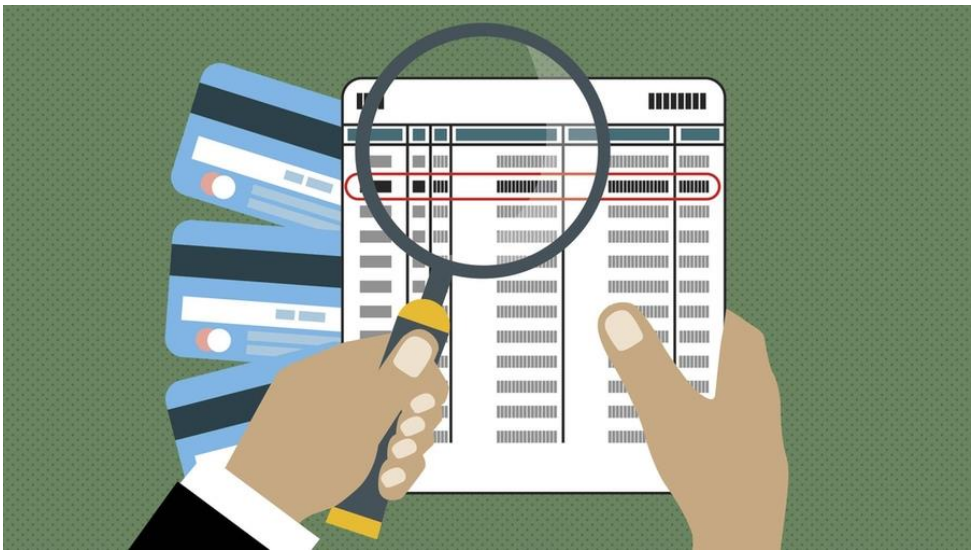


Default Credit Card Clients Prediction

CAPSTONE PROJECT REPORT

Dasun Kehelwala (DSA_0392)

Machine Learning Foundations Training
2022-11-20

Introduction (Problem Definition)

This project is focusing on predicting credit card clients who will default on their next month payment based on their demographic characteristics, past spending patterns and past repayment patterns. This is one of important business problems for banks which provide credit card facilities for Customers. This will be specifically useful to manage credit risks. This challenge is addressed as classification problem in this project. After deployment, customers should be able to access the service through API and also by sending batch input as csv file. Derived machine learning model predicts whether customer is going to default next month payment or not.

Dataset

The dataset used in this project was named as "Default of credit card clients Data Set". This dataset contains the default payment details in Taiwanese banking industry in year 2005. Dataset is available to download downloaded from UCI Machine learning repository through following link.

Link: https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients#

This dataset is Multivariate dataset with 24 attributes and 30,000 instances. All the attributes available in dataset is converted to Real Integer values. This dataset was donated to public access in 2016-01-26. Most notable observation about this dataset is its class imbalance. Table 1 Describes Attribute Details.

Table 1: Attribute Details

Attribute	Description
ID	Identifier for data entry
X1 (LIMIT_BAL)	Amount of the given credit (NT dollar): it includes both the individual consumer credit and his/her family (supplementary) credit. →Numerical
X2 (SEX)	Gender (1 = male; 2 = female). →Categorical variable mapped to integers
X3 (EDUCATION)	Education (1 = graduate school; 2 = university; 3 = high school; 4 = others). →Categorical variable mapped to integers

X4 (MARRIAGE)	Marital status (1 = married; 2 = single; 3 = others). →Categorical variable mapped to integers
X5 (AGE)	Age (year) →Numerical
X6 - X11 (PAY_0, PAY_2, PAY_3, PAY_4, PAY_5, PAY_6)	History of past payment. We tracked the past monthly payment records (from April to September 2005) as follows: X6 = the repayment status in September 2005; X7 = the repayment status in August, 2005; . . .; X11 = the repayment status in April, 2005. The measurement scale for the repayment status is: -2: No consumption; -1 = pay duly; 0: The use of revolving credit; 1 = payment delay for one month; 2 = payment delay for two months; . . .; 8 = payment delay for eight months; 9 = payment delay for nine months and above. → Categorical variables mapped to integers, but have ordinal nature as per definition
X12-X17 (BILL_AMT1 to BILL_AMT6	Amount of bill statement (NT dollar). X12 = amount of bill statement in September 2005; X13 = amount of bill statement in August 2005; . . .; X17 = amount of bill statement in April 2005. →Numerical
X18-X23 (PAY_AMT1)	Amount of previous payment (NT dollar). X18 = amount paid in September 2005; X19 = amount paid in August 2005; . . .; X23 = amount paid in April, 2005. →Numerical
Y (default payment next month_	default payment (Yes = 1, No = 0) → class variable - Categorical variables mapped to integers

Methodology (Solution Approach, Tools used)

As per problem definition, ML model need to predict this solution is approached as binary classification problem. Solution approach is described below as step-by-step process

1) Identifying and Loading Required Libraries

This was the first step in the process. Python libraries requires for Storing Data (Pandas), visualizing (Matplotlib, Seaborn), numerical processing (Numpy), Data Preprocessing and Feature Engineering (sklearn.preprocessing), Splitting dataset for training and testing components (sklearn.model_selection), Model Definition (sklearn,XGBoost), Oversampling (imblearn.) and Model Evaluation (sklearn.metrics) were imported. Model

was developed on Jupyter Notebook running on Google Collaboratory platform. Therefore, almost all these libraries were installed.

2) Loading Data and Viewing Basic Information About Dataset

Second Step is loading the data. Data file was stored at GitHub repository and directly loaded to pandas' data frame using "read_csv" function. After that, basic information about dataset was examined. All the data was available by default in numerical integer format.

3) Data Preprocessing

During this process, data cleaning and rearrangement functions were performed. Some columns were renamed for better consistency and interpretability. Checked and treated duplicate values and null values using the pandas' functions. Null values were not observed in this dataset. Also, categorical variables and numerical variables were identified and changed formats to preserve memory. Also out of range values were treated accordingly by referring data set definitions.

4) Exploratory Data Analysis

During Exploratory Data Analysis, detailed visualizations were performed to identify data set properties. Most notable observation was imbalanced nature of dataset. Positive class (Defaulted CX or value 1) represents only 22.1% of whole dataset. Figure 1 depicts the imbalanced Nature of Dataset.

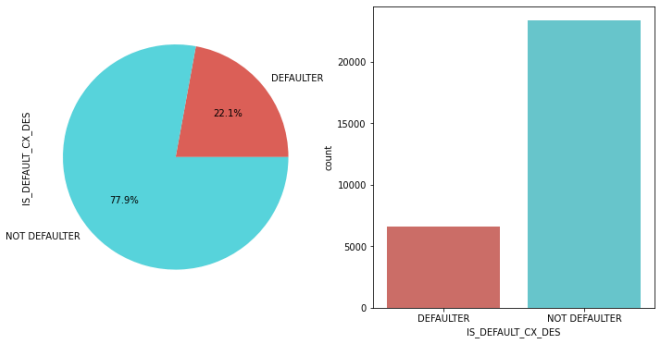


Figure 1 : Imbalanced Nature of Dataset

In addition to above observation, author has noticed significant correlation in bill statement related six variables through correlation matrix for numerical variables.

5) Feature Engineering, Feature Selection and Preparing for Machine Learning Model training

In this step, dataset was prepared to for machine learning model training. Non ordinal Categorical labels were encoded using label encoder. PAY_1 to PAY_6 variables initially set as categorical variables were converted back to integer as it has ordinal (ranking) meaning of client's payment status. Also, Dataset was split 70:30 ratio for Training and Testing. Feature vector which includes selected features used for model training is defined at this stage as pandas' series object.

6) Model Building and Evaluating

In this step, different models were developed along with different resampling methods to identify appropriate model. Initially Random Forest, XGBoost and Logistic Regression Classifiers were used with default hyperparameter settings. Results were evaluated by training models without any resampling technique. Then again tested similar models trained through resampled training set (Using random Over Sampling). Through this models, two feasible models for this application were identified. Results section explains this further. Function called "model_train" was developed by including these repetitive scores. To support feature selection, parameter named as feature vector was defined for this function. Precision, Recall, F1 Score, Weighted F1 score, and Area under ROC Curve primarily uses as evaluation purposes.

7) Hyperparameter Tuning and Selecting Best Model

During Step 6, Author has identified that XGBoost and Random Forest, Classifiers with Random Oversampling gave comparably better results. Therefore, manually tested those two models with different value combinations in feature space to identify best model. F1 score was primarily used for ranking. Model with highest F1 score was identified as best model. Results section explains this further.

8) Saving Best Model

Finally best model identified through the process was saved to a file for future use. "joblib" python library was used for this task.

Results

Initially three models were developed using Three classifiers: Logistic Regression, XGBoost and Random Forest, For these three models, default hyperparameters were used and test set with all 23 features was used for training without using resampling techniques. Since all three F scores were not satisfactory, Models with same settings was trained again with resampled dataset using Random Oversampling. Table 2 shows the results obtained through this initial run. XGBoost and Random Forest Classifiers with Random Oversampling gave better results.

Table 2: Results Through Initial Run

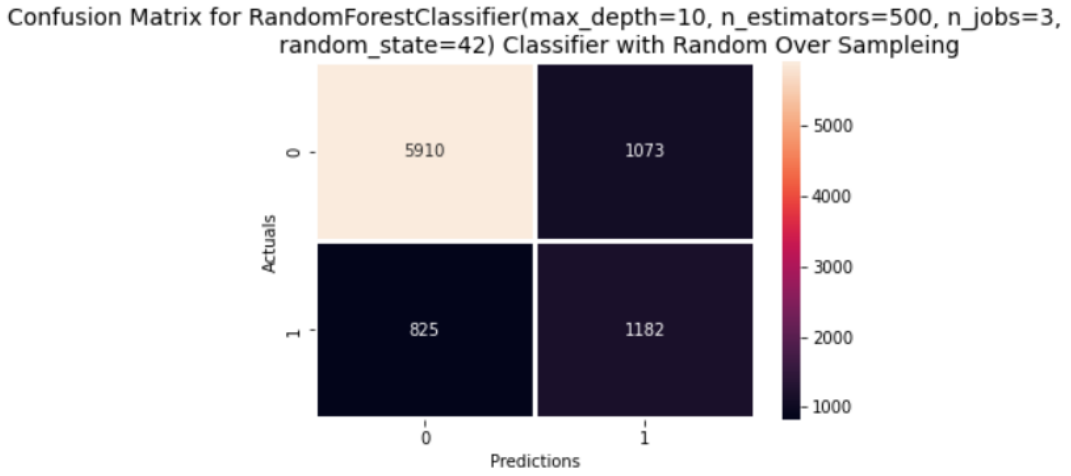
Model ID	Model	Resampeling method	Feature count	Accuracy score	Precision score	Recall score	F1 score	F1 score weighted	ROC AUC score
lgr_ns_01	LogisticRegression(random_state=42)	NO RESAMPELING	23	0.813126	0.714286	0.271550	0.393502	0.778805	0.727441
xgb_ns_01	XGBClassifier(random_state=42)	NO RESAMPELING	23	0.823248	0.697543	0.367713	0.481566	0.801508	0.789493
rf_ns_01	(DecisionTreeClassifier(max_depth=20, max_feat...	NO RESAMPELING	23	0.818020	0.672558	0.360239	0.469176	0.796197	0.782405
lgr_rovs_01	LogisticRegression(random_state=42)	Random Over Sampleing	23	0.720245	0.415333	0.620827	0.497703	0.737279	0.727212
xgb_rovs_01	XGBClassifier(random_state=42)	Random Over Sampleing	23	0.770523	0.489313	0.638764	0.554139	0.780456	0.786756
rf_rovs_01	(DecisionTreeClassifier(max_depth=20, max_feat...	Random Over Sampleing	23	0.807675	0.583534	0.483807	0.529011	0.800995	0.779527

Since XGBoost and Random Forest Classifiers with Random Oversampling have given better results in initial run, these two classifiers were used for second level evaluation. At second level, performance was tested using different hyperparameter settings. Table 3 shows the evaluation scores obtained through hyperparameter tuning.

Table 3 : Evaluation Scores Obtained Through Hyperparameter Tuning

Model ID	Model	Resampeling method	Feature count	Accuracy score	Precision score	Recall score	F1 score	F1 score weighted	ROC AUC score
xgb_rovs_02	XGBClassifier(colsample_bytree=0.5, gamma=9, m...	Random Over Sampleing	23	0.760734	0.473742	0.647235	0.547063	0.772604	0.790571
xgb_rovs_03	XGBClassifier(colsample_bytree=0.5, gamma=1, n...	Random Over Sampleing	23	0.763404	0.477794	0.643249	0.548312	0.774668	0.788177
xgb_rovs_04	XGBClassifier(colsample_bytree=0.5, gamma=0.5,...	Random Over Sampleing	23	0.763181	0.477424	0.642750	0.547887	0.774456	0.789252
xgb_rovs_05	XGBClassifier(colsample_bytree=0.5, gamma=0.5,...	Random Over Sampleing	23	0.765740	0.481257	0.633284	0.546902	0.776146	0.787701
rf_rovs_02	(DecisionTreeClassifier(max_features='auto', r...	Random Over Sampleing	23	0.809121	0.602972	0.424514	0.498246	0.796439	0.771253
rf_rovs_03	(DecisionTreeClassifier(max_features='auto', r...	Random Over Sampleing	23	0.810011	0.603892	0.432985	0.504353	0.798066	0.774043
rf_rovs_04	(DecisionTreeClassifier(max_depth=10, max_feat...	Random Over Sampleing	23	0.788877	0.524169	0.588939	0.554669	0.793111	0.785155
rf_rovs_05	(DecisionTreeClassifier(max_depth=10, max_feat...	Random Over Sampleing	23	0.788877	0.524211	0.587942	0.554251	0.793049	0.785135

According to results presented in Table 3, Model ID “rf_rovs_04” gave best F1 score (5.5547). Figure 2 describes the properties of model and depicts confusion matrix.




```
Model ID : rf_rovs_04
Model : RandomForestClassifier(max_depth=10, n_estimators=500, n_jobs=3,
                               random_state=42)
Resampeling method : Random Over Sampleing
Feature count  : 23
Accuracy score : 0.7888765294771969
Precision score : 0.5241685144124169
Recall score : 0.5889387144992526
F1 score : 0.5546691694040357
F1 score weighted : 0.793110632502284
ROC AUC score : 0.7851551504433039
```

Conclusion

This model was developed for business application which targets to predict defaulting credit card customers in next month. Even though this is important business application which delivers significant business value to customer, this is not a mission critical application like diagnostics application. As per author's understanding, this application requires to increase true positive count while maintaining false positives and false negatives at satisfactory level. Due to imbalanced nature of dataset, accuracy values could be high if algorithm classify majority of true negatives correctly. Therefore, F1 score was used to benchmark model performance and select best model for troubleshooting. Maximum F1 score achieved in this project was 0.5547. This can be considered as acceptable F1 score in this context as value is above 0.5. But F1 values above 0.8 are considered as good F1 score and above 0.9 F1 scores are considered as best. Therefore, further improvements in feature engineering and hyperparameter tuning is required.

Discussion

Author has faced few challenges during this project. First challenge was data preprocessing. There were some undefined levels for category variables. Specially, for variables which denotes History of past payment, majority of instances has undefined value 0 and -2. Author had to go through Kaggle discussion forum for this dataset and find relevant definitions for these two levels through verified sources. Second and most impacting challenge was handling imbalanced nature of this dataset. As a solution to handle imbalanced nature, Random Oversampling method was used.

In order to further improve accuracy in future, author has identified several optimizations for this model. First optimization further finetuning input feature set using feature engineering and feature selection. During exploratory data analysis, author has observed some high correlation in bill statement related six variables. That will be interesting to see model performance when single variable which sums up all six-bill settlement related variables. Also, instead of label encoding, one hot encoder can be also used for categorical variables which does not have ordinal meaning. Second optimization is trying with

different resampling technique like Synthetic Minority Oversampling Technique (SMOTE) and observe results. Due to available time limitations, author was able to try few Model types with limited feature steps. Therefore, third optimization is trying with more classifier types including Classifiers like Support Vector Machine and Deep Neural Networks (DNN). Instead of conventional DNN, Autoencoder based semi supervised learning can be also used for imbalanced dataset. Forth optimization is finetuning models by covering more combinations in feature space.

In order to improve the usability of model, this need to be accessed through web interface. Ideally model should be hosted in cloud or on-prem server and should be available to access via API. Front end web-based GUI should be provided to bank staff to enter individual customer details and get the results. Also, GUI should allow to upload csv files as batch input and get the results for multiple batches. Minor code rearrangements required to map this code to inference pipeline. Also, better to show feature importance and reasons for predictions to give better explainability to model.

References

- ✓ https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients#
- ✓ Yeh, I. C., & Lien, C. H. (2009). The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. Expert Systems with Applications, 36(2), 2473-2480.
- ✓ https://www.kaggle.com/datasets/uciml/default-of-credit-card-clients-dataset
- ✓ https://stephenallwright.com/good-f1-score/
- ✓ https://github.com/SumuduTennakoon/MachineLearningFoundations