***(Word count excluding code snippets & documentation table: 2,413)***

# **Introduction**

This case study project was related to predicting fraudulent claims in a claims dataset while utilizing driver information, policy variables, & vehicle data. Fraudulent claims make up ~16% of the claims so they are relatively sparse.

My approach will be outlined in detail below, but a large focus of mine was placed on cleaning the data (which had many issues) and creating new variables based off the claim notes. These both ended up being important when for the model performance.

For the modeling itself, a few different classification models were attempted, and, in the end, a lasso classification model was used for reasons that will be explained in more detail below.

All preparation, data analysis, & modeling building were performed in R.

# **Data Preparation and Exploratory Data Analysis**

As mentioned above, data preparation was an important step in the analysis. The first part of this was looking at the 4 different data sets. The first step was to analyze the variables that were going to be used in the joins to the claim dataset:

#########################################################  
## Data Checking & Joining  
#########################################################  
## Going to comment out sections that were previously used for checking  
## Can uncomment those sections to see the checks  
  
#### Policy Data Pre-Join Check  
## Checking the uniqueness of the variables that we are going to be joining on  
raw\_policy\_data %>%  
 count(policy\_id) %>%  
 filter(n > 1) %>%  
 arrange(desc(n))

## # A tibble: 142 x 2  
## policy\_id n  
## <chr> <int>  
## 1 P101444063304 2  
## 2 P102556840894 2  
## 3 P102703578880 2  
## 4 P103104864603 2  
## 5 P104193703875 2  
## 6 P104406377720 2  
## 7 P104953093800 2  
## 8 P105429252423 2  
## 9 P106032254976 2  
## 10 P106094769084 2  
## # i 132 more rows

*Note that many of these checking sections are commented out in the final\_code.R that was submitted as they slowed down performance (due to the use of the View() function). They can be uncommented as necessary by highlighting the rows and hitting Ctrl + Shift + C.*

For the policy (as well as vehicle, driver, & claims) datasets, the ID fields were not unique. This would cause issues/duplicates when joining the datasets together.

I looked at a sample Policy ID from this list and manually checked that there were no differences between any of the fields in the dataset (i.e. these were pure duplicates and not being caused by having different values for certain fields).

I double checked this by confirming that the unique function removed all duplicates i.e:

## Using unique fixes the issue  
raw\_policy\_data %>%  
 unique() %>%  
 count(policy\_id) %>%  
 filter(n > 1) %>%  
 arrange(desc(n))

## # A tibble: 0 x 2  
## # i 2 variables: policy\_id <chr>, n <int>

Since this solved this issue for all the datasets, unique versions of the policy, claim, vehicle & driver datasets were used when joining the datasets together.

## **Missing Variables**

In terms of missing variables (that were in the documentation but not the datasets):

1. Claims Data
   1. Report\_date
2. Driver Data
   1. Cell\_Usage
3. Policy Data
   1. Altitude
   2. Foggy\_Days
4. Vehicle Data
   1. Left\_Mile
   2. Braking\_Mile

Given that these fields are also missing from the assessment dataset, I simply moved forward without them.

## **Claim Notes**

The claim notes field had some extremely useful information within it. However, to make use of this information, the data had to be transformed into a more usable format. To do this, I first noticed that the formats of the notes were fairly consistent and written as separate sentences. B/c of this, I broke the notes apart separately by sentence:

#########################################################  
## Claim Notes  
#########################################################  
## Digging into the claim notes to try to text mine/find predictive phrases  
## Going to separate the notes by sentence  
separated\_claim\_data <- combined\_data %>%   
 select(claim\_id ,claim\_notes) %>%   
 separate\_wider\_delim(claim\_notes   
 ,delim = "."   
 ,names\_sep = "\_"  
 ,too\_few = "align\_start"  
 ) %>%   
 mutate\_at(vars(contains("claim\_notes")) ,str\_trim) %>%   
 mutate\_at(vars(contains("claim\_notes")) ,function(x) {if\_else(x == "" ,NA ,x)})  
  
  
## Joining the notes back onto the original dataset  
combined\_data <- combined\_data %>%   
 left\_join(separated\_claim\_data ,by = "claim\_id")

Once the notes were separated by sentence, I could mine the text more easily and search for phrases in the data. The main things searched for were:

1. Whether a pedestrian or cyclist was involved
2. Whether there were police (&/or medical) involvement
3. Details about the injury
4. Extent of the injury
5. Triple exclamation points
6. Any notes about fraud
7. If the vehicle listed in the claim note matched the make in the vehicle data
8. The introductory phrase of the claim note

Many of these were not predictive, but some were! In particular, the notes about fraud and the triple exclamation points were very useful. The methodology was to search for these in each sentence and then coalesce them together at the end to create a new variable. Just doing a simple one way analysis on the variables:

## Going to see how variables are correlated with fraud  
check\_fraud\_pct <- function(x) {  
 combined\_data %>%   
 group\_by(!!sym(x)) %>%   
 summarise(n = n()  
 ,fraud\_ind = sum(fraud\_ind)  
 ,fraud\_pct = fraud\_ind / n) %>%   
 arrange(desc(n))  
}  
  
check\_fraud\_pct("triple\_exclamation") ## Extremely predictive ... although few data points

## # A tibble: 2 x 4  
## triple\_exclamation n fraud\_ind fraud\_pct  
## <dbl> <int> <dbl> <dbl>  
## 1 0 103319 16326 0.158  
## 2 1 782 0 0

check\_fraud\_pct("fraud\_note\_ind") ## Extremely predictive ... although few data points

## # A tibble: 5 x 4  
## fraud\_note\_ind n fraud\_ind fraud\_pct  
## <fct> <int> <dbl> <dbl>  
## 1 no mention of fraud 96335 14482 0.150   
## 2 no signs of fraud 4178 564 0.135   
## 3 suspected fraud 2027 916 0.452   
## 4 probably not fraud 1292 95 0.0735  
## 5 fraud 269 269 1

For the triple\_exclamation, this is a 1 if the entire sentence in the claim note is “. !!! .” i.e. there are no other words in the sentence. Every single “. !!! .” ended up not being fraudulent – extremely valuable information to have!

An important distinction in the claim notes is that the phrase “Frauuuuud!!!” often appears. In terms of where this would appear above, the “fraud\_note\_ind” field captures this as “fraud”. This methodology differentiated b/w a “. !!! .” sentence and a “Frauuuuud!!!” statement, two of the most important variables in the model. Because of this separation both fields were extremely valuable & predictive.

While this makes it sound like all of the claim note fields were extremely useful, this was far from the case; there were some swings & misses!

Fields created such as the “vehicle\_mismatch\_flag” (which is 1 if the vehicle in the claim notes <> the make & model in the vehicle dataset) was a complete waste of time; there were no mismatches. (The theory behind this was that a claims examiner could be “in” on the fraud and embellish the claim notes with vehicle information that does not actually match the insured’s vehicle – luckily this never happened).

Other fields that were created to little success were: the presence (or lack thereof) of police/medical staff, the mention of drugs or alcohol, or the type of pedestrian involved in the accident (pedestrian, cyclist, or other). These all seemed potentially interesting but yielded little differentiation (with “differentiation” being judged by simple one-way checking using the “check\_fraud\_pct” function above. This check, while imperfect due to not considering correlations, being a one-way analysis, etc… was a good back-of-the-envelope check while exploring the claim notes. The fields were further examined in the modeling phase).

The exact methodology of the claim\_notes parsing can be seen in the code itself (in the “Claim Notes” section), but it was largely leveraging the “stringr” package and its regex functionality for detecting & extracting the patterns outlined above.

Overall, working with the “claim\_notes” field was labor intensive, but ended up yielding important insights into the claims and being a lift for the model’s performance.

## **Exploratory Analysis & Data Cleaning**

Now that the new claim notes variables had been created, the next step was to do some checking on the originally provided fields. To do this, I worked with the documentation to understand the possible values for each field in the data.

I performed this step in 3 sections: (1) Numeric fields (2) Character fields and (3) Date fields; these were separated for ease of checking. For example, for numeric fields I wanted to be able to bucket certain fields to better understand their relationship to fraud (whereas for character fields, I wanted to see each level in the data).

For the numeric checking, it soon become clear that certain fields had values which were being repeated too often (even if it fell into an “acceptable” range based off the documentation):

## Checking the numeric cols  
check\_fraud\_pct\_num2("num\_drivers") ## less drivers seems more fraudulent ... fix those with 17

## # A tibble: 11 x 4  
## num\_drivers n fraud\_ind fraud\_pct  
## <dbl> <int> <dbl> <dbl>  
## 1 1 36936 6485 0.176  
## 2 2 26695 4101 0.154  
## 3 3 19502 2820 0.145  
## 4 4 10753 1487 0.138  
## 5 5 4735 630 0.133  
## 6 6 1736 226 0.130  
## 7 7 480 60 0.125  
## 8 8 121 16 0.132  
## 9 9 15 2 0.133  
## 10 10 3 1 0.333  
## 11 17 3098 495 0.160

check\_fraud\_pct\_num2("clms\_flt1") ## Seems very predictive ... 1+ is very fraudulent ... fix those with 17

## # A tibble: 9 x 4  
## clms\_flt1 n fraud\_ind fraud\_pct  
## <dbl> <int> <dbl> <dbl>  
## 1 0 86967 11918 0.137  
## 2 1 8510 2036 0.239  
## 3 2 3701 1161 0.314  
## 4 3 1303 507 0.389  
## 5 4 390 156 0.4   
## 6 5 86 42 0.488  
## 7 6 18 8 0.444  
## 8 7 1 0 0   
## 9 17 3098 495 0.160

check\_fraud\_pct\_num2("viol\_mjr2") ## Seems very predictive ... 1+ is very fraudulent ... fix those with 17

## # A tibble: 9 x 4  
## viol\_mjr2 n fraud\_ind fraud\_pct  
## <dbl> <int> <dbl> <dbl>  
## 1 0 87079 11757 0.135  
## 2 1 8400 2118 0.252  
## 3 2 3736 1185 0.317  
## 4 3 1272 518 0.407  
## 5 4 391 199 0.509  
## 6 5 86 45 0.523  
## 7 6 15 8 0.533  
## 8 7 1 1 1   
## 9 17 3098 495 0.160

The “num\_drivers”, “clms\_flt1”, & “viol\_mjr2” fields all had a value of “17” stand out. Given the declining volume and then large spike at 17, I assumed these “17” values were incorrect and set those values to NA (even though the field was technically possible based off the documentation). Similar adjustments were made for other numeric fields:

## Fixing the columns that were identified as having issues above  
combined\_data <- combined\_data %>%   
 mutate(num\_drivers\_new = na\_if(num\_drivers ,17)  
 ,clms\_flt1\_new = na\_if(clms\_flt1 ,17)  
 ,late\_90d\_new = na\_if(late\_90d ,500)  
 ,outs\_bal\_new = na\_if(outs\_bal ,99)  
 ,viol\_mjr2\_new = na\_if(viol\_mjr2 ,17)  
 ,time\_bet10pm2am\_new = na\_if(time\_bet10pm2am ,0.75)  
 ,credit\_score\_new = na\_if(credit\_score ,1.5)  
 ,report\_lag\_new = na\_if(report\_lag ,99)  
 ,pop\_density = na\_if(pop\_density ,5) %>% as.factor()  
 )

In terms of the character column fixes, these were largely issues with having multiple values in the data which were not in the documentation. For example, when looking at gender:

#########################################################  
## Exploratory Data Analysis - Character Columns  
#########################################################  
## Checking their relationships to fraud  
check\_fraud\_pct\_num2("gender") ## Need to fix these fields

## # A tibble: 6 x 4  
## gender n fraud\_ind fraud\_pct  
## <chr> <int> <dbl> <dbl>  
## 1 Boy 8220 1298 0.158  
## 2 F 7626 1201 0.157  
## 3 Female 34727 5375 0.155  
## 4 Girl 7609 1189 0.156  
## 5 M 8348 1357 0.163  
## 6 Male 37547 5903 0.157

The same gender (male or female) appears in multiple rows with a different label (i.e. “Boy”, “M” & “Male”). Similar issues were seen in other columns such as “marital\_status”, “limits”, & “income”. The code for the corrections (& subsequent conversion into factors) can be seen below and also in the code itself:

## Fixing the inconsistent character fields  
combined\_data <- combined\_data %>%   
 mutate(gender\_new = case\_when(gender %in% c('Boy' ,'M' ,'Male') ~ 'Male'  
 ,gender %in% c('F' ,'Female' ,'Girl') ~ 'Female') %>%   
 factor(levels = c('Male' ,'Female'))  
 ,education\_new = factor(education ,levels = c('Some High School' ,'High School or GED'  
 ,'Bachelors' ,'Masters' ,'Doctorate'))  
 ,limits\_new = str\_replace\_all(limits ,pattern = "k" ,replacement = "000") %>%   
 str\_replace\_all(pattern = "000$" ,replacement = "k") %>%   
 factor(levels = c("15k" ,"20k" ,"25k" ,"50k" ,"100k" ,"200k" ,"250k" ,"300k" ,"500k"))  
 ,limits\_numeric = limits\_new %>%   
 str\_replace(pattern = "k" ,replacement = "000") %>%   
 as.numeric()  
 ,marital\_status\_new = case\_when(marital\_status %in% c('M' ,'Marr' ,'Married') ~ 'Married'  
 ,marital\_status %in% c('S' ,'Single') ~ 'Single') %>%   
 as.factor()  
 ,num\_cars\_new = case\_when(num\_cars == "Four" ~ "4"  
 ,num\_cars == "Three" ~ "3"  
 ,num\_cars == "Two" ~ "2"  
 ,TRUE ~ num\_cars  
 ) %>%   
 as.factor()  
 ,num\_cars\_band\_new = case\_when(num\_cars == "1" ~ "1"  
 ,num\_cars == "2" ~ "2"  
 ,num\_cars == "3" ~ "3"  
 ,num\_cars == "4" ~ "4"  
 ,TRUE ~ "5+") %>%   
 as.factor()   
 ,seat\_belt\_new = seat\_belt %>%  
 factor(levels = c('Never' ,'Rarely' ,'Occasionally' ,'Usually' ,'Always'   
 ,'Unknown'))  
 ,income\_new = case\_when(income %in% c('Mid' ,'Middle') ~ 'Middle'  
 ,income %in% c('Working' ,'Wrk') ~ 'Working'  
 ,TRUE ~ income) %>%   
 factor(levels = c('Poverty' ,'Working' ,'Middle' ,'Upper'))  
 ,note\_type\_new = case\_when(!(note\_type %in% c(":" ,"Fraud Suspected:" ,"Hit-and-run incident:")) ~ "All Other"  
 ,TRUE ~ note\_type) %>%   
 factor(levels = c(":" ,"All Other" ,"Hit-and-run incident:" ,"Fraud Suspected:"))  
 ,pedestrian\_type\_new = pedestrian\_type %>% tolower() %>% factor(levels = c("other" ,"cyclist" ,"pedestrian"))  
 ,injury\_type\_new = injury\_type %>% as.factor()  
 ,police\_type\_new = police\_type %>%   
 coalesce("no mention of police") %>%   
 factor(levels = c("officer on site" ,"police report"   
 ,"police and medical assistance"  
 ,"police notified" ,"no mention of police"))  
 ,alcohol\_or\_drugs\_new = alcohol\_or\_drugs %>% coalesce("All Other") %>% as.factor()  
 ,claim\_greater\_than\_limit = as.numeric(claimamount > limits\_numeric)  
 )

## Fixing the columns that were identified as having issues above  
combined\_data <- combined\_data %>%   
 mutate(num\_cars\_new = na\_if(num\_cars\_new ,"17"))

One note about these corrections is that they were fixed in a “\_new” column. Thus, when building the models later on, the original columns were dropped. It is possible to just overwrite the existing columns; however, this way allowed further reference/investigating into those original ones while still having new corrected versions.

The final step was looking at the dates:

#########################################################  
## Exploratory Data Analysis - Date Columns  
#########################################################  
## Converting to dates  
combined\_data <- combined\_data %>%   
 mutate(policy\_orig\_eff\_date\_new = ymd(policy\_orig\_eff\_date)  
 ,accident\_date\_new = ymd(accident\_date)  
 ,policy\_month = month(policy\_orig\_eff\_date\_new)  
 ,policy\_day = day(policy\_orig\_eff\_date\_new)  
 ,policy\_day\_of\_week = wday(policy\_orig\_eff\_date\_new ,label = TRUE)  
 ,acc\_month = month(accident\_date\_new)  
 ,acc\_day = day(accident\_date\_new)  
 ,acc\_day\_of\_week = wday(accident\_date\_new ,label = TRUE)   
 ,report\_date = accident\_date\_new + days(report\_lag)  
 )

## Warning: There were 2 warnings in `mutate()`.  
## The first warning was:  
## i In argument: `policy\_orig\_eff\_date\_new = ymd(policy\_orig\_eff\_date)`.  
## Caused by warning:  
## ! 574 failed to parse.  
## i Run `dplyr::last\_dplyr\_warnings()` to see the 1 remaining warning.

## Date variables that are missing:  
## 1) Report\_date - re-created it with the report lag

These dates were corrected using the “lubridate” library’s functionality. There were issues with certain dates being on non-existent days (i.e. Feb 30th); given that these days being non-existent did not have a large (one-way) relationship with fraud, I just kept these as NA at this point in the process. Report\_date (which was missing from the original dataset), was also replicated using the report\_lag variable.

At this point, all numeric, character & date columns were corrected and analyzing the relationships between the different variables was now possible (and their relationship directly with fraud).

## **Variable Relationships**

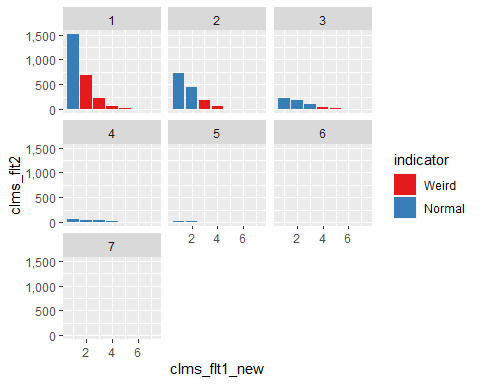
The various relationships that were looked at can be seen in the “Variable Relationships/Bucketing” section. One type of variable that I looked at were those which had a # of years components to them. In particular: “viol\_mjr#”, “clms\_flt#”, “viol\_mnr#”, & “clms\_naf#”. When looking at the documentation:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Variable Name | Variable Description | Variable Type | Variable Range | Variable Category |
| Clms\_flt# | Number of historical at-fault claims in the past # year(s) for the insured | Numeric | At least 0 | Policy |
| Clms\_naf# | Number of historical not at-fault claims in the past # year(s) for the insured | Numeric | At least 0 | Policy |
| Viol\_mjr# | Number of historical major violations in past # year(s) by insured | Numeric | At least 0 | Policy |
| Viol\_mnr# | Number of historical minor violations in the past # year(s) by insured | Numeric | At least 0 | Policy |

It shows these variables as the number of claims or violations in the past # year(s). I may be misinterpreting the documentation but, to me, this implied that clms\_flt2 should always be greater than or equal to clms\_flt1 (since clms\_flt2 is including claims for the past 2 years vs 1 year for clms\_flt1) i.e. clms\_flt5 >= clms\_flt4 >= clms\_flt3 >= clms\_flt2 >= clms\_flt1.

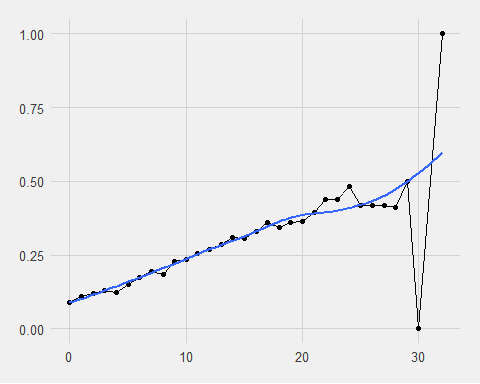
While it may be a misinterpretation of the documentation on my end, it did raise the question of whether that relationship was common or not in the data:

#########################################################  
## Variable Relationships/Bucketing  
#########################################################  
## Creating a function to see the relationship b/w different variables  
plot\_x\_y\_vars <- function(x , y) {  
 combined\_data %>%  
 filter(!!sym(x) != 0 & !!sym(y) != 0) %>%   
 mutate(indicator = if\_else(!!sym(x) > !!sym(y)  
 ,"Weird"  
 ,"Normal") %>%   
 factor(levels = c("Weird" ,"Normal"))  
 ) %>%   
 ggplot(aes(x = !!sym(x) ,fill = indicator)) +  
 geom\_bar() +  
 scale\_fill\_brewer(palette = "Set1") +  
 facet\_wrap(str\_c("~" ,y ,sep = " ")) +  
 scale\_y\_continuous(label = comma) +  
 ylab(y)   
}  
  
  
## Looking at one of the variables that has x years  
plot\_x\_y\_vars("clms\_flt1\_new" ,"clms\_flt2")



Situations where clms\_flt1 > clms\_flt2 in the chart above are shown in red so clearly it was occurring. (Similar relationships with clms\_flt1 (or clms\_flt2) being larger than the 3, 4, or 5 also occurred). This relationship was also checked for the "viol\_mjr", "viol\_mnr" & "clms\_naf" fields. The next question was whether this relationship was predictive of fraud or not:

## Plotting the fraud percentage by the # of "issues"  
## Where issues is defined as situations where the "viol\_mjr", "clms\_flt", "viol\_mnr" or "clms\_naf" fields  
## with a smaller figure where greater than that of a larger figure  
## i.e. clms\_flt1 > clms\_flt2 (or clms\_flt3 or clms\_flt4 or clms\_flt5 and so on ....)  
combined\_data %>%   
 group\_by(total\_issues) %>%  
 summarise(n = n()  
 ,fraud\_ind = sum(fraud\_ind)  
 ,fraud\_pct = fraud\_ind / n) %>%  
 ggplot(aes(x = total\_issues ,y = fraud\_pct)) +  
 geom\_point() +  
 geom\_line() +  
 geom\_smooth(se = FALSE) +  
 theme\_fivethirtyeight()



This issues field shows a very clear relationship with fraud with the more “issues” occurring being highly related to the percentage of fraudulent claims. A few other fields were looked at as well, but with this new “issues” field being created alongside the claim notes fields, I felt there was enough features to start building a model.

# **Model Development**

Since this was a classification problem, the immediate methods that came to mind were: lasso logistic regression, random forests, and boosting.

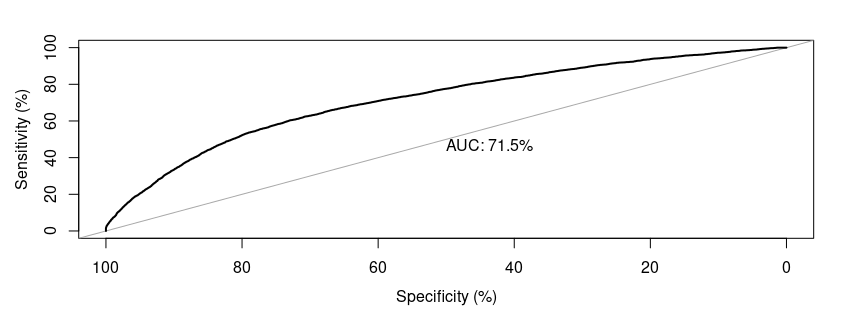
I started first by imputing any missing values with the median and removing the “old” uncleaned-up version of columns. I also removed many variables that were correlated (such as the "viol\_mjr", "viol\_mnr", “clms\_flt”, & "clms\_naf" fields – these relationships were now captured in the “issues” field described above).

The first model that was then trained was a lasso logistic regression. Before fitting the model, all numeric data was first scaled (as this could have an impact on the optimal penalty term). The optimal penalty term was then determined using 10 fold cross-validation.

The second model that was trained was a randomForest with 6 (out of 42) variables used at each split. This is the default for the randomForest function (which uses the square root of the number of columns for classification problems). The default values were used because I wanted to first compare the initial results before deciding which model to use and tuning that model.

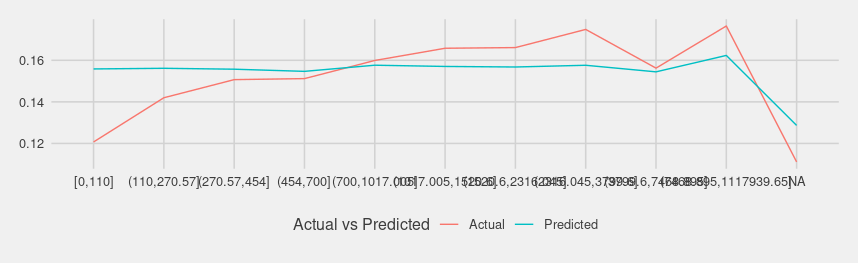
The final model was a generalized boosted regression model (GBM). The default parameters were also used to build this initial model.

Once all the initial models had been created, I looked at the ROC curve for all models. Lasso had the highest AUC at 71.5% while the random forest and GBM were at 70.2% and 71.2% respectively. The lasso ROC is shown below:

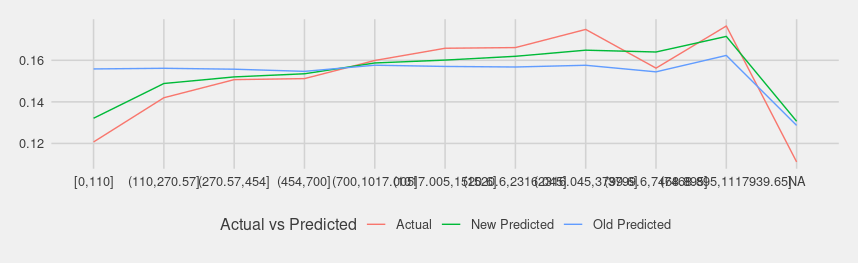


Given that lasso regression is already the most explainable (due to it not being a black box model) and it was performing the best on the first iteration, it was chosen as the model type. To iterate on this model, I looked at which variables were getting a factor of 0 and removed them following the parsimony principle for modeling.

After that, I created a function to plot the actual vs predicted fraud pct for all modeling variables used in the dataset (as well as some of the variables which had been dropped). This was to see if any relationships may not be getting captured (i.e. adding a polynomial, splines, etc… to improve the fit on certain variables). The one variable which had a particularly poor fit was the claim\_amount field (the x axis represents splitting the claim amount field into 10 equal sized buckets):



To improve this, the log of the claim\_amount was taken, rescaled, and used to fit a new model. The results of this new model can be seen in the green line below relative to claim\_amount:

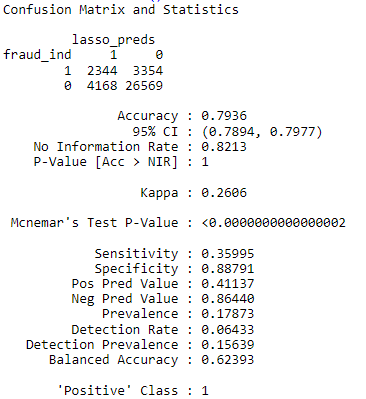


Once I felt comfortable that each of the variables used in the model were fitting the data well, I moved onto analyzing the model results.

# **Model Results**

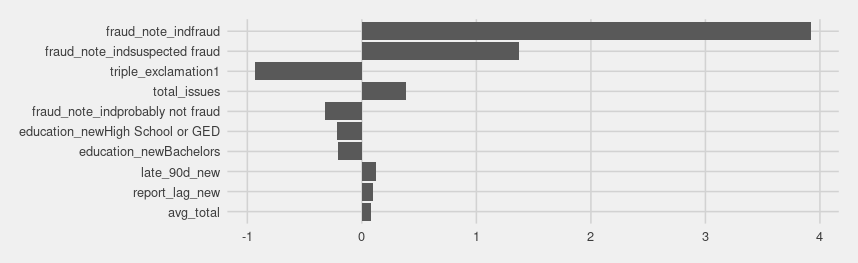
For the model results, I wanted to choose a reasonable threshold for determining what counted as a fraudulent claim. Given that the stakeholders were requiring at least one claim of every four flagged to be fraudulent, a natural threshold was 0.25. This was also reasonable given the shape of the ROC curve not being particularly steep (since a dramatic ROC shape could dictate a decision like this).

When looking at the confusion matrix of the test data using a 0.25 threshold:



The sensitivity is well above the 25% requirement at ~36% above. I do think it would be reasonable to lower the threshold further based off this (even as low as 13% had a specifity > 25% on the test set), but the 25% was kept. It would be recommended to have further discussions with the claims team & stakeholders to understand how aggressive/close to the 25% specificity requirement they would like to be.

One final aspect was understanding the relative magnitude of each of the variables. Looking at the 10 most impactful variables:



The 5 most important variables in the model were created from the claim\_notes or relationships b/w the “viol\_mjr#”, “clms\_flt#”, “viol\_mnr#”, & “clms\_naf#” fields; the feature engineering was successful!

Once this threshold was determined and the variable impacts understood, the model was applied to the assessment data (after first checking that the variable distributions were similar to the training data), and the results were scored.

# **Conclusions**

In conclusion, some of the most important predictors in the fraud analysis could be found in the claim\_notes. Other variables originally in the dataset (such as education) were also useful but required significant clean up. All of this eventually combined into a lasso logistic regression model which penalized many of the unimportant variables and resulted in a relatively simple, parsimonious model.

Conversations with business would still be recommended to determine optimal thresholds, but this work would serve as a solid foundation for detecting fraud and bringing value to stakeholders.