CREDIT DEFAULT PREDICTION

This project is carried out for the Kaggle Competition :

https://www.kaggle.com/competitions/amex-default-prediction/overview

Group Members

Shubhadeep Das (Leader)

Srijani Adhikary

Sahil Mallick

Introduction

A default occurs when a borrower stops making the required payments on a debt. Defaults can occur on secured debt, such as a mortgage loan secured by a house, or unsecured debt such as credit cards or a student loan. Credit default prediction is important for managing risk in a consumer lending business. Here American Express, the largest payment card issuer in the world, has organised a competition through Kaggle where the participants have to predict whether a customer will default in future based on an industrial scale data set.

Data Description

The dataset named 'train_data' provides information of the customers for a 13 months window (March, 2017 to March,2018). But for each customer all the 13 observations are not available. There are 190 variables in total out of which 11 variables are categorical. Features are anonymized and normalized, and fall into the following general categories:

- D_* = Delinquency variables
- S_* = Spend variables
- P_* = Payment variables
- B_* = Balance variables
- R_* = Risk variables

The 'train_labels' dataset provides 'target' for each customer ID. Here target is a binary variable which takes the value 1 if the customer defaults and takes the value 0 otherwise.

The dataset named ‘test_data’ provides information of 924621 customers for 18 months window (April, 2018 to October, 2019). Our objective is to predict the ‘target’ of these customers based on the model fitted on the ‘train_data’.

Difficulty in Handling Large Data

Originally the sizes of the train and test datasets were 16.39 GB and 33.82 GB in csv format. So memory error was showing up while reading these files in jupyter notebook. So I used the feather format of the datasets found in the following link : https://www.kaggle.com/datasets/munumbutt/amexfeather?select=train_data.ftr

Here the sizes of train and test datasets are 1.73 GB and 3.55 GB respectively. Though they are still large in size, atleast we were able to read the data. As the train data has 458913 rows and 192 columns (including customer_ID and target) any operation on it was very time consuming. So we had to be very careful while working on it because we could not afford correcting the mistakes and repeating the same operations again and again.

Feature Engineering And Feature Selection

First we separated the train data according to type of feature: Delinquency, Spend, Payment, Balance and Risk. We considered only the numerical features from each category. Then for each segment we grouped the data with respect to customer ID and aggregated them by mean, standard deviation, minimum, maximum and last value. Then considering 11

categorical features as a different segment we grouped them with respect to customer ID as earlier and aggregated them by count, number of unique observations and the last value. Then we created dummy variables for each of the aggregated categorical features. Now we opted for 5 methods of feature selection :

We fitted **decision tree**, **random forest** and **catboost classifier** in each of the segments and obtained the quantitative values of importance of the features. We sorted them in decreasing order, observed the cumulative importance and considered the features upto that point where 80 percent of the total importance was crossed.

We also used the **weight of evidence** method to visualise the predictive power of the features by observing their **information value**.

Then we applied the **Recursive Feature Elimination** technique where we were able to rank all the features by setting the no. of features to be selected to 1. As the process was taking very long time for full dataset we had to apply this method on a sample of 20 percent observations chosen randomly from the data. We used Random Forest classifier as the estimator here.

We decided to select 50 raw features in total. The numbers of total raw features are 86, 21, 3, 38, 28 and 11 respectively for the segments Delinquency, Spend, Payment, Balance, Risk and Categorical. So if we want to maintain approximately similar ratio of the raw features in our selected sample the numbers should be 23, 6, 1, 10, 7 and 3 respectively. But by intuition as our motive is to predict the risk of default, it seemed to us that Risk variables may have more importance than others. So we decided to take higher number of risk variables compensating by taking less number of Delinquency features. So comparing all the 5 methods we selected 15, 6,

1, 10, 15 and 3 raw features respectively. In the course of feature selection we took the original raw features of those aggregated features which were considered to be important by majority of the methods.

Feature Creation with Selected Features

Now we created the same aggregated features as created earlier on these 50 raw variables. We also created dummy variables for categorical features as before. Apart from those we built some new features like mean of last 3 values, flag variables denoting whether the last observation and the mean of last 3 observations are larger than the quantity mean+3*sigma. We imputed the missing values of these features by 0. Then we dropped the standard deviation for each raw variable because it was noted during feature selection that the importance of standard deviation for almost all raw variables are very low. So we end up with 398 derived features in total.

2nd Stage Feature Selection

Now we fitted catboost classifier to the data with 398 features to get the feature importances. We selected top 37 features which contributed to 90 percent of the total importance. We observed that all the derived flag variables were dropped in this process as they did not have much importance. We then proceeded to model fitting part with these 37 features.

FITTING MODEL

We used the library pycaret to train the data. It automatically imputes the missing values by mean. So we did not have to do anything to deal with the missing entries of the remaining columns.

Note that, previously whenever we fitted any model for feature selection we used pycaret. So we did not perform missing value imputation separately. Only while using random forest classifier in RFE we fitted the missing values of the numerical features by median because the random forest classifier of scikit-learn does not handle missing values on its own.

It came out from the comparison of different models by pycaret that catboost, lightgbm and xgboost are the top 3 classifiers which give the highest AUC's :

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
catboost	CatBoost Classifier	0.8906	0.9519	0.7840	0.7916	0.7878	0.7141	0.7141	38.1900
lightgbm	Light Gradient Boosting Machine	0.8903	0.9514	0.7968	0.7833	0.7900	0.7158	0.7158	4.2167
xgboost	Extreme Gradient Boosting	0.8896	0.9506	0.7855	0.7875	0.7865	0.7120	0.7120	59.4667
lr	Logistic Regression	0.8748	0.9490	0.8856	0.7059	0.7856	0.6988	0.7078	30.8767
gbc	Gradient Boosting Classifier	0.8860	0.9487	0.8265	0.7559	0.7896	0.7116	0.7130	105.5000
et	Extra Trees Classifier	0.8833	0.9480	0.8440	0.7412	0.7893	0.7091	0.7119	20.6033
rf	Random Forest Classifier	0.8843	0.9469	0.8346	0.7478	0.7888	0.7095	0.7115	45.7967
lda	Linear Discriminant Analysis	0.8635	0.9435	0.8909	0.6807	0.7717	0.6769	0.6894	1.4933
ada	Ada Boost Classifier	0.8748	0.9408	0.8260	0.7275	0.7736	0.6876	0.6902	20.3067
nb	Naive Bayes	0.8643	0.9266	0.7762	0.7210	0.7476	0.6550	0.6558	0.6067
qda	Quadratic Discriminant Analysis	0.8525	0.9188	0.7072	0.7188	0.7129	0.6137	0.6138	0.9433
knn	K Neighbors Classifier	0.8506	0.9097	0.8648	0.6618	0.7498	0.6459	0.6577	897.8667
dt	Decision Tree Classifier	0.8343	0.7942	0.7109	0.6695	0.6896	0.5767	0.5772	7.3800
dummy	Dummy Classifier	0.7411	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.3233
svm	SVM - Linear Kernel	0.8667	0.0000	0.9027	0.6838	0.7781	0.6855	0.6990	1.4667
ridge	Ridge Classifier	0.8636	0.0000	0.8909	0.6808	0.7718	0.6769	0.6894	0.9300

So we fitted these 3 models on 'train_data' and scored it on 'test_data'. On submission we got the scores 0.127, 0.3 and 0.54 respectively.

TRYING FOR IMPROVEMENT

Now we wanted to check the importance of the features segment wise. On comparing all models using pycaret, catboost came out to be the best model for each segment. So we fitted catboost classifier to each segment and obtained the following result :

SEGMENT	SCORE
Delinquency	0.539
Spend	0.442
Payment	0.434
Balance	0.509
Risk	0.443

Above result shows that delinquency variables are the most important among all the segments. So we decided to repeat the 1st stage feature selection process by taking all 87 delinquency variable and sample of raw variables from other segments. We chose 25 raw variables in total from all other segments : 4, 1, 9, 9, 2 from Spend, Payment, Balance, Risk and Categorical segments respectively. After that we repeated the same steps as earlier. In 2nd stage feature selection out of 810 derived feature we chose 46 which contributed to 90 percent of the total importance by catboost classifier. This time also Catboost, lightgbm and xgboost came out to be the best 3 models :

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
catboost	CatBoost Classifier	0.8927	0.9534	0.7921	0.7931	0.7926	0.7203	0.7203	55.4567
lightgbm	Light Gradient Boosting Machine	0.8919	0.9528	0.8031	0.7846	0.7938	0.7206	0.7207	5.6033
xgboost	Extreme Gradient Boosting	0.8911	0.9519	0.7915	0.7887	0.7901	0.7166	0.7166	74.7700
lr	Logistic Regression	0.8757	0.9500	0.8895	0.7064	0.7874	0.7012	0.7105	39.0333
gbc	Gradient Boosting Classifier	0.8870	0.9500	0.8309	0.7565	0.7920	0.7146	0.7161	127.9567
et	Extra Trees Classifier	0.8867	0.9498	0.8448	0.7496	0.7943	0.7166	0.7190	23.1700
rf	Random Forest Classifier	0.8858	0.9485	0.8403	0.7491	0.7921	0.7137	0.7159	47.2267
lda	Linear Discriminant Analysis	0.8685	0.9453	0.8943	0.6899	0.7789	0.6875	0.6993	1.7300
ada	Ada Boost Classifier	0.8767	0.9422	0.8233	0.7333	0.7757	0.6911	0.6933	24.7067
nb	Naive Bayes	0.8640	0.9257	0.7646	0.7252	0.7444	0.6519	0.6523	1.3467
qda	Quadratic Discriminant Analysis	0.8507	0.9186	0.6764	0.7277	0.7011	0.6018	0.6025	1.0833
dt	Decision Tree Classifier	0.8356	0.7968	0.7164	0.6711	0.6930	0.5809	0.5815	10.2433
knn	K Neighbors Classifier	0.5670	0.6063	0.5768	0.4411	0.4999	0.4306	0.4385	743.9300
dummy	Dummy Classifier	0.7411	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.9033
svm	SVM - Linear Kernel	0.8699	0.0000	0.8998	0.6910	0.7817	0.6912	0.7035	1.4767
ridge	Ridge Classifier	0.8685	0.0000	0.8944	0.6899	0.7789	0.6876	0.6993	0.8667

So we fitted these 3 classifier to our data and obtained the score 0.547, 0.538 and 0.541. So we achieved a little improvement. So we consider this catboost model as our final model whose performance measures are shown below :

Fold	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.8931	0.9537	0.7914	0.7948	0.7931	0.7210	0.7210
1	0.8923	0.9530	0.7922	0.7919	0.7920	0.7193	0.7193
2	0.8927	0.9535	0.7929	0.7927	0.7928	0.7204	0.7204
Mean	0.8927	0.9534	0.7921	0.7931	0.7926	0.7203	0.7203
Std	0.0003	0.0003	0.0006	0.0012	0.0004	0.0007	0.0007

CONCLUSION

Our main motive behind participating the Kaggle competition was not to be a top scorer. Our biggest challenge was handling the huge dataset. Then at first we performed model fitting after manual imputation of missing values without any feature selection. We splitted the 'train_data' into train and test and observed that xgboost classifier performed well in both training and testing data. But while scoring on the actual 'test_data' we ended up with a negative score. Then we gone through various methods of feature selection. We were introduced to the pycaret library. Gradually our score improved. Although we could not get very good score, it was a great learning experience for us.