IEORE 4523 Data Analytics

Santander Financial Product Prediction

System Breakers:

Yumeng Du; Yi Gong; Yixin Hu; Jielin Mao; Zhiyuan Ren

Instructor: Uday Menon

Abstract

Our team has decided upon the Santander Product Recommendation dataset from Kaggle to be our primary dataset for our final project. This dataset contains anonymized customer geodemographic data, anonymized financial account data, and around 1.5 years worth of customer purchase behavior data, where each row represents a consumer's purchase mix by a particular time stamp. Our goal is to build a robust model that can help Santander predict whether or not an individual will likely open-up a new credit card account with the bank. The key analyses we will conduct include data cleaning and visualization, feature engineering, exploratory data analysis, and machine learning model building.

Introduction

Having suitable financial products is critical for consumer's financial well-being, as well as profit generation for banks around the world. In this project, we use a real-world dataset provided by Santander (Spanish bank). We aim to derive critical information from customer demographics, account types, and engineering features to predict if the customer will open a credit card account in the next month.

The original dataset is retrieved from Kaggle: *train_ver2.csv* (13647309 rows). This dataset contains columns including 22 customer attributes (e.g., age, customer type, joining channel, etc.) and 24 account types (e.g., junior account, credit card, loans, etc.). The data is in chronological order, with monthly records of customer information and their owned financial products from 2015-02-28 to 2016-05-28. Each customer has a unique key *cust_code*. We further split the dataset into a training set (2015-02-28 to 2016-03-28), a testing set (2016-04-28) and a validation set (2016-05-28).

Data Preprocessing

The original dataset consists of two parts of information: customer features and account information. After reviewing the dataset, we first deleted two customer features with too many missing values: *spouse_index* and *address_type*, and then replaced other NaN values in customer attributes with non-NaN values from consecutive months (first backward filling, and then forward filling). Next, we winsorized extreme values in *age* with cutoff points 15 and 100. We also winsorized large *income* values after taking a log. Finally, we deleted variables that are highly unbalanced (major class takes up 99%): *emp_index*, *cust_type*, *residence_index*, *foreigner_index*, and *deceased*.

Our goal is to predict whether a customer will open a credit card account in the next month, so we added a new variable: *new_credit_card*. If a customer opens a credit card account, we would observe this variable

changing from 0 in this month to 1 in the next month. We also created following variables as predictors based on the historical behaviors of a customer:

Variable Name	Explanation
credit_card_cumlen	Cumulative length of months having a credit card
cc_close_lastmon	Whether closing the credit card last month
credit_card_close	Number of times closing a credit card account
l_cc	Whether having a credit account last month
ll_cc	Whether having a credit account two months ago

Feature Engineering

There are 3 main challenges when dealing with this dataset, which will be discussed in the subsections:

1. Unbalanced response variables (25:1 unbalanced ratio)
2. Features are overwhelmingly NaN or has near-null variance (99% of feature is one value)
3. Less than 10% of all variables are continuous, 90% of variables are binary

Response Variable Imbalances: Downsampling to Ensure Model Learning Environment

Our response variable is *new_credit_card*, that is to say, whether or not the customer opens a new credit account this month. However, after observing the data, it is easy to see that our data is extremely unbalanced (Figure 2.1). As mentioned in our "mines vs. rocks" example, we wish to prevent our model from predicting negatives constantly. Therefore, we decided to *downsample* our training data so that the labels are balanced (Figure 2.2). In doing so, we are able to ensure that our model is not predicting negatively constantly, rather actively teasing out the information provided by our feature set.

NaN-dominant Features: KNN Imputation to Retain Variance

A critical step in our feature engineering process was selecting the correct method of dealing with missing values in our data set. We have experimented with SimpleImputer and KNNImputer for our preprocessing pipeline. The ideal feature set would contain columns with a "good amount of variance" meaning that variances are not too high (worst case being each column has a unique identifier) and not too low (worst case being each column has one value). Therefore, we decide to use a KNN imputation to increase our variance (since a simple imputation will dramatically decrease the variance in our binary features with high NaNs). This resulted in better performance in out of sample prediction.

Binary-dominant Features: PCA to Introduce Continuity for Prediction

Finally addressing the final issue of binary features, we have decided to use industry-standard practice of *reintroducing continuity* through the PCA process. The issue with binary variables is that it fails to take account for subtleties in information (since the parameters on [0,1] variables essentially act as "level shifts" in a model). Therefore, we use PCA *only on the binary variables* to introduce continuity to this feature set, while keeping 95% of the total variance from the variables. Summarizing all of the items above, we construct a sklearn preprocessing pipeline (Figure 2.3).

Modeling and Results

Data Intuition

Probabilistic multiple logistic regression model demonstrates faster training and prediction speeds in comparison to more complex methods such as bagging/boosting when processing equivalent amounts of data. L1 regularization can improve the generalization performance of the model, since one hot encoding of the various categorical features will lead to many columns that do not hold a lot of information (sparsity). We expect the model less likely to overfit the training data and is more likely to perform well on unseen data, which is particularly important in this business scenario. Thus, we gathered information regarding precision/recall trade-off, model interpretability, and algorithmic efficiency and decided to utilize a penalized logistic regression as our baseline model.

L1-Regularized Multiple Logistic Regression (LASSO-Logistic)

In logistic regression, the predicted probability that an instance belongs to a particular class is transformed into a binary outcome using a threshold. The robustness of a classification model can be effectively evaluated through the threshold value as well as model reactivity to threshold changes. We use an optimal threshold of $p = 0.5$ (standardized data), whereas note that the model effects would be dramatically impaired if we increase the threshold by only 0.1 - 0.2, indicating potential drawbacks of the baseline model. The final model has an extremely low false positive rate of around 0.001% and AUC = 0.99 (Figure 3.1). The major problems we want to address at this step are the low F-score and trying to increase the threshold to make the model more robust when it comes to distinctive out-of-sample data.

Random Forest Classifier: Bagging

Random forest classifiers are relatively resistant to overfitting, which means that they are less likely to make poor predictions on new, unseen data compared to some other types of models. This makes them a good choice for our business case, as they can generalize well from the training data to new customers. And indeed as expected, even elevating our threshold to up to 0.92, the model can still generate very low

false positive rates and retrieve precious information about customers. Finally we set up the optimal threshold at 0.8 and achieved slightly less accurate predictions compared to our baseline model (Figure 3.2). Yet, our attempt to reinforce model robustness proved to be successful and we moved on to a better suited gradient boosting model in the last part.

XGBoost Classifier: Boosting

The final model achieves an optimal balance between model effectiveness and robustness by utilizing a relatively high threshold (0.8) and achieving slightly better out of sample performance as the baseline model did. The AUC = 0.99, along with the lowest false negative rate 1.26%, and the 0.001% false positive rate together formed our final model choice. (Figure 3.3)

Given that we want to find potential credit_card_account newcomers, we want the true positive rate to be as high as possible so that we can capture all potential customers within the dataset. This positive revenue, compared to the costs associated with reaching out to a false negative person (someone who would open a new credit card but we believed that they wouldn't) is advantageous. Since the people who have been categorized as false negatives would have opened the credit card accounts in Santander without needing additional incentives from our client's marketing strategies.

Conclusions and Reflection

By methodically going through our data preprocessing, choosing the suitable models and their parameters, and finally using the metrics that we have learned in this class, we are able to construct an end-to-end data processing pipeline which results in credit card predictions that align with business value (focusing on True Positives). There was a lot of experimenting that was not included in this final version of the notebook (can be viewed in our various versions on github). Notably using various architectures for the feature engineering process and experimenting with different models and parameters.

Future Improvements

Throughout our analysis, several steps could have been further optimized. First, we can consider building an end-to-end ML pipeline to automate data preprocessing. Second, the current Lasso estimator could be improved by alternating coefficients, and we can try other regularization approaches like Elastic Net. Finally, in the modeling part, we cannot completely overcome the innate disadvantages brought by an extremely imbalanced dataset, thus we could only acquire a relatively low F-score. Future improvements may focus on using an alternative training dataset to improve positive data ratio.

Appendix

Table 1. Feature Summary

Feature Type	Feature Name	Definition
Customer Information	*fetch_date*	Date of retrieving
	cust_code	Customer code (primary key)
	emp_index	Employee index
	country	Customer's country residence
	sex	Customer's sex
	age	Customer's age
	cust_date	Date of becoming holder of first bank account
	new_cust	Whether registering within last 6 months
	cust_seniority	Customer seniority (in months)
	indrel	Whether changing customer type
	last_date_as_primary	Last date as primary customer
	cust_type	Customer type at beginning of month
	cust_rel	Customer relation type at beginning of month
	residence_index	Whether residence country is the same as bank country
	foreigner_index	Whether birth country is different from bank country
	spouse_index	Whether being spouse of employee
	joining_channel	Channel of joining bank
	deceased	Whether having deceased
	address_type	Customer's address type
	prov_code	Customer's province code
	prov_name	Customer's province name
	activity_index	Whether being an active customer
	income	Gross income of customer's household
	segmentation	Customer's segmentation

Account Type	*savings_account*	Saving account
	guarantees	Guarantee account
	current_account	Current account
	derivative_account	Derivative account
	payroll_account	Payroll account (type I)
	junior_account	Junior account
	mas_account	Más particular account
	perticular_account	Particular account
	perticular_plus	Particular plus account
	st_deposit	Short-term deposits
	mt_deposits	Medium-term deposits
	lt_deposits	Long-term deposits
	e_account	E-account
	funds	Funds
	mortgage	Mortgage
	pension	Pensions (Type I)
	loan	Loans
	tax	Taxes
	credit_card	Credit card account
	securities	Securities account
	home_account	Home account
	payroll	Payroll account (Type II)
	pension2	Pensions (Type II)
	direct_debit	Direct debit account

Figure 1.1. Number of Customer Records in Each Month

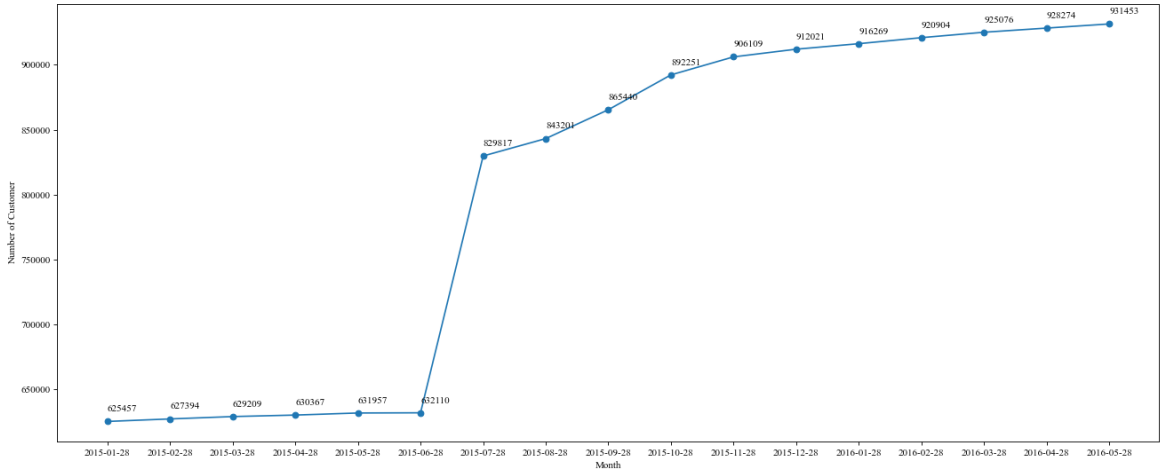
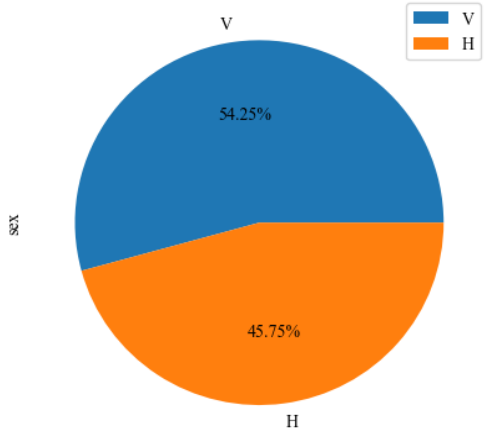


Figure 1.2 Gender Distribution

a) Proportion of Gender



b) The Distribution of Gender of Customers Opening an Account

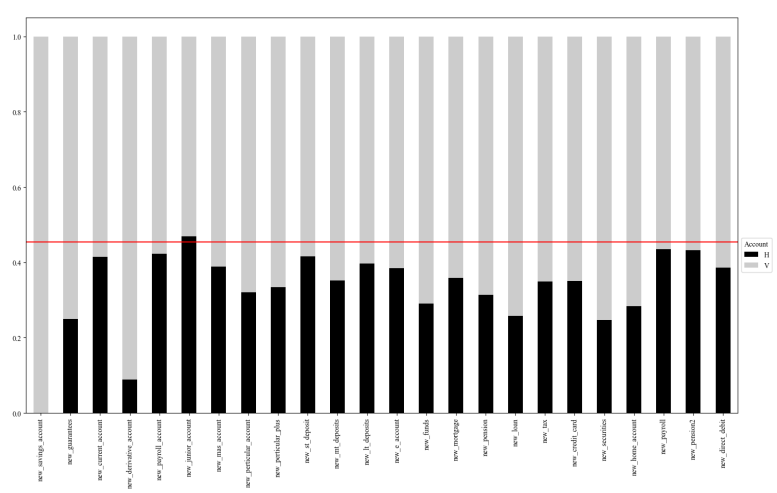
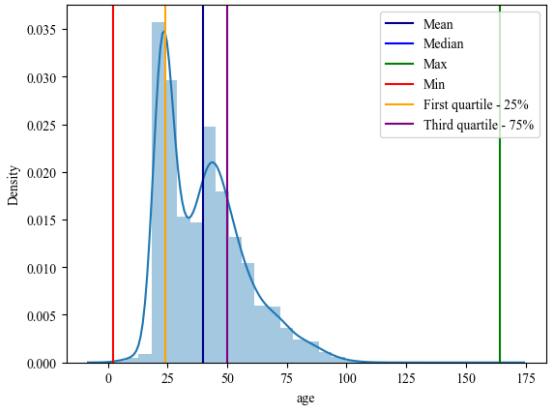


Figure 1.3 Age Statistics

a) Distribution of Age



b) The Distribution of Customer's Age with Account Opening

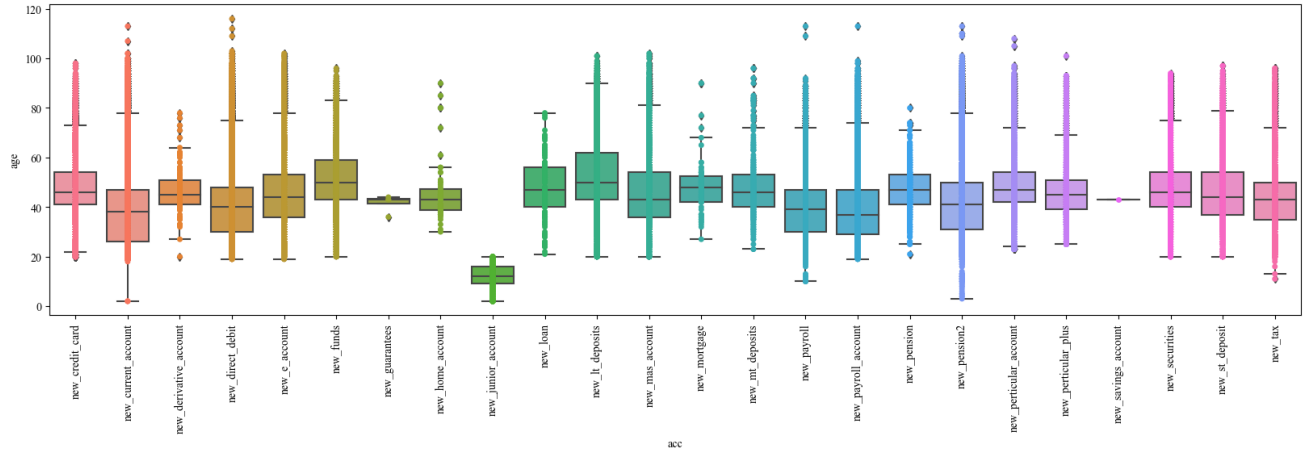
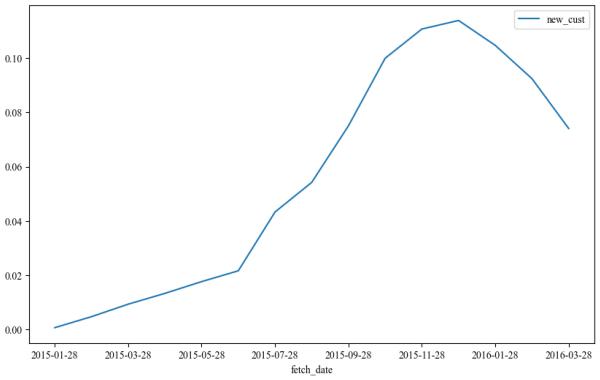


Figure 1.4 New Customer Summary

a) Ratio of New Customers in each Month



b) The Ratio of New Customers Opening an Account

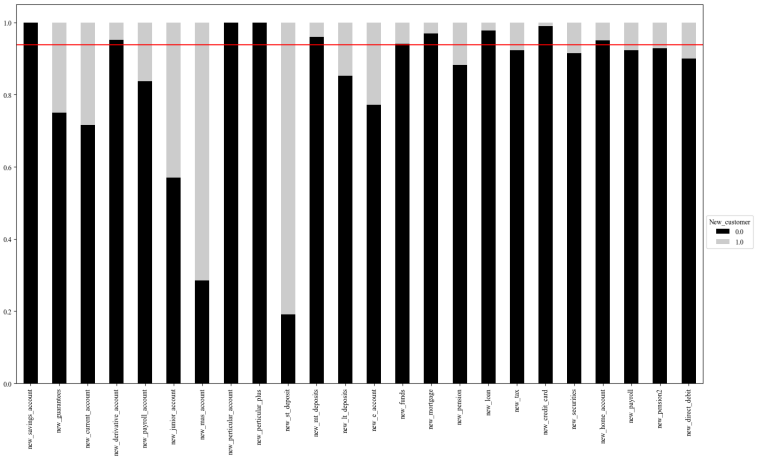
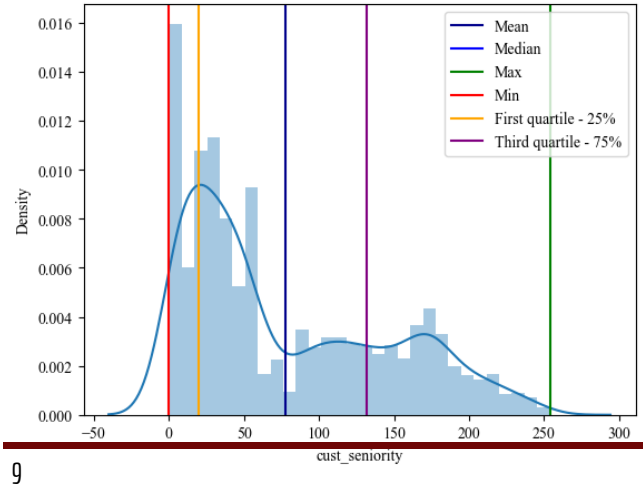


Figure 1.5 Customer Seniority Distribution

a) Distribution of Seniority



b) The Distribution of Customer's Sensority with Account Opening

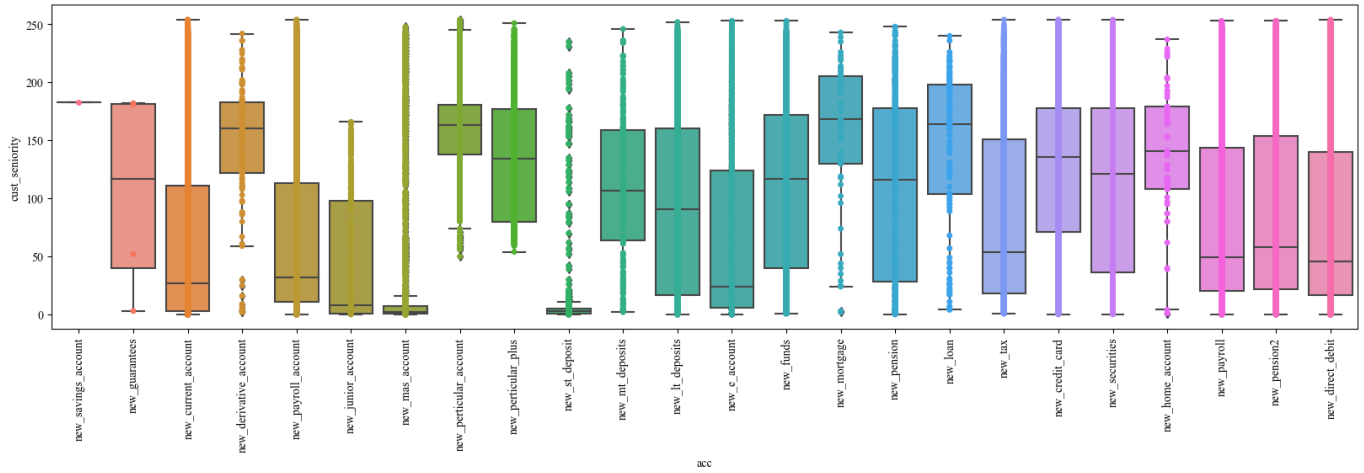
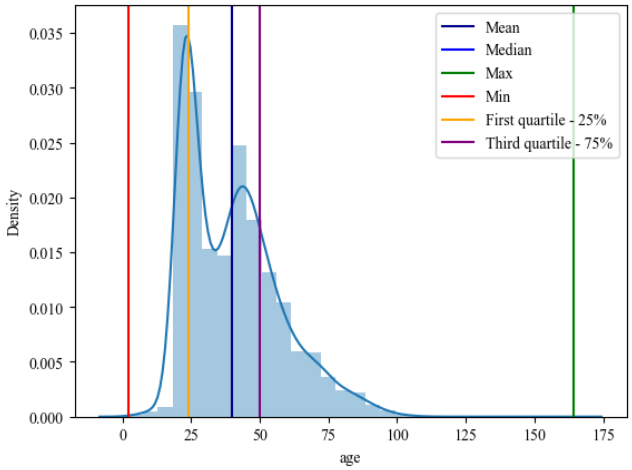


Figure 1.6 Income Statistics

a) Distribution of Income



b) The Distribution of Customer's Income with Account Opening

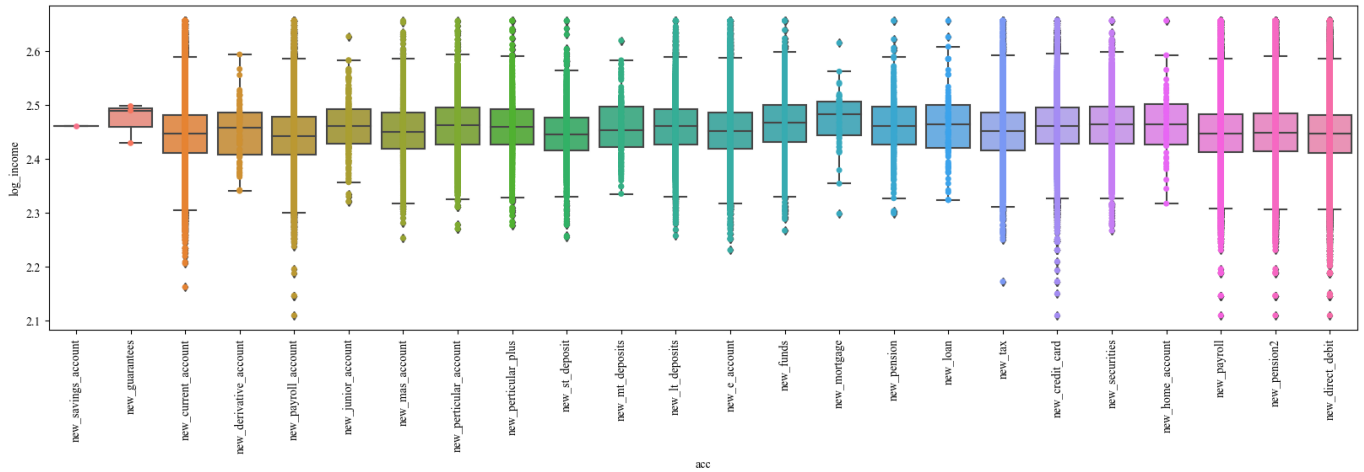


Figure 2.1 & 2.2 Distribution of New Credit Card before & after Downsampling

Fig.1 Raw labels distribution (*n*=925076)

Fig.2 Downsampled training labels (n=8804)

Figure 2.3 Graphical Representation of Feature Engineering Pipeline

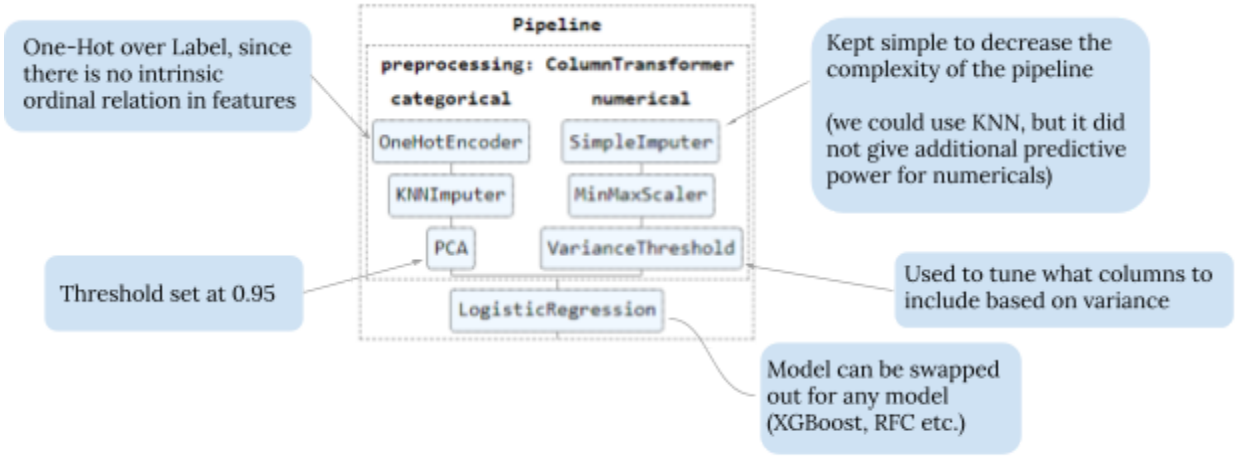


Figure 3.1 Out-of-Sample Performance: L1 Regularized Logistic Regression (Probabilistic Approach, threshold = 0.5 determined in the Train-Test phase)

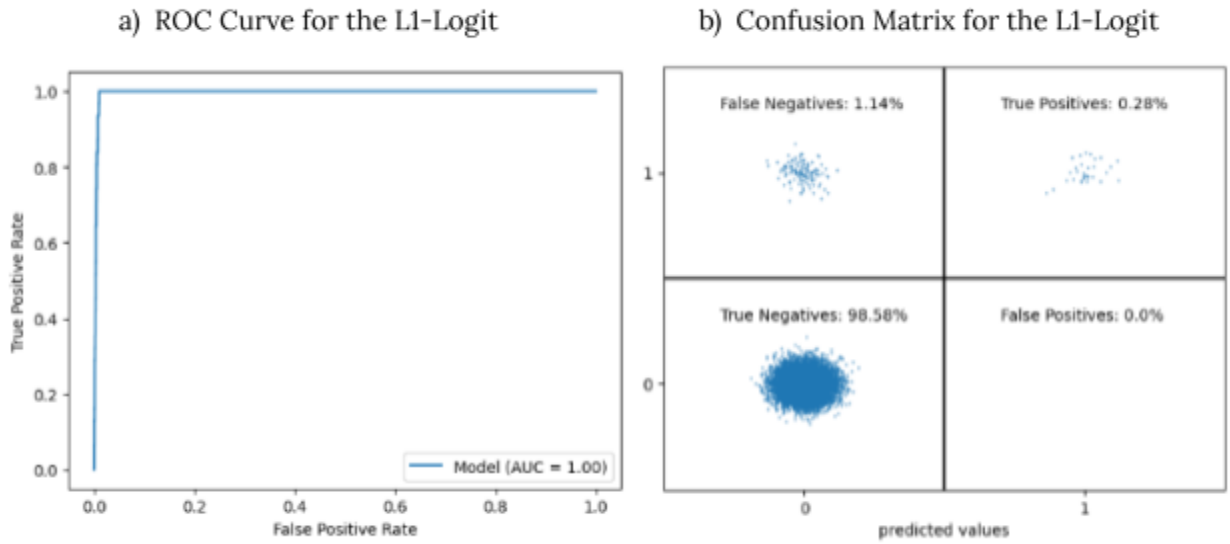


Figure 3.2 Out-of-Sample Performance: Random Forest Classifier (Probabilistic Approach, threshold = 0.8 determined in the Train-Test phase)

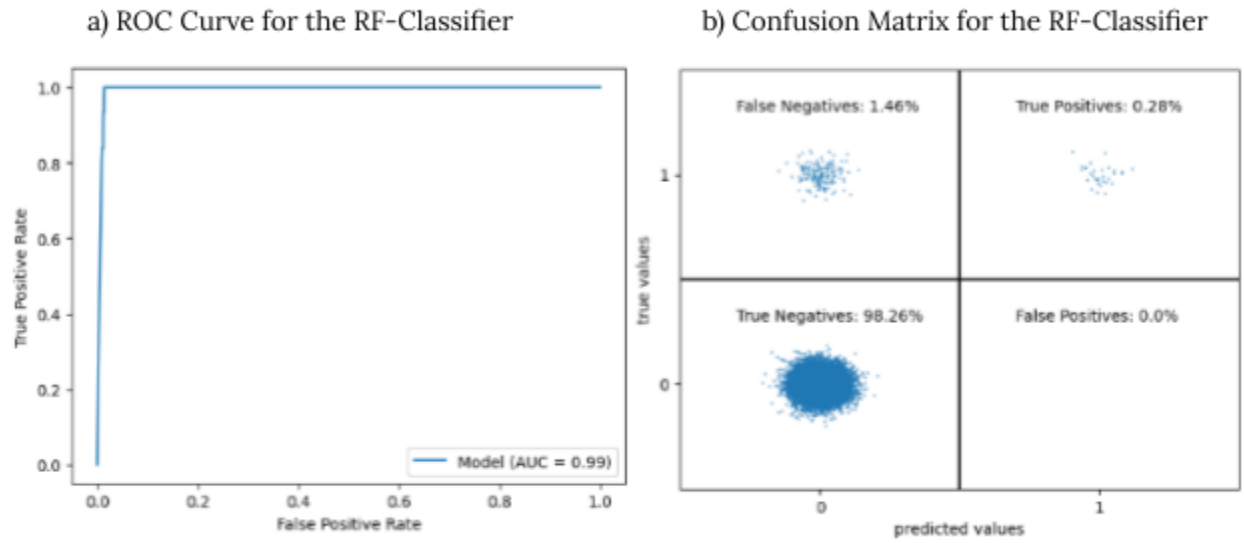


Figure 3.3 Out-of-Sample Performance: XGBoost Classifier (Probabilistic Approach, threshold = 0.8 determined in the Train-Test phase)