Predicting policy conversion in auto insurance

Tiancheng hou

November 27, 2019

Introduction

For auto insurance, any customer that asks for an estimated price for a policy (also known as a quote) will receive one. However, of those quoted, only a fraction will choose the company as their insurer. The percent of quoted policies the company issues, known as Conversion rates, have always been the focus of insurance company, as it directly related to the company's revenue. Our goal is to build a classification model to better identify the customers that are more likely to convert their policies and also find out the key characteristics of policies that determine the conversion rates. In this project, I mainly used XGboost for modeling and predicting. To make sure I got the best solution I also compared the result from XGboost with three other models, Lasso regression, Logistic regression and ridge regression. Based on the model, I find out the key characteristics of policies the company tends to write. The result of this project provide valuable information to insurance companies, helping them target the market and increase their sales.

keywords XGboost, Policy Conversion, Classification

Data description

The dataset comes from '2019 Modeling Competition'. There are three dataset describing the information of customers that asked for quote. The main dataset is the 'policy' dataset. Each row in this data give us the information of one particular policies. And two attached dataset provided extra information for each policies. One is on drivers level, gives us information of drivers in each policy, each policy could corresponds to multiple rows that describing multiple drivers in that policy. The third dataset is on vehicle level, gives us information of vehicles in each policy. Also, each policy could corresponds to multiple rows describing multiple vehicles in that policy. (To check explicated description of variables '2019 Modeling Competition')

Missing values

	Missing
Quote_dt	0%
discount	0%
Home_policy_ind	0%
zip	0.96%
state_id	0%
county_name	0%
Agent_cd	11.05%
quoted_amt	0.23%
Prior_carrier_grp	0%
credit_score	0.61%
Cov_package_type	1.57%
CAT_zone	0.51%
number_drivers	0%
num_loaned_veh	0%
num_owned_veh	0%
num_leased_veh	0%
total_number_veh	0%
primary_parking	0%

	Missing
policy_id	0%
gender	0%
living_status	0.1%
age	0%
safty_rating	0.16%
high_education_ind	1.1%

	Missing
policy_id	0%
car_no	0%
ownership_type	0%
color	2.18%
age	1.09%
make_model	0%
make_model_brand	0%

The table above shows the percentage of missing values in each variable. We notice many variables has missing values, but the proportions differ from each other. The factor variable `Agent_cd` has a lot of missing values, and it also has many levels. When I try to include this variable into the model, the AUC of the model is decreased, thus, I decided to remove this variable. Another factor variable, `Prior_carrier_grp`, also have a lot of missing values, however, when I run the GLM function, find many levels of this variable has significant effect. Then I assign all missing values as a new level, "Unknown". The other variable seems only have little proportion of missing value. A common way to deal with NAs is using KNN imputation. However, when there are a lot of variables, KNN would not be very trustworthy. In XGboost, there is another way to deal with missing values, which would be introduced later.

Feature engineering

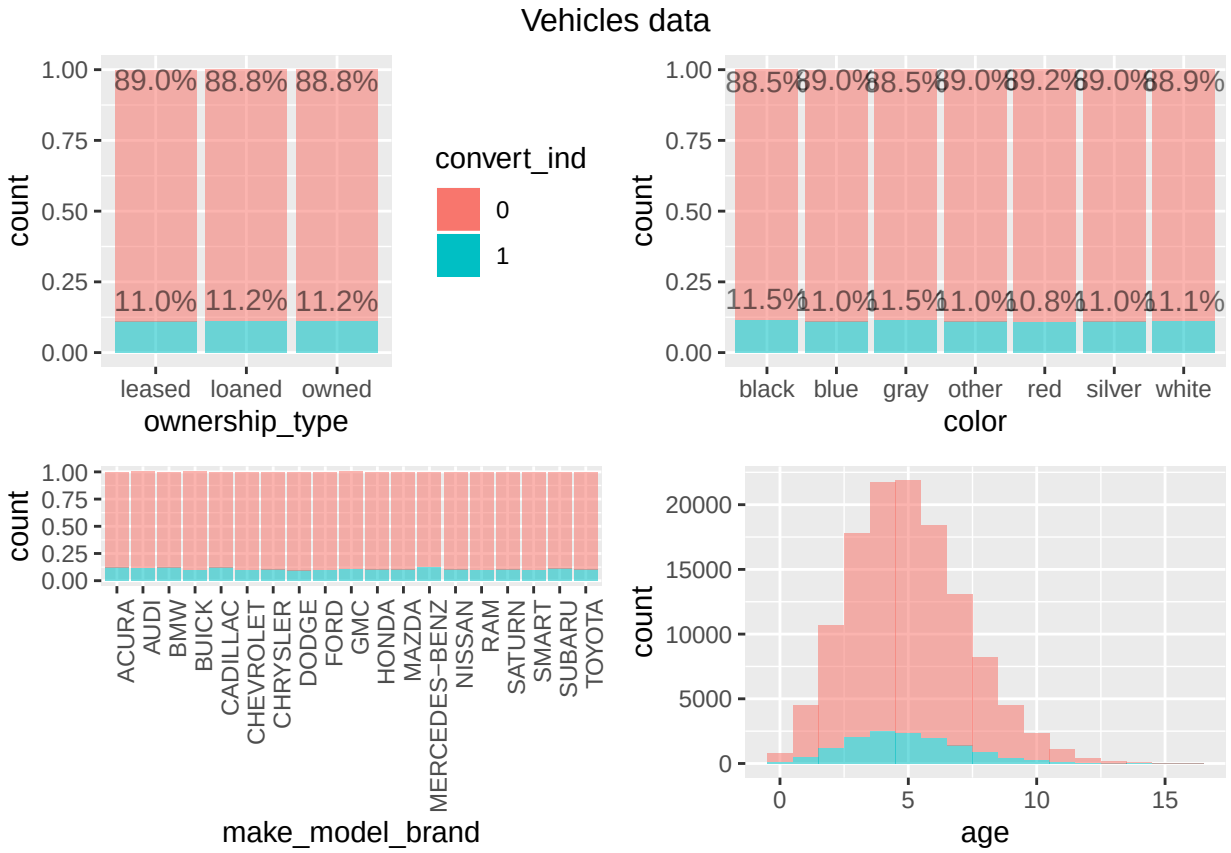
First, I want to exact some useful information from two attached datasets(drivers and vehicles) and create new features. Then, combine those variables with the main dataset(policies) for future modeling. To find out useful variables, I applied both data visualization and a rough logistic model for help.

Driver Dataset

In the 'Driver' dataset, `age` are significantly affect the response, and so as `high_education_ind`. Then we can calculate the mean of age and the ratio of drivers who have high education in each policy. The `living status` has three levels(own,rent,dependent), and we observe the two of them are very close. Then we can calculate the ratio of drivers living dependently in each policy. Although the gender does not seem to be important in the model, I decided not to remove this variable. Because when we construct the tree structure model, the interaction of age and other variables might be important. For example, young male might have higher conversion rate than young female.

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-2.3639	0.0570	-41.48	0.0000
genderM	-0.0452	0.0238	-1.90	0.0576
living_statusown	0.4565	0.0519	8.79	0.0000
living_statusrent	0.5200	0.0525	9.91	0.0000
age	-0.0054	0.0012	-4.51	0.0000
safty_rating	-0.0002	0.0008	-0.30	0.7643
high_education_ind1	0.2200	0.0288	7.63	0.0000

Vehicles Dataset



Similar to the 'driver' data, we find the `ownership_type` has three levels, two of them has a very similar coefficient. Then, I combine those two level into one. (Although doing a chi-square test before is preferred.) And according to the coefficient we can also classify the car's brand into three classes.

Based on previous plots and tables, we regenerate features as follow:

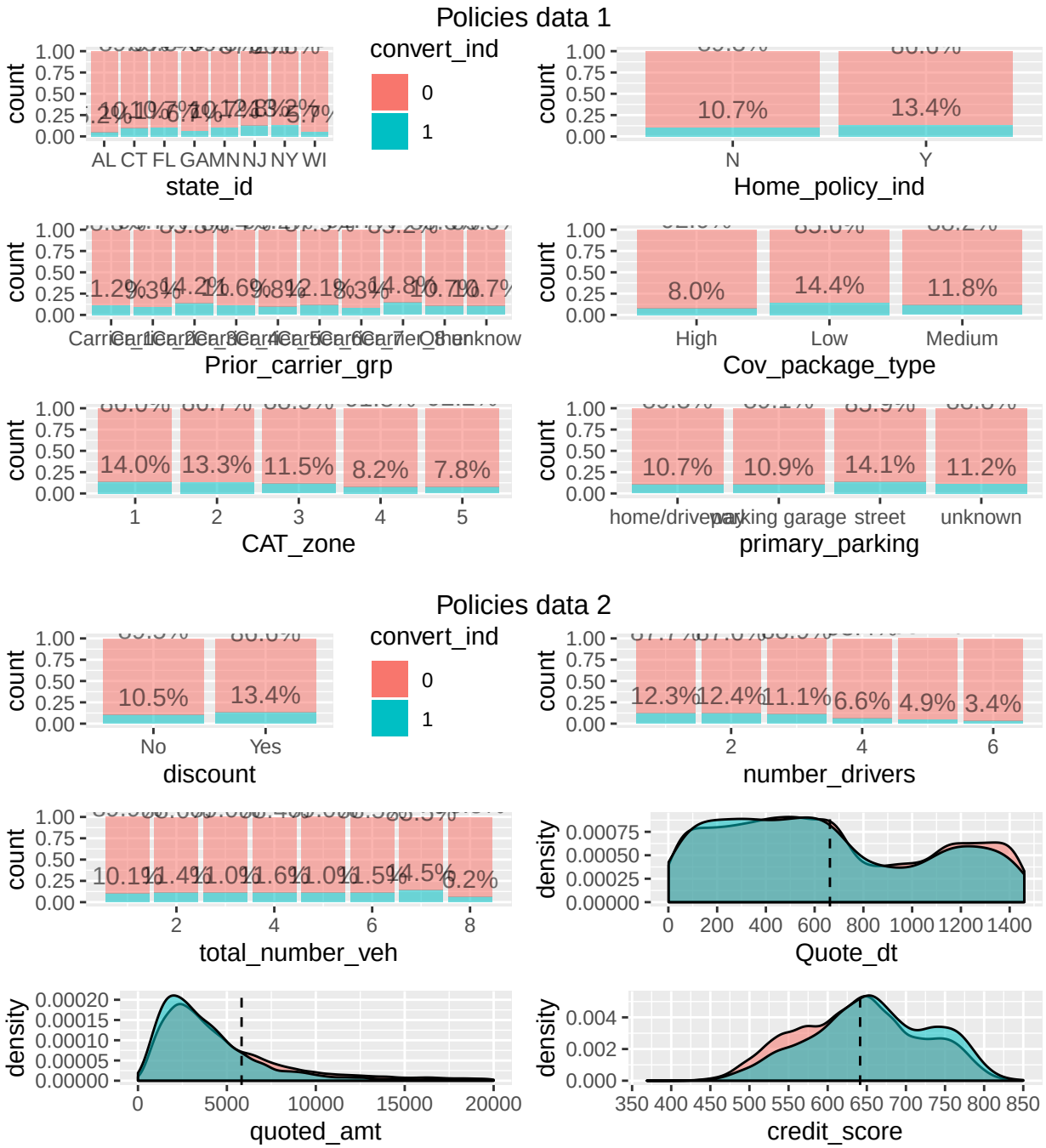
	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-1.8671	0.0684	-27.29	0.0000
ownership_typeloaned	0.0207	0.0279	0.74	0.4583
ownership_typeowned	0.0174	0.0271	0.64	0.5205
colorblue	-0.0515	0.0334	-1.54	0.1226
colorgray	-0.0013	0.0331	-0.04	0.9690
colorother	-0.0500	0.0333	-1.50	0.1337
colorred	-0.0717	0.0335	-2.14	0.0325
colorsilver	-0.0470	0.0334	-1.41	0.1588
colorwhite	-0.0437	0.0333	-1.31	0.1892
age	-0.0128	0.0040	-3.20	0.0014
make_model_brandAUDI	-0.0698	0.0681	-1.02	0.3057
make_model_brandBMW	-0.0382	0.0614	-0.62	0.5335
make_model_brandBUICK	-0.2599	0.1064	-2.44	0.0146
make_model_brandCADILLAC	-0.0226	0.0808	-0.28	0.7798
make_model_brandCHEVROLET	-0.2448	0.0660	-3.71	0.0002
make_model_brandCHRYSLER	-0.2075	0.0960	-2.16	0.0307
make_model_brandDODGE	-0.2767	0.0766	-3.61	0.0003
make_model_brandFORD	-0.2340	0.0664	-3.52	0.0004
make_model_brandGMC	-0.1700	0.0721	-2.36	0.0185
make_model_brandHONDA	-0.1883	0.0623	-3.02	0.0025
make_model_brandMAZDA	-0.1968	0.0854	-2.30	0.0212
make_model_brandMERCEDES-BENZ	0.0251	0.0621	0.40	0.6859
make_model_brandNISSAN	-0.1926	0.0730	-2.64	0.0083
make_model_brandRAM	-0.2481	0.1326	-1.87	0.0613
make_model_brandSATURN	-0.2205	0.0995	-2.22	0.0267
make_model_brandSMART	-0.2400	0.2042	-1.18	0.2399
make_model_brandSUBARU	-0.0954	0.0966	-0.99	0.3234
make_model_brandTOYOTA	-0.1816	0.0737	-2.46	0.0138

	make_model_brand	type
1	BMW	A
2	MERCEDES-BENZ	A
3	AUDI	A
4	ACURA	A
5	FORD	C
6	CHEVROLET	C
7	HONDA	B
8	DODGE	C
9	TOYOTA	B
10	MAZDA	B
11	CADILLAC	A
12	SATURN	C
13	NISSAN	B
14	GMC	B
15	BUICK	C
16	RAM	C
17	SUBARU	A
18	CHRYSLER	B
19	SMART	C

	color	col2
1	silver	light
2	white	light
3	other	light
4	gray	dark
5	red	light
6	blue	light
7	black	dark

	ownership_type	ownership_type2
1	owned	have
2	loaned	have
3	leased	leased

Policies Dataset



In the policies table, we find the variables `zip`,`state_id` and `county_name` gives us the same information. After fitting model use each one of them, I find using only `state_id` give us the best predictive result . The `total_number_veh` is the summation of the number of three different types of cars, so it can also be dropped. `Quote_dt` is given as a date format and can be converted into continuous numbers by days.

After data manipulation, we get the following 34 predictors.

```
##  [1] "Quote_dt"          "discount"          "Home_policy_ind"
##  [4] "state_id"          "quoted_amt"        "Prior_carrier_grp"
##  [7] "credit_score"      "Cov_package_type"  "CAT_zone"
```

```
## [10] "number_drivers"    "num_loaned_veh"    "num_owned_veh"
## [13] "num_leased_veh"    "total_number_veh"  "split"
## [16] "primary_parking"   "leased_N"          "car_age"
## [19] "light_N"           "A_N"               "C_N"
## [22] "leased_R"          "light_R"           "A_R"
## [25] "C_R"               "Male_N"            "Mean_age"
## [28] "dependent_N"       "mean_rating"       "High_edu_N"
## [31] "Male_R"            "dependent_R"       "High_edu_R"
```

Variables have a form like "_R" are ratio variables calculated from supplementary datasets. For example, "A_R" is the number of type A cars in the policy divided by total_number_veh in that policy. Variables have the form like "_N" are number variables. For example, "dependent_N" is the number of drivers living dependently in one policy, it is the interaction term of "number_drivers" and "dependent_R".

Modeling

Model description-XGboost

eXtreme Gradient Boosting (XGBoost) (Chen and Guestrin 2016)[1] is one of the most popular model. It famous of winning many data science competition on Kaggle.

The XGboost model is a tree ensemble model uses K additive functions to predict the output.

$$\hat{y_i} = \phi(x_i) = \Sigma_{k=1}^{K} f_k(x_i)$$

The objective function is:

$$Obj(\theta) = \Sigma_{i=1}^{n} l(y_i, \hat{y_i}) + \Sigma_{k=1}^{K} \Omega(f_k)$$

The first term is loss function. When we fit a binary classification model, it would be logistic loss:

$$l(y_i, \hat{y_i}) = y_i ln(1 + e^{-\hat{y_i}}) + (1 - y_i) ln(1 + e^{\hat{y_i}})$$

The second term is the regularization term:

$$\Omega(f_k) = \gamma T + \frac{1}{2}\lambda \Sigma_{j=1}^{T} \omega_j^2$$

where T is the number of leaves in the tree. ω is the score in the corresponding leaves.

At the t-th iteration,

$$Obj(\theta)^{(t)} = \Sigma_{i=1}^{n} l(y_i, \hat{y_i}^{(t-1)} + f_t(x_i)) + \Omega(f_t)$$

by using second-order approximation(Taylor expansion), and removing constant terms we get:

$$Obj(\theta)^{(t)*} = \Sigma_{i=1}^{n} [g_i f_i(x_i) + \frac{1}{2} h_i f_i^2(x_i)] + \Omega(f_t) = -\frac{1}{2} \Sigma_{j=1}^{T} \frac{G_j^2}{H_j + \lambda} + \gamma T$$

$$where: \ g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y_i}^{(t-1)}), h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y_i}^{(t-1)}) G_j = \Sigma_{i \in I_j} g_i, \ H_j = \Sigma_{i \in I_j} h_i$$

The change of objective after adding the split is:

$$Gain = \frac{1}{2}[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda}] - \gamma$$

To find best split point, XGboost will do the following to find the split point that has the biggest gain:

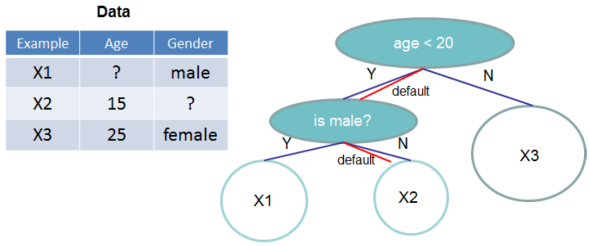
1.for each node, enumerate over all features.

2.for each feature, sorted the instances by feature value.

3.use a linear scan to decide the best split along that feature.

4.take the best split solution along all the features.

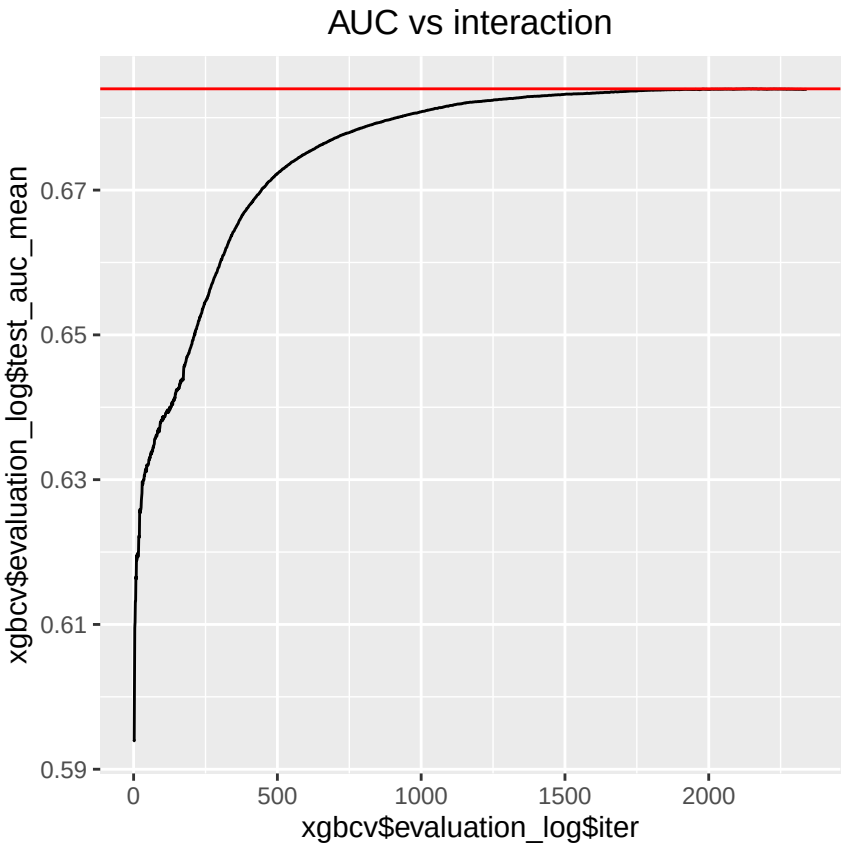
When predictor mareix is sparse. The XGboost will try to assign missing data into both directions to find the best solution. In this case, we do not need to impute missing values before modeling.



The plot is from paper XGBoost: A Scalable Tree Boosting System[1]

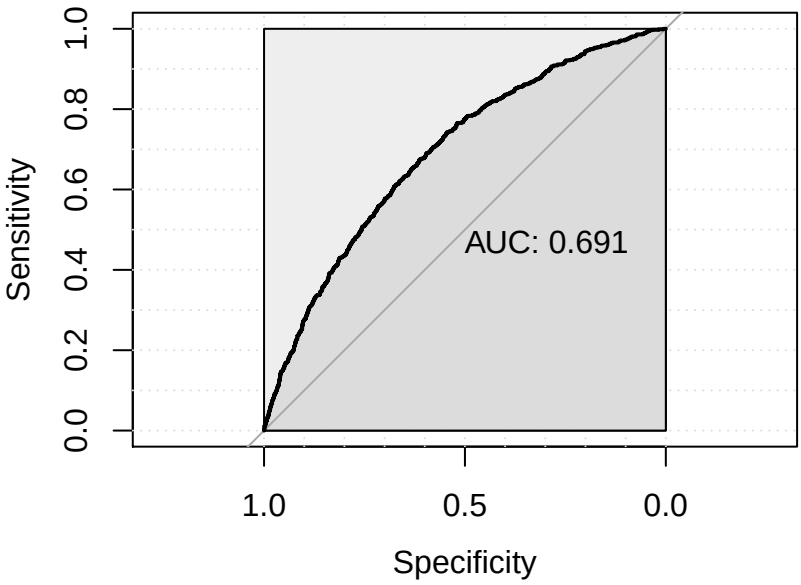
Model fitting and selection

There are many hyperparameters that need to be tuned in order to fully specify the model. In this project, I tried tunning some parameters manually, and do the cross validation to decided the number of iterations. To prevent overfitting, I set the shrinkage parameter, η, as well as maximum depth of a tree in very small values. [2]

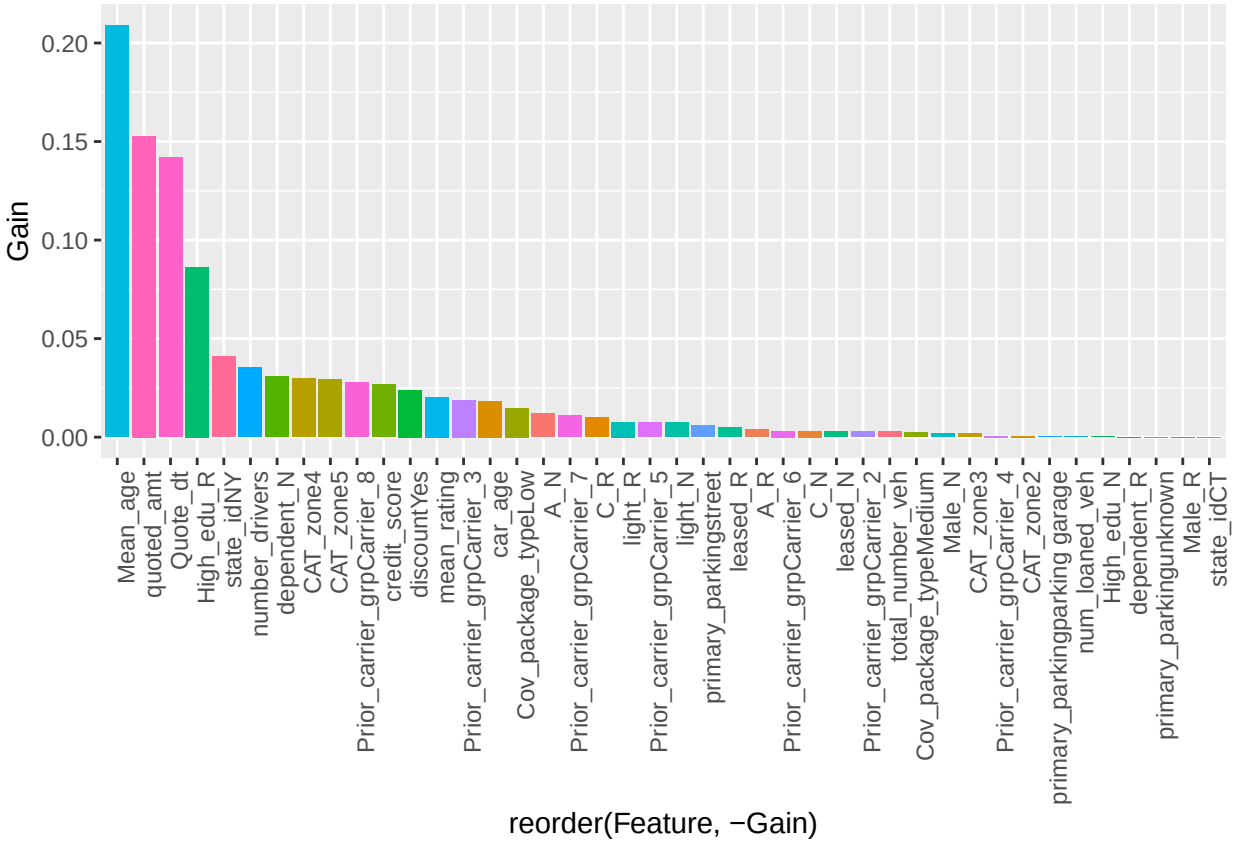


From the cross validation, we get the best number of iteration is 2141. And the mean of AUC is 0.684.

	Best iteration	Best AUC
1	2141	0.683988

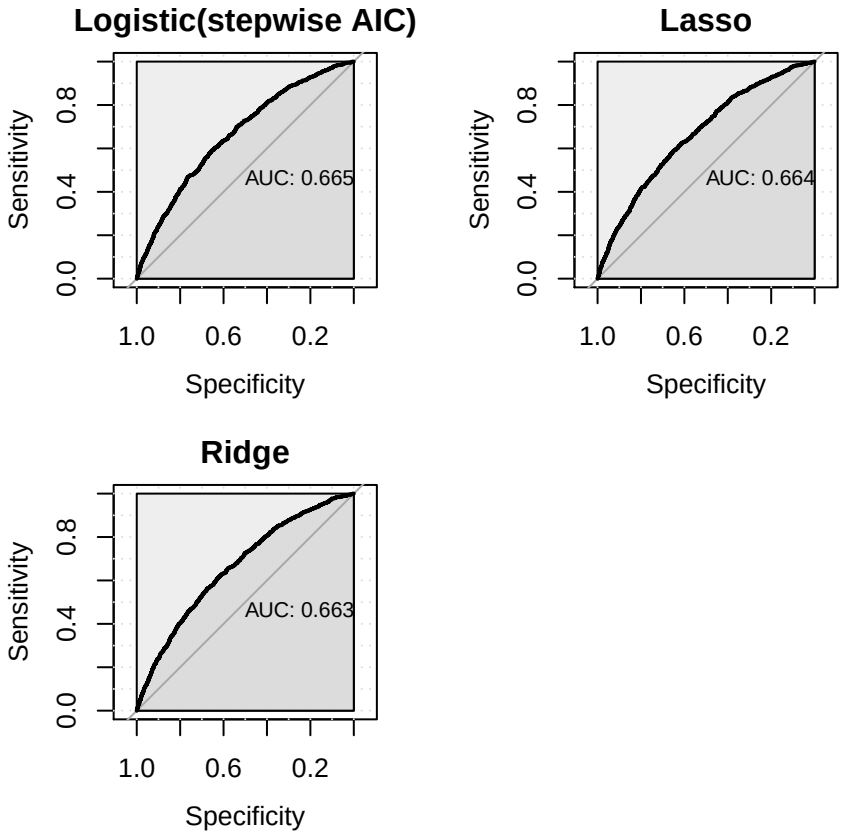


After we applied our model on testing dataset, we got the ROC plot and AUC score showing above. To find out the important features, we plot the "Gain" for each variable.



From the plot, we observe that `Mean_age`,`quoted_amt`,`quote_dt`,`High_edu_R` are very important determining whether the customer would convert their policies.

Then, compare with other model.



From the AUC plot, we notice that logistic regression, lasso regression, and ridge regression have a very similar predictive performance. On the other hand, XGboost, with AUC 0.691, performs better in predicting the customers' convert probability.

Conclusion

In this project, I first did some data visualization to find the connections between variables. Based on the plots generated in the previous step, I did some data manipulation to create some potentially useful features. Then, a very popular model in data science, XGboost, is introduced and briefly explained. To make the predictive result more accurate, I conducted a cross validation to find the best iteration. Latter, I summarise some important features that determines the customer's convert probabilities, which provides very useful information on the insurance company. Finally, I compared XGboost models and other model for prediction to make sure the model gives the best predictive result.

Reference

[1] Tianqi Chen. Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. ACM. ISBN 978-1-4503-4232-2/16/08, 2016.

[2] https://xgboost.readthedocs.io/en/latest/parameter.html