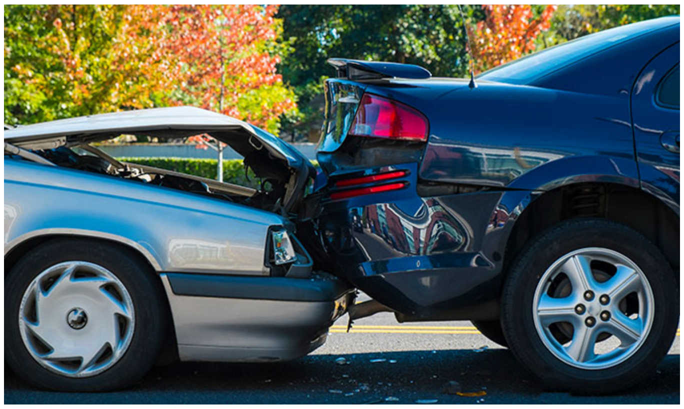
Assignment #2 - MSDS 411 Summer 2018

Andrew Knight



Introduction

My complete work for this assignment can be found in the Jupyter Notebook file **411_Assignment_2_Andrew_Knight.ipynb**.

In this assignment, we seek to build a predictive model using logistic regression to determine the likelihood that a given person will file an insurance claim in the future. The analysis and ultimate logistic model will be scored against the other models for performance.

The objective is to accurately predict if a driver in the test set will be involved in a car crash and if so, determine the damage amount. This requires two distinct models, one model will use logistic regression to determine if a person is likely to be in a crash and a separate linear model to determine the damage amount. The focus for this report is on the crash prediction and so a very simple linear model will be used to estimate the damage amount.

An insurance company would obviously want to know the probability that a person will be involved in a crash and if they are, the amount they will need to pay to fix the damage to the car. The scored data file output will provide a set of predictions for each test record based on the logistic regression analysis.

Data Exploration (EDA)

First some basic exploratory analysis of the data. I found 8161 records in the training data and 2141 records in the test data set. The target flag and target amount variables contained no null values in the training set. The test set did not contain any values for either of these response variables.

The first thing I did during my exploratory analysis was to fill all NA values with zero and then use the info() and describe() functions to spot problem areas. I found that it helps to analyze the test data alongside the training data for consistency. These data contain a mix of continuous data and categorical data and there are several simple data cleaning actions that will help before moving on with both training and test sets. I also looked at the values for count, mean, min and max for each variable. I also found the value_counts() feature useful for some categorical variables to see how many different values are present and any oddly named values. Some are highlighted below.

Then, I converted the variables that contained currency value references from strings to integers without the '$' and ' , ' characters. These included income, bluebook, home_val, and oldclaim. This was done for both data sets. No other variable text manipulation was required.

My observations specific to the training and test sets are given below.

The Training Data:
Six of the 23 predictor variables contain at least one null value which will need to be addressed.
I also noticed that the variable car_age contained a erroneous value of -3 for one of the training records. This is evident in both the boxplot and the histogram. So I decided to check all variables for any negative values. Finding no others, I simply replaced the negative car_age with zero rather than deleting this record altogether.

The Test Data:
The test data contain the same variables as the training data however it does not contain the response variables **target flag** and **target amount** as these will be predicted by the models. In the test data set, I did not find any negative values as I did in the training set.

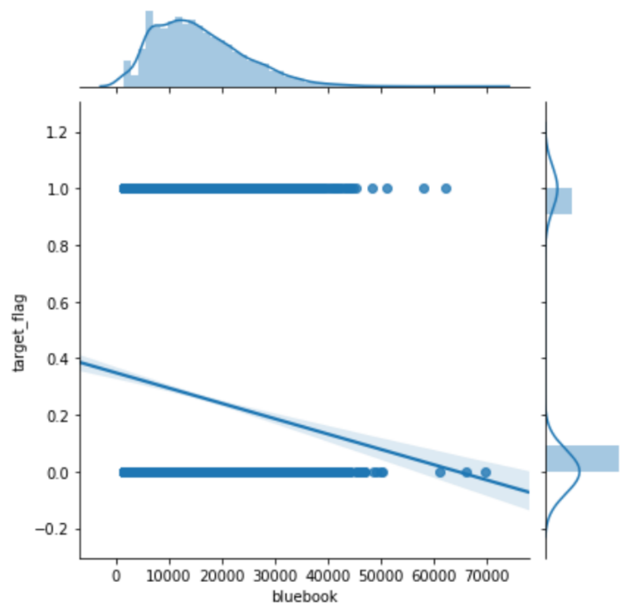
Data Preparation
First, I worked to address the null values present in both train and test datasets. After converting nulls to zeros and verifying no additional negative values, I looked for extreme outliers in the data. I primarily used the describe() feature to find high max values.

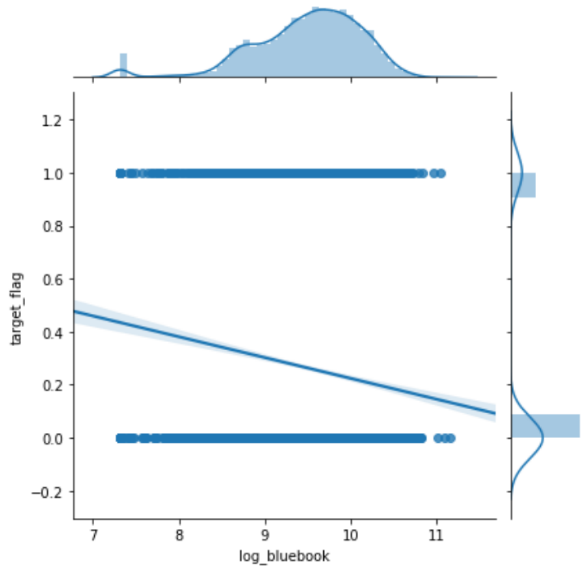
Second, I created some new variables to use in my models. I created a series of dummy, log-transformed and category encoded variables for some variables that I wanted to use. Here is a table of new variables created.

Original Variable	**New Variable**	**New Variable Type**
age	age_1hot	Numeric binary (1 = Age >= 25, 0 = Age < 25
car_use	car_use_1hot	Numeric, binary (1 = Private, 0 = Commercial)
education	education_not_hs	Numeric, binary (1 = All education defined as NOT High School (greater than), 0 = HS education or less)
urbanicity	urban_high_1hot	Numeric binary (1 = high urban area, 0 = all other areas)
revoked	revoked	Converted Y / N to binary (1 = revoked (Y), 0 = not revoked (N))

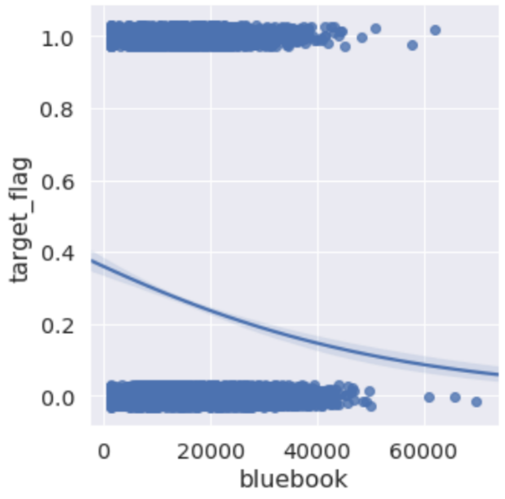
mstatus	mstatus	Converted Y / N to binary (1 = married (Y), 0 = not married (N))
red_car	red_car	Converted Y / N to binary (1 = drives red car (Y), 0 = car not red (N))
car_type	car_type_cat	Created category codes for categorical string values
job	job_cat	Created category codes for categorical string values
job	job_nonstudent	Created new binary variable (1 = all jobs NOT defined as student, 0 = students)
age	log_age	log(age), zeros replaced by mean(age) rather than removing record altogether
income	log_income	log(income), zeros replaced by mean(income)
bluebook	log_bluebook	log(bluebook), zeros replaced by mean(bluebook)
home_val	log_home_val	log(home_val), zeros replaced by mean(home_val)
car_age	log_car_age	log(home_val), zeros replaced by mean(home_val)
tif	log_tif	log(tif)
travtime	log_travtime	log(travtime)

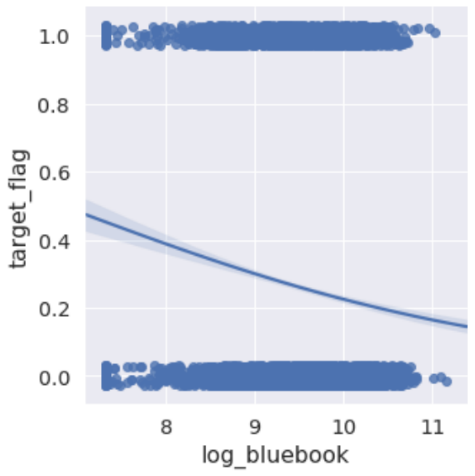
There were a few surprises after testing some of my new variables during the model building step. Variables that I thought might be relevant like job_cat and red_car turned out not to be useful as they usually lacked significance in according to the p-values obtained.
I created some log-transformed variables for some that had a distribution heavily skewed to low-end towards zero or for those that deviated significantly from normality. They are listed in the new variables table above and I also compared a plot verses the response target_flag.
The bluebook variable, as an example shows a similar fit but a more narrow confidence band on the outliers. The example plots of the predictors verses the response are given here. The distributions of each are shown as well.





I tried the same plot using the seaborn logistic lmplot, again comparing the variables, bluebook example given again here. These show a better fitting model for the binary response.

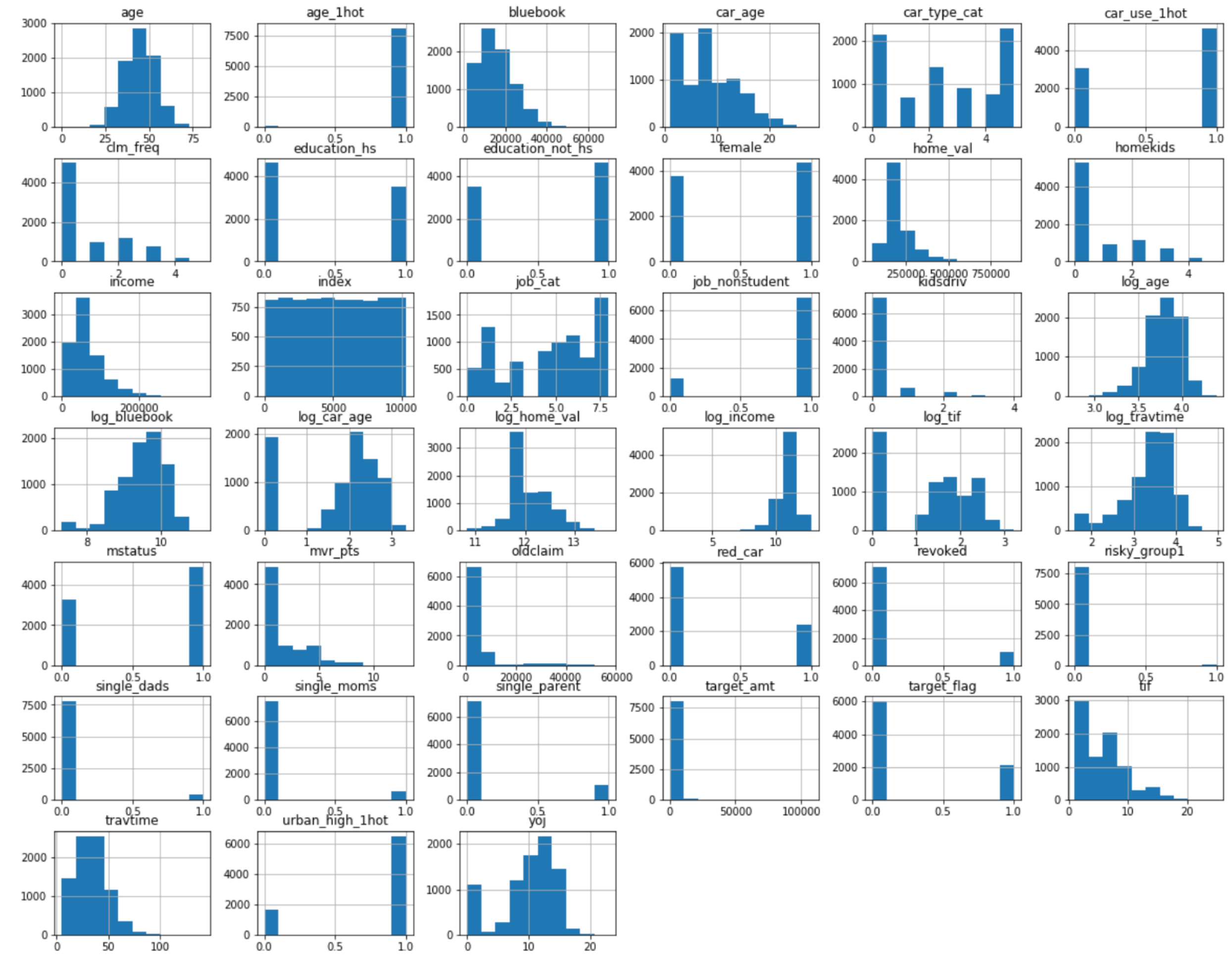




These log-transformed variables were also more statistically significant than their non-transformed counterparts. The transformation allowed me to use them in my models when I would not have been able to without it.

After the data were cleaned and the new variables were added, the histograms were plotted again.

```
Length: 39, dtype: object
```



The three types of new predictors can be seen clearly, the binary 'dummy' variables for indicating yes/no of certain conditions (like single parent or red car), the categorical state variables like homekids or car_type_cat and finally the log-transformed numerics.

Build Models

I looked at several different models and I started by comparing simple logistic regression models to more complicated forms. For each model I decided to focus on three metrics, the Log-Likelihood value, the Pseudo R-Squared value and total area under the ROC curve.

The following table provides the observed results. For each trial I considered the coefficient p-values, the public score on Kaggle and the three metrics to determine the best models to hone in on. The full model description for each can be found in Section 3 of the python code jupyter notebook.

Model	**Log-Likelihood**	**Pseudo R-Squared**	**ROC Curve Area**	**Public Kaggle Score**
Trial 1	-4375.8	0.07076	0.69	0.40865

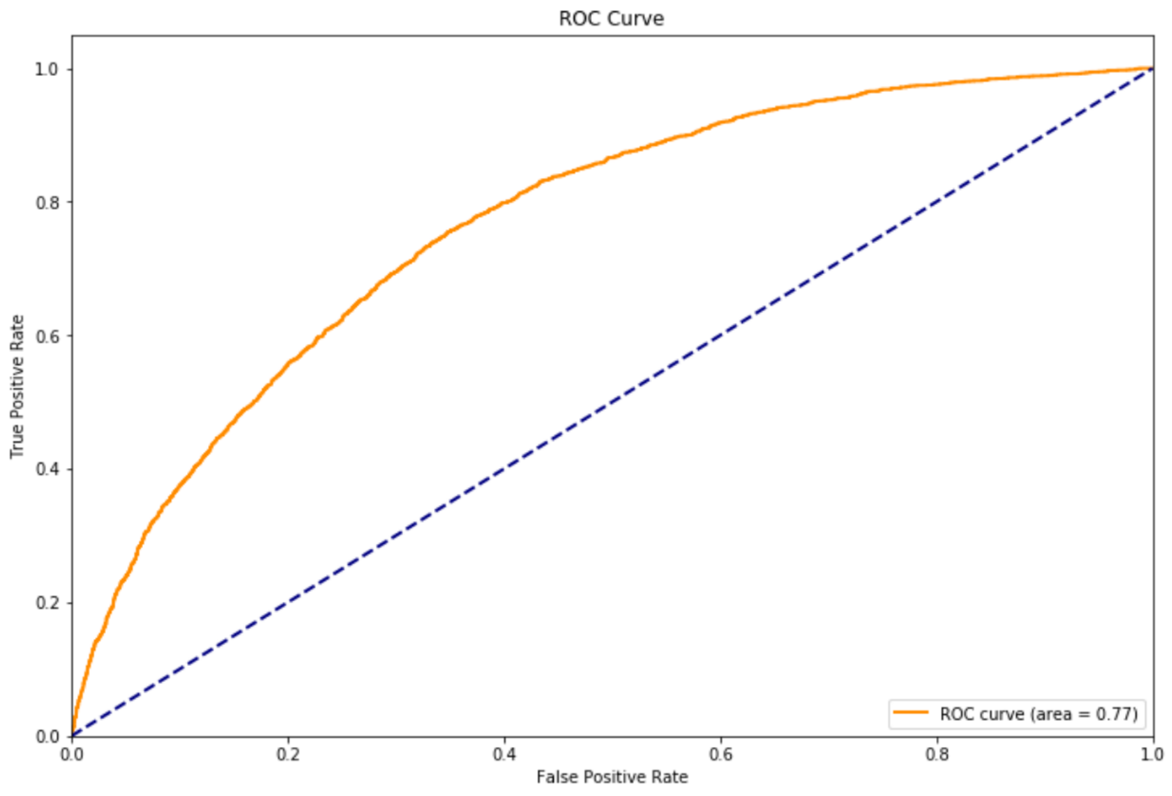
Trial 2	-4315.8	0.0835	0.70	0.41513
Trial 3	-3964.9	0.1580	0.77	0.39259
Trial 4	-3847.6	0.1829	0.77	0.39263
Trial 5	-3846.8	0.1831	0.62	0.39025
Trial 6	-3719.7	0.2101	0.62	0.37904
Trial 7	-3712.4	0.2116	0.62	0.37846
Trial 8	-3712.2	0.2117	0.62	0.37826
Trial 9	-3703.5	0.2135	0.62	0.38012

After applying additional variable transformations, testing some new variables and comparing the scored data file for the trial models given above, I decided to focus in more closely on three models. In addition to the three primary metrics in the table above, I looked at the ROC curve and the Hosmer-Lemeshow Lack of Fit plots.

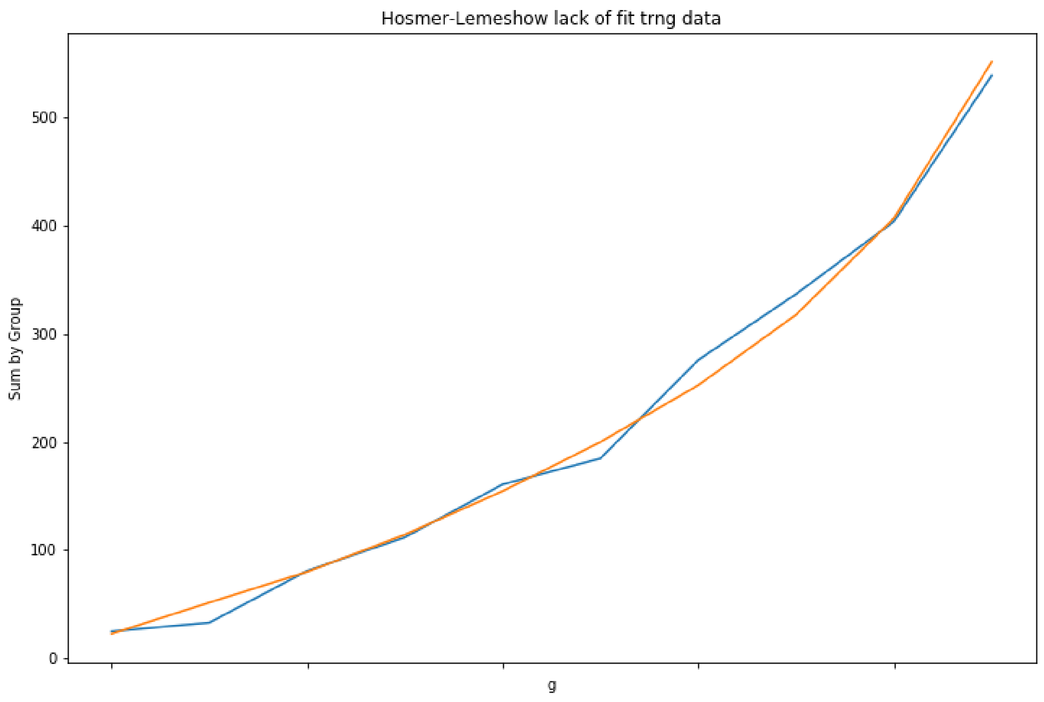
Model 1 (Trial 4):
“target_flag ~ log_age + car_use_1hot + log_home_val + log_bluebook + travtime + homekids + urban_high_1hot + tif + clm_freq + mvr_pts + education_not_hs + yoj + car_type_cat”

The ROC curve:

```
0.7679865845384665
```



The H-L Lack of Fit:

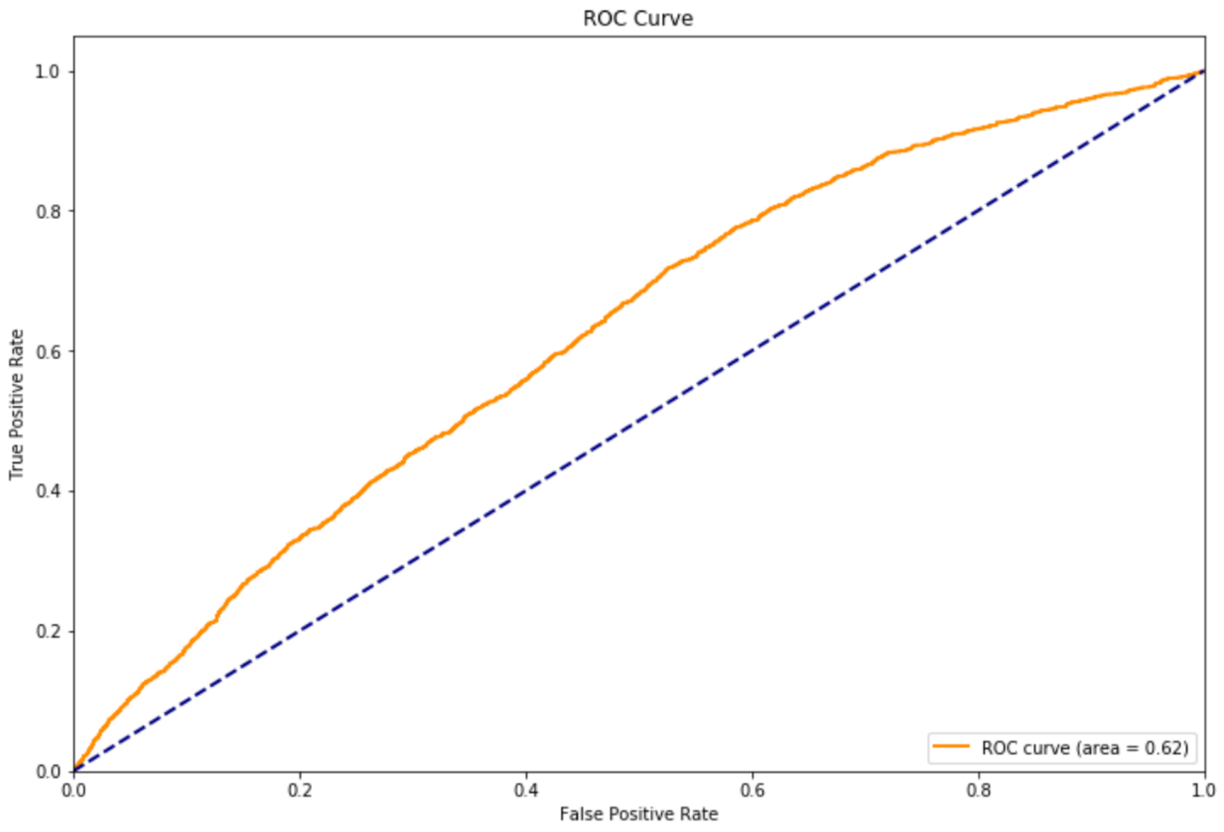


This model shows a better ROC curve than model 2 but also shows a more noticeable lack of fit to the data. Based on the Log Likelihood and R-Squared values, does not perform as well as the other two. The better ROC curve fit seems to be due to the fact that it is a simpler model than the others tried. Clearly the gains accuracy gains associated with more predictor variables is relevant.

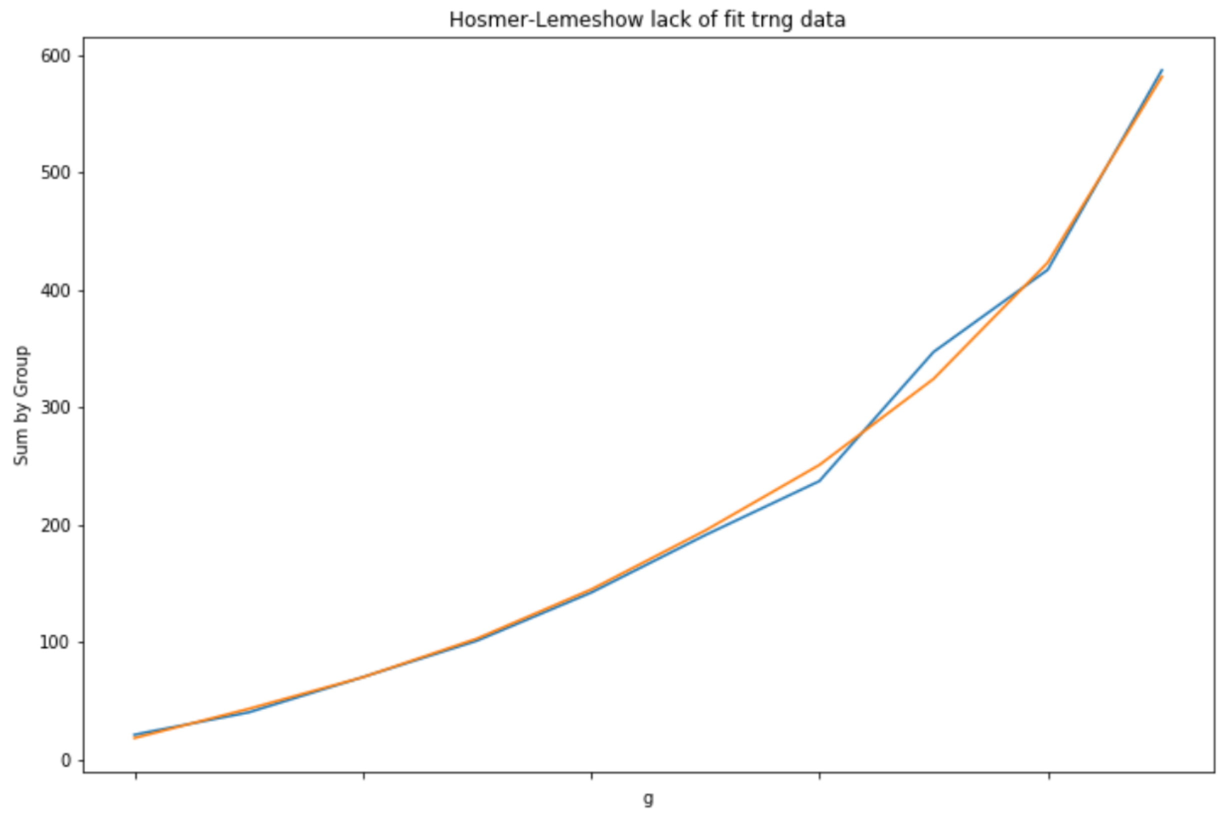
Model 2 (Trial 6):
"target_flag ~ age_1hot + car_use_1hot + travtime + homekids + urban_high_1hot + tif + clm_freq + mvr_pts + education_not_hs + yoj + car_type_cat + bluebook +
  home_val + oldclaim + revoked + mstatus + red_car + kidsdriv"

The ROC curve:

```
0.6248062654345994
```



The H-L Lack of Fit:

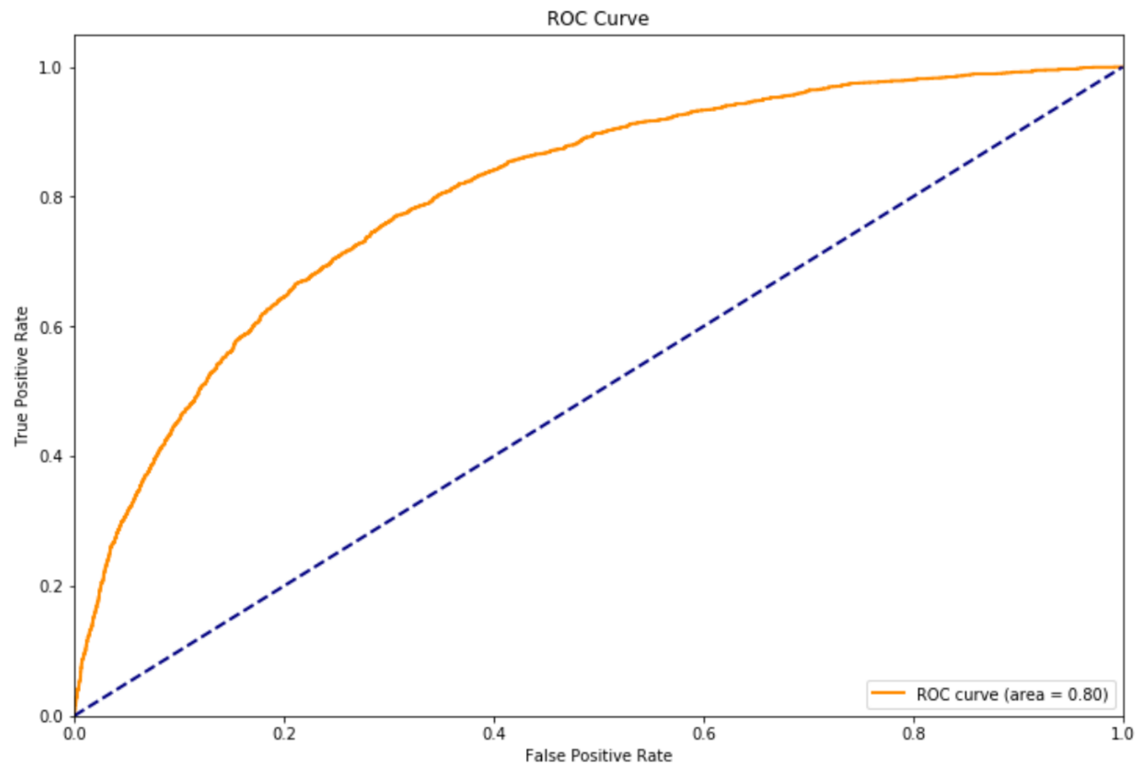


While the ROC curve area value is worse, this model shows an improvement in the Log Likelihood and R-Squared values over Model 1 and the predictions seems to fit the data better as shown in the Hosmer-Lemeshow plot.

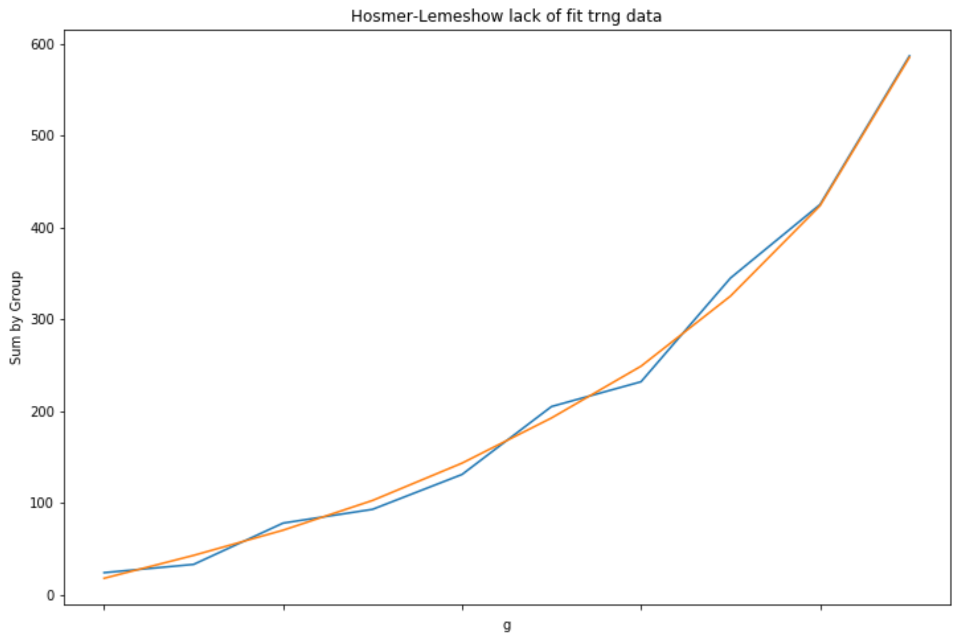
Model 3:
“target_flag ~ log_age + car_use_1hot + log_home_val + log_bluebook + log_travtime + urban_high_1hot + log_tif + clm_freq + mvr_pts + education_not_hs + yoj + car_type_cat + oldclaim + revoked + mstatus + kidsdriv + single_parent”

The ROC curve:

```
0.8044432009836088
```



The H-L Lack of Fit:



I also decided to try standard OLS linear regression model performance to compare with my logistic regression models. The respective metrics obtained for the OLS-equivalent models are compared here:

	Log-Likelihood	**R-Squared**	**ROC**
Model 1 (logit) Model 1 (ols)	-3847.6 -4040.1	0.1829 0.189	0.77
Model 2 (logit) Model 2 (ols)	-3719.7 -3900.9	0.2101 0.216	0.62
Model 3 (logit) Model 3 (ols)	-3701.5 -3884.5	0.2139 0.219	0.80

This comparison between logistic and linear regression was somewhat surprising. The R-Squared (shown above) and the Adjusted R-Squared values from the OLS model are actually better than the Logistic regression model however the Log-Likelihood values are worse.

I ran a test of the OLS predictions using Kaggle and found that the scores were worse than their equivalent logistic predictions. This was expected but it tells me that the Log-Likelihood value is a better indicator of the overall score than the R-Squared value. Model 3 has the best overall score for each metric of the three despite increased complexity.

<u>Select Model</u>
This report is ideally suited for a probability / severity model in which the final probability prediction is given by the final selected logistic regression model and the damage estimate severity model is given by a simple linear regression.

The final logistic regression model chosen is… Model 3. The full regression summary is given here.

```
                           Logit Regression Results
==============================================================================
Dep. Variable:            target_flag   No. Observations:                 8161
Model:                          Logit   Df Residuals:                     8143
Method:                           MLE   Df Model:                           17
Date:                Mon, 06 Aug 2018   Pseudo R-squ.:                  0.2139
Time:                        02:43:15   Log-Likelihood:                -3701.5
converged:                       True   LL-Null:                       -4709.0
                                        LLR p-value:                     0.000
====================================================================================
                       coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------------
Intercept            7.6731      1.180      6.503      0.000       5.360       9.986
log_age             -0.3049      0.152     -2.004      0.045      -0.603      -0.007
car_use_1hot        -0.8779      0.061    -14.342      0.000      -0.998      -0.758
log_home_val        -0.5587      0.091     -6.142      0.000      -0.737      -0.380
log_bluebook        -0.3757      0.049     -7.698      0.000      -0.471      -0.280
log_travtime         0.4128      0.051      8.115      0.000       0.313       0.512
urban_high_1hot      2.2623      0.111     20.297      0.000       2.044       2.481
log_tif             -0.2251      0.030     -7.460      0.000      -0.284      -0.166
clm_freq             0.1884      0.026      7.119      0.000       0.137       0.240
mvr_pts              0.1194      0.013      8.865      0.000       0.093       0.146
education_not_hs    -0.6225      0.064     -9.740      0.000      -0.748      -0.497
yoj                 -0.0113      0.006     -1.767      0.077      -0.024       0.001
car_type_cat         0.1312      0.015      8.527      0.000       0.101       0.161
oldclaim         -1.633e-05   4.04e-06     -4.044      0.000   -2.42e-05   -8.41e-06
revoked              0.9328      0.091     10.257      0.000       0.755       1.111
mstatus             -0.4924      0.070     -7.070      0.000      -0.629      -0.356
kidsdriv             0.4086      0.055      7.477      0.000       0.302       0.516
single_parent        0.4455      0.098      4.548      0.000       0.254       0.637
====================================================================================
```

This probability prediction model given by the logistic regression has the best Log_likelihood value of the compared models. While it has more independent variables than other models, making it more complex, the model has the best fit to the training date.

The model would be written out as: model = 7.6731 - 0.3049 X1 - 0.8779 X2 - 0.5587 X3 - 0.3757 X4 + 0.4128 X5 + 2.2623 X6 - 0.2251 X7 + 0.1884 X8 + 0.1194 X9 - 0.6225 X10 - 0.0113 X11 + 0.1312 X12 - 0.000016 X13 + 0.9328 X14 - 0.4924 X15 + 0.4086 X16 + 0.4455 X17 where the variables X1 - X14 are

X = ['log_age', 'car_use_1hot', 'log_home_val', 'log_bluebook', 'log_travtime', 'urban_high_1hot', 'log_tif', 'clm_freq', 'mvr_pts', 'education_not_hs', 'yoj', 'car_type_cat', 'oldclaim', 'revoked', 'mstatus', 'kidsdriv', 'single_parent'].

The coefficient values above seem to make sense for the most part. For instance, higher values coefficient values on predictors such as revoked and car_use indicate they more heavily weigh in on the model and would seem natural as items that have high influence on probability of a crash. Negative values were a bit harder to discern as they appeared on both positive and negative contributing items, although the log-transformations made these a bit harder to decipher directly. The comparatively smaller values on items like age and oldclaim variables were surprising though; I would expect some of those to be higher.

When testing different models, it seemed like minor changes (for instance changing one variable to a log version) sometimes had a big impact on the performance when using less complex models of fewer variables. On the other hand, when making small changes to more complex models, only slightly improved the model fit metrics. This is expected and I assume it is a good way to 'narrow in on' the 'best' model as part of the validation step.

The crash probability predictions are given in the output file **andrewknight_hw02_predictions_model3.csv** which is the same submitted to Kaggle.

Even though this was not the best score obtained in the public Kaggle score, this is being used as the final model due to exhibiting the best Log Likelihood metric and the best ROC curve area value for the training data given. One other idea that I did not explore is using the linear response target_amount as a predictor in the logistic regression model. This could be a useful input but it is not available in the test data set so its effect would not be scored, however it seems like it could be an interesting test.

Severity Model:
The linear model used only considers ***bluebook*** value and the ***car_age*** of the primary vehicle. The reason is that if the person is involved in a crash and files a claim, we need to estimate the payout required. We will not be considering liability of harm to other people or property. This model will also not be considering the severity of the accident, the medical expenses to the insured or any other factor that most auto insurance companies would have to consider. The severity model then is given by:

Target_amt ~ bluebook + car_age

The observed results are not good with an Adjusted R-Squared of only 0.014 but this demonstrates the concept of the severity model. The predicted results are provided in the output file **andrewknight_hw02_severitypreds.csv**.

Scored Data File

The final scored data file for the logistic model contains two columns. The first is the index reference and the second is the p_target column containing the predicted probability of a future crash for each record. These were rounded to three decimal places. There are a total of 2141 output values and no invalid or null values are present.
The final scored file for the linear severity model also contains two columns. The Index reference and the predicted amount of the damage. This is a really simple and inaccurate linear model.

Conclusion

In conclusion this was an interesting exercise with a simple real-world use case. It's evident that working with logistic regression to achieve a predictive model on a binary response poses unique challenges during the EDA, data cleaning and model building process. Many of the visual tools used in linear OLS regression do not translate well to the general form and it take a different approach.

The takeaway for me was that simplicity is elegance when building models with many variables but when we want accuracy of fit we have to try many different iterations and continue to add variables where possible. The Log Likelihood metric is very important in determining the logistic regression model performance. Building a similar model for a larger, realistic data set for an insurance company including much more data and many more variables would be difficult but at least we know how to start!