**Policy Cancellation Rate Prediction**

for Kangaroo auto insurance company

Yunhao Huang(yh3090), Yiming Tan(yt2633), Xiaoxi Zhao(xz2740)

Multivariate Statistical Inference Final Project Srping 2019

1. **Abstract**

Policy cancellation prediction is a widely discussed problem in machine learning. In this report, we performed multivariate data analysis to build a prediction model to the insurance policy data for the Kangaroo auto insurance company. The analysis is performed in Python with exploratory data analysis, data cleaning, feature engineering, model fitting (XGBoost, Random Forest, Light GBM, Adaboost), and stacking methods. We generate business insights from the model and give our suggestions to the insurance company.

Key words: machine learning, classification, insurance policy.

1. **Introduction**
2. **Background**

The insurance companies are always interested in the policy cancellation and the reasons behind it. Competing in the NESS 2019 Statathon Competition on Kaggle <https://www.kaggle.com/c/TRVstatathon>, we aim to create a retention model based on historical policy data for Kangaroo Auto Insurance Company. Based on the model, we will predict those house insurance policies that are most likely to cancel as well as interpret the results and find out which features contribute the most into a policy cancellation.

1. **Motivation**

All of our team members are well interested in performing data analysis and implementing what we’ve learned from multiple classes to the real world data. This data set includes several predictors where we can perform multivariate data analysis and further it is a good chance for us to participate in a Kaggle competition. We want to further our understanding, to learn more from real world experiences, which can help us in our future career as a data science.

1. **Data Analysis**
   1. **Exploratory Data Analysis**

In the training dataset, we have 792026 non-cancelled policies (75.5%), 253097 cancelled policies (24.1%)(Figure1). It is an imbalanced data classification problem, thus we will use AUC as our metrics.

We can also see from Figure 1 that our target variables include 3452 -1’s, which is of no meaning to our dataset and only a small part of our whole dataset(0.4%), so we delete them.

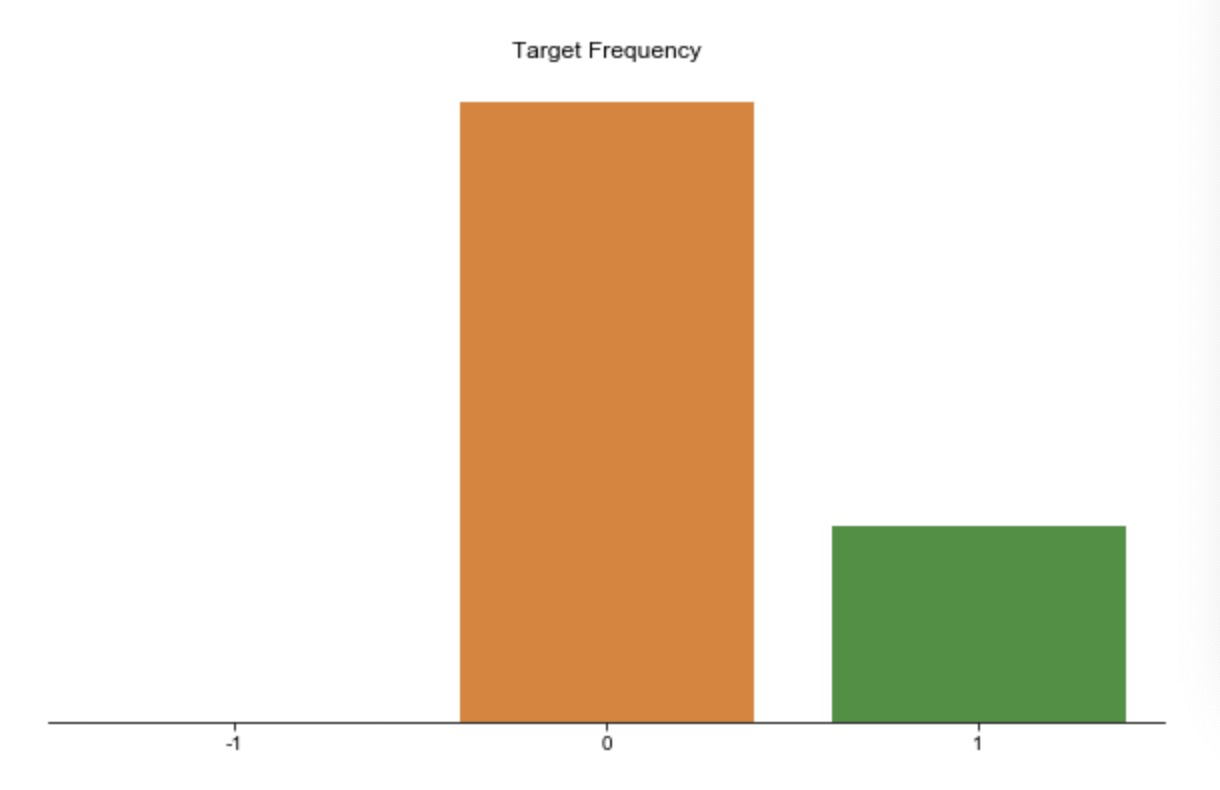
****

Figure1: Targer Data Distribution

According to the boxplot of the numerical data (Figure 2) and the barplot of the categorical data(Figure 3) in the test and training set, both the numerical data and categorical data share very similar distributions in these two separate datasets, meaning that we can generalise our model from the training set to the test set.

The correlation plot(Figure 4) shows that most of the features are uncorrelated with each other, which makes the PCA non effective. We try to reduce the number of features to reduce the variance of our prediction by Principle Component Analysis, but it doesn’t work well since the small correlation between predictors.

Further, the features don’t seem to have a linear relationship with the response variable. So the linear models for classification, for example logistic regression, LDA, and QDA might not have a good prediction performance in this case. To analyze this dataset, we will generate some nonlinear features, and focus mainly on the tree based ensemble models and stack the classification results together.

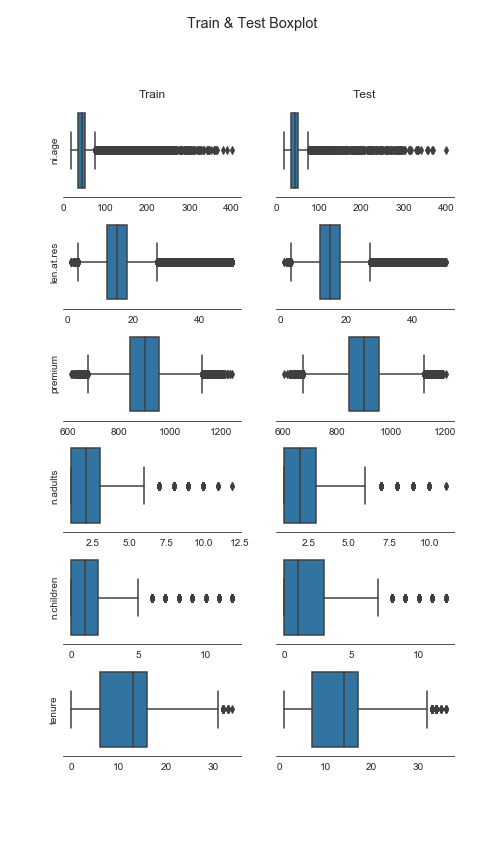


Figure 2: Boxplot of numerical data for test and training set

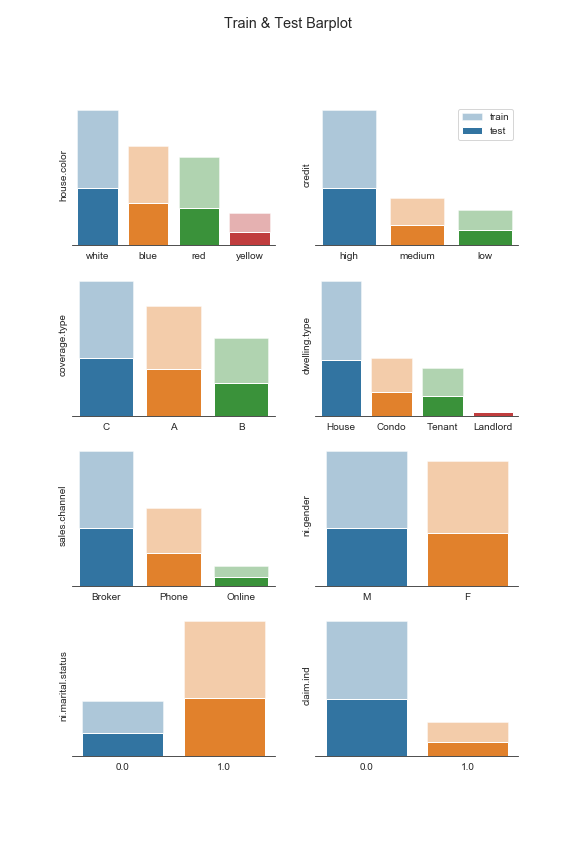


Figure 3: Barplot of categorical data for test and training set

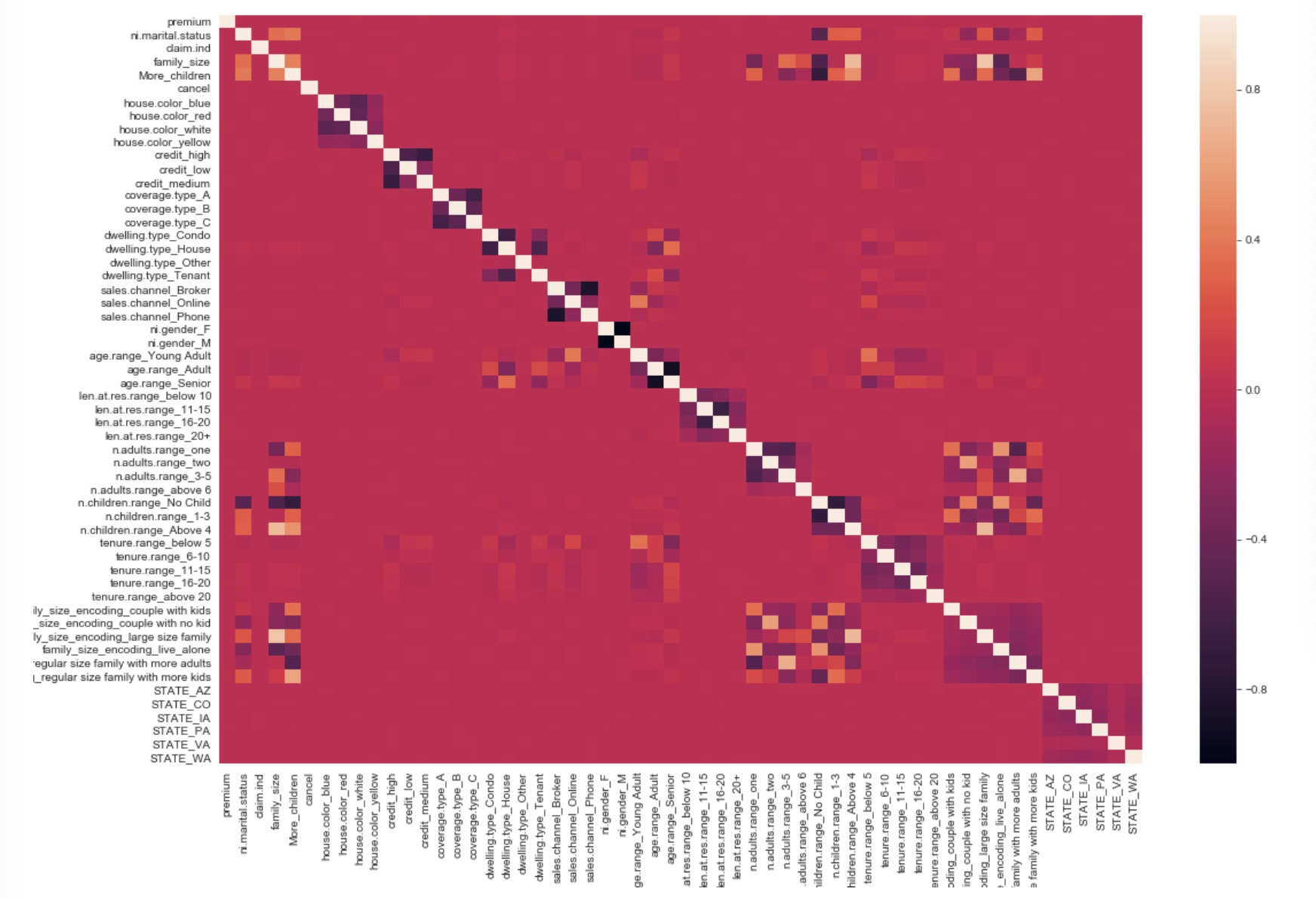


Figure4: Correlation plot

* 1. **Feature Engineering**

**Mean encoding**: Instead of using one-hot encoding, we replaced the categorical data with the mean response of each category to include more meaningful information, and also to deal with the features with too many categories.

**Premium per person/adult:** Premium value itself seems to be an important feature, but it does not perform well in the model. It is more reasonable to consider premium divided by family size and number of adults. Thinking about the features from data understanding and real life is sometimes an effective way in feature engineering.

**Zip code:** Consider neighborhood effect by mapping the zip codes to corresponding cities, states, longitude, latitude and average income. As for cities and states, we did mean encoding as before.

* 1. **Model fitting**

**Metrics: AUC**

To evaluate the performance of the model, we used AUC(Area under curve), which is given by



Since the dataset is imbalanced, it is not fair to use classification accuracy to measure the model, because even a random guess can achieve a nearly 75% accuracy. Therefore, we used AUC as our metrics when we were training and evaluating the model. Instead of giving a decision threshold and the predicted results, AUC can measure the result of the model given all possible thresholds.

**Base Models: Random Forest, XGBoost, LightGBM and AdaBoost**

We could hardly find any linear relationship between the predictors and our response and linear model did not perform well, so we basically tried tree-based models. We first tuned parameters and trained each of the model separately to make sure each has an acceptable performance. Since the feature importances in the models vary, which means that models are making decisions from different aspects of the data, we could use stacking to combine the model to slightly improve the performance.

**Stacking**

Stacking is an ensemble learning technique to combine multiple classification models via a meta-classifier. Above the base models level, we built up a stacking level to combine the prediction results of the base models, where predicted probabilities from the base models were considered as predictors, and a logistic regression model was built on them to predict the target probabilities. We used Google Cloud to train the models.

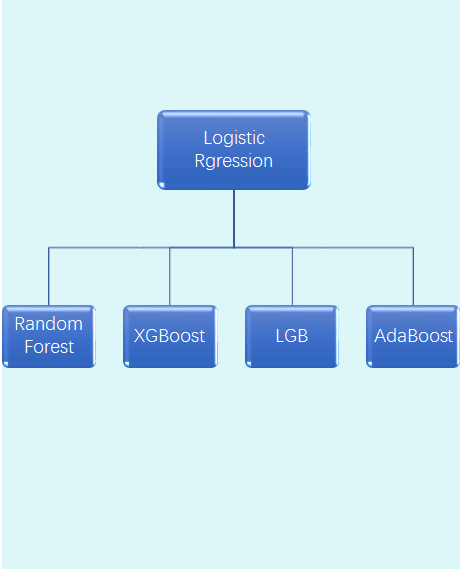


Figure5: Stacking and Base Models

1. **Results**

After finishing modeling, we plot the feature importance for XGB, random forest, lightGBM and adaboost (Naive Bayes don’t have feature importance, so we do not plot it). The plot is as following:

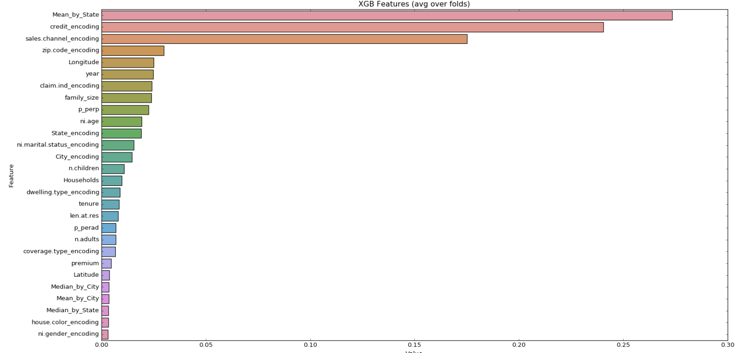
****

Figure 4: Feature importance of Xgboost

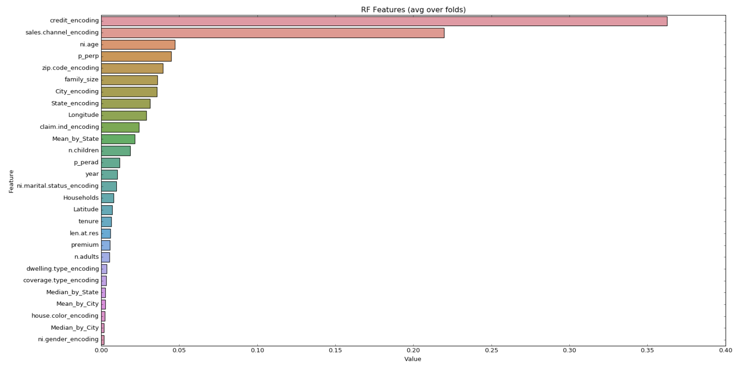
****

Figure 5: Feature importance of Random Forest

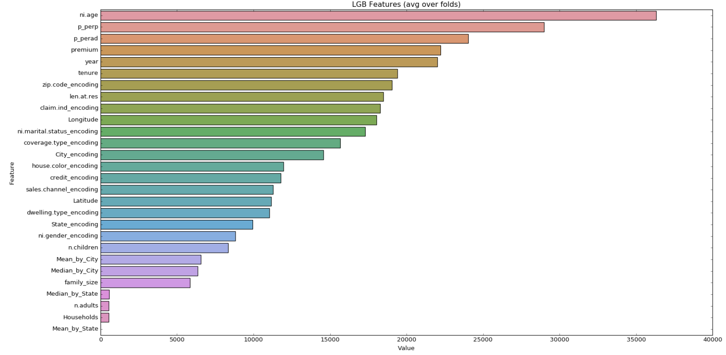
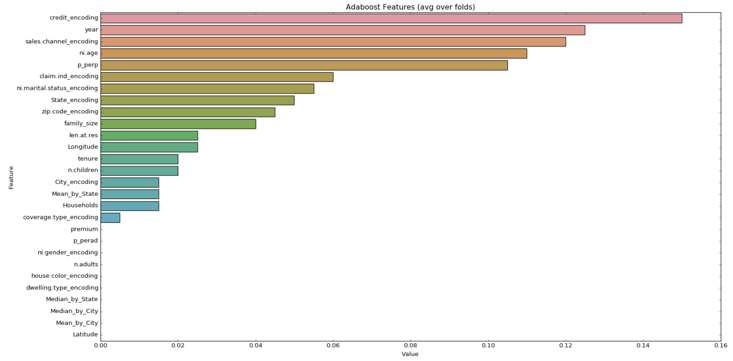
****

Figure 6: Feature importance of lightGBM

****

The top 5 feature importance of each model are as follow:

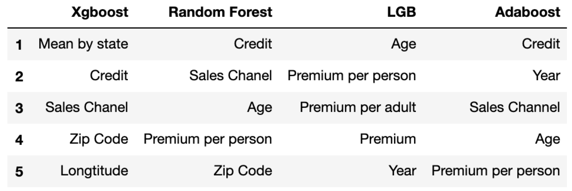


Figure 7: Summary of feature importance

In summary, the importance features are:

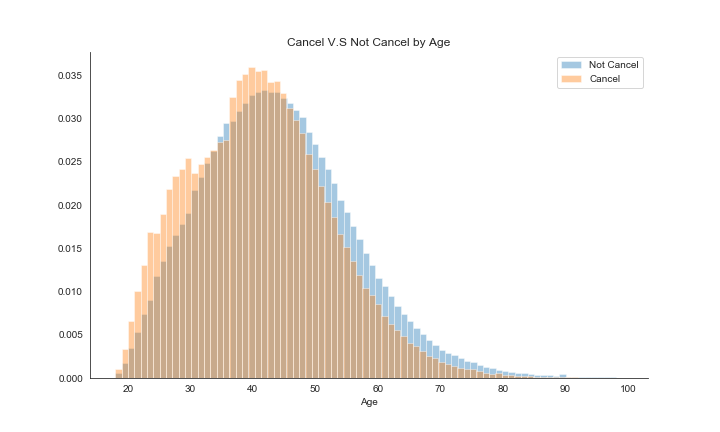
* First Tier: Credit, Age, Sales channel, Premium per person
* Second Tier: Zip Code, Year
* Thrid Tier: Mean by state, Longitude, Premium per adult, Premium.

In more details Credit, Age, Sales channel and Premium per person appear in 3 out of 4 models, Zip code and year appear in 2 models, Mean by state, Longitude, premium per adult and premium appear in 1 model. So we can say Credit, Age, Sales channels and Premium per person can be the most important factors. Next we will going to interpret these factors.

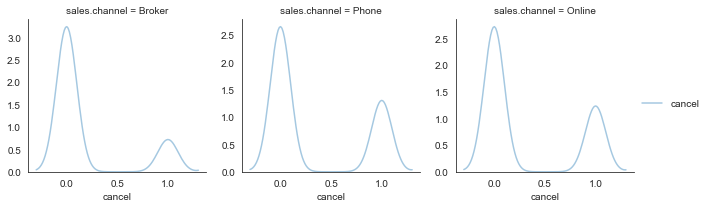
* 1. **Business Insights**

The first factor is the financial credit of the client. This factor can be interpret along with mean income group by state. If the client has good financial credit, which means that they are not likely to wrong about the financial status. So these group of clients will care more about their quality of live, so these client are more likely to live in a luxury house, if there are some damage in their house, they will pay a lot of money to fix the damage without insurance. So these people are not likely to cancel the policy in the condition of sufficient money. In the same reason, if the state has high mean income, which means that we are more likely to see high financial credit client in the state.

The second factor is age. If we plot the age without target, we can observe people between 20 to 45 are more likely to cancel the policy, but people who are older are not likely to cancel the policy. That can be interpret in this way: young people are good at using social media, they have more information of this kind of policy. Once they find the better alternative choices, they may cancel this one and buy the other one in some conditions. However, for elder people, they don’t have many choices because they are not good at using social medias. So they are less likely to find a good alteration for this policy.



The third factor is sales channel, which can be divided as phone, online. This factor is also reasonable since your are more likely to trust people you familiar with. If the clients bought the product from broker, they actually discuss the product with the broker in person, so they are not likely to cancel the policy because the product is recommended by the people they familiar. However, if they bought the product by phone or from online, they are more likely to cancel because it is less trusty.



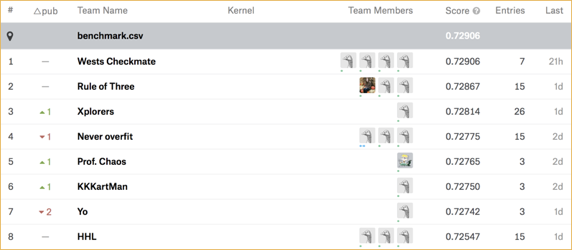
The Fourth factor is premium per person, this variable is equal to premium divided by family size, where family size is equal to the number of children plus the number of adults.This factor can be interpret as how much money of this product each family member should afford. This factors can reflect financial status of the family from the other aspect. If this value is large, which means each family member should afford more money of this policy, and they could cancel this policy if this value is too high.

1. **Conclusion**

Our Final result is as following:

* Training AUC: 73.32%
* Validation AUC: 73.10%
* Test AUC: 72.775%

And we finally got 4th place in the competition.



In conclusion, our project has great business value, we conclude that personal information(financial credit, client age), product information(sales channel, premium), location(zip code, longitude, mean income group by state) and time period(year) are importance for cancellation prediction.

In this project, we made some mistakes at the beginning. The first one is that, we only looked at test AUC for our first model. After finishing data cleaning, we fitted our first model without tuning. The test AUC score is around 50%, which is really bad. We were confused and nearly gave up. However, after we looking at training score, it is around 99%, which implied overfitting. So we learned that we should spend more time on tuning model rather than seeking data insight at the beginning of this project, because if we fitted a good model, it can be easier to find data insight after we plot the feature importance.

The second thing we learned is that run time is a big problem. Our raw dataset has around 1 million rows and 18 columns, it is not a big dataset but it takes a whole data to run Xgboost algorithm without parallel computing. So we had to use Google Cloud Platform to run the algorithm. But still, we spent a hard time on tuning parameters because Xgboost has lots of parameters to be tuned. So we decided to use random search rather than grid search to tune all the parameters of Xgboost. However, random search are not guaranteed to find best parameters like grid search, since in each iteration, random search can only choose one random set of parameters in a selected range. The performance of random search can be worse than grid search, but the speed is much faster. We learn that we have to sacrifice either performance or speed when we build models.

We had tried other methods, such as subsampling and oversampling. Although these methods are not used in this project because the final performance is not good, we also learned that there are some drawbacks of these two method. For subsampling, we could loss a lot of information because we need to delete a large amount of data in majority class, which introduce large bias. For oversampling, such as SMOTE, we need to generated a lot of fake data for minority class, which may introduce noise and lead to large bias. Moreover, when we were doing over samping and cross validation. There is a common mistake, which we made as well at beginning--We can’t not do oversampling first, then doing cross validation, it will result in overwhelming good score. That is because we leak the information from validation set to training set in each iteration of cross validation. The right way is that, we should do oversample in each iteration of cross validation rather than outside the loop. This point is very important, although it is not included in our report, we can use this technique in our future.

**7. Appendix**

1. **Contribution Notes**

Yunhao Huang: Data cleaning & EDA, modeling, feature engineering brainstorm

Yiming Tan: Feature Engineering & Modeling

Xiaoxi Zhao：Brainstorming and Modeling

1. **Data Overview**

****

The whole dataset can be accessed through <https://github.com/multivariate-no-overfitting/Final-PJ/blob/master/train.csv>

1. **Github Page**

<https://github.com/multivariate-no-overfitting/Final-PJ>

1. **Code**

Due to the page length, we will not include the code in this report, but the code can be checked through github link <https://github.com/multivariate-no-overfitting/Final-PJ/blob/master/Whole%20Procedure.ipynb>

**8. Bibliograpghy**

Chen, T., & Guestrin, C. (2016). XGBoost. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD 16*. doi:10.1145/2939672.2939785

Guolin Ke, Qi Meng and Thomas. (2017) LightGBM: *A Highly Efficient Gradient Boosting Decision Tree.*