Data Analysis on Credit Card Transactions Fraud Detection

Reyhaneh Saffar_99222064

1. Data Exploration

This is a simulated credit card transaction dataset containing legitimate and fraud transactions from the duration 1st Jan 2019 - 31st Dec 2020. It covers credit cards of 1000 customers doing transactions with a pool of 800 merchants.

Here are the column names and their data types:

Column Name	Data Type	Description
trans_date_trans_time	object	Date and time of transaction
cc_num	int64	Credit card number
merchant	object	Merchant name
category	object	Category of merchant
amt	float64	Transaction amount
first	object	First name of credit card holder
last	object	Last name of credit card holder
gender	object	Gender of credit card holder
street	object	Address of credit card holder
city	object	City of credit card holder

Column Name	Data Type	Description
state	object	State of credit card holder
zip	int64	Zip code of credit card holder
lat	float64	Latitude of credit card holder
long	float64	Longitude of credit card holder
city_pop	int64	Population of city of credit card holder
job	object	Job of credit card holder
dob	object	Date of birth of credit card holder
trans_num	object	Transaction number
unix_time	int64	Unix time of transaction
merch_lat	float64	Latitude of merchant
merch_long	float64	Longitude of merchant
is_fraud	int64	Whether the transaction is fraudulent or not

2. Data Processing

- **Null values:**

In the Data Processing section, specifically in the analysis of null values, a deep copy of the original dataset is created for further examination. The analysis involves inspecting each column to gather key information such as the number of unique values, the total count of null values, the percentage of null values relative to the total entries, and the data type of the column. This comprehensive examination is conducted across all columns in the dataset.

The results of the analysis reveal insightful details for each column. For instance, the 'amt' column has 52928 unique values and contains no null values, indicating completeness. Similar observations are made for other numerical columns such as 'lat,' 'long,' 'merch_lat,' 'merch_long,' 'cc_num,' 'zip,' 'city_pop,' and 'unix_time,' all of which exhibit no null values.

Additionally, categorical columns like 'is_fraud,' 'trans_date_trans_time,' 'merchant,' 'category,' 'first,' 'last,' 'gender,' 'street,' 'city,' 'state,' 'job,' 'dob,' and 'trans_num' also display a lack of null values, while providing insights into their respective unique value counts and data types.

This thorough analysis is instrumental in understanding the integrity of the dataset, ensuring that the subsequent stages of the statistical analysis are built upon a foundation of clean and complete data.

- **Sampling:**
  In the Sampling phase of the Data Processing section, an initial assessment of the test dataset reveals its substantial size, consisting of 555,719 entries across 22 columns. To address the challenge posed by the dataset's magnitude, a stratified sampling approach is employed. The dataset is partitioned into two subsets based on the 'is_fraud' column values, distinguishing between fraudulent and non-fraudulent transactions.

  The 'df_fraud' subset comprises 2,145 entries, representing instances of fraudulent transactions, while the 'df_not_fraud' subset contains 553,574 entries denoting non-fraudulent transactions. To create a balanced representation for model training, a custom sampling function is implemented. This function generates a sample from the non-fraudulent subset, adjusting its size based on a specified multiplier. In this case, the non-fraudulent sample size is set to ten times the size of the fraudulent subset.

  The resulting 'df_not_fraud_sample' consists of 21,450 entries, effectively balancing the representation of fraudulent and non-fraudulent transactions in the test dataset. These samples are then combined into 'df_test_sample,' serving as a representative subset for model evaluation.

  Subsequently, the training and test datasets are prepared, with features ('X') and labels ('Y') separated. To address the imbalanced nature of the training dataset, the RandomUnderSampler from the imbalanced-learn library is applied. This technique mitigates the class imbalance by randomly removing instances from the majority class, ensuring a more equitable distribution for model training.

  This strategic sampling process contributes to the robustness of the model by enabling it to effectively learn from both fraudulent and non-fraudulent instances, enhancing its predictive capabilities in real-world scenarios.

- **Data Scaling:**
  In the Data Scaling phase, a copy of the preprocessed dataset is created to facilitate the application of a scaling transformation. Leveraging the StandardScaler from the scikit-learn library, numerical columns within the dataset are selected for scaling. The scaling process involves transforming the numerical features to a standard scale, ensuring uniformity and comparability among variables.
  A custom function is implemented to apply the scaling transformation to the specified numerical columns in the dataset. This function utilizes the StandardScaler to standardize the selected columns, ensuring that they have a mean of zero and a standard deviation of one.
  The numerical columns subjected to scaling include "amt," "lat," "long," "merch_lat," "merch_long," and "city_pop." The same scaler instance is employed for both the initial dataset and an additional dataset, 'X_test,' ensuring consistency in the scaling process.
  The scaled dataset, 'df_scale,' is then printed, revealing the standardized values of the selected numerical columns. This transformation ensures that the numerical features are on a consistent scale, mitigating potential issues caused by varying magnitudes among different variables. This standardization contributes to the effectiveness of machine learning algorithms, particularly those sensitive to the scale of input features, by enhancing model convergence and interpretability.

- **Data Encoding:**
  In the Data Encoding phase, a new dataset, 'df_encoded,' is created as a copy of the previously scaled dataset, and a similar copy, 'encoded_test,' is made for additional processing. The focus of this phase is on encoding categorical variables to facilitate their incorporation into machine learning models. Identified categorical columns, including "cc_num," "merchant," "category," "gender," and "job," are earmarked for encoding.
  The One-Hot Encoding technique is applied using the OneHotEncoder from scikit-learn. This process transforms categorical variables into a binary matrix representation, creating columns for each unique category and assigning binary values (0 or 1) to indicate the presence or absence of a category. The encoder is configured to handle unknown categories and produce a dense output matrix.
  A ColumnTransformer is employed to specify the columns and transformers for the encoding process, with the 'cat' transformer assigned to the OneHotEncoder and applied to the designated categorical columns. The 'remainder' parameter is set to 'passthrough,' ensuring that non-categorical columns remain unaltered.
  The transformation is applied to both the training dataset ('df_encoded') and an additional dataset ('X_test'), resulting in 'X_encoded' and 'X_test_encoded,' respectively. The dimensions of these transformed datasets are displayed, revealing the expansion of the feature space due to the one-hot encoding process.
  The resulting shapes of (15012, 2193) for 'X_encoded' and (23595, 2193) for 'X_test_encoded' indicate the creation of new binary columns for each unique category in the specified categorical columns. This expanded feature set is instrumental in capturing the categorical information effectively, enabling machine learning models to leverage these encoded representations for improved performance and accuracy in predictive tasks.

3. Model:

In the Model Evaluation phase, various classification algorithms are applied to the preprocessed and encoded data to assess their performance in predicting fraudulent transactions. The models include Logistic Regression, Linear Support Vector Classification (LinearSVC), K-Nearest Neighbors (KNN), Decision Tree, Random Forest, and Naive Bayes.

For each model, the following evaluation metrics are computed and reported: Accuracy, Area Under the Receiver Operating Characteristic curve (AUC), and F1 Score. These metrics provide a comprehensive view of the model's ability to correctly classify instances, its discriminatory power, and the balance between precision and recall.

After fitting each model to the training data and predicting on the test data, the metrics are printed, showcasing the model's performance. Additionally, a Receiver Operating Characteristic (ROC) curve is plotted for each model, visualizing the trade-off between true positive rate and false positive rate across different classification thresholds.

The generated ROC curve plot illustrates the performance of each model, with a legend indicating the AUC for each. Models achieving higher AUC values are generally better at distinguishing between positive and negative classes. The plot is saved as "roc_curve.eps."

In the specific output presented:

- Logistic Regression achieves an accuracy of 67.51%, an AUC of 72.38%, and an F1 Score of 30.48%.
- LinearSVC achieves an accuracy of 73.09%, an AUC of 75.05%, and an F1 Score of 34.35%.
- KNN achieves an accuracy of 63.23%, an AUC of 61.32%, and an F1 Score of 22.58%.
- Decision Tree achieves an accuracy of 80.08%, an AUC of 67.14%, and an F1 Score of 31.90%.
- Random Forest achieves an accuracy of 82.81%, an AUC of 78.97%, and an F1 Score of 43.99%.
- Naive Bayes achieves an accuracy of 46.41%, an AUC of 29.15%, and an F1 Score of 2.66%.

These metrics collectively offer insights into the strengths and weaknesses of each model, aiding in the selection of the most suitable model for predicting fraudulent transactions in the given dataset.