Machine Learning Final Project - Team HCH

Jing Neng Hsu
b08902022@ntu.edu.tw

Chih Chien Chang
b08902128@ntu.edu.tw

Yao Ting Huang
b08902136@ntu.edu.tw

1 Introduction

In this project, we are going to predict customer's service status based on some labeled data. For this multi-class classification task, we will introduce our work in the following order:

1. **Data Analysis:** This section includes our analysis about the dataset, including correlation matrix of the dataset.
2. **Data Preprocessing:** This section includes preprocessing of data before data becoming a valid dataset(no loss data and numerical), such as data encoding and data imputation.
3. **Feature Engineering:** This section includes processing on features, such as feature selection, feature transformation and feature extension.
4. **Model Selection:** This section includes selection of different models in training, such as classic classification model and automated machine learning.
5. **Conclusion:** This section concludes the best approach to deal with this task and what we can do to further improve the performance of this task in the future.
6. **Job allocation and Reproduce:** This section will includes our job allocation and the github url for reproduction.

2 Data Analysis

2.1 Unbalanced data

First of all, target categories are unbalanced, over 70 percents of data are labeled as No Churn, about 12 percents are Competitors. The remaining only takes up 15 percents. Thus, the technique of subsampling may be take into consideration during preprocessing data.

2.2 Covariance matrix

To find out the relationship between each feature and customers categories, we compute the covariance matrix of all the features and customers categories by training data. Below are some observations of it.

2.2.1 Strongly related. Given there are more than 30 features, we first consider features that are strongly related. The result is demonstrated in Figure 1. However, the resulting relations are obvious. For example, zip code, latitude and longitude has high covariance due to zip code being determined by latitude and longitude. And those accumulative data such as total charges, total long distance charges and total revenue highly depends on tensure in months. The only relation that is not so trivial is Monthly charge has something to do with total charges and total revenue.

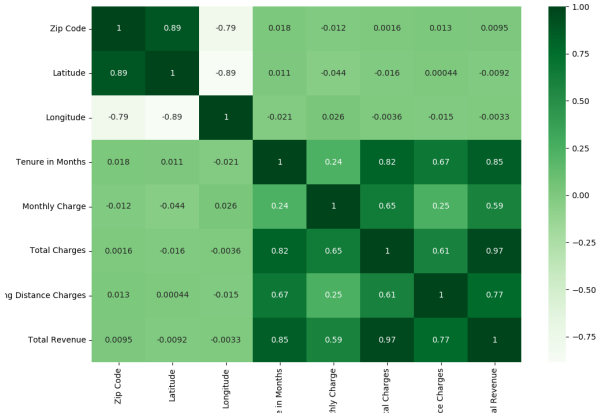

Figure 1. those attributes has high covariance (bigger than 0.6) to some other attributes.

2.2.2 Medium related. Next, we consider features that are medium related. The result is demonstrated in Figure 2. Those attributes indicating if the customer use the service has medium covariance with monthly charges.And those who use internet service downloads more things.Those remains are some trivial relations:(Age, Under 30, senior), (Dependents,Number of Dependents), (No churn, Competitor). The last one is because they are two most frequently appears category, and they are disjoint.

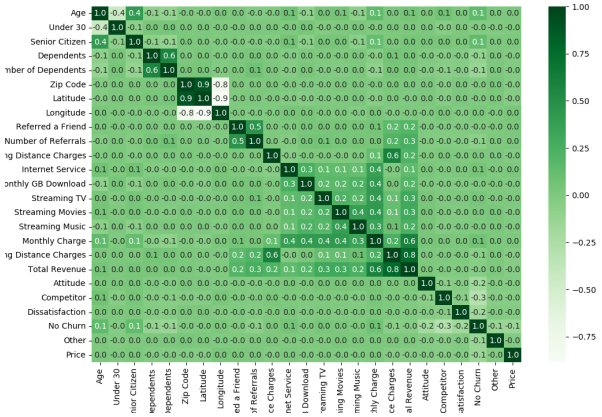

Figure 2. those attributes has medium covariance (bigger than 0.3) to some other attributes.

2.2.3 Weakly related. Since there is no high correlation between features and target categories yet, we turn our attention to those features that have weak relation(>0.1) with the target categories. The full covariance matrix is shown in Figure 3. The only one such relationship is between satisfaction score and no churn. The result accord to our intuition. Satisfied customers are more likely to stay.

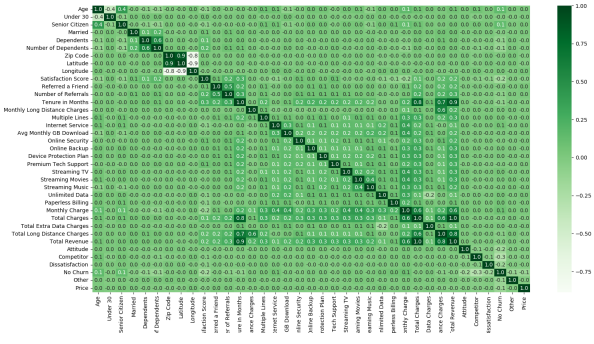

Figure 3. the full covariance matrix.

2.3 summary

As the correlation matrix shows, it is difficult to predict test data based on the value of other columns, so machine learning is suitable for this task.

3 Data Preprocessing

In this section, we will discuss and experiment several different ways to make our data eligible for training, including data encoding and data imputing.

3.1 Data Encoding

The original dataset is generated by gathering all 6 different csv file given by TA. However, there are some not-numerical feature which is not acceptable for machine learning model, the encoding of not-numerical feature is needed. In this section, we proposed two approaches: Heuristic Encoding and Categorical Encoding.

Heuristic Encoding. One heuristic approach to encode the data is to encode all feature no matter it is numerical or not, this method saves a lot of coding time at the expense of correctness. Plus, this method can deal with loss data since NaN is served as a special type of feature which will be encoded.

Categorical Encoding. Another approach to encode is to encode all the categorical data. This method treats NaN in numerical feature as the mean value of the column.

Experiment Result. The following Evaluation will based on F-score of three models(ADAboost, LightGBM and Random Forest, the introduction of these three model will be included in section 5)

Score	Encoding Method and	ADAboost	LightGBM	Random Forest
Public	Heuristic Encoding	0.23226	0.27739	0.27660
Public	Categorical Encoding	0.28426	0.32498	0.27075
Private	Heuristic Encoding	0.23003	0.30992	0.27052
Private	Categorical Encoding	0.26655	0.32755	0.26216

Table 1. F-score of different encoding method

As the table shows, the Categorical Encoding method has better performance than Heuristic Encoding on most model since the former is a more logical encoding method. One interesting thing is that in Random Forest Model, two method share almost the same performance, the possible cause is that tree-based model is not affected by value of a certain feature that much. For example, encoding tuple "1.5, 12.5" to "1, 2" may not affect the prediction of tree-based model but number-related model like ADAboost or LightGBM are likely to be affected.

3.2 Data imputation

Since there are loss data in every feature, some imputation method is needed. We tried 3 different imputation methods: Most Frequent Imputation, Mean Imputation and kNN Imputation.

Most Frequent Imputation. Most Frequent Imputation is a method that imputes missing feature as the most frequent value in that feature column. We think that this method can perform will on the dataset since our data is imbalanced and most features are categorical.

Mean Imputation. Mean Imputation is a method that imputes missing feature as the mean value of the feature column. We think that this method might not perform will since most features are categorical, the mean value might not be that representative.

kNN Imputation. kNN Imputation is a method that puts data onto a vector space and imputes missing data based on the nearest neighbor in the vector space. If the dataset is pretty sparse, the imputation method may perform will.

Experiment Result. The following Evaluation will based on F-score of three models(ADAboost, LightGBM and Random Forest, the introduction of these three model will be included in section 5), and feature extension is applied(see section 4).

Score	Method on categorical	Method on numerical	ADAboost	LightGBM	Random Forest
Public	MF	MF	0.29957	0.30552	0.31774
Public	MF	Mean	0.29957	0.30552	0.31774
Public	MF	kNN	0.29957	0.30552	0.31774
Public	Mean	MF	0.29451	0.31093	0.29709
Public	Mean	Mean	0.29451	0.31093	0.29709
Public	Mean	kNN	0.29451	0.31093	0.29709
Public	kNN	MF	0.29451	0.31093	0.29709
Public	kNN	Mean	0.29451	0.31093	0.29709
Public	kNN	kNN	0.29451	0.31093	0.29709
Private	MF	MF	0.26320	0.30760	0.28026
Private	MF	Mean	0.26320	0.30760	0.28026
Private	MF	kNN	0.26320	0.30760	0.28026
Private	Mean	MF	0.25621	0.29949	0.28288
Private	Mean	Mean	0.25621	0.29949	0.28288
Private	Mean	kNN	0.25621	0.29949	0.28288
Private	kNN	MF	0.25621	0.29949	0.28288
Private	kNN	Mean	0.25621	0.29949	0.28288
Private	kNN	kNN	0.25621	0.29949	0.28288

Table 2. F-score of different imputing method

As the table shows, the imputation on numerical data doesn't affects the performance of the model(this might owing to the number of numerical feature is small) and for the categorical data, the best imputation method is to use the Most Frequent Imputation method. However, performance using these imputation methods are no better than simple encoding, we speculate that this might due to the fact that our imputation method alter the distribution of the categorical data and resulting in poor performance.

3.3 summary

According to the experiment results, the best preprocessing method is to encode the categorical data and NaN. We think the underlying reason behind it is the size of missing data is relatively large, which takes 12 percents of each feature. In this situation, using imputer is worse than simple encoding because the imputation will alter the distribution of dataset too far from the original data.

4 Feature Engineering

In this section, we are going to discuss and experiment several different ways to select/delete/generate features.
In this section, we do some rule based imputation beforehand. Including using feature "Age" to impute "Under 30" and "Senior Citizen", and using "Number of Dependents" to impute "Dependents". This is the correct imputation as the relation showed in description of dataset on kaggle. Plus, all the preprocess of the dataset is the categorical encoding method in section 3.

4.1 Feature Selection

In this section, we do 6 different rough feature selection: Select all feature, drop demographic related feature, drop location related feature(include population and location), drop population related feature, drop satisfaction score and drop service related feature.

Score	drop	ADAboost	LightGBM	Random Forest
Public	none	0.23986	0.31558	0.29290
Public	demographic	0.29760	0.29973	0.28681
Public	loca + pop	0.22746	0.34155	0.27606
Public	population	0.26300	0.32520	0.26210
Public	satisfaction	0.25173	0.22654	0.18872
Public	service	0.25269	0.29697	0.29764
Private	none	0.23150	0.30564	0.29708
Private	demographic	0.27375	0.28686	0.30681
Private	loca + pop	0.23478	0.30469	0.27859
Private	population	0.22505	0.28021	0.27326
Private	satisfaction	0.21118	0.21804	0.18686
Private	service	0.24992	0.28592	0.27170

Table 3. F-score of different feature selection

From the table above, we can see the satisfaction score is the most important feature. Other then that, there is no clear feature selection methods outrun others in all three models.

4.2 Feature Transformation

In this section, we are going to include 3 different feature transformation methods: Polynomial Transform, Normalization, Standardization.

Polynomial Transform. This method will perform a polynomial transformation on features set, generating more features.

Normalization. This method will normalize each row of dataset. We speculate that this method will not lead to a good result since many categorical data lay inside this dataset.

Standardization. This method will make each column in feature set has mean equals to 0 and standard deviation equals to 1. We speculate that this method will also not lead to a good result since many categorical data lay inside this dataset.

Score	Method	ADAboost	LightGBM	Random Forest
Public	none	0.23986	0.31558	0.29290
Public	2-polynomial	0.22742	0.34262	0.26859
Public	normalize	0.24530	0.29634	0.27353
Public	standardize	0.24917	0.30212	0.28516
Private	none	0.23150	0.30564	0.29708
Private	2-polynomial	0.28060	0.27929	0.28104
Private	normalize	0.27208	0.27924	0.26996
Private	standardize	0.23514	0.32729	0.30927

Table 4. F-score of different feature transformation

Experiment Result. According to table above, the feature transformation approaches proposed are not having strong influence on the performance of all three model.

4.3 Feature Extension

In this section, we propose a simple feature extension method we think which can improve the performance of our model. This simple method is to add three features: "Monthly Refunds", "Monthly Extra Data Charges" and "Monthly Long Distance Charges", which are the average of "Total Refunds", "Total Extra Data Charges" and "Total Long Distance Charges" respectively using feature "Tenure in Months".

Score	Feature extension	ADAboost	LightGBM	Random Forest
Public	Yes	0.28979	0.32875	0.25659
Public	No	0.23986	0.31558	0.29290
Private	Yes	0.27569	0.31073	0.28176
Private	No	0.23150	0.30564	0.29708

Table 5. F-score of using feature extension

From table above, we have ADAboost and LightGBM with better performance using feature extension, while Random Forest has poorer performance using feature extension. One possible explanation of this result is the interference of bad random seed in Random Forest model.

4.4 Summary

After examining different feature engineering method, we think simple feature extension is the most reasonable one and it indeed improve the performance of some models. Furthermore, we guess there are several reasons why other feature engineering methods are not working. First reason is the size of the dataset. Dataset for this task is relatively small and has a lot of feature comparing to other dataset. Some feature engineering methods may require the size of dataset over certain threshold because of probable reason. Second reason is the missing data of the dataset. Dataset for this task has 12 percent of missing rate in most features, performing feature engineering on dataset with loss data might increase the chance of making the distribution of feature engineered dataset having a more diverse distribution comparing to original data.

5 Model Selection

In this section, we are going to introduce several common classification methods, including ADAboost, Naive Bayes, Decision Tree, LightGBM, QDA, Random Forest and XGBoost. Then introduce an automated machine learning pipeline package called Auto-sklearn, and compare performance of different models.

5.1 Common classification method

ADAboost. ADAboost is a boosting method proposed by Freund and Schapire in 1995. [7] The key concept of ADAboost is combining a number of classifiers(weak learners) with different weight. Generally, this method is considered to be a solution to tackle overfitting.

Decision Tree. Just like what the professor taught in class, Decision Tree classifier is a tree shaped classifier proposed by Hunt in 1966 [8]. Each internal node of decision tree represents a "test" on a feature, different test result will lead to different rules. Though Decision Tree may reach high accuracy, the chance of overfitting is also quite likely. We expected to see Decision Tree having poor performance since our dataset is relatively small.

Random Forest. To deal with overfitting in Decision Tree classifier, Random Forest was proposed by Ho et al. [5] in 1995. Random Forest classifier's methodology is quite straightforward: train a number of Decision Tree separately and do a equal-weighted voting. This ensemble of Decision Tree classifier can effectively cope with the overfitting problem. However, Random Forest model requires a huge amount of training data, which might not be feasible in this dataset, thus we think Random Forest may not be that good comparing the case where the size of dataset is large.

XGBoost. XGBoost is a Gradient Boosting Decision Tree (GBDT) model. The main idea of GBDT is to find the best place to split the dataset for certain feature, i.e. find the best feature an threshold to create a internal node for Decision Tree. The method used to optimize this process is called Gradient Boosting. Unlike NN based models, it is not feasible to use gradient descent to do optimization. Gradient Boosting [4] is proposed to tackle this problem.
XGBoost(Extreme Gradient Boost) [2] is a method proposed by Chen et al. in 2016. It uses some greedy based algorithm to find the best point to split the dataset: first sort the dataset base on the label of each data and then find the best splitting point by traversing the dataset. This method can not only achieve a good result but also largely save the time of find a good ensemble of decision trees. Because of these advantages, XGBoost is often used as a classifier for classification task. We expected XGBoost has a good performance on the dataset for this task.

LightGBM. Although XGBoost has good performance and efficiency, the step where sorting all data based on their label takes a lot of space and is not cache friendly. LightGBM [6] is proposed by Ke et al. in 2017 to solve the problem. The main differences of LightGBM and XGBoost are histogram based algorithm and the leaf-wise algorithm.
In XGBoost, if a feature consists of continuous value, it takes a enormous amount of time to find a good split point. In LightGBM, we will discretize the values using histogram. In

this way, though the accuracy might decrease a little, the efficiency is largely improved.
In XGBoost, we often have to go through each node level by level(which is called level-wise), even if some nodes only splitting 2 data. This is clearly inefficient. In LightGBM, the order of traversal of internal nodes are based on the size of the data it splits, this leaf-wise algorithm is more efficient than the level-wise algorithm.
Based on the fact that LightGBM is also a ensemble method similar to XGBoost, we expect LightGBM to achieve the similar good performance on the dataset for this task.

Naive Bayes. Unlike other classifiers, Naive Bayes [1] use both feature and label to construct a probability distribution model. With this model and the assumption that each feature is independent, we can use Bayes theorem to estimate the corresponding label of testing data. Based on the result of section 2, we think this classifier might perform well on the given dataset.

QDA. QDA(Quadratic Discriminant Analysis) is a transformation based classifier. The main concept of QDA is to transform high dimensional data onto a low dimensional space with the variance of each group is minimized and the variance between different group is maximized and quadratic line can split different group. In the dataset of this task, we think QDA can have decent performance since the dataset is high dimensional.

Experiment Result. The following section's data are preprocessed with categorical encoding(section 3) and feature extension(section 4).

Model	Public score	Private Score
ADAboost	0.28979	0.27569
Naive Bayes	0.23185	0.23971
Decision Tree	0.28032	0.29949
LightGBM	0.32875	0.31073
QDA	0.28159	0.28716
Random Forest	0.25659	0.28176
XGBoost	0.29448	0.28234

Table 6. F-score of different model

As predicted, ADAboost, LightGBM and XGBoost has good performance. QDA also perform well, this may caused by the fact that QDA doesn't need much data to achieve a good performance. Naive Bayes perform poorly might because missing data rate is too high, causing the probability estimator to make wrong prediction. The reason why Decision Tree outperform Random Forest may due to insufficient data size. Random Forest is a model that needs a large number of dataset to achieve a good performance, while our dataset for this task is relatively small.

5.2 Auto ML

Auto ML is a task whose goal is to automate the process of machine learning task(s). For example, for a two class classification task, an auto ML model will encode features, select suitable models and hyperparameters and optimize the model selected. In our experiment, we used Auto-sklearn [3], thus in the following paragraph, we will introduce the component and methodology of Auto-sklearn.

Auto-sklearn. Auto-sklearn [3] is an automated pipeline structure for classic machine learning task proposed by Feurer et al. in 2015. The classic machine learning task can be splitted into several subtasks: data preprocessing, feature preprocessing, model selection, hyperparameter selection and model training. The Auto-sklearn deal with these problem in the following ways:

- **data preprocessing:** Auto-sklearn will perform some simple method on a given dataset such as one-hot encoding and mean imputation.
- **feature preprocessing:** Auto-sklearn will perform some simple method on a given feature set such as normalization, feature selection and feature transformation.
- **model selection and hyperparameter selection:** Auto-sklearn will perform some Bayesian optimization method to find several good models and hyperparameter for given task. This process is called Combined Algorithm Selection and Hyperparameter optimization (CASH).
- **model training:** Auto-sklearn will train those models, ensemble models and output the result using models and hyperparameters found in the previous step.

Since it is hard to determine the convergence of Auto-sklearn, we will use different training time to see the performance between them.

Experiment Result. The following section's data are preprocessed with categorical encoding(section 3) and feature extension(section 4).

Training time	Public score	Private Score
10 min	0.24790	0.25069
60 min	0.31198	0.32210
100 min	0.30587	0.30284
600 min	0.32187	0.27506

Table 7. F-score of different training time of Auto-sklearn

The Auto-sklearn model with 10 min training time has poor performance clearly because 10 min is to short for model to fit. 60 min and 100 min training time has decent performance. In the case where training time is 600 min, the underlying reason yielding a poor performance is caused by the ensemble weight of Random Forest model increasing over training time, and we know Random Forest might not have

a good performance in the dataset for this task in section 5.1.

5.3 Summary

There are several reasons that can explain why the expected good model has a poor performance. First reason is the size of the dataset. Model that requires a lot of data will not have a good performance on this dataset due to its size. Second reason is the distribution of the dataset. Dataset for this task is mainly comprised of categorical discrete features, which makes some Gaussian distribution based models perform not that well comparing to running on numerical continuous data.

6 Conclusion

After experimenting on different aspect of this task, we conclude that using categorical encoding and simple feature extension plus LightGBM will have a relatively good result. Though almost all models run pretty quickly owing to the small size of dataset, LightGBM maintains a good performance while faster than XGBoost. As for scalability, LightGBM is designated for big data classification/regression task, which makes it quite scalable. These good traits make LightGBM our final model recommendation.

There are still some room of improvement, such as we can train a model focusing on imputing this dataset, or we can use more detailed feature selection method etc. One of the most important thing we got from this dataset is that Autosklearn might have its limit in a small dataset, and it is crucial to select model based on the size of dataset.

7 Job Allocation

- **Jing Neng Hsu** - Model training, experiment, report
- **Chih Chien Chang** - Data analysis, report
- **Yao Ting Huang** - Data analysis, report

8 Reproduce

You can find my code at https://github.com/jonathan-hsu123/Machine-Learning-Final-Project-NTU-CSIE-2021, it includes all the dataset and tools we used.

References

[1] Tony F. Chan, Gene H. Golub, and Randall J. LeVeque. 1979. *Updating Formulae and a Pairwise Algorithm for Computing Sample Variances.* Technical Report. Stanford, CA, USA.

[2] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Francisco, California, USA) *(KDD '16)*. ACM, New York, NY, USA, 785–794. https://doi.org/10.1145/2939672.2939785

[3] Matthias Feurer, Aaron Klein, Jost Eggensperger, Katharina Springenberg, Manuel Blum, and Frank Hutter. 2015. Efficient and Robust Automated Machine Learning. In *Advances in Neural Information Processing Systems 28 (2015)*. 2962–2970.

[4] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics* (2001), 1189–1232.

[5] Tin Kam Ho. 1995. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, Vol. 1. IEEE, 278–282.

[6] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems* 30 (2017), 3146–3154.

[7] Robert E Schapire. 2013. Explaining adaboost. In *Empirical inference*. Springer, 37–52.

[8] Xindong Wu, Vipin Kumar, J Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J McLachlan, Angus Ng, Bing Liu, S Yu Philip, et al. 2008. Top 10 algorithms in data mining. *Knowledge and information systems* 14, 1 (2008), 1–37.