☰ Contents

Introduction

Research Question

Credit cards are now an extremely common means of transaction that most of the adult consumers possess these days. It is therefore very important for the credit card issuing companies to be able to predict and work with the possibilities of their customers not being able to make their default payments. With this in mind, our research question that we aim to answer is: given characteristics (gender, education, age, marriage) and payment history of a customer, is he or she likely to default on the credit card payment next month?

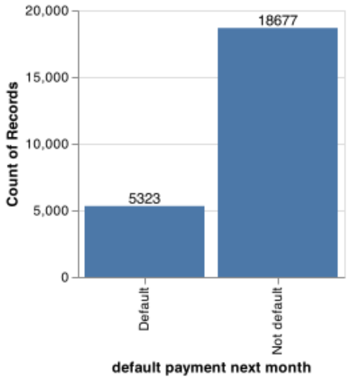
Data

The data set that we used was put together by I-Cheng Yeh at the Department of Information Management, Chung Hua University, in Taiwan. The data set itself was sourced from the UCI Machine Learning Repository and can be found here https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients. Each row in the data set represents variables associated with a customer and his or her credit card payment information, including a boolean value of default. There are 30,000 observations in the data set and 23 features. There are no observations with missing values or duplicated rows in the data set.

Initial EDA

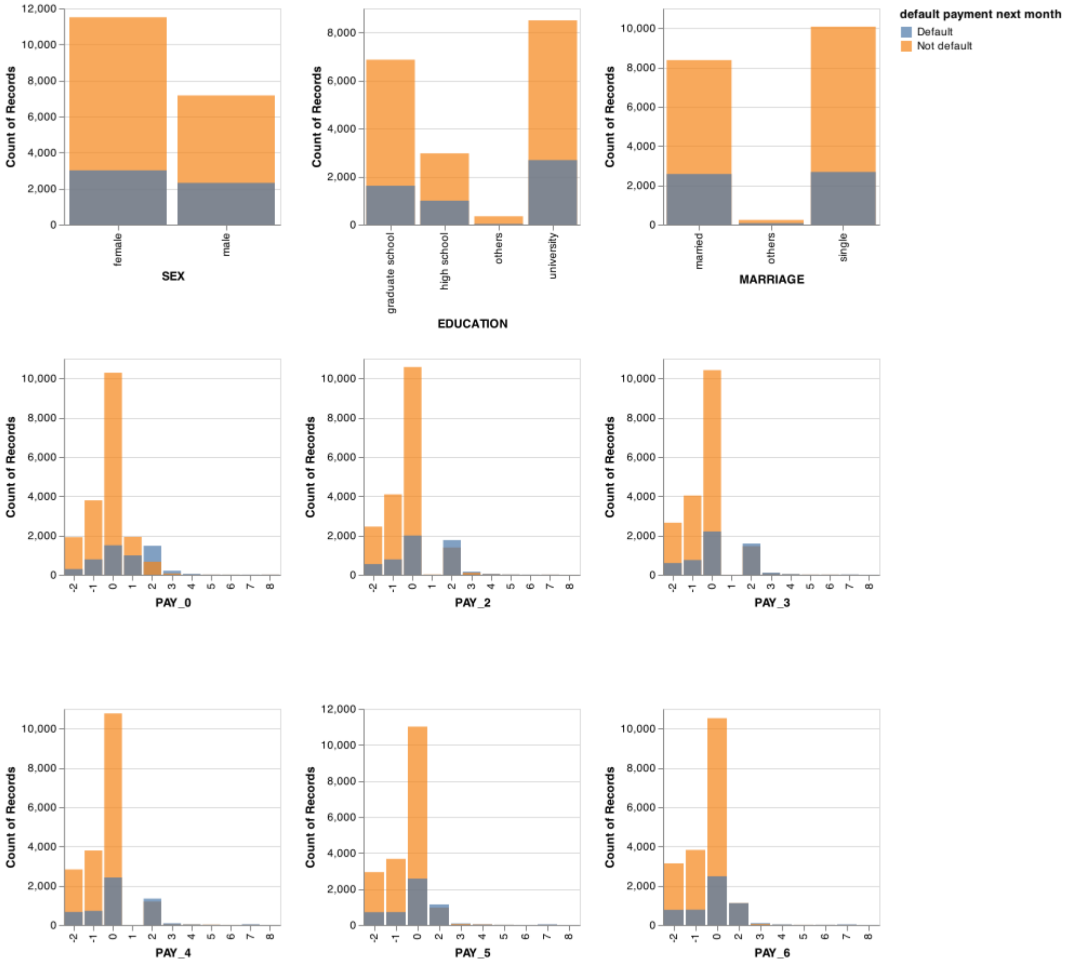
Distribution of target variables

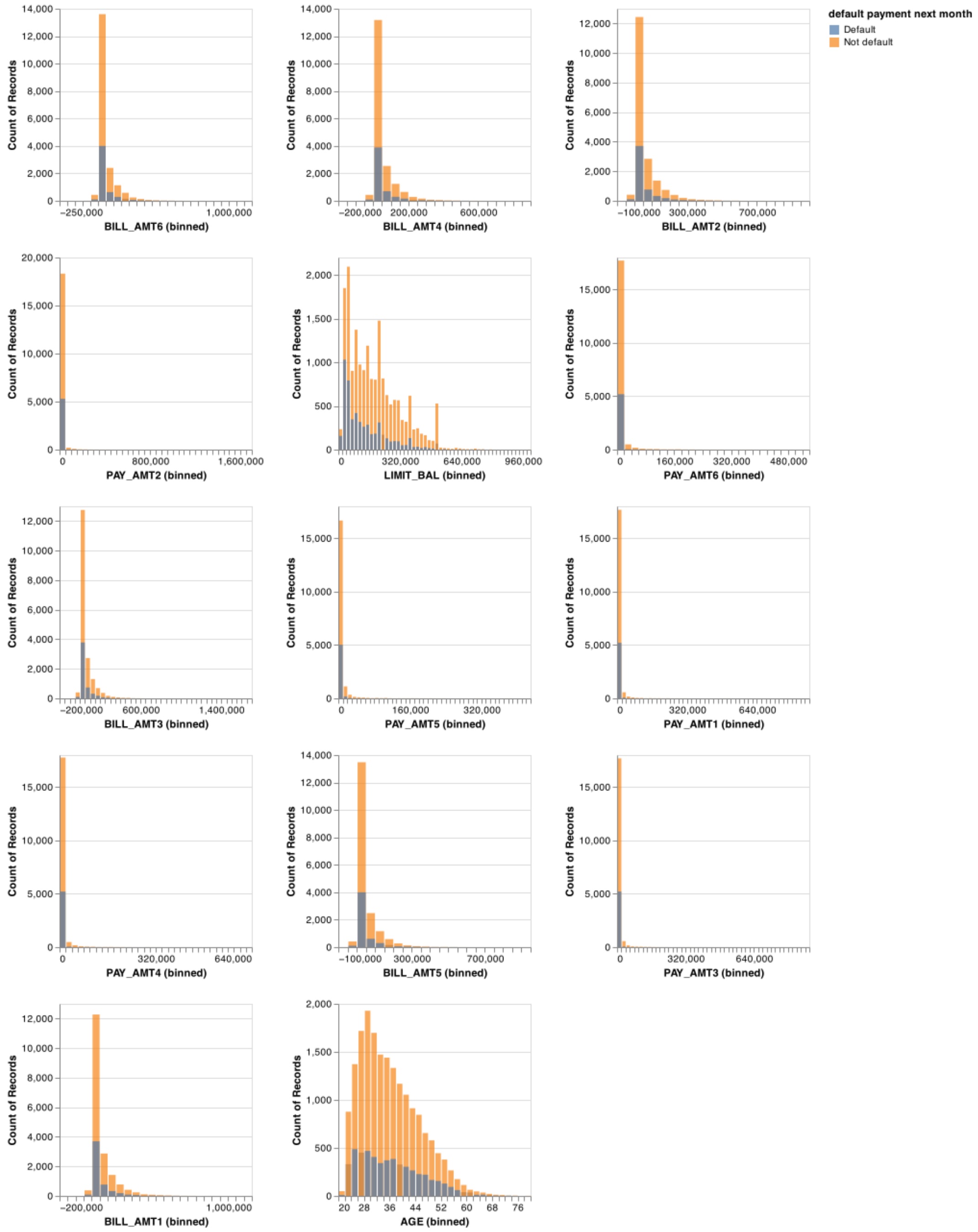
The first thing we spotted from the data was that there is a minor class imbalance. For instance our training data contained only 22.3% of class 1 in the target column. We decided to employ a few techniques to deal with this class imbalance during the analysis later.



Distribution of numeric and categorical features by target variable

To explore how each feature contributes to the prediction of the default class, we plotted the distribution of each numeric and categorical feature from the train data set and colored the distribution by class (default: blue and not default: orange). We see that the distributions below overlap for the two classes and they look quite similar to the human eye.


default payment next month
Default
Not default
Count of Records
12,000
10,000
8,000
6,000
4,000
2,000
0
female
male
SEX
Count of Records
8,000
6,000
4,000
2,000
0
graduate school
high school
others
university
EDUCATION
Count of Records
10,000
8,000
6,000
4,000
2,000
0
married
others
single
MARRIAGE
Count of Records
10,000
8,000
6,000
4,000
2,000
0
-2
-1
0
1
2
3
4
5
6
7
8
PAY_0
Count of Records
10,000
8,000
6,000
4,000
2,000
0
-2
-1
0
1
2
3
4
5
6
7
8
PAY_2
Count of Records
10,000
8,000
6,000
4,000
2,000
0
-2
-1
0
1
2
3
4
5
6
7
8
PAY_3
Count of Records
10,000
8,000
6,000
4,000
2,000
0
-2
-1
0
1
2
3
4
5
6
7
8
PAY_4
Count of Records
12,000
10,000
8,000
6,000
4,000
2,000
0
-2
-1
0
2
3
4
5
6
7
8
PAY_5
Count of Records
10,000
8,000
6,000
4,000
2,000
0
-2
-1
0
2
3
4
5
6
7
8
PAY_6



Analysis

Splitting and cleaning the model

We split our data into train and test data frames with the default setting of 0.2 split ratio. We then converted the categorical features to contain more meaningful strings as their values and the outcome file is saved in the data folder as train_visual.csv file.

Preprocessing

Since our data was relatively clean, we applied Standard Scaling on the numeric features and One Hot Encoding on the categorical features.

Choosing the best model

We trained and cross-validated the training dataset on Decision Tree, SVC, Random Forest and Logistic Regression. We also utilized class_weight parameter and set it as 'balanced' to deal with the class imbalance that was observed during the initial EDA. According to our model training, Logistic Regression gave the best validation scores using ROC_AUC as the scoring method.

	Dummy Classifier	Decision Tree	RBF SVM	Random Forest	Logistic Regression
fit_time	0.022215	0.347246	33.637723	2.617017	0.607800
score_time	0.015800	0.015001	19.555780	0.173685	0.014914
test_accuracy	0.778208	0.732208	0.775208	0.814750	0.774958
test_f1	0.000000	0.397375	0.531159	0.451018	0.531616
test_recall	0.000000	0.398270	0.574111	0.343226	0.575805
test_precision	0.000000	0.396708	0.494466	0.659219	0.493954
test_roc_auc	0.500000	0.612886	0.765369	0.766240	0.768365

Hypertuning the model

On our selected model, we tuned the parameters class_weight and C of the Logistic Regression. We obtained our best parameters and the best model which is saved as the pickle file.

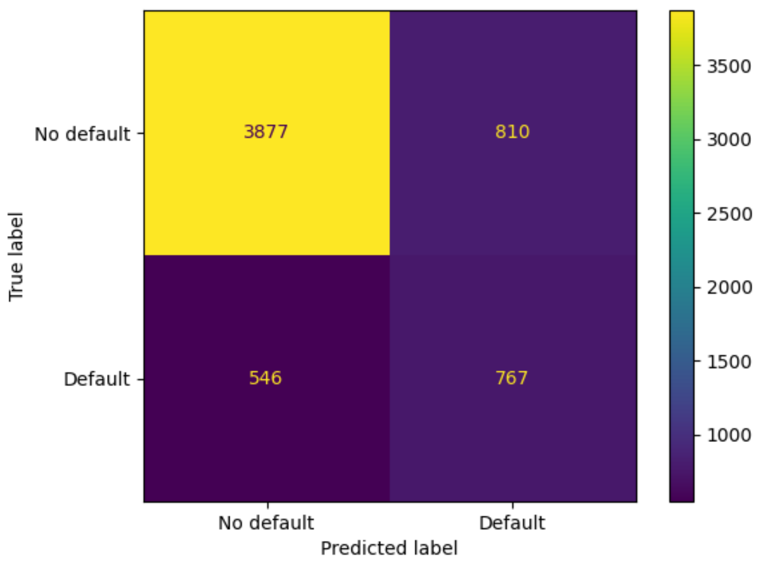
C: float, default=1.0
Inverse of regularization strength; must be a positive float.

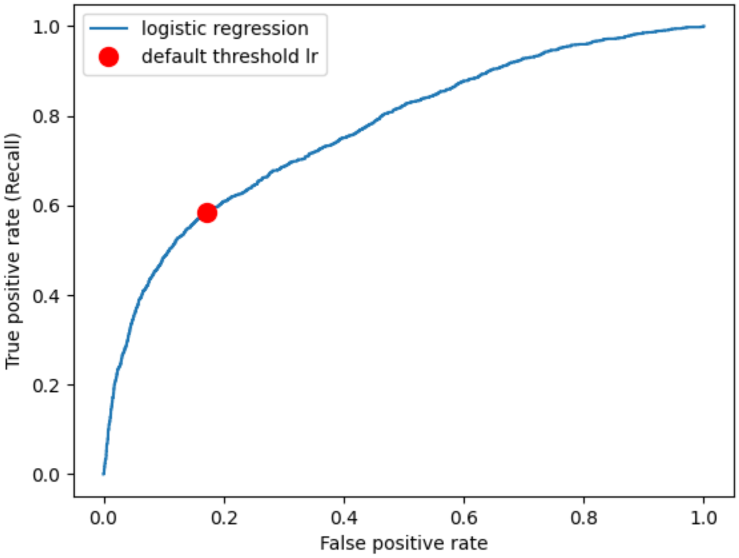
class_weight: dict or 'balanced', default=None
Weights associated with classes in the form {class_label: weight}.

Results

Model Results

We evaluated the model from pickle on the test dataset and we obtained comparable test scores to the validation score. We plotted the Confusion Matrix and the ROC-AUC curve corresponding to the model on the predicted labels.





Reservations and Suggestions

Major limitation of this project is that the data was collected in 2005. Consumers' spending behaviours and tastes must have changed since then so the results of this project should not be taken for granted and be blindly applied to the current setting. To further improve this model in the future, we suggest including more features such as income, vocation, size of the household, and debt to asset ratio. With more relevant features to base the predictions on, we should be able to predict our target class with more accuracy.

References

Dua, D., & Graff, C. (2017). UCI Machine Learning Repository. Opgehaal van http://archive.ics.uci.edu/ml

Python Core Team. (2019). Python: A dynamic, open source programming language. Opgehaal van Python Software Foundation website: https://www.python.org/

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. Computing in science & engineering, 9(3), 90.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825–2830.

McKinney, W. (2010). Data Structures for Statistical Computing in Python. In S. van der Walt & J. Millman (Reds), Proceedings of the 9th Python in Science Conference (bll 51–56).

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., … Oliphant, T. E. (2020). Array programming with NumPy. Nature, 585(7825), 357–362. doi:10.1038/s41586-020-2649-2

By MDS DSCI 522 Group 18