**Instructions to run the code:**

(Assume user is using mac and has python3 installed)

Please pull down the [dataset](https://github.com/CapitalOneRecruiting/DS), and unzip it to the C1 directory (local folder).

In terminal, run ‘make install’ to install the required packages.

**Q1+Q2) EDA and Data Quality Assessment**

In terminal:

* Run ‘make profiling’ to run pandas profiling. This takes a couple of minutes to complete, the output has been saved for you transactions\_output.html
* Run `make run\_EDA` to generate eda report. This runs QC logic that removes problematic records, and saves a dataset for future analysis

Check find\_EDA.py for code logic.

A pandas profiling report can give greater detail into the EDA. From that report, I have highlighted the most important characteristics and insights. The basic summary statistics and counts of variables can be found as output in the code, I have omitted this information in an effort to keep this readout succinct.

The raw dataset contains 786,363 records of credit card transactions. The dataset is extremely unbalanced, with approximately 1.6% of transactions being fraudulent. The majority of the transitions occur in the US. Each transaction has an associated date. A fraud indicator is also present. Information about the account is also present.

There are 13 categorical columns, with some blank or missing values (see data quality report below). Many of these categorical columns are related to the type of transaction, or descriptions of the merchant. These columns have high cardinality. Merchant name had a transaction number (ex #1234) which I removed in an attempt to clean the dataset and reduce the cardinality of this column. The merchant Fresh Flowers had the highest rates of fraud (6.5% of transactions).

If I had more time, I would investigate these merchant transaction numbers to see if there was any association with the likelihood of fraud (maybe create an indicator variable if there was a re-current transaction for example).

Below are some data quality issues I found with the dataset and subsequently corrected. I took a very conservative approach and dropped records from modeling if a single feature had any missing values. If I had more time, I would have explored feature imputation strategies to fill in the missing values.

1. **Empty Columns**: columns below are empty. These columns dropped from analysis/modeling.
   1. echoBuffer
   2. merchantCity
   3. merchantState
   4. merchantZip
   5. posOnPremises
   6. recurringAuthInd
2. **Inliers:**
   1. Many of the transaction features (transactionAmount, availableMoney, currentBalance have many zeros). This is concerning in that the mechanisms that generated this dataset may have errors where a missing value is set to zero. This is something to investigate in more detail.
   2. There is a single account (380680241) and customer (380680241) Inlier, making up ~4.2% of all transactions. This account/customer has been dropped from dataset for modeling.

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

**3. Missing values**: acqCountry, merchantCountryCode, posEntryMode, posConditionCode had blank values. These records have been removed from the dataset.

4. **Outliers in numeric features:** numeric values more than 5 sd from mean were dropped from analysis. This affected columns:

* 'transactionAmount'{2057} outliers removed
* 'currentBalance' {8115} outliers removed
  + Note: A log transform of these right tailed values may be a more appropriate method to identify outliers

**5. Outliers in Date features:** accountOpenDate and dateOfLastAddressChange have records that are misformatted and occur before 2002.

* remove records with accountOpenDate less than '2002-05-13'
* remove records with dateOfLastAddressChangeless less than '2002- 11-05

After applying these filters, around 52,000 records were removed for a final record count of 734,000. This is around 7% of the original dataset. The class imbalance got slightly worse using this filtering method, going from 1.7% to 1.4%.

**Q2) Analysis of TransactionAmount feature**

Check find\_EDA.py for code logic (Q2: Transaction Amount).

After dropping records that didn’t pass the data quality checks(~7% of original data), I analyzed the structure of the transaction amount column. I noticed a right tail skew. There were a lot of inliers at 0, indicating possible data corruption (one would expect non zero transaction amounts). There was low correlation between this feature and fraud.

![Chart, histogram

Description automatically generated]() ![Chart, histogram

Description automatically generated]()

I hypothesize that this feature is lognormal. I applied a lognormal transformation below. If we ignore the inlier at zero, the resulting transformation looks normal. To test my hypothesis that his is lognormal, I took the added 1 and took the log of transactionAmount. I then ran a normality test (with the null hypothesis that the log-transformed data is normal. I got a P value of 0 which was less than alpha of 0.001, so I could reject the null hypothesis that the data is normal. Therefore, the data is not log-normal based on the statistical test. Maybe exponential?

If I had more time I would dig into this and apply the correct transform. The benefit of this analysis is that by applying the correct data transformation, we can use statistical classifiers without violating their underlying assumptions, and correctly infer accurate P-values and confidence intervals. Since we are primarily focused on maximizing predictive accuracy using machine learning approaches, this descriptive statistical analysis has not been performed.

**Q3) Duplicate Transactions Analysis**

Run `make find\_duplicate\_transactions`. This runs the duplicate transaction function that programmatically identifies records that are duplicates.

Check find\_duplicates.py for code logic.

I used the original dataset, and re-ran some preprocessing on it:

* Created a date-time transform on the transactionDate
* Removed number from merchant name
* I sorted the dataset by customerID, transactionID, and date to ensure

**Steps to calculate reversals:**

I counted up the number of reversals by identifying them using transactionType = ‘REVERSAL’. To investigate further, I looked at the top customers based on reversals, and saved customer #2 to Q3\_reversal\_example.csv. I then manually investigated when reversals occur, at first glance it appears most of them happen more than a day later. Sometimes they occur in the same day.. If I had extra time, I would verify that transactionType is in fact correct, it could be there are reversals in the dataset that are not recorded by the transactionType column. Custom logic could be developed to look for transactions that are the from the same merchant, amount, customer; but occur later

**Total reversals: 20,303 total amount: $ 2,821,792.5**

To calculate number of multi-swipes, I created an algorithm that runs in O(N) time and space. I did this by creating 2 dictionaries that are populated using a specific key. To generate this key, I assumed that a multiswipe occurs when:

1. it happens on the same day
   1. (this is not always true, a multiswipe can occur between midnight and 1AM and we would miss this),
2. To the same customer/account
3. To the same merchant
4. for the same amount.

A multiswipe also cannot be a reversal.

If I had more time, I would improve this logic by hashing the transaction datetime such that it is in customizable blocks. For example, we would define a multiswipe as a transaction that occurs within X minutes. Then, we can vary X and get upper/lower bounds of the number of multiswipe transactions.

**Total multi-swipes: 7,900 , total amount: $ 1,103,357.75**

**Q4) Modeling**

* Run `make build model`. This creates features, sets up modeling pipelines, and generates the output.
* Note: this takes a while to run. The logistic regression model requires a lot of compute.

**Feature engineering approach**

After dropping records that didn’t pass the data quality checks (see EDA report section above for details as to what was dropped, ~7% of original data).

I generated 8 new features related to these general ideas:

1. Information about transactions. Does someone spend over their limits or balances?
2. Information about the time the transaction occurred. Is there a particular monday or hour where there is more fraud?
3. Features related to the length the account is open, or how long has it been since their address changed.
4. Future work if I had more time: I would focus on the merchant names. I would choose the top X merchants as categories, or create a merchant embedding vector to incorporate the merchant type.

New feature generation logic:

X['cvv\_match'] = X.cardCVV == X.enteredCVV

X['utilization\_rate'] = X.currentBalance / X.creditLimit

X['transaction\_over\_limit'] = X.transactionAmount / X.creditLimit

X['transaction\_over\_balance'] = X.transactionAmount / (X.currentBalance + 1)

X['transactionDateTimeHour'] = pd.to\_numeric(X.transactionDateTime.str.split("T", expand=**True**).iloc[:,1].str.split(":", expand=**True**).iloc[:,0])

X['transactionDateTimeMonth'] = pd.to\_numeric(X.transactionDateTime.str.split("T", expand=**True**).iloc[:,0].str.split("-", expand=**True**).iloc[:,1])

X['address\_open\_diff'] = (X['dateOfLastAddressChange'] - X['accountOpenDate']) / np.timedelta64(1, 'D')

X['transaction\_address\_diff'] = (X['transactionDateTime'] - X['dateOfLastAddressChange']) / np.timedelta64(1, 'D')

**Modeling Methodology**

After generating the features, I just focused on logistic regression. To deal with the class imbalance, I employed sampling strategies from imbalanced-learn package. If I had more time, I would try more advanced strategies such as SMOTE, although more accuracy is more likely if we focus on feature engineering, particularly with the text features.

**Final Accuracy**

These numbers were generating using sklearn pipelines. I also utilized imbalanced-learn packaged for quick sampling strategies. Numeric features were centered and standardized.   
  
Below are accuracy statistics on a cross validation set. I was able to get a model with accuracy a dumb majority class classifier. Overall, the logistic regression with under sampling is best, with a ROC on the Cross validated set being equal to 0.66. There is definitely more room for improvement, and this would be the starting baseline to beat. More features, more models, and more hyperparameter tuning will undoubtably improve this baseline.

Text

Description automatically generated

Thank you for taking the time to review this assignment, it was a lot of fun for me!