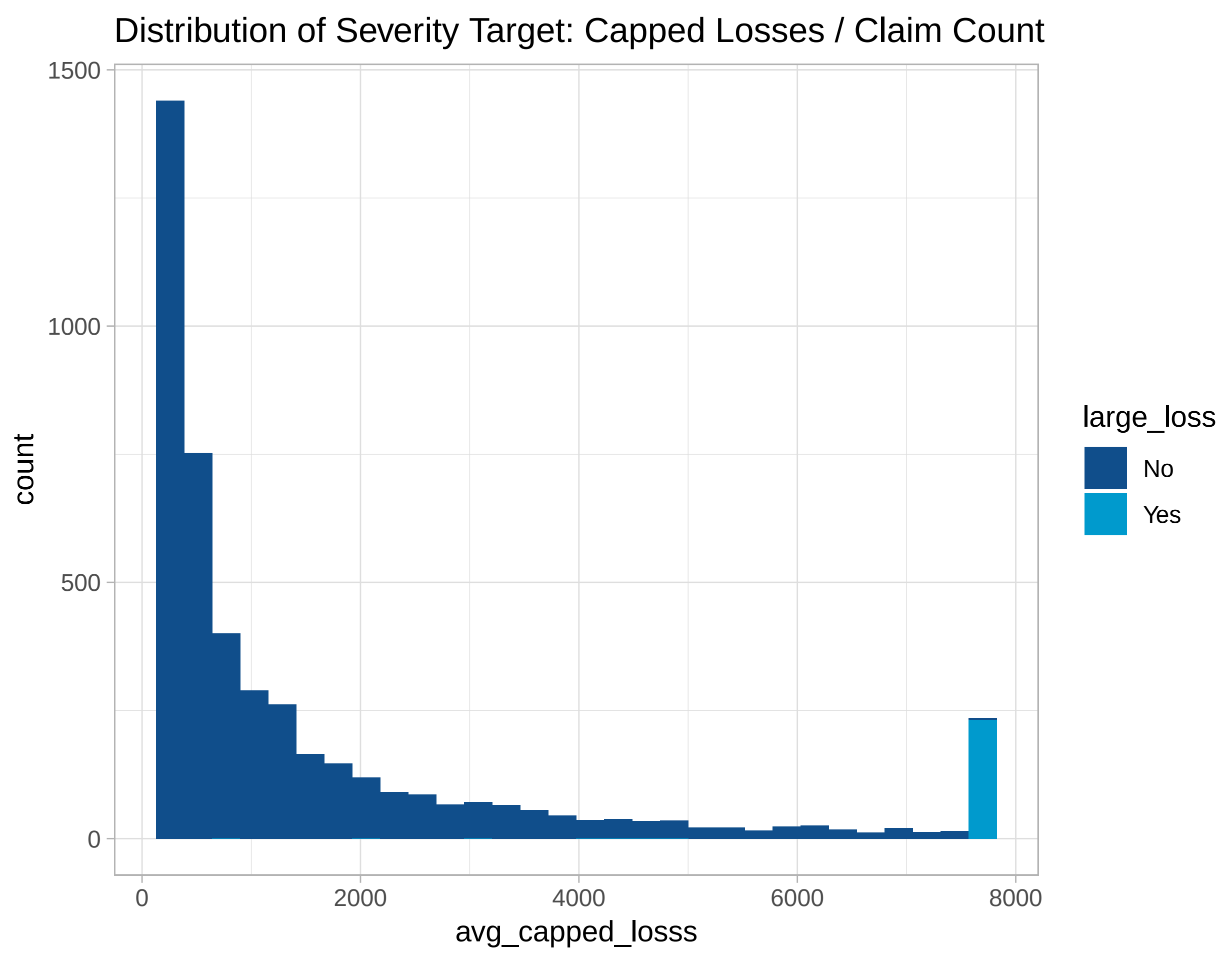
```r
car <- car %>%
  mutate_if(is.factor, fct_infreq)
```

When we look at the data we see that there are several losses which are much larger than the others. If these are left in the model as-is, they will have too much influence on the model's parameters and will result in a poorer fit. These are being capped at the 90% quantile.

```r
avg_loss_cap <- quantile(severity$avg_loss, 0.95)

severity <- severity %>%
  mutate(
    avg_capped_losss = ifelse( avg_loss < avg_loss_cap, avg_loss, avg_loss_cap),
    large_loss = as.factor(ifelse(avg_loss > avg_loss_cap, "Yes", "No"))) %>%
  select(-clm, -X_OBSTAT_) # drop features which provide no additional information

severity %>%
  select(large_loss) %>%
  summary()
```

```
##  large_loss
##  No :4392
##  Yes: 232
```

```r
severity %>%
  ggplot(aes(avg_capped_losss, fill = large_loss)) +
  geom_histogram() +
  ggtitle("Distribution of Severity Target: Capped Losses / Claim Count") +
  scale_fill_manual(values=c("dodgerblue4","deepskyblue3")) +
  theme_light()
```



Model Design

Finally, we have enough information to build a clean data set for modeling. This will help to smooth out the data to make it easier to model. To summarise, the changes that we have made have been

- Capped losses at the 95% percentile of non-zero claims. We will model large losses seperately
- Removed claims at deductible amounts. We will also model deductible losses seperately.
- Split the data into a set for modeling `severity`, which excludes zero-pay claims, and for frequency, which includes all claims

The model design will be as follows

$$\text{E[Future Losses]} = \text{E[Frequency]E[Severity]} + \text{E[Number of deductible losses][Deductible]} + \text{E[Number of capped losses][Cap]}$$

There will be four final models selected: frequency, severity, and count models for the number of deductible claims and the number of capped claims For the count models we can use Poisson regression.

We split the data into a training and testing set.

```r
set.seed(1)
severity_train_index <- createDataPartition(
  severity$avg_capped_losss, times = 1, p = 0.7, list = FALSE) %>% as.vector()
```

```r
train_severity <- severity %>%
  slice(severity_train_index) %>%
  filter(deductible == "no_deductible", large_loss == "No")

test_severity <- severity %>%
  slice(- severity_train_index) %>%
  filter(deductible == "no_deductible", large_loss == "No")

frequency_train_index <- createDataPartition(
  car$numclaims, times = 1, p = 0.7, list = FALSE) %>% as.vector()

train_frequency <- car %>%
  slice(frequency_train_index)

train_frequency <- car %>%
  slice(- frequency_train_index)
```

GLM Models for Severity

Theory

To explain Generalized Linear Models (GLMs), let's first start with an explanation of a simpler version, Ordinary Least Squares (OLS).

The item we are trying to prdict is Y, and the data that we have is a matrix X where each column is a feature. For a given data set X, we want to be able to make an accurate guess as to Y. The error of the guess is $\mathcal{E}$, which is centered at zero.

The first assumption is that Y is a linear function of the features. That is, using the data, we can make predictions about X using only addition and multiplication.

$$Y = X\beta + \mathcal{E}$$

Because X and β are both constants, their expected value is just themselves. The error term has expected value zero because we expect the estimates to "miss" above and below the target an equal number of times.

$$\mu = E[Y] = E[X\beta] + E[\mathcal{E}] = X\beta$$

The value β is determined by minimizing the error term. Using calculus, this is as simple as taking the derivative of $(Y - E[Y])^T(Y - E[Y])$ with respect to β, setting it equal to zero, and solving.

The second main assumption is that the response y is normally distributed That is, $\dagger \sim N(\mu, \sigma^2)$.

$$P[Y = y|X] = f(y;\mu) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp(-\frac{(y-\mu)^2}{2\sigma^2})$$

Making this above guess at the shape of Y has little basis in reality and is for convenience. To make a GLM, all that we need is to relax the previous two assumptions. Instead of assuming that Y is directly a linear function of X, we allow for certain functions to connect the two. We call this the *link* function $g(.)$.

$$g[E[Y]] = X\beta$$

We do still place limits as to what $g(.)$ can be. In fact, often times this is just the natural logarithm. This is because $log(.)$ has several nice properties, such as the fact that positive values stay positive and the variance decreases.

When relaxing the constraing on the shape of y, we allow for any member of the exponential family of distributions. This includes any random variable with a PDF which can be written as $e^{\text{Something}}$. For example, if modeling count data, a Poisson distribution is the common fit, and we can write the Poisson's PDF in expoential form with some clever algebra. For $y \sim Pois(\lambda)$,

$$f(y; \lambda) = \frac{\lambda^y e^- \lambda}{y!}$$

Recall that $e^{log(\text{Something})} = \text{Something}$ which implies

$$f(y; \lambda) = \exp(-\lambda + ylog(\lambda) - log(y!))$$

And this is of the exponential form $e^{\text{Something}}$, which means that we can model Poisson distributed Y's. Other distributions in the exponential family are the binomial, normal, gamma, negative binomial, and the Tweedie, which is used specifcally in insurance claims contexts.

Model Fitting

We test out many combinations of models and evaluate them using the performance on the holdout set. Traditionally in insurance, the gamma distribution is used as a the response distribution for claim severity.

```r
#using only the mean response
glm_baseline <- glm(
  formula = avg_capped_losss ~ 1,
  family = Gamma(link = "log"),
  data = train_severity,
  weights = train_severity$numclaims)

#using only the length of time of coverage
vars_1 <- "exposure"
glm_severity_1 <- glm(
  formula = avg_capped_losss ~ exposure,
  family = Gamma(link = "log"),
  data = train_severity,
  weights = train_severity$numclaims)

vars_2 <- "exposure, log(veh_value)"
glm_severity_2 <- glm(
  avg_capped_losss ~ exposure + log(veh_value + 0.001),
  family = Gamma(link = "log"),
  data = train_severity,
  weights = train_severity$numclaims)

vars_3 <- "exposure, log(veh_value), veh_age, gender"
glm_severity_3 <- glm(
  avg_capped_losss ~ exposure + log(veh_value + 0.001) + veh_age + gender,
  family = Gamma(link = "log"),
  data = train_severity,
  weights = train_severity$numclaims)
```

```r
vars_4 <- "exposure, log(veh_value), numclaims, veh_age, gender"
glm_severity_4 <- glm(
  avg_capped_losss ~ log(veh_value + 0.001) + exposure + numclaims + veh_age + gender,
  family = Gamma(link = "log"),
  data = train_severity,
  weights = train_severity$numclaims)

vars_5 <- "veh_value, exposure, numclaims, veh_age, gender, area"
glm_severity_5 <- glm(
  avg_capped_losss ~ veh_value + exposure + numclaims + veh_age + gender + area,
  family = Gamma(link = "log"),
  data = train_severity,
  weights = train_severity$numclaims)
```

Creating lists of models makes comparisons easier.

```r
severity_glms <- list(glm_severity_1, glm_severity_2, glm_severity_3, glm_severity_4, glm_severity_5)
```

Using the package `broom` from the Tidyverse, we can quickly create a data table with these model summaries. Based on this output, we see that the best fit is in the 4th model, the one with the parameters `log(veh_value) + exposure + numclaims + veh_age + gender`. The p-values associated with `area` where all large, so this is not being included in the model.

We compare all of the models based on several fitting criteria. These are

Deviance: This is a generalization of the sum of squares to GLMs which compares the likelihood of the current model to the likelihood of a saturated model with all parameters included.

For the predicted value μ of response y, the deviance D is

$$D(y, \mu) = 2(log(p(y|\theta_s) - log(y|\theta_0)))$$

Where θ_s is the the set of parameters of the fully saturated model. This is the model whic fits the data perfectly by having one parameter for each observation. θ_0 is the current set of parameters for the model being tested.

Akaike Information Criterion (AIC): AIC tries to select a model which matches an unknown, high-dimensional reality. A lower AIC is better. The formula is

$$\text{AIC} = 2k - log(L)$$

where k is the number of parameters and L is the value of the maximum value of the likelihood function for the model. AIC penalizes overfitting because as more features are added to the model, the number of parameters k increases, which results in a higher AIC.

Bayesian Information Criterion (BIC): This assumes that within the set of candidate models is a "true" model. Although this is an unrealistic assumption it does allows us to compare models side-by-side. Just like AIC, lower is better. The formula for BIC is

$$\text{BIC} = log(n)k - 2log(L)$$

There is a lot of debate about whether AIC or BIC should be used. A good rule of thumb is to use both, and if they disagree in that AIC is larger but BIC smaller or vice versa, to use an alternative method to differentiate such as cross-validation.

In the summary below, `model 4` has the best fit. Although the Deviance is slightly lower in `model 5`, the `area` variable adds an 4 parameters, which is they the AIC and BIC are lower.

```r
GLM_summaries <- data_frame(features = c(vars_1, vars_2, vars_3, vars_4, vars_5)) %>%
  cbind(
    severity_glms %>%
    map_dfr(glance) %>%
    select(-null.deviance, -df.null, -logLik)
  ) %>%
  mutate(final_model = ifelse(row_number() == 4, 1, 0))

GLM_summaries %>%
  kable()
```

features	AIC	BIC	deviance	df.residual	final_model
exposure	41644.83	41662.01	1985.871	2261	0
exposure, log(veh_value)	41628.59	41651.49	1972.994	2260	0
exposure, log(veh_value), veh_age, gender	41623.74	41658.08	1966.771	2258	0
exposure, log(veh_value), numclaims, veh_age, gender	41597.48	41637.55	1947.027	2257	1
veh_value, exposure, numclaims, veh_age, gender, area	41594.47	41663.16	1937.997	2252	0

```r
summary(glm_severity_4)
```

```
##
## Call:
## glm(formula = avg_capped_losss ~ log(veh_value + 0.001) + exposure +
##     numclaims + veh_age + gender, family = Gamma(link = "log"),
##     data = train_severity, weights = train_severity$numclaims)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2118  -0.9762  -0.4303   0.3114   3.4592
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)             7.58722    0.10025  75.686   <2e-16 ***
## log(veh_value + 0.001) -0.06746    0.03689  -1.829   0.0675 .
## exposure               -0.15729    0.08004  -1.965   0.0495 *
## numclaims              -0.20262    0.04428  -4.576    5e-06 ***
## veh_age                 0.05299    0.02380   2.227   0.0261 *
## genderM                 0.05181    0.04097   1.265   0.2061
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Gamma family taken to be 1.001899)
##
##     Null deviance: 1995.7  on 2262  degrees of freedom
## Residual deviance: 1947.0  on 2257  degrees of freedom
## AIC: 41597
##
## Number of Fisher Scoring iterations: 6
```

In order to compare GLMs with GBMs in the next section, we need a common metric. We do so with the Root Mean Squared Error evaluated on the testing set. As a check, we can comopare the RMSE with our final GLM with that of a model with no coefficients, the intercept-only model. As we expected, there is reduction in error.

```r
#Get the root mean squared error for a glm from test data.
get_GLM_RMSE <- function(input_model, test_data) {
  mu <- predict.glm(input_model, test_data, type = "response", se.fit = T)$fit
  y <- test_data$avg_capped_losss
  #return the square root of the squared error
  sqrt(sum((y - mu)^2/length(y)))
}

get_GLM_RMSE(input_model = glm_severity_4, test_data = test_severity)
```

```
## [1] 1626.97
```

```r
get_GLM_RMSE(glm_baseline, test_data = test_severity)
```

```
## [1] 1640.889
```

GBMs for Severity

Gradient Bosted Machines (GBMs) are one of the best off-the-shelf models in the world (Elements of Statistical Learning, 302). There are many variantions of GBM algorithms, but the overall structure is consistent and looks as follows.

1. Initialize the observation weights $w_i = \frac{1}{N}, i = 1, 2, ..., N$
2. For $m = 1$ to M:

a. Fit a weak learner $G_m(x)$ to the training data using weights for the observations w_i
b. Compute an error rate for this learner which measures how well the weak learner predicts on the test set
c. Use this error rate to update the weights $\{w_1, ..., w_N\}$

3. Output a final set of predictions $G(x)$ as a linear combination of $\{G_1(x), ..., G_m(x)\}$

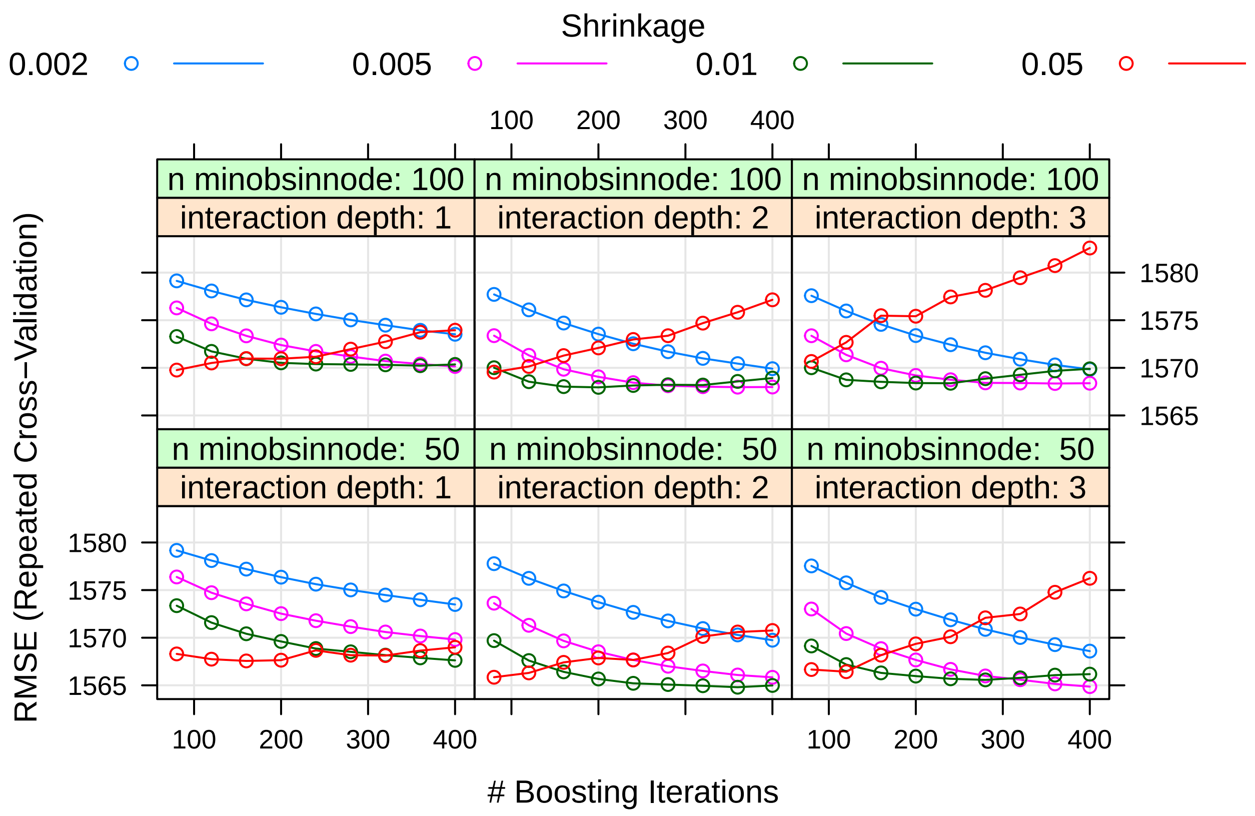
The "weak learner" $G(.)$ is usually a regression or classification tree. The tuning parameters for a GLM are as follows.

- The number of iterations, i.e. trees, which is m, defined as `n.trees`
- The complexity of the tree, called the interaction depth, `interaction.depth`.
- The learning rate: how quickly the algorithm adapts, called `shrinkage`. The lower the learning rate, the better the model, but the more trees are required, and for each additional tree that is added there is a diminishing return of added performance. Compute power is the main limitaiton to the number of trees that can be used.
- The minimum number of training set samples in a node to commence splitting, `n.minobsinnode`. If this is too small, then the model will overfit to specific observations; too high and the model will miss picking up signal.

The model tuning process involves computing many combinations of these parameters and choosing the model with the highest performance on the test data. We test out several values of the input parameters and evaluate the cross-valideated RMSE. After testing out different combinations with a grid search, I found that 400 trees with an interaction depth of 2, a learning rate of 0.01, and 50 minimum observations per node to give the best results on the test set.

```r
fitControl <- trainControl(## 10-fold CV
                           method = "repeatedcv",
                           number = 10,
                           ## repeated ten times
                           repeats = 2)
```

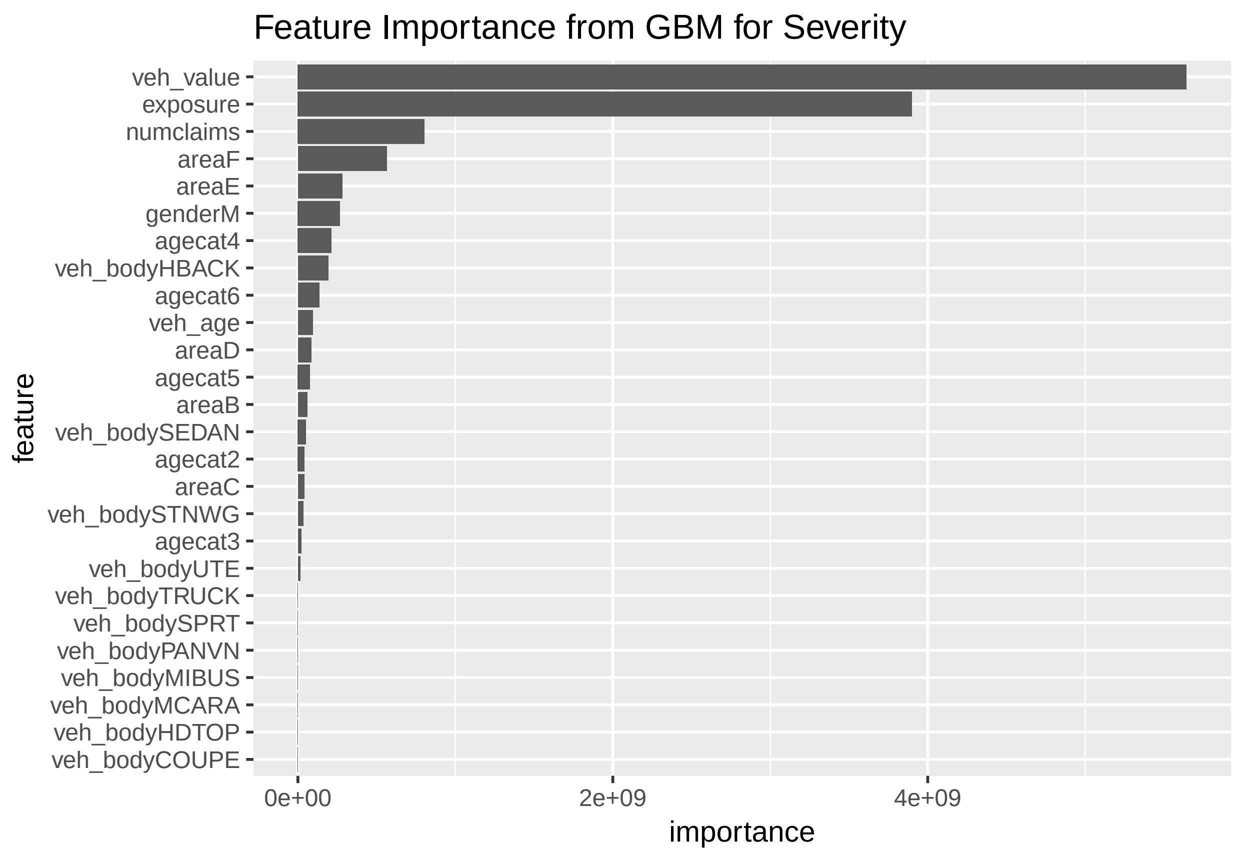
```r
gbmGrid <-  expand.grid(interaction.depth = c(1, 2), #most of the time this is sufficient
                        n.trees = (2:10)*40,
                        shrinkage = c(0.002, 0.005, 0.01, 0.05),
                        n.minobsinnode = c(50, 100))

#
# gbm_severity <- train(
#   avg_capped_losss ~ veh_value + exposure + veh_age + gender + area + agecat +
#     numclaims + veh_body,
#   data = train_severity,
#   method = "gbm",
#   trControl = fitControl,
#   tuneGrid = gbmGrid,
#   verbose = FALSE)

#test save and reload
# saveRDS(gbm_severity, "gbm_severity.rds")
# rm(gbm_severity)
gbm_severity <- readRDS("gbm_severity.rds")
#saveRDS(gbm_severity, file)
plot(gbm_severity)
```



We can compare the predictive power of the different features from the GBM by looking at the variable importance. This is based on the number of times that a variable is used for splitting, and weighted by the squared improvement in the model as a result of the split, and averaged over all the trees Elith et al. 2008, A working guide to boosted regression trees veh_value and exposure are the two most influential features. The value of the vehicle being influential makes sense because expensive cars are more expensive to repair. It is not clear to me why the lenth of coverage would be predictive of the average claim amount. See the dependency plots below. We also see that numclaims and areaC, whether or not the policyholder is in area "C", wherever that is, are important.

Model Interpretation

```
#there must be a better way of getting these factor levels be sorted on the graph...
importance_order <- data_frame(
  feature = rownames(varImp(gbm_severity$finalModel)),
  importance = varImp(gbm_severity$finalModel)$Overall) %>%
  arrange(importance) %>%
  select(feature) %>%
  unlist() %>%
  as.character()

data_frame(
  feature = rownames(varImp(gbm_severity$finalModel)),
  importance = varImp(gbm_severity$finalModel)$Overall) %>%
  mutate(feature = fct_relevel(feature, importance_order)) %>%
  ggplot(aes(feature, importance)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  ggtitle("Feature Importance from GBM for Severity")
```

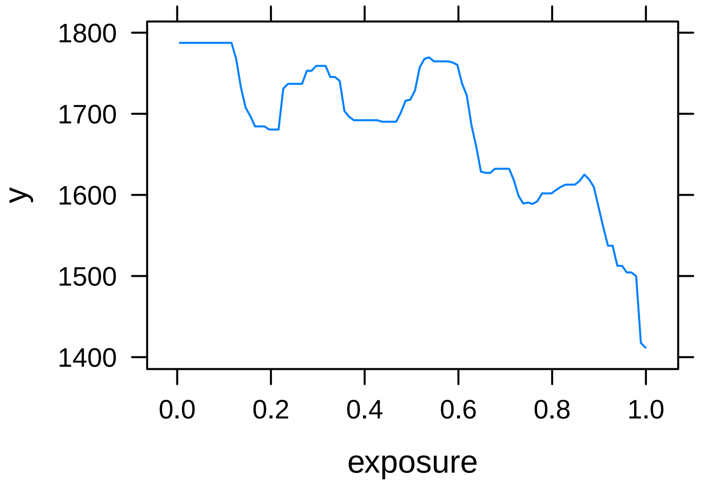


The partial dependence plots show the impact of the predictors on the data. From the `exposure` graph, you can see a seasonal pattern in the fraction of the year of coverage. Is this because once a driver is in an accident they are less likeloy to cancel their coverage?

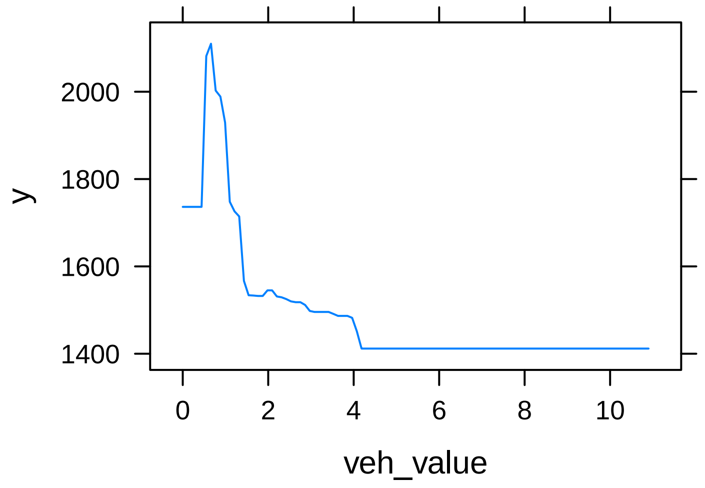
Is this because policyholders who have been insured longer are in more severe accidents or less severe accidents? Perhaps riskier drivers buy longer coverage terms, or maybe commercial drivers who are constantly on the road have policies which are auto-reknewed by their employer?

- **Exposure**: There is a cyclycal pattern in the risk of an accident as exposure, which measures a fraction of a 12-month policy year, increase. Do these dates refer to the time that a claim was reported to the insurance company or the date of the accident? We know from the data description that all of these policies were taken out in 2004 or 2005. Is this exposure pattern capturing a seasonal trend? We need more information in order to answer this.
- **Vehicle Value**: Vehicles valued less than $20,000 are at a higher risk of accident. We are not given information as to whether these claims include bodily injury. Could it be that because cheaper cars are lighter weight and have fewer safty features that these drivers are at a higher risk of sustaining personal injury?
- **Vehicle Age**: Drivers are careful when driving new cars. For vehicles less than 1.5 years old, there is lower risk of an accident.

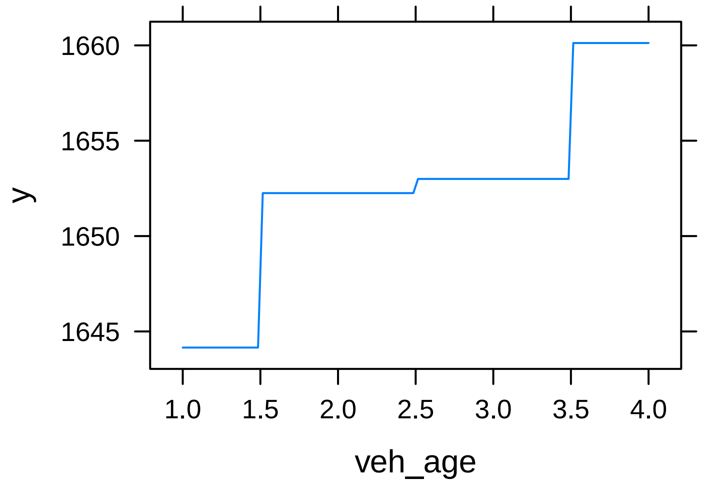
```
vars <- c("exposure", "veh_value", "numclaims", "areaC", "genderM", "veh_age")
plot.gbm(gbm_severity$finalModel, i.var = c("exposure"))
```



```
plot.gbm(gbm_severity$finalModel, i.var = "veh_value")
```



```r
plot.gbm(gbm_severity$finalModel, i.var = "veh_age")
```



We can compare the GLM with the GBM using RMSE evaluated on the holdout set. The GBM has a lower RMSE.

```r
#Get the root mean squared error for a glm from test data.
get_GBM_RMSE <- function(input_model, test_data) {
  mu <- predict.train(input_model, test_data)
  y <- test_data$avg_capped_losss
  #return the square root of the squared error
  sqrt(sum((y - mu)^2/length(y)))
}

data_frame(`GLM RMSE` = get_GLM_RMSE(glm_severity_4, test_data = test_severity),
           `GBM RMSE` = get_GBM_RMSE(gbm_severity, test_severity)) %>%
  kable(., digits = 3)
```

GLM RMSE	GBM RMSE
1626.97	1620.321

Model for Frequency

We create the target variable as the average number of claims per unit of exposure. Mathematically, this is equivalent to adding an offset term. An offset is formally defined as a predictor whose coefficient is constrained to be 1.

$$\log\left(\frac{\text{Number of Claims}}{\text{Exposure}}\right) = X^T\beta \rightarrow \log(\text{Number of Claims}) = X^T\beta + \log(\text{Number of Claims})$$

```r
frequency_1 <- glm(
  formula = numclaims ~ exposure,
  family = poisson,
  data = car,
  offset = log(exposure))

summary(frequency_1)
```

To be continued…

References:

Casualty Actuarial Society GLM Paper: https://www.casact.org/pubs/monographs/papers/05-Goldburd-Khare-Tevet.pdf

Casualty Actuarial Society Presentation: https://www.casact.org/education/rpm/2012/handouts/Session_4738_presentation_1068_0.pdf

https://www.casact.org/education/rpm/2012/handouts/Session_4736_presentation_895_0.pdf