Machine Learning Final Project Report

游家權 R10942152　黃義翔 B08502063　藍瑩 B07705055

Preprocessing

Evaluation:

To evaluate the performance of numerous data processing methods, we decided to use sklearn.linear_model.LogisticRegression[1](Figure 1) to evaluate our performance, and choose the best method based on 10-fold cross validation scores.

```
#using logistic regession
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
pipe= Pipeline([('scaler',StandardScaler()),('clf', LogisticRegression(,multi_class='ovr',class_weight='balanced'))])
scores = cross_val_score(pipe, X, y, cv=10,scoring=make_scorer(f1_score, average='macro'))
scores.mean()
```

Figure 1

Imputation and Encoding:

First, we can fill the missing data in “Latitude” and “Longitude” features with “Lat Long” feature, which reduces nearly half of the missing data in “Latitude” and “Longitude”. Second, through evaluation, we select sklearn.impute.IterativeImputer[2] to fill in the rest of the missing data. Then, for categorical features, the data is rounded to the nearest category, and all negative values in the rest of the data are assigned to 0. Third, for categorical features, we can use one-hot or frequency encoding[3] to encode our data. However, location features like “City” or “Zip Code” have more than a thousand categories. It will cause **curse of dimensionality**[4] if we use one-hot to encode those two features, and the data would be nearly identical if we use frequency encoding. To reduce the number of categories, we use k-means clustering[5] to divide customers into k groups based on latitude and longitude of customer’s residence(Figure 2), and choose k by the validation score. All combination of encoding methods and the corresponding validation scores are shown below:

combination of encoding methods	biggest 10-fold cross validation f1 score(k from 1 to 20)	public score	private score
drop all categorical features and location feature	0.31341233977868066	0.28595	0.33778
one-hot encoding(categorical features)+drop location feature	0.3120524431026143	0.29167	0.35013
frequency encoding(categorical features)+drop location feature	0.3139292727558664	0.27942	0.33883
one-hot encoding(categorical features and location feature)	**0.33575388754240243(k=13)**	**0.30465**	**0.35114**
frequency encoding(categorical features and location feature)	0.32400913594478636(k=5)	0.29403	0.31988
one-hot encoding(categorical features)+frequency encoding(location feature)	0.3190769736738338(k=5)	0.30186	0.34684
frequency encoding(categorical features)+one-hot encoding(location feature)	0.3293012906330455(k=13)	0.29600	0.33790

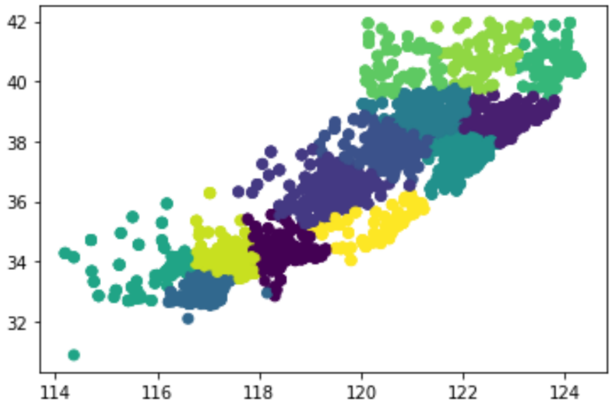


Figure 2: k-means clustering of customer's rersidence location, k=13

Gender	Age	Married	Number of Dependents	Satisfaction Score	Referred a Friend	Number of Referrals	Tenure in Months	Phone Service	Avg Monthly Long Distance	Multiple Lines	Internet Service	Avg Monthly GB						
Online Security	Online Backup	Device Protection Plan	Premium Tech Support	Streaming TV	Streaming Movies	Streaming Music	Unlimited Data	Paperless Billing	Monthly Charge	Total Charges	Total Refunds	Total Extra Data Charges						
Total Long Distance Charges	Total Revenue	No Offer	Offer A	Offer B	Offer C	Offer D	Offer E	Cable	DSL	Fiber Optic	No Internet	Month-to-Month	One Year	Two Year	Bank Withdrawal	Credit Card	Mailed Check	0 1 2
3	4	5	6	7	8	9	10	11	12									

Figure 3: all features readied to be trained

training:

After deciding all preprocessing methods, we can decide C in logistic regression model. The result is listed below:

C	10-fold cross validation f1 score	public score	private score
0.001	0.31234327145586177	0.26407	0.32724
0.01	0.3377813467620546	0.29793	0.33939
0.02	0.3404127998900565	0.29002	0.35276
0.03	**0.3413112734679104**	**0.29738**	**0.35071**
0.04	0.3379492944313618	0.29918	0.35511
0.1	0.3372012612649843	0.30502	0.35235
1	0.33575388754240243	0.30465	0.35114

Permutation Feature importance[6](top 5):

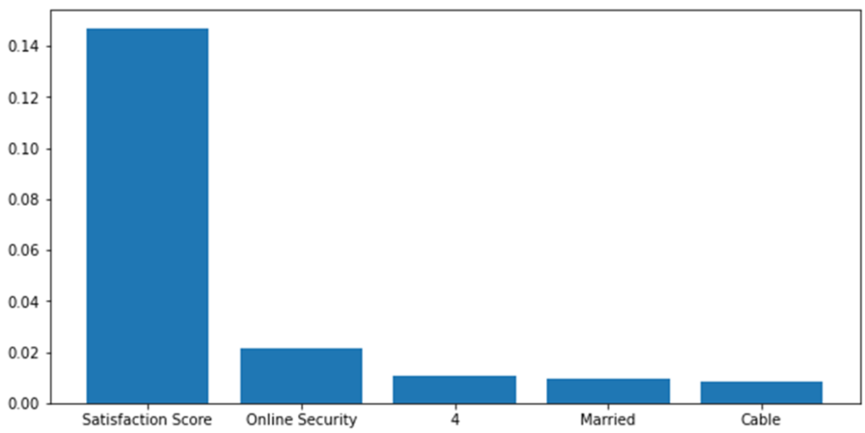
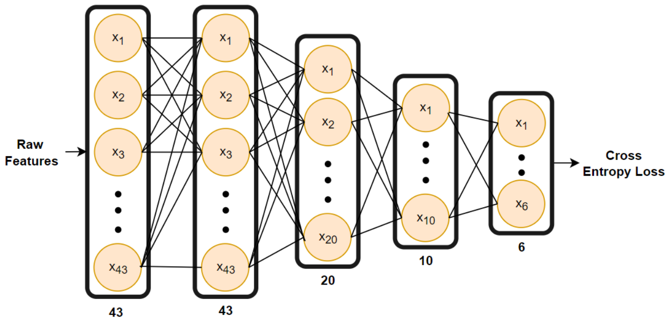


Figure 5: Permutation feature importance

Experiment of MLP(Multi-Layer Perceptron)

We attempt to use MLP to solve this multi-class problem. We design our network through a series of ablation studies, trying to choose the best or the most reasonable setting possible. Adam[8] is used as our optimizer, and Cross Entropy Loss is used to update our network in the experiments below. All models below are trained with 50 epochs.

Structure of our MLP model



Network Design:

Nerwork Structure	Learning Rate	Batch Size	Imbalance Techniques	Public Score	Private Score
(43, 43, 43, 43, 6)	0.0005	4	-	**0.33232**	0.33008
(43, 43, 20, 10, 6)	0.0005	4	-	0.30767	0.32669
(43, 6)	0.0005	4	-	0.29928	**0.33471**

We've tested server possible network structures, and it turns out (43, 43, 43, 43, 6) with 5 fully connected layers have the best performance in general. Although the single layer network (43, 6) has the best f1 score in the private dataset. We believe it's just pure luck, since private dataset is just 50% of the public data.

Batch Size:

Nerwork Structure	Learning Rate	Batch Size	Imbalance Techniques	Public Score	Private Score
(43, 43, 43, 43, 6)	0.0005	4	-	0.33232	**0.33008**
(43, 43, 43, 43, 6)	0.0005	8	-	0.32739	0.31994
(43, 43, 43, 43, 6)	0.0005	32	-	**0.34978**	0.30710

Batch size influences network performance greatly. Small batch size allows the network to update more frequently and randomly, while big batch size lets it update more stable. In this experiment, we choose to use a smaller batch size in our model, because we believe randomly updating will help the network tackle the imbalance problem. It allows minor classes to "dominate" update direction in some lucky draws. We observed this learning behavior by monitoring minor classes f1 scores respectively.

Deal with Imbalance Dataset:

We used two different techniques to deal with the imbalance dataset in the MLP. First, we used a **uniform sampler**[9] to sample training data with high probability if it's in a minor class. Secondly, we used a **weighted loss function**[10] to make the network pay more attention to the minor class loss.

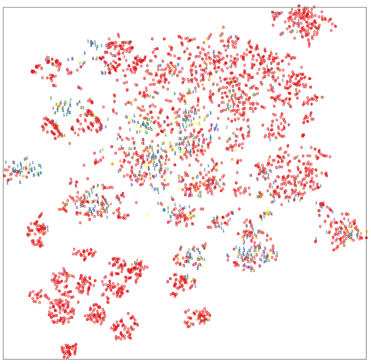
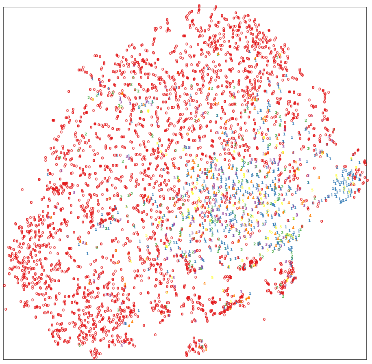
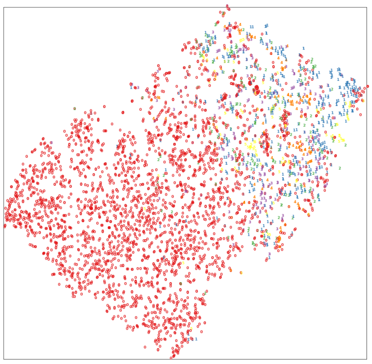
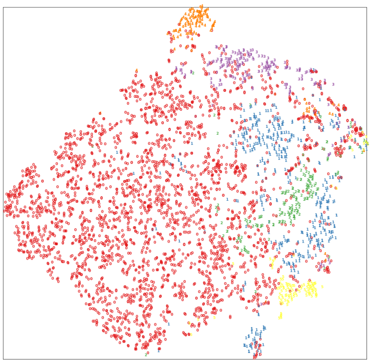
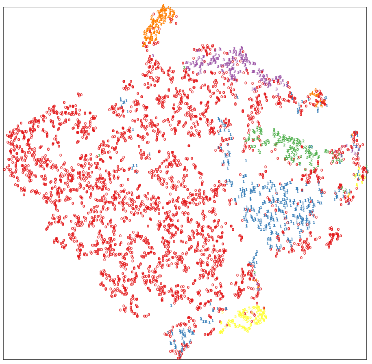
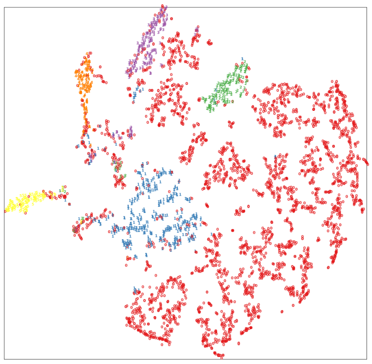
Nerwork Structure	Learning Rate	Batch Size	Imbalance Techniques	Public Score	Private Score

(43, 43, 43, 43, 6)	0.0005	4	-	0.33232	0.33008
(43, 43, 43, 43, 6)	0.0005	4	Weighted Loss	**0.35439**	**0.38497**
(43, 43, 43, 43, 6)	0.0005	4	Uniform Sampler	0.29749	0.26130

Our experiment results show weighted loss has a positive impact on our model training process. On the other hand, the uniform sampler doesn't help our model at all. We think it's because the uniform sampler changes the data distribution inside a batch, making our network assume the input data will always be uniform distribution, which is definitely not true.

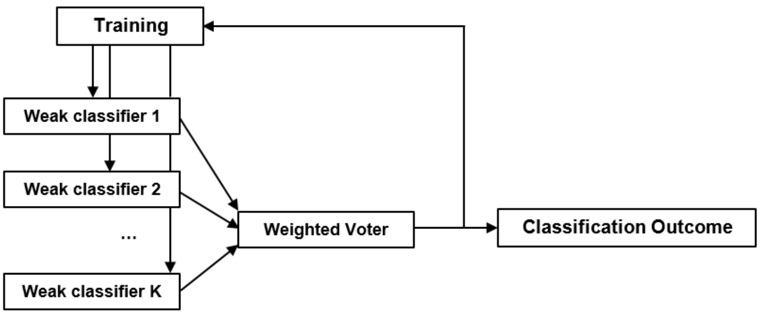
<u>Visualize Features in MLP</u>

We use TSNE[11] to reduce the dimension of features and print them on 2D plots for the purpose of visualizing features in the network. As shown below, "No churn" is colored in red, while "Competitor", "Dissatisfaction", "attitude", "Price", "Other" are colored in blue, green, purple, orange, and yellow. As we can see, raw features are inseparable at first; however, after going through MLP layer by layer, we can see that the same label data points gradually become a cluster, making them easier to be classified by a simple decision boundary. These visualization results illustrate the ability to classify that MLP learned from training dataset.

Raw Feature	1st layer output	2nd layer output
		
3th layer output	4th layer output	Final layer output
		

Experiment of AdaBoost

Structure of AdaBoost model



Design:

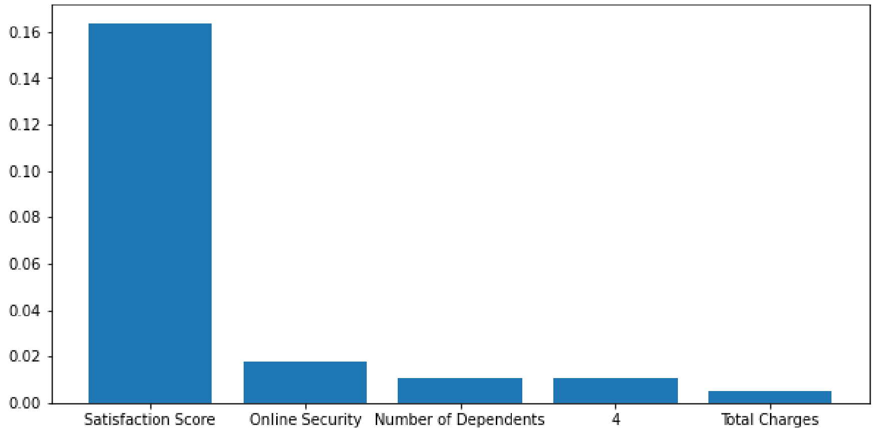
The Adaptive Boosting algorithm can combine several weak classifiers with weighted voters, into a stronger classifier, and hardly needs parameter adjustments. Furthermore, thanks to the python scikit-learn's ensemble.AdaBoostClassifier[12], the Adaptive Boosting algorithm can be conveniently developed, it might be a good choice for a beginner.

Experiments:

n_estimators	base_estimator	Private Score	Public Score	n_estimators	base_estimator	Private Score	Public Score
50	None	0.30209	0.28602	50	Logistic Regression	0.30230	0.27816
200	None	0.25806	0.29092	200	Logistic Regression	0.31661	0.30682
500	None	0.28181	0.26410	500	Logistic Regression	0.32664	0.29255

The parameters that have been experimented are the n_estimators and the base_estimator, which are the maximum number of estimators at which terminated, and the boosted ensemble was built from. If the 'weak' classifier base is stronger than default, the result would be better. Moreover, the maximum number of estimators seems not proportional to the outcome result. It is because if the estimators are many, the possibility of overfitting might occur. Although the Adaptive Boosting algorithm also states that it seldom occurs overfitting, if the estimator number is outrageously increased, the outcome result would probably show the phenomenon of overfitting.

Permutation Feature importance(top 5):



Conclusion

Comparison of three different model:

Efficiency: total training time
Scalability: (total training time with doubled data)/(original training time)

Method	Efficiency	Scalability	interpretability	Overfitting	Number of Hyperparameters	private F1 score
Logistic Regression	Good (0.849s)	Good (1.53)	Good	Hard	Few	0.35071
Multi-Layer Perceptron	Bad (46.18s)	Middle (1.73)	Bad	Easy	Many	0.38497
AdaBoost	Good (1.32s)	Good (1.14)	Good	Medium	Medium	0.32664

In terms of efficiency, logistic regression definitely has the shortest training time, while MLP has the longest since it needs to update multiple steps. Logistic regression is very easy to understand and explain, but MLP and AdaBoost are much more sophisticated. MLP is very easy to overfit, especially when training data is not rich enough, while Logistic regression and AdaBoost are less likely to happen. Logistic regression has only one hyperparameter to tune, making it the easiest model to tune up, while MLP and AdaBoost have lots more hyperparameters needed to deal with.

Final selection of models:

Though MLP achieved high score on the private leaderboard, the hyperparameters of MLP are very hard to tune, and the lack of cross validation caused by long training time makes us hard to detect overfit. On the contrary, though logistic regression didn't perform well on the public leaderboard, cross validation scores give us promising results, and the logistic regression indeed performs well on private leaderboard. Thus, we decided to choose logistic regression as our final selection of models.

Pros and Cons of Logistic Regression:

Pros:

Easy to implement
Not easy to overfit
Very few parameters needed to be tuned
Short training time

Cons:

May underfitting
Less model flexibility
Cannot easily achieve high score on test set

Students	游家權 R10942152	黃義翔 B08502063	藍峦 B07705055
Workloads	MLP	Preprocessing, logistic regression	AdaBoost

Reference:

[1]https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
[2]https://scikit-learn.org/stable/modules/generated/sklearn.impute.IterativeImputer.html
[3]https://towardsdatascience.com/all-about-categorical-variable-encoding-305f3361fd02
[4]https://en.wikipedia.org/wiki/Curse_of_dimensionality
[5]https://en.wikipedia.org/wiki/K-means_clustering
[6]https://scikit-learn.org/stable/modules/permutation_importance.html
[7] Build MLP with pytorch: https://www.itread01.com/content/1542450988.html
[8] Adam Optimizer: https://pytorch.org/docs/stable/generated/torch.optim.Adam.html
[9] Uniform Sampler: https://androidkt.com/deal-with-an-imbalanced-dataset-using-weightedrandomsampler-in-pytorch/
[10] Weighted loss function:
https://discuss.pytorch.org/t/weights-in-weighted-loss-nn-crossentropyloss/69514/3
[11] TSNE: https://mortis.tech/2019/11/program_note/664/
[12]https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html