Capital One - Data Science Challenge Submission

CID: C2475091 **Recruiter:** Brianne Wade **Version:** v20.01

For this challenge, the scripts are written in Python language. Data is loaded using the JSON library in python and saved in a list. The list is then converted to a pandas data frame. The data is then preprocessed, in which the columns 'merchantCity', 'merchantState', 'merchantZip', 'posOnPremises' and 'recurringAuthInd' are dropped due to insufficient data and the data structure of the 'transactionDateTime' is modified.

For visualization of 'transactionAmount' structure, matplotlib library is used. In the next step reversed and multiswiped transactions are programmatically identified. In the case of reversed transactions, the 'purchase' transaction is identified using the transaction amount and the date & time of the transaction. The assumption here is that the 'Purchase' transaction is not absent from our dataset. The condition set for identifying reversed transactions is that the transaction amount should be the same for 'Purchase' and 'Reversal', while the customer ID and account number should also match. And the data & time of 'Reversal' should be later that the date & time of the 'Purchase'.

To identify duplicate transactions, past 120seconds transactions are analyzed for a specific customer id and account number. If the transaction amount of the past transaction matches the present transaction amount, the present transaction is classified as a duplicate transaction. Assumptions here are that a customer will not make two different transactions of the exact same amount within a short span of time.

In the next step, the data is preprocessed again for the fraud detection model in which the boolian fields are converted into binary value fields. An additional column is added which takes '1' value if the entered CVV and card CVV match, and '0' if they do not match. Then the columns 'echoBuffer', 'cardLast4Digits' 'merchantName', 'accountOpenDate', 'transactionDateTime', 'currentExpDate', 'customerId','dateOfLastAddressChange', 'accountNumber','enteredCVV','cardCVV' are dropped. Assumption here is that the information in these columns will not affect the result of Fraud detection model significantly. Furthermore, the categorical values are one-hot encoded and missing values are filled with the most frequent values. For this task, sklearn library is used. RandomUnderSampler function from the imblearn package is used to randomly sample the data.

This cleaned data is then split into test and train sets using sklearn traintestsplit function. For training the model Gridsearch of hyperparameters is used. For this assignment, a logistic regression model is trained.

Size of train data:- (18625, 45) Fraud entries - 9350, Non-Fraud entries - 9275

Size of test data:- (6209, 45) Fraud entries - 3067, Non-Fraud entries - 3142

Answer 1:

Data is loaded using the JSON python library as saved in a list. This data is further converted into a pandas data frame. In the dataset given for this challenge, we have there are 786,363 data points, each representing one credit card transaction. Listed below are the fields in each datapoint and their corresponding data structures:

Name of Field	Data Structure	Name of Field	Data Structure

accountNumber	Integer	enteredCVV	Integer
customerId	Integer	cardLast4Digits	Integer
creditLimit	Integer	transactionType	String
availableMoney	Integer	echoBuffer	String
transactionDateTime	Date Time format	currentBalance	Float
transactionAmount	Float	merchantCity	String
merchantName	String	merchantState	String
acqCountry	String	merchantZip	Integer
merchantCountryCode	String	cardPresent	boolean
posEntryMode	Integer	posOnPremises	String
posConditionCode	Integer	recurringAuthInd	String
merchantCategoryCode	String	expirationDateKeyInMatch	boolean
currentExpDate	Date	isFraud	boolean
dateOfLastAddressChange	Date	accountOpenDate	Date
cardCVV	Integer		

Listed below are the summary stats of each field in the data:

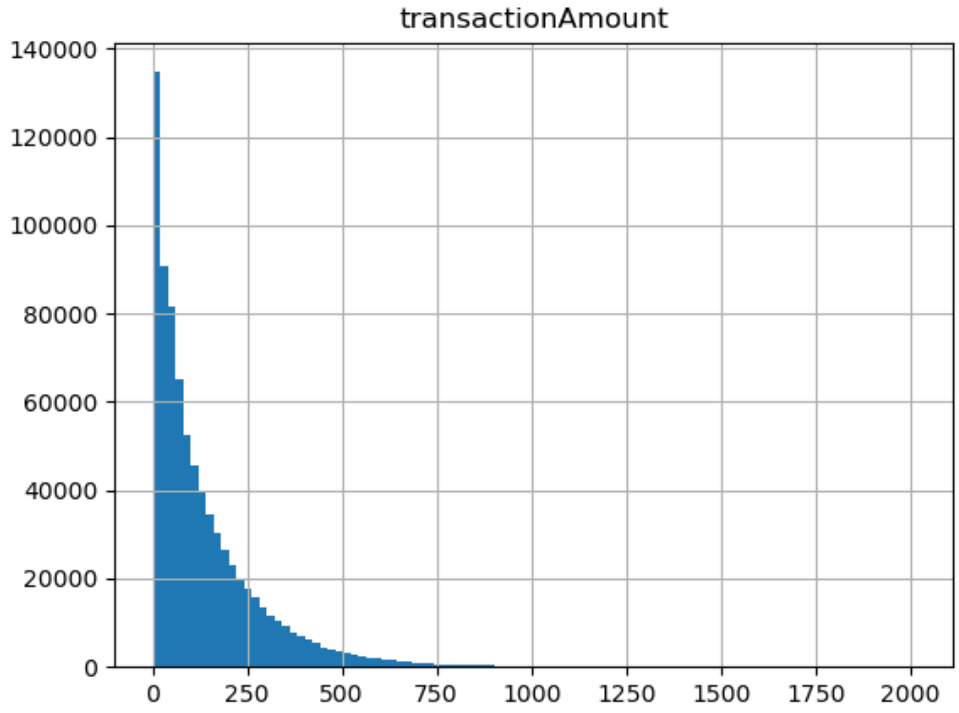
Name of Field	**Summary Statistics**
accountNumber	Unique vals - 5000, Null vals - 0
customerId	Unique vals - 5000, Null vals - 0
transactionAmountcreditLimit	Max credit limit - 50000, Min credit limit - 250, Null vals - 0, Unique vals - 10, Avg credit limit - 10759, Std dev - 11636
availableMoney	Maximum - 50000, Minimum- -1005, Null vals - 0, Average available money - 6250, Std dev - 8880
transactionDateTime	Start date - 01-01-2016, End date - 12-30-2016, Null vals - 0
transactionAmount	Total entries - 786363, Unique vals - 66083, Max val - 2011.54, Min val - 0, Mean vals - 136, Std dev - 147

merchantName	Null vals - 0, Unique vals - 2490
acqCountry	Null vals - 4562, Unique vals - 4, Unique Entries - ('US', 'CAN', 'MEX', 'PR')
merchantCountryCode	Null vals - 724, Unique vals - 4, Unique Entries - ('US', 'CAN', 'MEX', 'PR')
posEntryMode	Null vals - 4054, Unique vals - 5, Unique Entries - ('02', '09', '05', '80', '90')
posConditionCode	Null vals - 409, Unique vals - 3, Unique Entries - ('01', '08', '99')
merchantCategoryCode	Null vals - 0, Unique vals - 19
currentExpDate	Null vals - 0, Min date - 01-2020, Max date - 12-2032
accountOpenDate	Null vals - 0, Min date - 08-22-1989, Max date - 12-31-2015
dateOfLastAddressChange	Null vals - 0, Min date - 08-22-1989, Max date - 12-30-2016
cardCVV	Null vals - 0, Range - [100, 998]
enteredCVV	Null vals - 0 , Range - [000 - 998]
cardLast4Digits	Null vals - 0, Range - [0, 9998]
transactionType	Null vals - 689, Unique vals - 3, Unique Entries - ('Purchase', 'Address Verification', 'Reversal')
echoBuffer	All null values
currentBalance	Null values - 0, Max value - 47498.81, Min value - 0
merchantCity	All null values
merchantState	All null values
merchantZip	All null values
cardPresent	Unique values - 2, Unique entries - (True, False)
posOnPremises	All null values
recurringAuthInd	All null values
expirationDateKeyInMatch	Unique values - 2, Unique entries - (True, False)
isFraud	Unique values - 2, Unique entries - (True, False)

Fields which have all null values like merchantCity, merchantState have been removed from the data in the preprocessing step.

Answer 2:

Here is what the histogram of the processed amounts of each transactions looks like:



Observation:- The histogram is right skewed. There are a higher number of transactions with the low transaction amount, and low transactions with higher transactions

Hypothesis:- Looking at the structure of the histogram, we can hypothesize that credit cards are mostly used for day-to-day transactions like buying groceries, paying bills, and paying for travel. People do not prefer using credit cards to buy higher-value goods or pay high amounts of money.

Answer3 :

The total number of reversed transactions in this dataset is 43625 and the total amount of reverse transactions is 2,777,228.26$

The total number of duplicate transactions could not be computed due to high code complexity. The current code has high time complexity and requires significant time to run. This can be further improved by optimizing the code.

Answer4 :

For this assignment, the logistic regression model is used to identify if a given transaction is Fraud or not. The results of the model are as follows:

True Positive: 2095 samples
True Negative: 2176 samples
False Negative: 1002 samples
False Positive: 936 samples
Precision : 0.6911
Recall : 0.6764
Accuracy : 0.6878

Improvements over this model can be done by including more fields in the data and using a better strategy to fill the missing data. Additionally, models such as Random Forest, SVM, and XGBoost can be used to improve accuracy.

Instructions to run the code: Download the 'dsc_C2475091.zip' folder and unzip the contents.
Download the Capital One data and move the contents to a 'Data' folder

1. To run the trained model:- A trained model of logistic regression is saved in the model directory
    a. Run the 'python Scripts/main.py' command

This will run the script with the saved model. It will print the NaN values for all Entries at first. Then it will print the Min, Max, Null values and number of unique values for all entries. Finally, it will print, true positive, true negative, False positive, and False Negative values for the test dataset, along with precision, recall, and accuracy.

2. To re-train the model and evaluate the performance:
    a. Run the 'python Scripts/main.py –train' command

This will re-train the logistic regression model and save the new model params in a pickle file named 'logistic_regression_model'. It will re-evaluate the new model and print new true positive, true negative, False positive, and False Negative values for the test dataset, along with precision, recall, and accuracy metrics.

3. To evaluate the number of reverse transactions:
    a. Run the 'python Scripts/main.py –reverse_transactions' command

This will compute the number of reverse transactions and the amount in $ of reverse transactions and print out the values.

4. To evaluate the number of duplicate transactions:
    a. Run the 'python Scripts/main.py –duplicate_transactions' command

This will compute the number of duplicate transactions and the amount in $ of duplicate transactions and print out the values. The time complexity of this operation is high and needs improvement.

The requirements.txt file can be used to install all necessary libraries to run the above scripts.