Dataset A (BI)

DSCoE Model Documentation



Taylor Smith (fp7y)

Lucas Allen (d0gk)

TAble of  
Contents

Dataset A (BI) i

Overview 1

Model overview 1

Data prep 2

Overview of data 2

Special variables 3

Omitted Variables 3

created variables 3

Exploratory Data Analysis 4

Preprocessing 4

Tools 5

Development and Validation 6

Model building 6

Pipelines 7

Cross validation 8

Grid Search CV 9

Final model structure 11

Fitting the Final Model 11

Appendix 12

Figure 1: Schema 12

Figure 2: Pairwise Interactions 15

Figure 3: Multicollinearity 16

# Overview

## Model overview

The Data Science Center of Excellence’s BI model for DataSet A is a Gradient Boosting Machine (GBM). The initial data aggregation, split and feature engineering was performed by the PCA team. Several grid searches were performed (building several hundred models in total) in order to select the model that maximized the validation set lift metric. All code written for the DataSet A analysis can be found at the DSCoE’s [Gitlab](https://sfgitlab.opr.statefarm.org/data-science/PCModExpTelematics/tree/master/dataseta).

# Data prep

## Overview of data

All variables created in the data prep stage were based on a PCA-provided data prep Hive script, however some of the features were changed to reference the BI equivalent of some PD variables. The Hive query for variable creation can be found in the DSCoE [Gitlab repository](https://sfgitlab.opr.statefarm.org/data-science/PCModExpTelematics/tree/master/dataseta). The data was split at the Hive level to avoid introducing validation data too early. The training set contained 1,745,767 observations, and the validation set was comprised of 1,309,573. The resultant schema included 80 floating point features, 28 int features (generally indicator variables) and one string feature (Figure 1). All missing value imputation was handled in the Hive code.

## Special variables

The following are features that were specially handled during all stages of the model process (i.e. did not receive any sort of pre-processing):

|  |  |
| --- | --- |
| **Variable Name** | **Description** |
| pp\_bi\_sy02c\_cn00x | The target feature |
| caryears\_bi\_sy02c | The exposure and weights column |
| ilae\_bi\_sy02c | The loss variable |
| bi\_glm\_offset\_tn00x | The GLM offset variable |

## Omitted Variables

The following variables were excluded from the analysis altogether for the provided reasons:

|  |  |
| --- | --- |
| **Variable Name** | **Description** |
| apf912\_0\_0\_hoh | InsurView variable which uses income |
| apt002\_0\_0\_hoh | InsurView variable which uses income |
| apf912\_0\_0\_hoh\_nn00n | InsurView variable which uses income |
| inccnt\_bi\_sy02c | Claim count during experience period (leakage) |

## created variables

Several variables were created as the result of a set of operations on existing features, the first of which were the outcome of pairwise interactions between the features shown in Figure 2. In addition to the interaction operation, the VSD column (which was previously categorical) was manually one-hot encoded so that it could be treated as a numeric column for correlation and near-zero variance analysis.

# Exploratory Data Analysis

## Preprocessing

After the all of the features were engineered, several preprocessing and feature selection steps were administered to the data in order to maximize model performance while simplifying the feature space. In particular, two feature selection techniques were found to perform well on the dataset:

* Filtering multicollinearity (Figure 3)
  + Filter out features with a correlation greater than the provided threshold. When a pair of correlated features is identified, the mean absolute correlation (MAC) each feature is considered, and the feature with the highest MAC is discarded. For the purpose of GBMs, multicollinearity is not necessarily very impactful on the model fit, however, it does impact the variable importance scores and the complexity of a model.
* Near-zero variance
  + Identify and remove any features that exhibit a variance below a certain threshold.

Pre-processor thresholds were determined in a grid search (see Development and Validation), and the hyper-parameters that contributed to the highest lift of the estimator fit were retained.

## Tools

The models were built in python using the following packages:

* H2O 3.10.0.7
  + Distributed dataframe capabilities
  + Distributed machine learning algorithm implementations
* skutil 0.1.0
  + For a more sklearn-esque grid search over H2O estimators and pre-processors
  + General H2O, Pandas and sklearn utilities
* Pandas 0.18.0
  + For aggregation and group-by functions in dataframes
* sklearn 0.18
  + For preprocessors and machine learning estimators
* numpy 1.10.4
  + For mathematic and scientific operations in python

# Development and Validation

## Model building

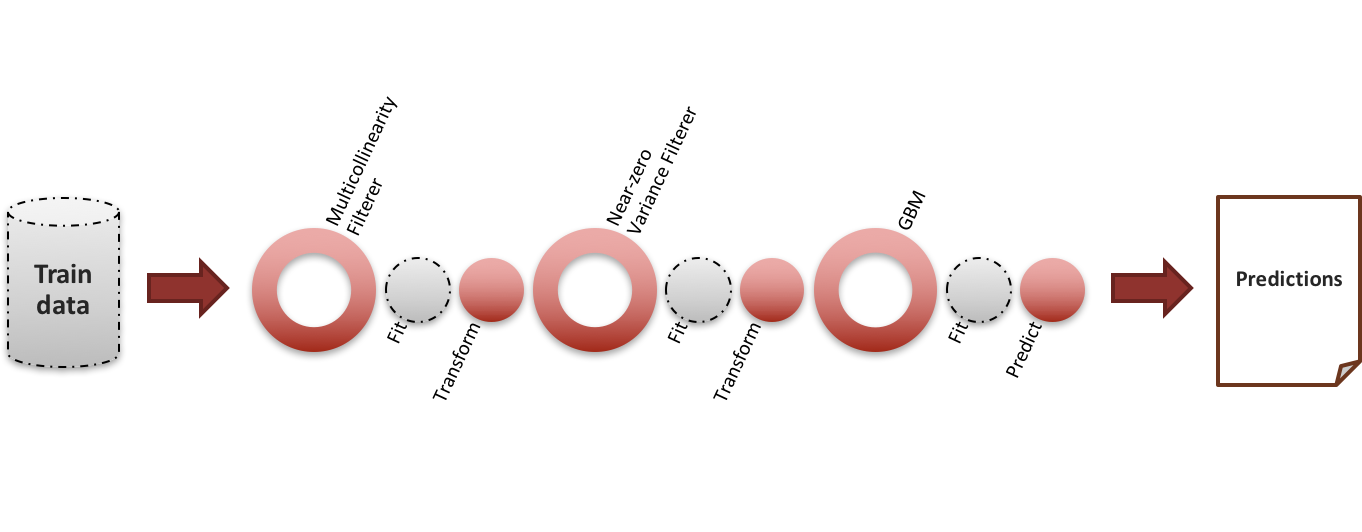
Since the data was very large, we leveraged H2O in order to fit the models in parallel across our Hadoop cluster. H2O provides support for randomized grid searches, but only at the estimator level. Since our pipeline included several pre-processors whose thresholds impacted the model fit, we leveraged the skutil package, which provides a sklearn-esque pipeline and grid search functionality for H2O, which allowed us to search over hyper-parameters on the pre-processors as well as the estimator.

Furthermore, skutil’s grid search leverages KFold cross validation with shuffling, so our mean training scores (performed on the held-out fold) closely resembled that of our final validation score. The following section will outline core components of our workflow, and will more specifically document the specific model-building process.

## Pipelines

At the core of our model is a pipeline estimator object that allows us to string together pre-processors and H2O algorithms. There are two important stages in the pipeline:

1. Fit—the training data is fed sequentially through each pre-processor in the pipeline, fitting each step before being transformed and serving as the input to the next pre-processor. The last stage of the pipeline is an estimator, and the input is the transformed data that is used to fit the estimator.



1. Predict—after the individual pieces of the pipeline are fit, a data frame can be pushed through it, being transformed at each stage and finally being scored against the estimator object.

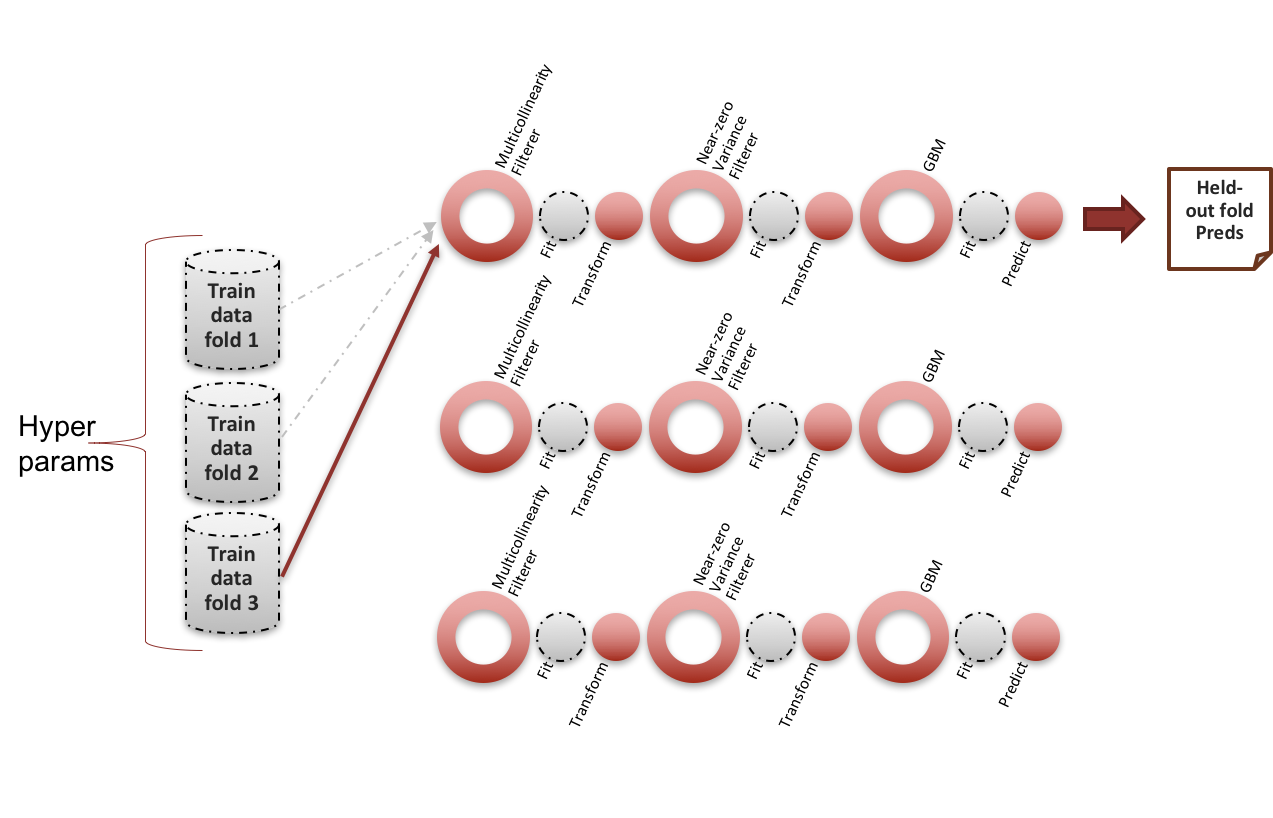


Leveraging pipelines allowed us to observe the effects of manipulating pre-processor thresholds and hyper-parameters on the model score.

## Cross validation

A concept common in the data science sphere is that of cross validation. The idea is to split the training set into *k* shuffled “folds” and to fit *k* models with the same hyper-parameters, where each model leaves a separate fold out as its impromptu validation set. Finally, all of the models are scored against the held-out folds, and the mean cross validation score is used to represent the hyper-parameter selection. One of the attractive results of cross validation is that the model score will closely resemble the validation set score, and the model is very unlikely to be overfit. However, it can take a very long time even to fit a small number of models.

The following diagram depicts a three-fold cross validated pipeline, and shows how the first two folds are used to train the first model, while the final fold is held out as an impromptu validation set.

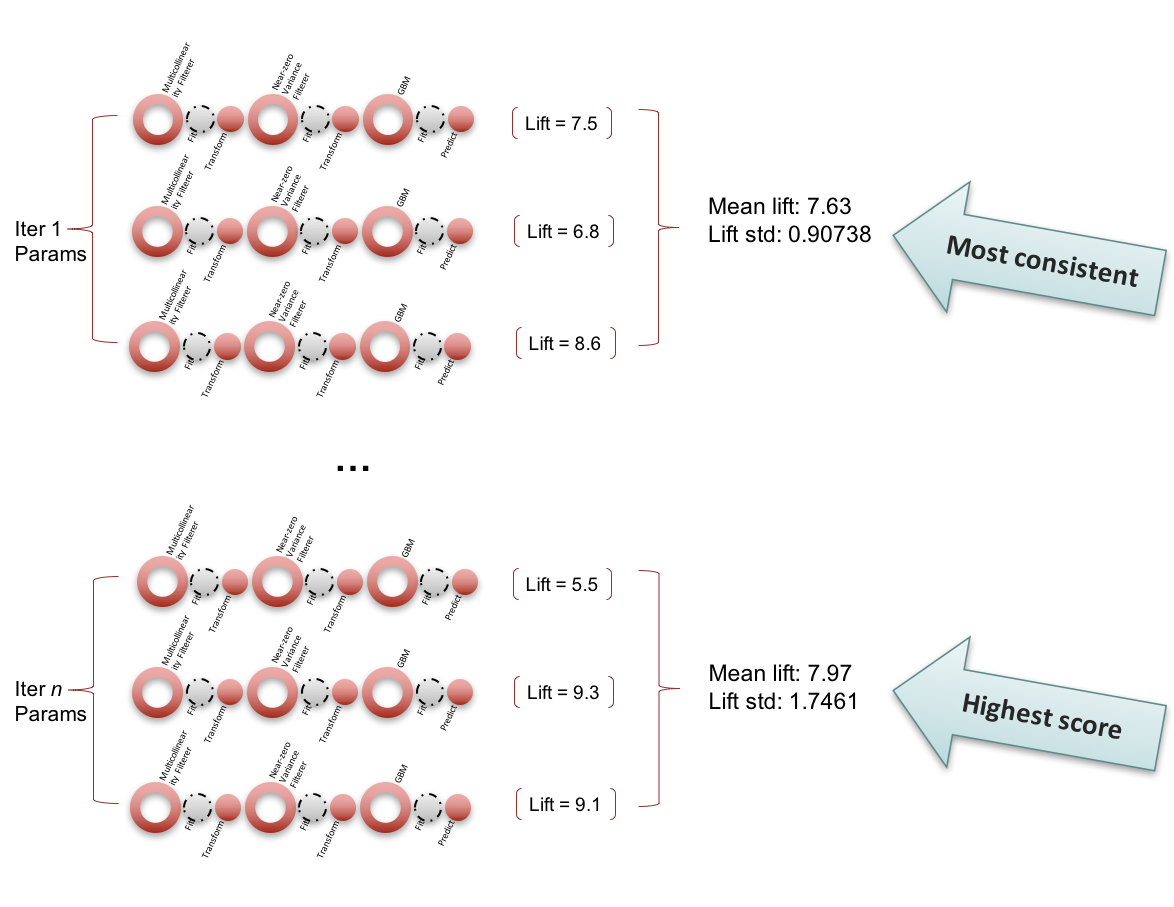


Three-fold cross validation is used to fit the first of three Pipeline models

We used five-fold cross validation in conjunction with our grid searches over pipelines in order to produce the most robust set of models possible. Furthermore, it allowed us significant flexibility in the model selection process, as we were able to select either the models that maximized the mean cross validation lift score (minimize bias), or the models that minimized the cross validation score standard deviation (minimize variance).

## Grid Search CV

We performed randomized grid searches over our pipeline (150 iterations at five-fold CV, meaning 750 total models were built on 150 unique sets of hyper-parameters)



The hyper-parameter initialization point was instated as follows:

hyper\_params = {

'nzv\_\_threshold' : uniform(1e-8, 1e-2),

'mcf\_\_threshold' : uniform(0.75, 0.24),

'gbm\_\_tweedie\_power' : uniform(1.5, 0.48),

'gbm\_\_ntrees' : randint(150, 300),

'gbm\_\_max\_depth' : randint(3, 7),

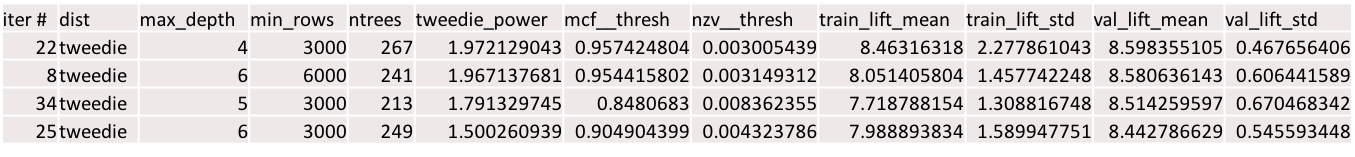
'gbm\_\_min\_rows' : [1,3000,6000,10000],

'gbm\_\_distribution' : ['tweedie'],

'gbm\_\_seed' : [rand\_state]

}

The top five models (selected on the basis on the maximum validation lift):

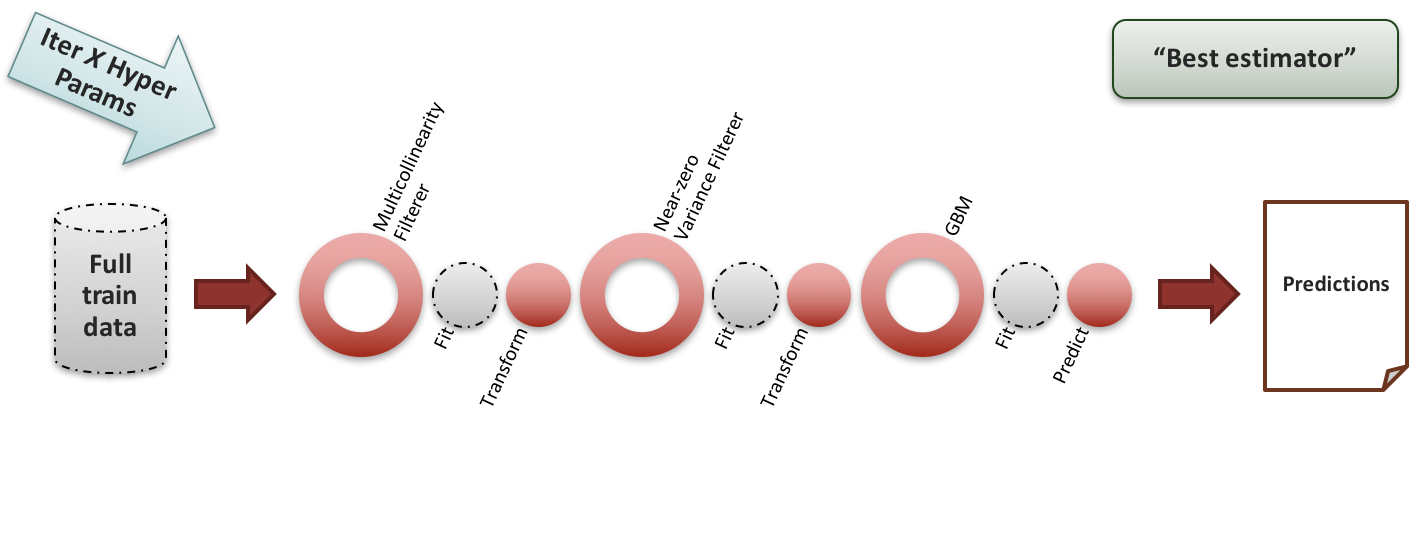


Recall that “train” scores are actually the held-out fold scores and are similar to validation scores. These are also sorted by top validation lift, and do not show the gini value in this view.

# Final model structure

## Fitting the Final Model

After the grid search is fit, the model hyper-parameters that maximize some specified criteria (we maximized mean cross-validation lift) is fit on the *entire* train set (where before it was fit on *k*-1 folds of the train data). The final structure of the model, then is the fit pre-processing pipeline that feeds into fit model:



# Appendix

## Figure 1: Schema

|  |  |
| --- | --- |
| **Variable Name** | **Type** |
| pp\_bi\_sy02c\_cn00x | float |
| caryears\_bi\_sy02c | float |
| ilae\_bi\_sy02c | float |
| inccnt\_bi\_sy02c | float |
| bi\_glm\_offset\_tn00x | double |
| carage\_when\_purchased | float |
| how\_long\_owned\_yrs | float |
| total\_last\_3yr\_major | float |
| total\_last\_3yr\_minor | float |
| total\_last\_5yr\_major | float |
| total\_last\_5yr\_minor | float |
| pop\_per\_sq\_mi\_nn00b | float |
| hh\_sum\_lap\_cnt\_pols | int |
| hh\_sum\_lap\_cnt\_pols\_90 | int |
| hh\_sum\_lap\_cnt\_pols\_drv | int |
| hh\_sum\_lap\_cnt\_pols\_drv\_90 | int |
| hh\_sum\_lap\_cnt\_pols\_comp | int |
| hh\_sum\_lap\_cnt\_pols\_comp\_90 | int |
| no\_cp\_insurer | int |
| credit\_0a11\_score | float |
| nohit\_ind | int |
| driver\_cnt | float |
| no\_veh\_hh\_n | float |
| carage\_at\_eff | float |
| own\_purchase\_inter | float |
| atfgm3 | float |
| atfgm5 | float |
| nafgm\_py03c | float |
| nafgm\_py05c | float |
| aft\_matrix\_bi\_py05 | float |
| naf\_coll\_matrix\_bi\_py05 | float |
| naf\_comp\_matrix\_bi\_py05 | float |
| naf\_ers\_matrix\_bi\_py05 | float |
| vsd | string |
| liability\_rating\_group\_num | int |
| rate\_irg\_bn10x | int |
| youthfulyn | float |
| hh\_new\_car\_ind | float |
| max\_how\_long\_licensed | float |
| min\_how\_long\_licensed | float |
| min\_hh\_age | float |
| max\_hh\_age | float |
| newly\_licensed\_0mo\_6mo\_ind | float |
| newly\_licensed\_7mo\_12mo\_ind | float |
| newly\_licensed\_13mo\_18mo\_ind | float |
| newly\_licensed\_19mo\_24mo\_ind | float |
| newly\_licensed\_25mo\_30mo\_ind | float |
| newly\_licensed\_31mo\_36mo\_ind | float |
| newly\_licensed\_37mo\_42mo\_ind | float |
| newly\_licensed\_43mo\_48mo\_ind | float |
| youth\_age\_16\_ind | float |
| youth\_age\_17\_ind | float |
| youth\_age\_18\_ind | float |
| youth\_age\_19\_ind | float |
| youth\_age\_20\_ind | float |
| youth\_age\_21\_ind | float |
| youth\_age\_22\_ind | float |
| youth\_age\_23\_ind | float |
| youth\_age\_24\_ind | float |
| youth\_age\_25\_ind | float |
| veh\_gt\_drv | float |
| drv\_gt\_veh | float |
| leased\_ind | int |
| lien\_ind | int |
| pir\_min\_cov\_ol\_bi\_min | float |
| pir\_max\_cov\_ol\_bi\_max | float |
| pir\_max\_cov\_ol\_pd\_max | float |
| pir\_max\_cov\_deduct\_co | float |
| pir\_min\_cov\_deduct\_co | float |
| pir\_max\_cov\_deduct\_cp | float |
| pir\_min\_cov\_deduct\_cp | float |
| hh\_max\_plcy\_age\_12\_nn00g | float |
| hh\_max\_plcy\_age\_24\_nn00g | float |
| hh\_max\_plcy\_age\_36\_nn00g | float |
| hh\_max\_plcy\_age\_60\_nn00g | float |
| hh\_max\_plcy\_avg\_tenure\_nn00g | float |
| hh\_max\_plcy\_max\_tenure\_nn00g | float |
| hh\_max\_plcy\_min\_tenure\_nn00g | float |
| pir\_miss | float |
| ln\_miss | float |
| cs\_score\_num | float |
| nt\_score\_num | float |
| apf912\_0\_0\_hoh\_nn00n | float |
| addrstability\_hoh\_nn00n | float |
| assetowner\_hoh\_nn00g | float |
| businessinactassociation\_hoh\_ind | int |
| curraddrapplicantowned\_hoh\_ind | int |
| curraddrlenofres\_hoh\_nn00n | int |
| derogcount\_hoh\_nn00b | float |
| derogseverityindex\_hoh\_nn00b | float |
| edattendedcollege\_hoh\_nn00g | float |
| evictioncount\_hoh\_ind | int |
| felonycount\_hoh\_ind | int |
| highriskcreditactivity\_hoh\_ind | int |
| inputaddrnotprimaryres\_hoh\_ind | int |
| inputaddrphonecount\_hoh\_ge2 | int |
| inqpersonalfinancerecent\_hoh\_ind | int |
| liencount\_hoh\_ind | int |
| prevaddroccupantowned\_hoh\_ind | int |
| propertyowner\_hoh\_ind | int |
| propownedcount\_hoh\_ge1 | int |
| recentactivityindex\_hoh\_nn00n | float |
| srcsconfirmidaddrcnt\_hoh\_nn00n | float |
| ssnaddrcount\_hoh\_nn00n | float |
| subjectssncount\_hoh\_nn00b | float |
| subjectssncount\_hoh\_nz\_nn00b | float |
| verifieddob\_hoh\_nover\_ind | int |
| verifiedname\_hoh\_ind | int |
| missing\_riskview | int |

## Figure 2: Pairwise Interactions

|  |
| --- |
| **Variable Name** |
| newly\_licensed\_0mo\_6mo\_ind |
| newly\_licensed\_7mo\_12mo\_ind |
| newly\_licensed\_13mo\_18mo\_ind |
| newly\_licensed\_19mo\_24mo\_ind |
| newly\_licensed\_25mo\_30mo\_ind |
| newly\_licensed\_31mo\_36mo\_ind |
| newly\_licensed\_37mo\_42mo\_ind |
| newly\_licensed\_43mo\_48mo\_ind |
| youth\_age\_16\_ind |
| youth\_age\_17\_ind |
| youth\_age\_18\_ind |
| youth\_age\_19\_ind |
| youth\_age\_20\_ind |
| youth\_age\_21\_ind |
| youth\_age\_22\_ind |
| youth\_age\_23\_ind |
| youth\_age\_24\_ind |
| youth\_age\_25\_ind |
| veh\_gt\_drv |
| drv\_gt\_veh |
| leased\_ind |
| lien\_ind |
| driver\_cnt |
| no\_veh\_hh\_n |

## Figure 3: Multicollinearity

