CONVOLVE 3.0

TEAM MEMBERS:

AARUSH SINGH

KESHAV BANSAL

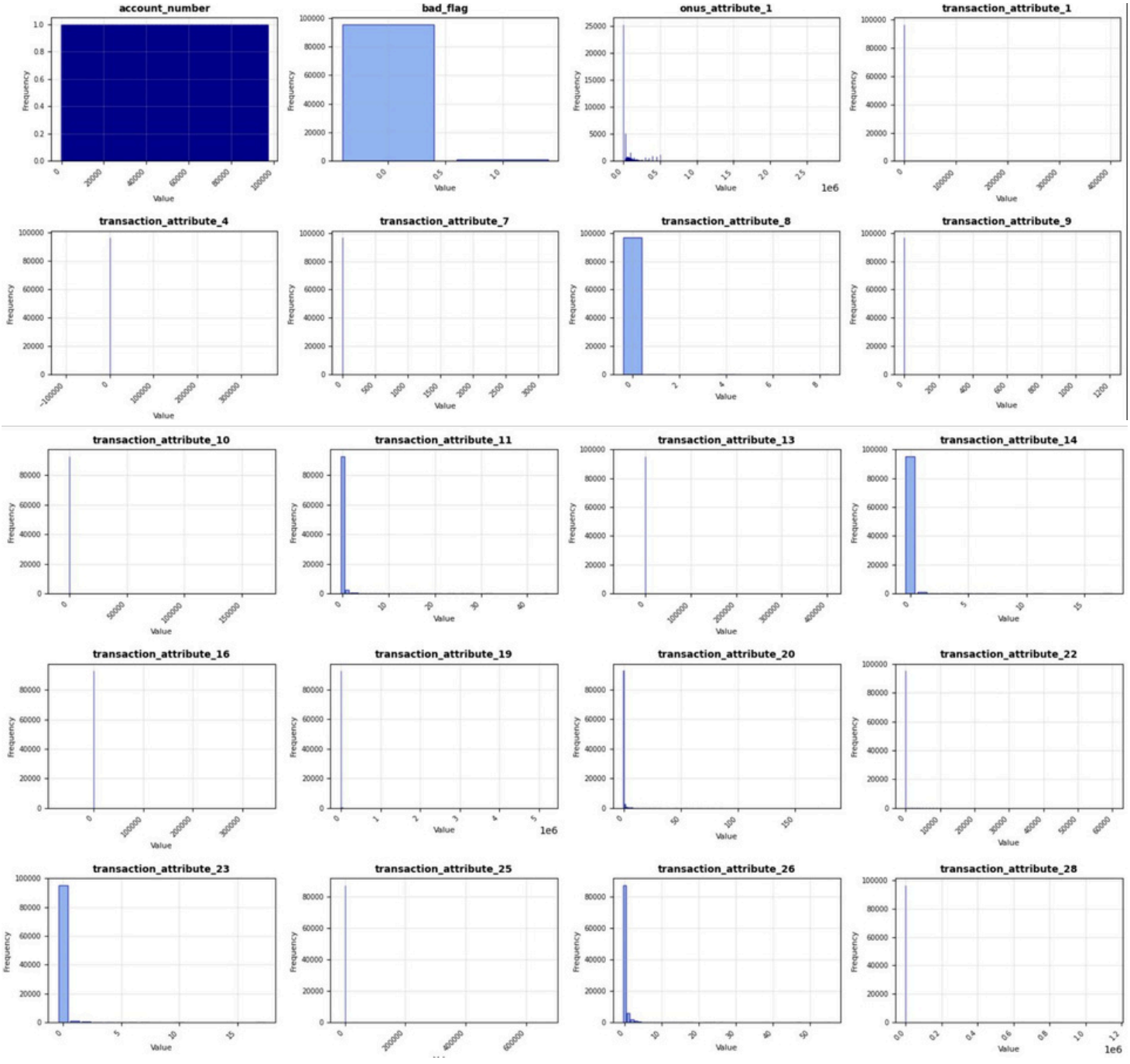
KUSHAGRA TIWARI

Table Of Content

Exploratory Data Analysis

The notebook incorporates several essential EDA steps aimed at understanding the dataset and extracting initial insights. Here's a summary of the findings and methodologies:

1. **Distribution of the Target Variable (bad_flag)**
   - A count plot revealed that the target variable, bad_flag, exhibits a highly imbalanced distribution.
   - Numerical representation of the imbalance was computed, highlighting the proportion of each class.



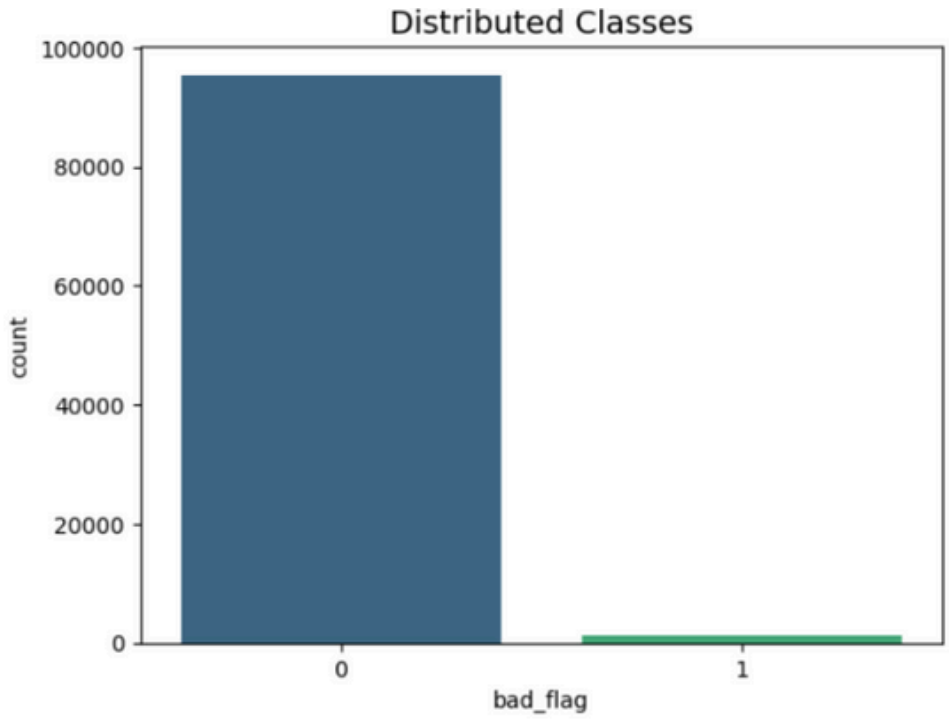
Exploratory Data Analysis

2. **Different Categories were acutely observed (transaction, onus, and bureau attributes)**

- Both Bureau and Onus likely represent categorical data which has been encoded.
- What kind of imputations were used? A lot of rows suffer from a lot of NaN values, these are then imputed via a dummy category represented by “999.0”, the reason modal imputation and linear regression based imputation were ignored is that, given the dataset, it is unlikely or atleast uncertain that modal conditions ever lead to bad_flag ever being 1, hence, in order to catch outliers modal or regression based imputation can lead to false negatives
- What about transaction attributes?: Most attributes are either highly correlated and have a lot of zero’s. Since these attributes represent card transactions, a lot of rows from

Transaction_attribute_1 to transaction_attribute_580(approximately) are likely some niche form of card transactions that most users either don't avail, or are transactions which are not applicable for their card type, as evidenced by the higher proportion of zeroes and NaNs

Hence, Imputation by 0 is the most logical step, because act of making a transaction is what we care about, it is irrelevant whether a transaction was not made out of choice or due to card settings, no additional information is revealed.



Exploratory Data Analysis

Memory Size Reduction

```python
def optimize_dataframe(df):
    """
    Downsize numeric columns in the DataFrame to minimize memory usage without losing accuracy.
    NaN values are preserved during the conversion.
    """
    optimized_df = df.copy()

    for col in optimized_df.select_dtypes(include=['int64', 'float64']).columns:
        col_data = optimized_df[col]

        if pd.api.types.is_integer_dtype(col_data) or np.all(col_data.dropna() == col_data.dropna().astype(int)):
            # If the column is integer or float but can safely be cast to integer
            if col_data.min() >= np.iinfo(np.int8).min and col_data.max() <= np.iinfo(np.int8).max:
                optimized_df[col] = col_data.astype('Int8')  # Nullable integer type
            elif col_data.min() >= np.iinfo(np.int16).min and col_data.max() <= np.iinfo(np.int16).max:
                optimized_df[col] = col_data.astype('Int16')  # Nullable integer type
            elif col_data.min() >= np.iinfo(np.int32).min and col_data.max() <= np.iinfo(np.int32).max:
                optimized_df[col] = col_data.astype('Int32')  # Nullable integer type
            else:
                optimized_df[col] = col_data.astype('Int64')  # Nullable integer type
        else:
            # For floats, determine the smallest float dtype
            if col_data.min() >= np.finfo(np.float32).min and col_data.max() <= np.finfo(np.float32).max:
                optimized_df[col] = col_data.astype('float32')
            else:
                optimized_df[col] = col_data.astype('float64')

    return optimized_df
```

All of the feature columns are of the dtype float64 but a large majority of features can be downsized to lower memory sizes without any loss in accuracy at all,any column which can be downsized has been downsized.This resulted in quicker processing of the data.

Crucial and Valuable insights

1) Cluster wise correlation between transaction attributes:

This is an excerpt from the table showcasing high correlation amongst first three columns of transaction attributes, similar clustering is observed throughout the data, possibly signalling the total amount paid over different time intervals with slightly varying interest.

Some columns were also noted to be the sums of others, although a simple collinearity check with iterative thresholds dealt with such columns.

2) Empty/ Ghost Rows:

1. Nearly 2,000 rows in the training set were that of people, with a significantly smaller fraction of fraudsters, having majority or all of their attributes empty. This is likely caused due to customers who sign up for a card but end up not using it, given that such people are fundamentally irrelevant to our model, we get rid of them.

3) High overlap in Bureau attributes:

1. Given that bureau attributes represent largely encoded categorical data, they also share a lot of encoded labels across themselves signalling some common toggle options in different aspects like Daily credit limit is ON/OFF for International payments, vs the same being ON/OFF for E commerce payments.

	transaction_attribute_1	transaction_attribute_2	transaction_attribute_3
transaction_attribute_1	1.000000	0.989940	1.000000
transaction_attribute_2	0.989940	1.000000	0.989940
transaction_attribute_3	1.000000	0.989940	1.000000
transaction_attribute_4	-0.000806	-0.000814	-0.000806
transaction_attribute_5	-0.001118	-0.001129	-0.001118
...	...	...	...
transaction_attribute_660	0.022778	0.018278	0.022778
transaction_attribute_661	-0.015431	-0.014491	-0.015431
transaction_attribute_662	-0.000789	-0.000797	-0.000789
transaction_attribute_663	0.001785	0.002205	0.001785
transaction_attribute_664	-0.001673	-0.001690	-0.001673

Feature Selection and Extraction

1. Iterative Correlation Analysis

The first step involves examining the pairwise correlation between features to detect multicollinearity, which can negatively impact model interpretability and performance.

- **Methodology:**
  - A correlation matrix (df.corr()) was computed to assess the relationships between numerical features.
  - Upper triangular values of the matrix were extracted to avoid redundancy.
  - Features with a correlation exceeding predefined thresholds (0.75, 0.8, and 0.85) were identified and flagged for removal.
- **Outcome:**
  - Redundant features were systematically removed, retaining only those with lower correlations to ensure reduced multicollinearity.
  - This step decreased the number of input features while maintaining the diversity of information.

2. Mutual Information (MI) Scoring

The notebook further evaluates the predictive power of features using Mutual Information (MI), a metric that quantifies the dependency between each feature and the target variable (bad_flag).

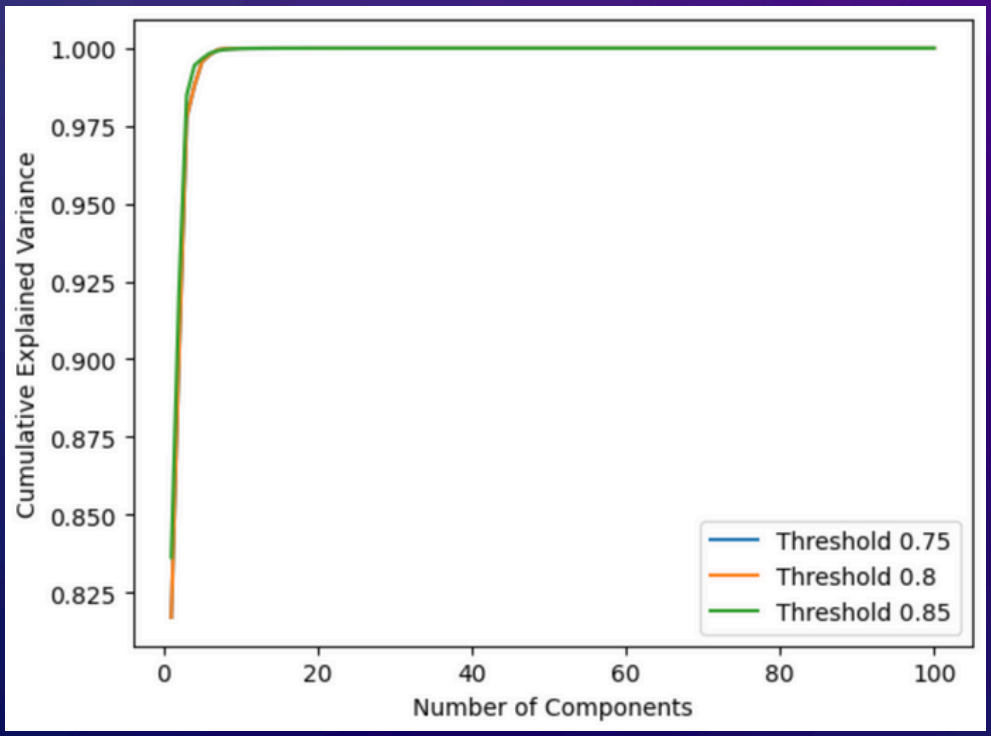
- **Methodology**:
  - Features were ranked based on their MI scores.
  - Only the top features, as ranked by MI, were retained for further analysis. This ensures that irrelevant or less informative features are discarded.
- **Outcome**:
  - A clear ranking of features by their contribution to predicting the target variable.
  - A smaller, more focused dataset optimized for model input

Feature Selection and Extraction

3. Dimensionality Reduction Using PCA

Principal Component Analysis (PCA) was employed to further refine the dataset by transforming the features into a new set of uncorrelated components while preserving maximum variance.

- **Methodology**:
  - PCA was applied to the dataset after MI-based selection.
  - Cumulative explained variance was calculated to determine the number of principal components required to retain **95%** and **99%** of the variance.
  - Features corresponding to these thresholds were retained as part of the dimensionality reduction process.
- **Outcome**:
  - Identification of the optimal number of features required for the model:
    - Features required for **95% variance**.
    - Features required for **99% variance**.
  - A plot was generated showing the cumulative explained variance against the number of components.



MODEL

Implemented a Ensemble Model consisting of XGBOOST,CATBOOST AND LIGHTBOOST

TRAIN TEST SPLIT

Implemented a Train-Test split of 80:20 and while doing so it was made sure that the proportion of zeros and ones remains the same in both the train and test set so as to achieve a fair evaluation.

```
Training set class distribution:
bad_flag
0    0.985822
1    0.014178
```

```
Test set class distribution:
bad_flag
0    0.985849
1    0.014151
```

METRIC USED FOR COMPARING MODELS

Used PR-AUC and Mean Absolute Error as evaluation metric of choice.The reason for preferring PR-AUC over ROC-AUC being that given the highly imbalanced(98.5%+ of cards having not defaulted) nature of the dataset.

HYPERPARAMETER TUNING

Final Hyperparameters for all three models were selected using RandomSearchCV . Minimisation of the Mean Absolute Error on the test set was the goal of the Hyperparameter tuning.

```
params = {
'iterations': 800,
'learning_rate': 0.07,
'depth': 5,
'loss_function': 'Logloss',
'eval_metric': 'Logloss',
'random_seed': 42,
'verbose': False
```

```
# LightGBM parameters
lgb_params = {
    'objective': 'binary',
    'metric': 'mae',
    'boosting_type': 'gbdt
    'num_leaves': 10,
    'learning_rate': 0.07,
    'max_depth': 5,
    'subsample': 0.2,
    'seed': 42,
    'verbose' :-1
}
```

```
xgb_params = {
    'n_tress': 4600,
    'objective': 'binary:logistic'
    'eval_metric': 'mae',
    'max_depth': 3,
    'seed': 42,
    'learning_rate': 0.01,
    'max_delta_step':1,
    'subsample': 0.3
```

```
ensemble_preds = 0.15 * preds_lgb + 0.7 * preds_cat + 0.15 * preds_bst
```

Above is the weightage given to each individual model in the Ensemble Model