Module 5 - Case Study: Transaction Fraud Detection

Note: This case study is part of  your submission due in at the end of module 5!

Deliverable:

**Systems Planning and Requirements** (Deliverable A): Use the System Planning Framework to submit a design document. For this submission, use a markdown document titled `SystemsPlan.md` in your repository. You can may use the [Design Document Template.](https://jhu.instructure.com/courses/66217/pages/resource-design-document-template)

**Exploratory Data Analysis (**Deliverable B): Analyze the transaction data to engineer features. Below is the list of questions your supervisor would like you to analyze. For every question, provide figures and tables supporting your response, and ***explain how your findings will influence how you would design your features, the fraud detection model, or the overall system.*** Report your findings (analysis and design decisions) in a jupyter notebook named `analysis/exploratory\_data\_analysis.ipynb`.

1. What is the distribution between fraudulent and non-fraudulent transactions?
2. Which age groups are more susceptible to fraudulent transactions?
3. If a fraudulent transaction has been committed, what is the expected number of transaction committed per day with the same victim? (Temporal component asking do fraudulent transactions occur multiple times on the same customer)
4. For each purchase "category," plot the mean and standard "amt" between fraudulent transactions and non-fraudulent transactions
5. What is the distribution between time transactions between fraudulent transactions and non-fraudulent transactions? What is the most likely time of a fraudulent transaction?
6. Is there a relationship between the distance between customer location (lat/lon) and merchant location (merch\_lat/merch\_lon) and fraudulent transactions?
7. Are certain states, cities, and zip codes more susceptible to fraudulent transactions?
8. What is the distribution of fraudulent and non-fraudulent transactions occurring for each day of the week (i.e., Sunday, Monday, etc.), each month of the year (i.e., January, February)?
9. Compared to any other time of the year, how prevalent are fraudulent transactions during the holidays (11-30 to 12-31)? During post-holidays (1-1 to 2-28)? During the summer (05-24 to 09-01)?
10. Is there a relationship between between "city\_pop" and incidence of fraud?
11. Are certain "job" types susceptible to fraudulent transactions?
12. Think of at least 4 more questions and provide the answers to them.

**Data Engineering Pipeline**(Deliverable C): Prototype a ETL pipeline to transform the data using the knowledge discovered in your Exploratory Data Analysis. Describe this pipeline in your design document titled `SystemsPlan.md` (under the methodology section). Use the following directions:

1. In a module named `data\_pipeline.py`, define an class called `ETL\_Pipeline`
2. The `ETL\_Pipeline` class must contain at least 3 class methods: `extract()`, `transform()` and `load()`
3. The `extract()` method will take at least one parameter called `filename` indicating the .csv file carrying data we are looking to extract.
4. The `transform()` method will contain all the processes needed to clean, process, and prepare the data for modeling
5. The `load()` method will contain the processes to write the pre-processed dataset to a specified .csv file. (pandas `to\_csv()` function)
6. Use as many methods/functions or classes as needed to keep your code organized and readable. For example, every transformation can be wrapped within a function. Document your methods/functions using [DocstringsLinks to an external site.](https://realpython.com/documenting-python-code/" \t "_blank).

**Dataset Partitioning**(Deliverable D): Design the partitioning strategy you will use to evaluate your model and argue for its validity. Describe this pipeline in your design document titled `SystemsPlan.md`(under the methodology section).

1. In a module named `dataset.py`, define an class called `Fraud\_Dataset()`.
2. The `Fraud\_Dataset()` class should contain all processes needed to split a dataset.
3. The `Fraud\_Dataset()` should include at least 3 functions `get\_training\_dataset()`, `get\_testing\_dataset()`, `get\_validation\_dataset()`. These functions will return a training, validation, and testing dataset, respectively.
4. It is recommended that you design this class to adapt to a k-fold cross validation evaluation method.
5. Use as many methods/functions or classes as needed to keep your code organized and readable. Document your methods/functions using [DocstringsLinks to an external site.](https://realpython.com/documenting-python-code/" \t "_blank).

**Metrics Pipeline**(Deliverable E): Design the model metrics and argue for why they are pertinent to our system/model goals in you design document titled `SystemsPlan.md` (under the methodology section). *Hint: Include advanced strategies to evaluate your model*. Use the following directions:

1. Implement these metrics in a module named `metrics.py`. This module will define a class called `Metrics`.
2. The `Metrics` class must contain a function called `generate\_report()`. This function will take in at least two arguments: arguments y\_prediction, and y\_label (the models output and the ground truth label, respectively). When called, the function will generate a report which can be either be a .txt file or a file format of your choosing stored in a directory called `results`.
3. The `Metrics` class should also contain all functions needed to generate the reports. It's acceptable to use open source functions.
4. Use as many methods/functions or classes as needed to keep your code organized and readable. Document your methods/functions using [DocstringsLinks to an external site.](https://realpython.com/documenting-python-code/" \t "_blank).

The output of the report generated by run() should be:

Model Results:

Precision: W%  
Recall: X%  
Sensitivity: Y%  
Specificity: Z%  
...

This could be of the form:

def run(...):

# Generate Report

print(f"Precision: self.precision(...)%", ...)  
print(f"Recall: self.recall(...)%", ...)  
print(f"Sensitivity: self.sensitivity(...)%", ...)  
print(f"Specificity: self.specificity(...)%", ...)  
...

# Generate Figures

Remember, this report should be written out to a text file!

**Model Selection Analysis** (Deliverable F):  Analyze the performance of three candidate models of your choosing. Argue for which model you select using the criteria you design in . Report how the selected model's performance will influence the systems design in a jupyter notebook titled analysis/model\_performance.ipynb.

1. In a module named `model.py`, define an class called `Fraud\_Detector\_Model()`. This class will construct your model and handle the necessary logic to take raw input data, and produce an output. Include a method called train(...) to train your model. and a method called test(...) to call your metrics module. Document your methods/functions using [DocstringsLinks to an external site.](https://realpython.com/documenting-python-code/" \t "_blank).

**Deployment Strategy** (Deliverable G): Deploy your system. Explain your deployment strategy in SystemsPlan.md.

1. Using the same module named `model.py` in Deliverable F, define an class called `Fraud\_Detector\_Model()`. Again, this class will construct your model and handle the necessary logic to take raw input data, and produce an output. Include a method called predict(...) which will take in inputs as an argument and produce a string output: "Not Fraud" or "Fraud".
2. Construct a Dockerized system architecture that enables the model to receive and process input data from requests sent via Postman. This input will need to be in the format of the row in the dataset provided. *Hint: 1) use your ETL\_Pipeline to transform your data into usable features. 2) you may use a .json file to send files (see example below).*
3. Package your system into a docker image and publish it in DockerHub.
4. Check in your final code submission to your GitHub repository (in your FraudDetection directory). Include a ReadMe.md to describe its contents and instructions to run the system along with a test case.

Example content of .json file:

{

"trans\_date\_trans\_time": ... ,  
"cc\_num": ...  
"merchant": ... ,  
"category": ... ,  
"amt": ... ,  
...  
"merch\_long": ... ,

}

Module 2 - Case Study: Transaction Fraud Detection

Note: This case study will be the basis for your submission in Module 5.

Background:

You are employed as a newly hired AI/ML developer for a young and rapidly expanding bank holding company specializing in credit card services. Recently, there has been a rising number of customer reports regarding fraudulent transactions--which is slowly eating into the profitability of your company. While you know that your company uses a machine learning system to automatically detect fraud,  you suspect that the system's model prediction performance has started to deteriorate. This suspicion has lead you to investigate the existing model. You discover that these models are now only achieving an expected precision and recall score of ~40% and ~70%, respectively. After reporting your findings to the CEO,  she has tasked you to prototype a new model to improve upon the current model's performance. Her office has provided you data on the historical transactions of 1000 random customers from 2019-01-01 to 2020-12-31 (today is January 1, 2021). This dataset includes both 1) personal information, 2) transaction information, and 3) whether or not the transaction was fraudulent. The dataset can be downloaded [here](https://jhu.instructure.com/courses/66216/files/9594135?wrap=1)[Download here](https://jhu.instructure.com/courses/66216/files/9594135/download).

|  |  |
| --- | --- |
| **Column** | **Description** |
| trans\_date\_trans\_time | Transaction date and time |
| cc\_num | Unique customer number/ID |
| merchant | Merchant/vendor name for transaction |
| category | Category of purchase (e.g., entertainment, gas\_transport, food\_dining, etc.) |
| amt | Total amount of transaction |
| first | Customer first name |
| last | Customer last name |
| sex | Customer's sex |
| street | Street address of customer |
| city | City address of customer |
| state | State of customer residency |
| zip | Zip code of customer |
| lat | latitude coordinate of customer address |
| long | longitude coordinate of customer address |
| city\_pop | Population of city |
| job | Customer's employment title |
| dob | Customer's date of birth |
| trans\_num | Unique transaction number |
| unix\_time | Timestamp of transaction |
| merch\_lat | Latitude of merchant/vendor |
| merch\_long | Longitude of merchant/vendor |
| is\_fraud | 1=fraudulent transaction, 0=non-fraudulent transaction |