Convolve Epoch-2

Team - **epoch0**

January 28, 2024

1 Data Analysis

- Addressing missing values, we identified four columns with notably high NaN percentages, namely ['others_42', 'others_43', 'others_44', 'others_45'], each exceeding 90%. To handle these missing values, we utilized the SimpleImputer method from the sklearn.impute module. Categorical features underwent imputation with the most frequent values, while numerical features were filled using the mean, adhering to standard procedures.
- Regarding encoding strategies, empirical observations led us to favor label encoding over one-hot encoding for eight columns with object data types. Notably, the label-encoded dataset yielded higher accuracy values during experimentation.
- Moving on to a more detailed analysis, correlation graphs were employed, taking into account the inherent imbalance in the dataset. Correlation matrices are traditionally calculated on balanced datasets, where the impact of each column is considered equal. To address this, we opted to use correlation plots on both the imbalanced and subsampled datasets.
- In creating a balanced dataset, we performed subsampling, selecting 2,000 data points for both target values (0 and 1). This ensured an equal representation of both classes, enhancing the clarity of correlation matrices for subsequent feature analysis.
- During the computation of Pearson correlation coefficients, certain columns possessed NaN values due to a singular value type within the column, resulting in a standard deviation of zero. Recognizing the insignificance of these features in modeling, we opted to remove them. The list of discarded features includes ['demog_7', 'demog_12', 'txn_35', 'txn_36', 'txn_49', 'txn_50', 'txn_65', 'txn_70', 'txn_71', 'txn_72', 'demog_38', 'demog_10'].

2 Submission Model

- We divided the dataset into training and testing sets with a 4:1 ratio, using 80,000 samples for training and the rest for testing. Opting for XGBoost as our base model after experimenting with SVM, AdaBoost, LightGBM, and other models, we achieved highest accuracy with XGBoost. Hyperparameter tuning further improved the accuracy, reaching 99.9955%(before dropping the Primary Key) in our best attempt.

3 Model 1

- After removing the Primary key and testing different models, we achieved a peak accuracy of 99.78

The hyperparameters of the model are as follows:

1 'objective': 'binary: logistic',

2 'eval_metric': 'logloss',

3 'max_depth': 5,

4 'learning_rate': 0.1,

5 'subsample': 0.8,

6 'colsample_bytree': 0.8,

7 'seed': 42,

	Precision	Recall	F1-Score	Support
0	1.00	1.00	1.00	19600
1	0.92	0.94	0.93	400
Accuracy			1.00	
Macro Avg	0.96	0.97	0.96	20000
Weighted Avg	1.00	1.00	1.00	20000

F1 Score: 1.00

$$\begin{bmatrix} 19567 & 33 \\ 25 & 375 \end{bmatrix}$$

After getting the preliminary accuracy we moved on to eliminating more columns based on Cramer score. We have calculated the cramers score to eliminate some variables from the categorical features.

1 country_code Cramér's V: 0.0

2 demog_2 Cramér's V: 0.061008823183767415

3 income Cramér's V: 0.02829268508574592

4 city_tier Cramér's V: 0.04026841172246118

5 occupation Cramér's V: 0.04011163988296899

6 demog_4 Cramér's V: 0.010669417145235942

7 demog_22 Cramér's V: 0.007179092449478876

8 os Cramér's V: 0.021186563891304612

9 email_domain Cramér's V: 0.01849801990083717

10 demog_40 Cramér's V: 0.2014080319565834

11 demog_43 Cramér's V: 0.1349684791008429

Based on the above scores we can eliminate categorical columns like ['country_code', 'demog_22'] have fewer crammers scores.

After removing the less correlated score the accuracy improved to **99.75**.

- We proceeded to eliminate highly correlated features from the dataset by setting a threshold of 0.95. For pairs of columns with a correlation coefficient exceeding 0.95, one column was removed. This step aimed to mitigate redundancy in the dataset, preventing the introduction of noise into the model predictions. The resulting probability remained at 99.76
- The columns that were dropped include: 'demog_11', 'demog_26', 'demog_29', 'demog_33', 'demog_34', 'demog_35', 'demog_41', 'others_1', 'others_26', 'others_29', 'others_31', 'txn_4', 'txn_46', 'txn_75', 'txn_76', and 'txn_80'.

4 Model 2

- In addition to manual feature selection, we explored the application of ANOVA for feature selection in our dataset. We conducted training on the initial 100 features using this statistical method, resulting in an accuracy of **99.78%**.
- ANOVA, or analysis of variance, is a statistical technique employed to assess the significance of individual features in predicting the target outcome. It operates on the premise that values within a given group should exhibit similarity to each other and divergence from values in other groups.
- This method aids in identifying features that contribute significantly to the target prediction, allowing for a more refined and optimized feature set in our model. The integration of ANOVA into our feature selection process contributes to the overall enhancement of model accuracy, complementing the manual selection approach previously employed.

5 Experiments during the course:

We have tried various models present in the hope of better accuracy. I am presenting the accuracies of all those in the table format.

Model	Accuracy
Decision Tree	99.32
RandomForest	99.60
LightGBM	99.62
SVM	99.54
Extra Trees	99.62
KNN	99.46
Logistic Regression	99.54
AdaBoost Classifier	99.54
SVM with Linear Kernel	99.54
Multilayer perception	99.54
Bagging Classifier	99.50
LDA	98.74
Passive-Aggressive	99.14
CatBoost classifier	99.68
2-Layered DL model	96.88

Table 1: Example Table

- In the course of our data analysis, a notable observation revealed that 76 columns in the dataset contained 25,794 NaN values each. Further investigation unveiled that these NaN values were consistently associated with the same set of samples. Among these columns, a majority of the samples had a value of 0 in many features, rendering them largely non-informative for the model learning process. Consequently, we identified and subsequently dropped 42 columns with such characteristics.
- Given the context of the dataset, which pertains to bank accounts, we developed a theory regarding the nature of these NaN values:
  - NaN values may not indicate missing data but rather the absence of a particular feature for a given account.
  - For instance, certain accounts might not be subject to TDS payments, leading to a null entry under the 'text' column.
- To address these NaN values, we adopted the following strategies:
  - Replacing NaN values in numerical features with -1, a value not encountered in actual data, facilitates model learning.
  - Replacing NaN values in 'object' data types with the entry 'Unknown.'
- To make use of fact that accuracy should ideally increased if an independent non-uselful column is dropped:
  - Systematically removed each column one at a time and monitored whether the accuracy increased in comparison to when all columns were included.
  - Columns that contributed to an increase in accuracy were systematically removed one by one, with the goal of identifying the combination that yielded the highest overall accuracy improvement.
  - After identifying the optimal combination, the loop is iterated until no further improvement in accuracy is observed.

After implementing these NaN value handling techniques, we proceeded to one-hot encode and scale the final DataFrame using the StandardScaler() method. The resulting model achieved a peak accuracy of 99.69%, with the training process focusing on approximately 100 features.

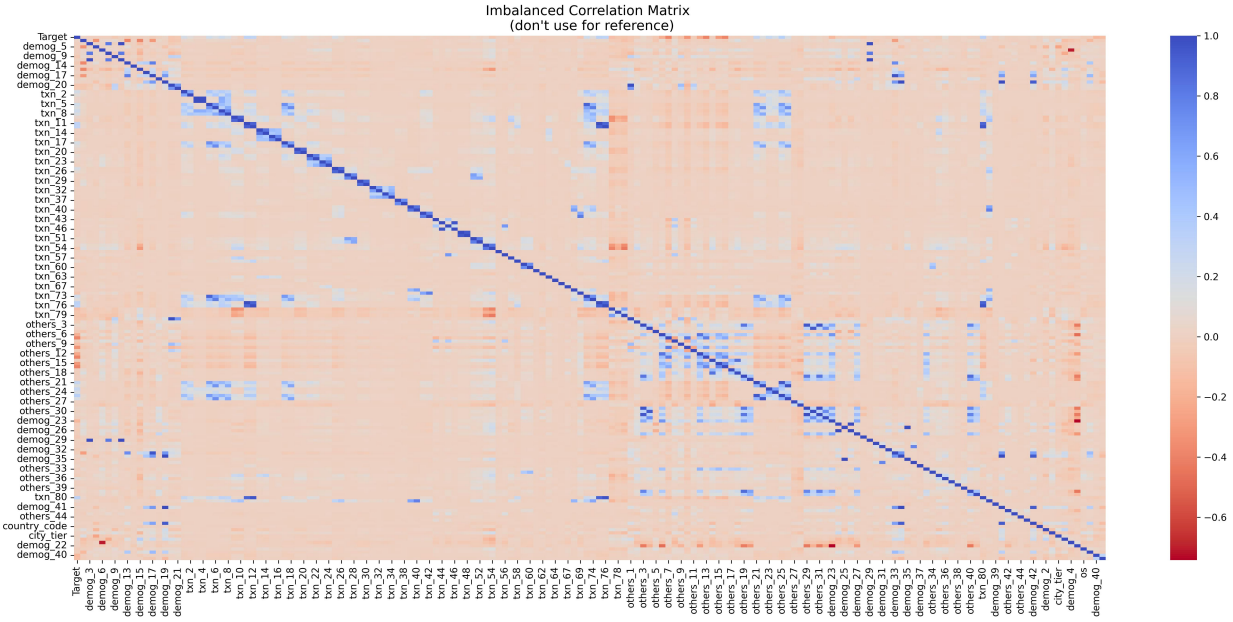

Figure 1: Correlation matrix for imbalanced dataset

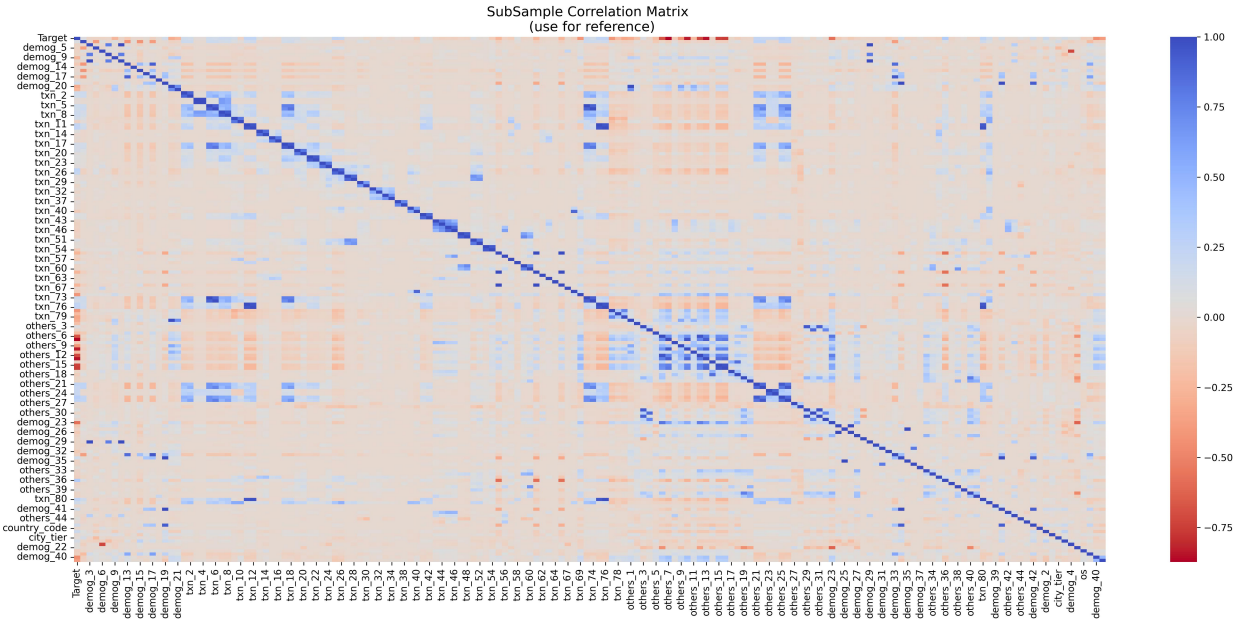

Figure 2: Correlation matrix for SubSampled dataset

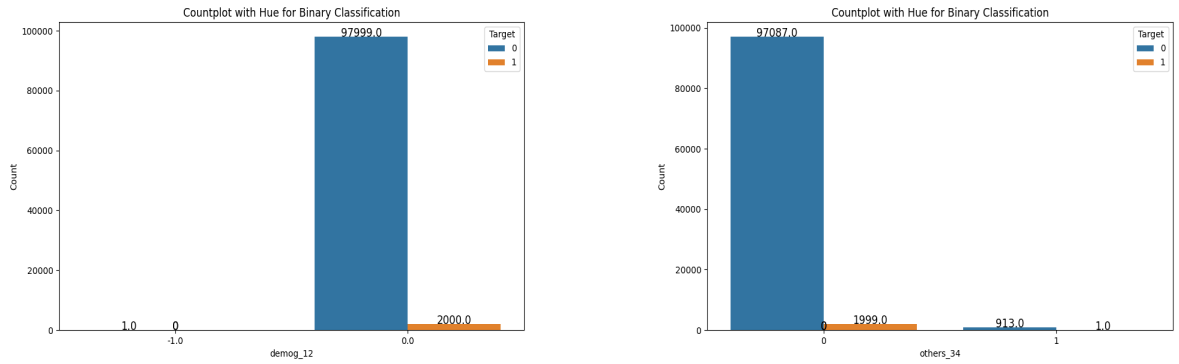

Figure 3: Marginal Columns