HTML 2021 Final Project - Churn Category Classification

Hsiao-Tzu Hung r08922a20, Chun-Yen Chen r09222037, Di-En Sung 60941006s

1 Introduction

Attracting new customer and retaining existing customer is important to a telco company to grow their revenue. However, most of the research shows that it costs about 6 to 7 times more to get a new customer while keeping the customers already have needs less effort. The company needs to figure out the reasons why the customer decide not to subscribe the service or buy the product of the company. There are various of reasons causing the customer terminate the contract such as the lower price offer providing by competitor, the bad experience of the service, etc.

If the churn rate is high, it will hinder the growth of the company revenue. As the data scientists in the telephone company, churn analysis must be done by us. We need to clarify which factor drives the customer break off the contract and determine the method to predict churn on the individual customer churn or not.

2 Data Exploration

The data sets for this classification problem are processed from the IBM telco customer churn data. The problem is a multi-class classification problem, where the goal is to predict the customers' category: {No Churn, Competitor, Dissatisfaction, Attitude, Price, Other}. *No Churn* means that the customer will keep using the service, while the other five categories indicate the reason that the customer leaves the service. There are totally 4226 customers' data used for training, and 1409 customers' data need to be predicted.

2.1 Imbalanced dataset

The distribution of the target label is imbalanced. 73.4 % of the training data belongs to *No Churn*, and the class with fewest sample is *Other*, with only 2.7 %. As a result, the key to improve the classification performance will be to handle the problem of imbalanced data.

2.2 Dive into data

We first start by take a look at the relation between the features. There are some interesting observation as following.

1. *Contract*: Most of customer with Month-to-Month Contract opted to move out as compared to that of customers with One Year Contract and with Two Year Contract. Major customers who moved out were having Bank Withdrawal as Payment Method.

2. *Internet Type*: A lot of customers choose the Fiber optic service and it's also evident that the customers who use Fiber optic have high churn rate, this might suggest a dissatisfaction with this type of internet service.
3. Most of the senior citizens churn.
4. Most customers churn in the absence of online security.
5. Customers with Paperless Billing are most likely to churn.
6. Customers that doesn't get married are more likely to churn.
7. Customers with No Premium Tech Support are most likely to churn.
8. Customers with higher Monthly Charges are also more likely to churn.
9. New customers are more likely to churn.

2.3 Correlation of charge information in service data

Service.csv offers information about the types of service customers use and how they are charged. Among these data, the data describing the type of service is categorical, and the data relating to the charge is numerical. Some numerical data represent the average cost, and some describe the total cost. By plotting their paired correlation, we can find some correlation between them. For example, the *Total Long Distance Charges* is correlated to *Avg Monthly Long Distance Charges*. As a result, we only use part of the features for training. The used features are listed in Section 3.3.

3 Missing Value Handling

3.1 Simple intuition between features

The following brackets of the missing values can be filled by each other. {*'Age', 'Under 30', 'Senior Citizen'*}, {*'Dependents', 'Number of Dependents'*}, {*'Lat Long', 'Latitude', 'Longitude'*}, {*'Referred a Friend', 'Number of Referrals'*}, {*'Total Long Distance Charges', 'Avg Monthly Long Distance Charges', 'Tenure in Months'*}, {*'Unlimited Data', 'Total Extra Data Charges'*}, {*'Total Refunds', 'Total Charge', 'Total Long Distance Charges', 'Total Extra Data Charges', 'Total Revenue'*}.

3.2 Bayesian method

We used the Bayesian probability to fill the missing value. First, we find the Bayesian probability of each data. Second, we filled the missing value by probability based on the existing value. Finally, pick the highest probability to be the filling value.

3.3 Nearest Neighbor imputation

The k-Nearest Neighbor(KNN) algorithm is a popular machine-learning-based model for imputation [2]. Following the tutorial written by Jason Brownlee [1] and Kyaw Saw Htoon [2], we work on the imputation in the following steps:

[1]`https://machinelearningmastery.com/knn-imputation-for-missing-values-in-machine-learning/`
[2]`https://medium.com/@kyawsawhtoon/a-guide-to-knn-imputation-95e2dc496e`

1. Select the features that need to be imputed. We choose: *'Tenure in Months', 'Avg Monthly Long Distance Charges', 'Avg Monthly GB Download', 'Total Extra Data Charges', 'Total Refunds', 'Offer', 'Phone Service', 'Multiple Lines', 'Internet Service', 'Internet Type', 'Online Security', 'Online Backup', 'Device Protection Plan', 'Premium Tech Support', 'Streaming TV', 'Streaming Movies', 'Streaming Music', 'Unlimited Data', 'Contract', 'Paperless Billing', 'Payment Method', 'Referred a Friend', 'Gender', 'Senior Citizen', 'Married', 'Dependents', 'Under 30'.*

2. Transfer all the categorical data to one-hot representation.

3. Because KNN Imptuer is a distance-based method, we need to normalize all the data. Here we scale the variables to have values between 0 and 1.

4. Apply the KNNImputer provided by Scikit-learn [3] with *n_neighbors=10* and *weights=distance* to the normalized data.

5. For *'Satisfaction Score', 'Latitude', 'Longitude'*, we don't use knn imputer. Instead, we only fill the missing value with the median of the feature.

4 Approaches

4.1 Approach 1: Logistic regression with class weight

Logistic regression is easy to realize and achieves good performance with linearly separable classes. As a result, it is natural to start with this method. As we discussed in the lecture, we need data to help the logistic regression to decide the boundary and update the model by calculating the error. However, if the dataset is imbalanced, then the process will failed to update the model by the data from minority classes. To address this problem, we use the weighted Logistic regression model to take into account the skewed distribution of the classes.

To be more specifically, in the *LogisticRegression* function of scikit-learn, we set the *class_weight* of to *balanced*. In the setting, the weights of classes is $w_j = \frac{N_{total}}{K \times n_j}$, where w_j is the weight for class j, N_{total} is the total number of the dataset, K is the number of the output class, and n_j is the number of the samples in class j. This means that we give higher weights to the the minority class and lower weights to the majority class. The weights will be used in the log loss function, and give the loss penalty.

4.2 Approach 2: Adaboost with 2 stage classification

From the previous approach, logistic regression doesn't seem to work so well. One of the reasons might be that the data is not linear separable. Hence, we want to introduce a more powerful algorithm, Adaboost, which can deal with nonlinear data. After training with Adaboost, the F1-score on the private data is 0.261 which is still not good enough. We used to classify the customer to the category in one time but this classification strategy seems too difficult for a model to make right prediction. All things considered, to simplify the classification task, we first transform the problem into predicting whether the customer is churn or not. Since there is correlation between 'Satisfaction Score' and 'Churn', it might be more easier to classify if a customer is churn or not at first. Once the customer is classified as churn, then we'll break down the churn category, i.e., stage 2 to predict which reason trigger the customer churn . In both stage, we use Adaboost to do the classification.

The parameter of the Adaboost is shown in Table 1.

Parameter	n_estimators	booster	max_depth	learning rate	subsample	eval_metric
Value	500	Decision Tree	2	1	0.8	auc

Table 1: The parameters of the Adaboost.

4.3 Approach 3: XGboost with oversampling

XGBoost (eXtreme Gradient Boosting) [1] is a machine learning library under the Gradient Boosting framework, and it is designed for speed and performance. The library provides a system for use in various computing environments, such as parallelization using multiple CPU cores and distributed Computing for training huge models utilizing a cluster of machines[3]. XGBoost is also widely used by machine learning practitioners to create state-of-the-art data science solutions[4].

In this work, we use the Python implementation[5] to run the experiment. The parameter we used for the XGboost is shown in Table 2. We choose to use *auc* as evaluation metric because our dataset is imbalanced and we calculate the F1-score as a measure of performance. According to the document, if auc is used for eval_metric, then we need to specify *multi:softprob* to objective, because AUC is calculated by 1-vs-rest with reference class weighted by class prevalence. Other parameters are chosen by using cross validation.

Parameter	n_estimators	booster	max_depth	eta	subsample	objective	num_class	eval_metric
Value	100	gbtree	24	0.2	0.8	multi:softprob	6	auc

Table 2: The parameters of the XGboost.

As for handling the imbalanced dataset, we use the *RandomOverSampler* provided by *imblearn*[6] toolkit with a random seed to be 42. The default sampling stratedy is *not majority*, which means it will resample all classes but the majority class.

5 Discussion

5.1 Missing value handling strategy

We fill in missing value by Section 3.1 & 3.3 respectively. In the training process, we both trained the Logistic regression model with imbalance data and then predict the category in one time. The F1-scores of fill-in method 3.1 and 3.3 on the Kaggle private data are 0.141 and 0.283 respectively. Thus, we choose to use Section 3.3 (Nearest Neighbor imputation) for the following discussion.

5.2 Classification performance

Table 3 shows the F1-score of all approaches. Among the first three models, XGboost achieves the best result with 0.308, however the difference is not substantial. Next, we choose the XGboost to combine with the two strategies that handling the imbalanced data problem. the result shows that we can improve the classification performance by the strategies. Among these two strategies, oversampling improves most, which can reach 0.350.

[3]https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/

[4]https://github.com/dmlc/xgboost/tree/master/demo#machine-learning-challenge-winning-solutions

[5]https://xgboost.readthedocs.io/en/stable/python/python_intro.html

[6]https://imbalanced-learn.org/stable/

Approach	F1-score
Adaboost	0.261
Logistic regression with class weight	0.297
XGboost	0.308
XGboost w/ 2-stage classification	0.322
XGboost w/ oversampling	**0.350**

Table 3: The F1-score of the approaches.

5.3 Efficiency

	Logistic Regression	Adaboost	XGboost
Train time complexity	$O(nd)$	$O(ndw)$	$O(KD\|\|x\|\|_0 logn)$
Test time complexity	$O(d)$	$O(dw)$	$O(KD)$

Table 4: The time complexity of the three methods.

As we can see in Table 4, Training with Logistic regression takes $O(nd)$, where n is the number of samples, d is number of feature.[7] Prediction for a new sample takes $O(d)$. When it comes to training Adaboost, we will go through n variables for d features to find out the decision stump, and then calculate their loss function and to update the weights. In this process, it takes $O(nd)$ operations. The same calculation is repeated for w number of weak learner so it takes $O(ndw)$ operations. While testing, the model predicts depending on the decision stump or the weak learner which comprises only one feature. This operation is repeated for each weak learner and each weak learner only takes one feature into consideration which means that this process takes $O(dw)$.[8] As for XGBoost, it takes $O(KD||x||_0 logn)$[1], where K is the number of trees, D is the depth of the trees, and x is the number of non-missing entries in the training data. Prediction for a new sample takes $O(KD)$. All the methods can be trained within 30 seconds on the training set even with oversampling strategy.

5.4 Scalability

Both Logistic regression and XGboost can be parallelized during training. Using the Logistic regression in Scikit-learn [9], we can simply assign *n_jobs* for multi-core training. As for XGboost, one of its biggest contribution is that it offers a scalable system that support parallel Learning, distributed Computing, and Out-of-Core Computing.

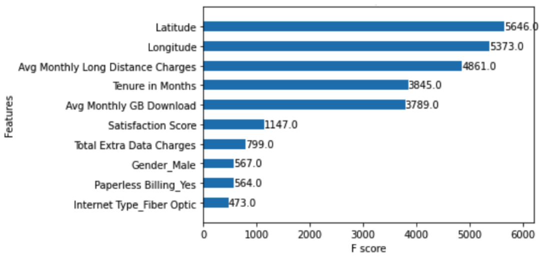
On the other hand, we could hardly train Adaboost with multi-core because weightings of the examples in the next iteration of the algorithm depends on the performances of the previous iteration[10]. That is, a new iteration cannot be started before the previous one being done. Despite this, we still can make the classification parallelized because the weak learner can work independently.

5.5 Interpretability

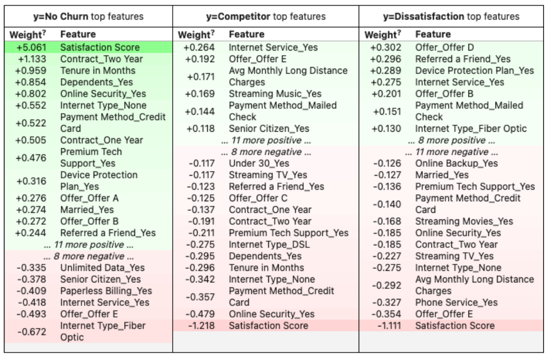
XGboost API offers an easy way to print out the importance of the features. From Figure 1(a) we can see that the top five features are Latitude, Longitude, Average Monthly Distance Charges, Tenure in Months, and Avg Monthly GB Download. As for the Logistic regression, we can use additional toolkit called *eli5* to show the weights of the features. Figure 1(b) shows an example of weights in the Logistic regression when predicting *No Churn, Competitor, Dissatisfaction.* There are some interesting

[7]https://levelup.gitconnected.com/train-test-complexity-and-space-complexity-of-logistic-regression-2cb3de762054
[8]https://medium.com/@chaudhurysrijani/tuning-of-adaboost-with-computational-complexity-8727d01a9d20
[9]https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
[10]https://www.researchgate.net/post/Which_is_the_best_way_for_parallelizing_the_Adaboost_algorithm

observation we can get from this result, and some of them are also aligned to the observations mentioned in Section 2.2. First of all, when the target is *No Churn*, *Satisfaction Score* and *Contract_Two Year* are the top two features. This matches our common sense. When the target is *Dissatisfaction*, *Offer_Offer D* has the largest weight, and the *Internet Type_Fiber Optic* also has a positive weight. This is aligned with the observation mentioned in Section 2.2 that customers choosing *Fiber optic service* have high churn rate. This might offers us some insight when making business decision.



(a) XGboost



y=No Churn top features	
Weight?	Feature
+5.061	Satisfaction Score
+1.133	Contract_Two Year
+0.959	Tenure in Months
+0.854	Dependents_Yes
+0.802	Online Security_Yes
+0.552	Internet Type_None
+0.522	Payment Method_Credit Card
+0.505	Contract_One Year
+0.476	Premium Tech Support_Yes
+0.316	Device Protection Plan_Yes
+0.276	Offer_Offer A
+0.274	Married_Yes
+0.272	Offer_Offer B
+0.244	Referred a Friend_Yes
	... 11 more positive ...
	... 8 more negative ...
-0.335	Unlimited Data_Yes
-0.378	Senior Citizen_Yes
-0.409	Paperless Billing_Yes
-0.418	Internet Service_Yes
-0.493	Offer_Offer E
-0.672	Internet Type_Fiber Optic

y=Competitor top features	
Weight?	Feature
+0.264	Internet Service_Yes
+0.192	Offer_Offer E
+0.171	Avg Monthly Long Distance Charges
+0.169	Streaming Music_Yes
+0.144	Payment Method_Mailed Check
+0.118	Senior Citizen_Yes
	... 11 more positive ...
	... 8 more negative ...
-0.117	Under 30_Yes
-0.117	Streaming TV_Yes
-0.123	Referred a Friend_Yes
-0.125	Offer_Offer C
-0.137	Contract_One Year
-0.191	Contract_Two Year
-0.211	Premium Tech Support_Yes
-0.275	Internet Type_DSL
-0.295	Dependents_Yes
-0.296	Tenure in Months
-0.342	Internet Type_None
-0.357	Payment Method_Credit Card
-0.479	Online Security_Yes
-1.218	Satisfaction Score

y=Dissatisfaction top features	
Weight?	Feature
+0.302	Offer_Offer D
+0.296	Referred a Friend_Yes
+0.289	Device Protection Plan_Yes
+0.275	Internet Service_Yes
+0.201	Offer_Offer B
+0.151	Payment Method_Mailed Check
+0.130	Internet Type_Fiber Optic
	... 8 more positive ...
	... 11 more negative ...
-0.126	Online Backup_Yes
-0.127	Married_Yes
-0.136	Premium Tech Support_Yes
-0.140	Payment Method_Credit Card
-0.168	Streaming Movies_Yes
-0.185	Online Security_Yes
-0.185	Contract_Two Year
-0.227	Streaming TV_Yes
-0.275	Internet Type_None
-0.292	Avg Monthly Long Distance Charges
-0.327	Phone Service_Yes
-0.354	Offer_Offer E
-1.111	Satisfaction Score

(b) Logistic regression

Figure 1: Feature importance for XGboost and Logistic regression

6 Conclusion

In this project, we investigated three classification models and two strategies for imbalanced dataset. Among these approaches, XGboost provides an easy way to interpret what model have learned and is well established in system design, which is efficient and scalable. For the classification performance, XGboost with oversampling training strategy can achieve the best performance among the approaches listed here. As a result, we recommend XGboost with oversampling as the model to use practically.

7 Contribution

- Hsiao-Tzu Hung r08922a20: kNN imputation, weighted Logistic regression, XGboost with oversampling, interpret models, fight on the score board, report writing(45%)
- Chun-Yen Chen r09222037: Simple intuition between features, Bayesian method, report writing(10%)
- Di-En Sung 60941006s: data exploration, Adaboost with 2 stage classification, logistic regression with two kinds of missing value handling, report writing (45%)

References

[1] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *CoRR* (2016).

[2] Wei-Chao Lin and Chih-Fong Tsai. "Missing value imputation: a review and analysis of the literature (2006—2017)". In: *Artificial Intelligence Review* (2020).

[3] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.