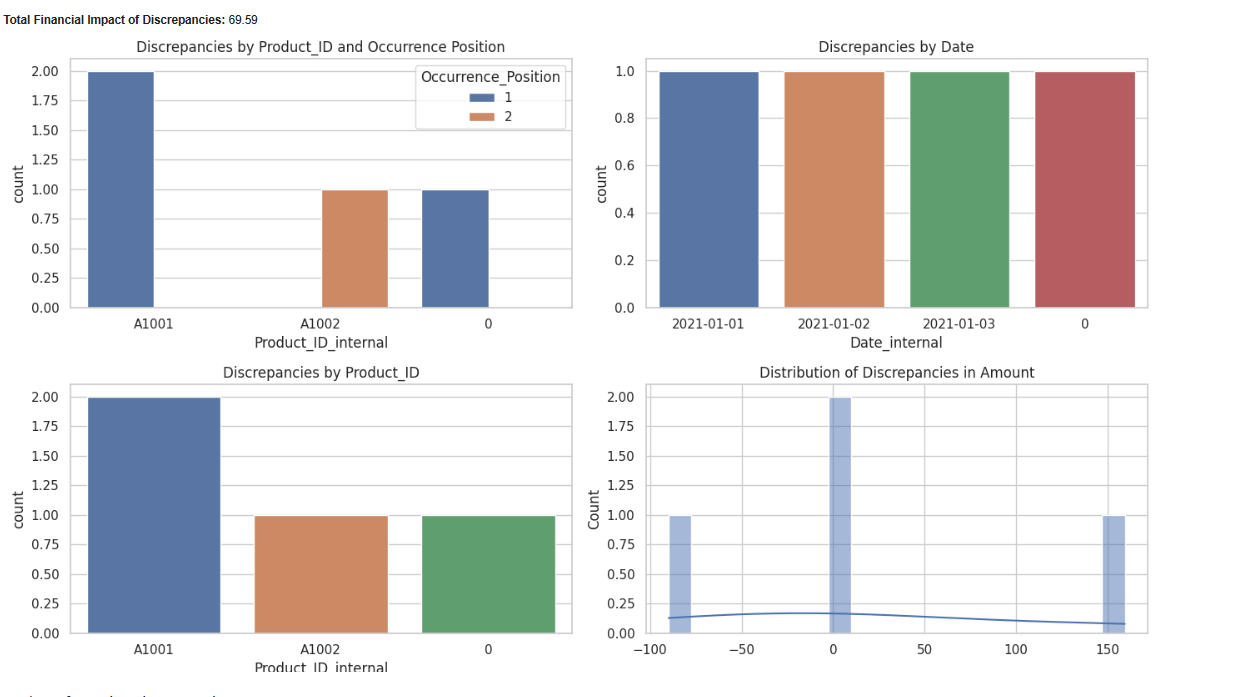
## **Reconciliation Analytics for E-commerce Transactions**

* **Background:**
  + You're working for an e-commerce company that utilises multiple payment gateways and internal systems to record transactions. Occasionally, discrepancies arise between what's recorded in the internal system versus what the payment gateways report. Reconciliation is crucial to ensure financial accuracy.
* **Objective:**
  + Your task is to analyse the discrepancies, create a model to predict when and where discrepancies might occur in the future, and develop a streamlined data pipeline to facilitate regular reconciliation efforts.
* **Accompanying file**: Credrails.ipynb
  + Has the code for this assessment

**Understanding The Problem and Building the Data Set**

* We are trying to reconcile the payment records from the payment gateways to the records in the internal system
* But first let us understand the problem. What really is a discrepancy??. In the given document:
  + Transaction T0001 has a slight discrepancy in the Amount.(clear)
  + Transaction T0003 also has a slight discrepancy in the Amount.(clear)
  + Transaction T0006 in gateway\_transactions.csv has a discrepancy, while the one in internal\_transactions.csv has a different amount for the same product and date.
    - while the one in internal transactions has a different amount for the same product and date? Meaning even the internal transactions can have conflicts within itself??
    - Here I am going to assume that because there is no stakeholder to ask this question
  + gateway\_transactions.csv contains an extra transaction T0007.(clear)
  + What about T0005, is it not a discrepancy because it is missing in the gateway\_transactions.csv??
    - This is one of the questions where I would have asked the stakeholder for more clarity because of the ambiguity nature. But since they are not here I am gonna assume its not and fill it with dummy values
  + So final assumption:
    - T001, T003, T006 and T007 are the ones with discrepancies

**Exploratory Data Analysis**



***Results of Exploratory Data Analysis***

* We only have one transaction present in gateway transactions that is not in internal transactions: T007
* Products A1001 and A1002 are the ones affected by discrepancies
* Product A1001 has 2 discrepancies while Product A1002 has only one
* Each and every day recorded a discrepancy
* Products A1001 discrepancies occured when the transaction was the first for the day
* Product A1002 discrepancy occured when the transaction was the second for the day
* Discrepancy amount ranges between -100 and +150. The high range is due to missing internal transactions present in gateway transactions

### **Machine Learning**

* Preprocess the data for modelling (handle missing transactions, mismatches, feature engineering, etc.).
* Develop a model to predict potential discrepancies in future transactions based on identified patterns.
* Evaluate your model's performance using appropriate metrics. (Optional) Consider creating a clustering model to group transactions by their likelihood or type of discrepancy

##### 

##### **Thoughts before Modelling**

* This kind of data doesn't require complex machine learning. Why?
  + the data is minimal (very) - wish we were provided with a more comprehensive dataset with hundreds of rows
  + based on my EDA a simple if statement can be used here and it will have a over 80% accuracy why?
    - we only have two products with discrepancies that is product A1001 and A1002 and a product that is missing in internal transactions definitely is a discrepancy
    - Also from the pattern studied in our EDA , occurence position plays a huge impact in product 1001. Maybe it is by coincidence or maybe not. This is the time I could prefer more clean and quality data to answer this question. Not synthetic data since it will definitely be biased
* All the above conclusions are based on the identified patterns. But this might be misleading. We are only assessing 7 rows in gateway and 6 rows in internal which is a very small dataset to draw conclusion from
* A better way is to generate a synthetic dataset, which may be biased based on the patterns established in the given small dataset
* If machine learning is necessary then:
  + use linear models e.g logistic regression for a simple
  + do not use decision trees , gradient boosting decision trees and neural networks. They need massive amount of data and will easily overfit to small datasets
* So what now?
  + Build a simple if program
  + generating synthetic dataset
  + Build a linear regression model and evaluate it
  + Fit a Kmean clustering algorithm

**Building a simple if model**

### 

### 

### This is a simple rule-based model that makes predictions based on certain conditions in the data. Here’s a breakdown of what it does:

### **Initialize Predictions:** The model starts by adding a new column to the dataframe called ‘prediction’ and sets all its values to 0.

### **Iterate Through Rows:** The model then iterates through each row of the dataframe.

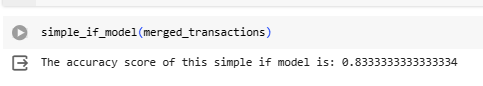
### **Check Product Amounts:** For each row, the model checks if the ‘Product\_ID\_internal’ matches one of the four specified product IDs (‘A1001’, ‘A1002’, ‘A1003’, ‘A1004’) and if the ‘Amount\_internal’ does not match the known correct amount for that product. If both conditions are met, it sets the ‘prediction’ for that row to 1.

### **Check Occurrence Position:** The model also checks if the ‘Product\_ID\_internal’ is ‘A1001’ and the ‘Occurrence\_Position’ is 1, or if the ‘Product\_ID\_internal’ is ‘A1002’ and the ‘Occurrence\_Position’ is 2. If either condition is met, it sets the ‘prediction’ for that row to 1.

### **Calculate Accuracy:** Finally, the model calculates the accuracy of its predictions by comparing them to the actual labels in the ‘Label’ column of the dataframe.

### This model is quite simple and may not perform well on complex tasks, but it can be a good starting point for more sophisticated models. It’s also very interpretable, as you can easily understand why it makes each prediction. However, it’s important to note that this model could be improved by considering more features or using a more sophisticated algorithm. Therefore it is supposed to act as a baseline model

***Results***



It gets 83% accuracy but this result might be misleading as there are only 6 transactions we are basing our results on. If more data was provided by the stakeholders then we would have a more generalizable result

**Building a Logistic Regression Model**

We really do not have enough data for our experiments. I would have asked more data from the stakeholders but since they are not here, I am going to build a synthetic dataset , create a baseline for it using a simple if model then build and evaluate our logistic regression model

***The Data***

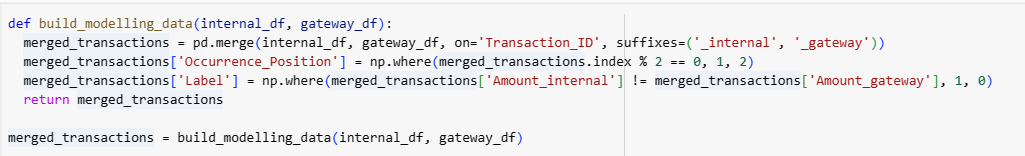
We are going to use a script that generates two dataframes: internal and gateway transactions. Internal contains 100 rows while gateway has 110 rows. The product Id’s A1001 and A1002 are more likely to have discrepancies.

For A1001 , if it’s the first transaction for the day, the amount is 299.89 (a discrepancy) else 299.99 (not a discrepancy).

For A1002, if it’s the second transaction for the day, the amount is 159.40(a discrepancy) otherwise it’s 159.50(not a discrepancy)

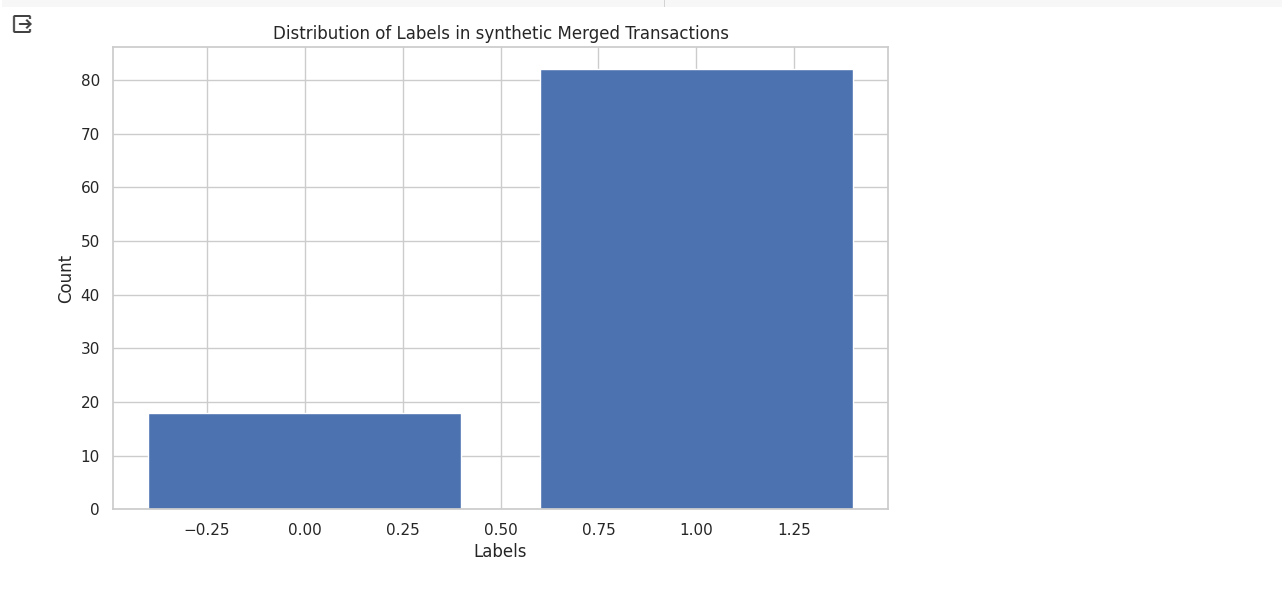
The discrepancies are randomised

***The Training Data***



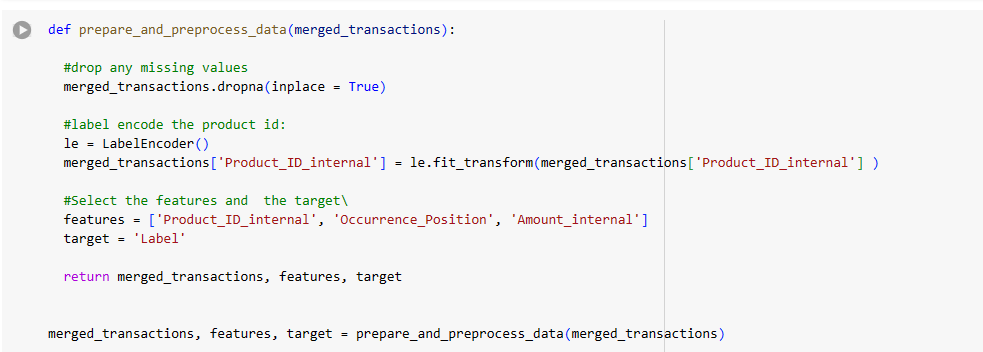
1. **Merge DataFrames**: The function starts by merging two input dataframes, internal\_df and gateway\_df, on the column ‘Transaction\_ID’. The suffixes ‘\_internal’ and ‘\_gateway’ are added to overlapping column names from the two dataframes.
2. **Create Occurrence Position**: It then creates a new column ‘Occurrence\_Position’ in the merged dataframe. This column is set to 1 for rows with an even index and 2 for rows with an odd index.
3. **Create Labels**: The function also creates a new column ‘Label’ in the merged dataframe. This column is set to 1 if the ‘Amount\_internal’ is not equal to the ‘Amount\_gateway’, and 0 otherwise. This could be used as a target variable for a machine learning model, indicating whether there is a discrepancy between the internal and gateway amounts.
4. **Return Result:** Finally, the function returns the merged dataframe with the new ‘Occurrence\_Position’ and ‘Label’ columns.

***The Label Distribution of the Synthetic Dataset***



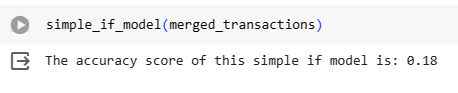
* This distribution is highly imbalanced
* Therefore we have to use a stratified cross validation for our model to generalise better

***Data Preparation and Preprocessing***



1. **Drop Missing Values:** The function starts by dropping any rows in merged\_transactions that contain missing values. The inplace=True argument means that the changes are made directly to the original DataFrame.
2. **Label Encode Product ID:** Next, it creates a LabelEncoder object (from the sklearn.preprocessing module) and uses it to transform the ‘Product\_ID\_internal’ column. Label encoding is a way of converting categorical variables into numerical form, which is easier for machine learning algorithms to work with. The transformed column replaces the original ‘Product\_ID\_internal’ column in the DataFrame.
3. **Select Features and Target:** The function then defines a list of feature column names and a target column name. The features are the columns that the model will use to make predictions, and the target is the column that the model will try to predict.
4. **Return Results:** Finally, the function returns the preprocessed DataFrame, the list of feature names, and the target name.

***Creating a baseline model using our simple if model***



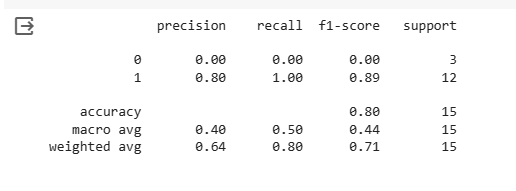
* well our simple if model scores 0.18 in our generated synthetic dataset, therefore not good enough, let’s try to beat it

***Building our Logistic Regression Model and Evaluating it***



1. **Split the Data**: The train\_test\_split function from sklearn.model\_selection is used to split the data into training and testing sets. The features (merged\_transactions[features]) and the target (merged\_transactions[target]) are passed as arguments. The test\_size parameter is set to 0.15, meaning that 15% of the data will be used for the test set. The stratify parameter is set to the target, which means that the split will preserve the proportion of classes (labels) in both the training and testing sets. The random\_state parameter is set to 42 to ensure that the splits are reproducible.
2. **Scale the Data:** The StandardScaler function from sklearn.preprocessing is used to standardize the features by removing the mean and scaling to unit variance. This is done to ensure that all features have the same scale, which can improve the performance of many machine learning algorithms. The scaler is fitted on the training data and then used to transform both the training and testing data.
3. **Fit the Model:** A LogisticRegression model from sklearn.linear\_model is created and fitted on the scaled training data. Logistic regression is a type of classification algorithm that predicts the probability of a binary outcome.
4. **Make Predictions:** The fitted model is used to make predictions on the scaled testing data.
5. **Evaluate the Model:** The classification\_report function from sklearn.metrics is used to print a report showing the main classification metrics for the model on the testing data. This includes precision, recall, f1-score, and support for each class, as well as the overall accuracy of the model.

***Results***



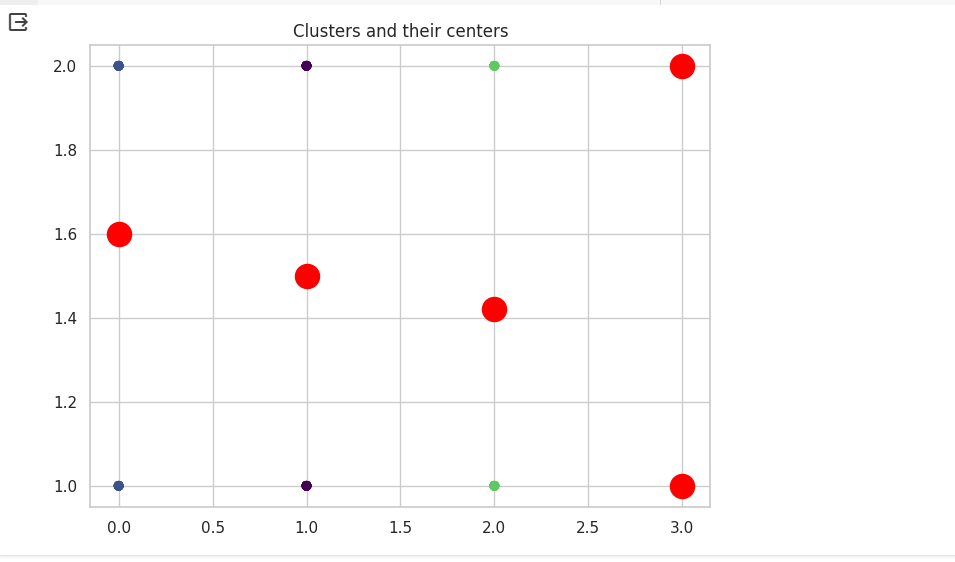
* The accuracy has improved from 0.18 in our simple if function to 0.80 for both logistic regression and random forest but the best metric for such cases is F1 score since our data was highly imbalanced
* The best we can get is 0.44 F1 in both logistic regression and RandomForest
* Well this is synthetic data, We can't base our results on this. Another chance to beg for more quality data

**KMEANS CLUSTERING**

****

1. **Define Features**: The variable X is defined as the features columns of the merged\_transactions DataFrame.
2. **Define Function**: The function perform\_kmeans\_clustering is defined to perform KMeans clustering on a given DataFrame X.
3. **Calculate Silhouette Scores**: Inside the function, it first initialises an empty list silhouette\_scores. It then defines a range of potential cluster numbers K from 2 to 9. For each k in K, it performs KMeans clustering and calculates the silhouette score, which is a measure of how similar an object is to its own cluster compared to other clusters. The silhouette scores are stored in the silhouette\_scores list.
4. **Find Optimal k**: The optimal number of clusters k is the one that gives the highest silhouette score. This is found by finding the index of the maximum score in silhouette\_scores and getting the corresponding k from K.
5. **Perform KMeans Clustering**: KMeans clustering is then performed again using the optimal k. The random\_state parameter is set to 42 to ensure that the results are reproducible.
6. **Visualise Clusters**: The clusters and their centres are visualised using a scatter plot. The data points are coloured according to their cluster labels, and the cluster centres are marked in red.
7. **Call Function**: Finally, the function perform\_kmeans\_clustering is called with X as the argument.

***Results***



* The plot shows three distinct clusters of data points, each represented by a different color (blue, purple, and green). Each cluster groups together data points that are similar to each other based on certain characteristics or features.
* The colored dots at the corners of the plot (blue, purple, and green) represent the centers of their respective clusters. These are calculated as the mean (or sometimes median) of all the data points belonging to a cluster.
* The red dots scattered across the plot represent individual data points. Each data point is assigned to the cluster whose center is closest to it.

**DATE ENGINEERING PIPELINE**

Well Since there is no real data sources and data, I will implement a simple data pipeline in python that mimics the data pipeline

The code illustrated in the notebook in the data engineering pipeline section defines a data pipeline for identifying and reconciling discrepancies in transaction data from two sources: internal and gateway. Here’s a breakdown of what each function does:

1. **data\_pipeline:** This is the main function that orchestrates the entire pipeline. It takes as input the training data, the internal and gateway transactions data, the features to be used for training the model, and the target variable. It trains a logistic regression model, identifies discrepancies between the internal and gateway transactions, reconciles these discrepancies, and returns the corrected transactions.
2. **build\_logistic\_regression\_model**: This function takes as input the data, the features, and the target. It splits the data into training and testing sets, scales the data, fits a logistic regression model, makes predictions, evaluates the model, and returns the trained model.
3. **flag\_discrepancies**: This function takes as input the trained model, the internal and gateway transactions, and the features. It uses the model to predict potential discrepancies in the transactions and returns the flagged transactions.
4. **reconcile\_data**: This function takes as input the internal and gateway transactions and the discrepancies. It corrects the discrepancies in the transactions based on a predefined dictionary of correct product prices and returns the corrected transactions.
5. **build\_modelling\_data**: This function merges the internal and gateway transactions into a single DataFrame, adds an ‘Occurrence\_Position’ column, and adds a ‘Label’ column indicating whether there is a discrepancy in the transaction amounts.

The pipeline is designed to be run periodically (e.g., daily or weekly) to keep the internal and gateway transactions data in sync. The logistic regression model is trained on historical data and then used to flag potential discrepancies in new transactions. These discrepancies are then reconciled based on the known correct product prices. The corrected transactions can then be used for further analysis or reporting.

**ASSESSMENT CONCLUSIONS**

I have struggled with some ambiguity in the data which could have been answered by more data and stakeholder support

In future assessments kindly provide comprehensive datasets that can lead to better EDA and model building. Synthetic datasets are usually filled with biases and might not lead to the best results

Thank you