# **Prototype / MVP Document**

## **1. Introduction**

The MVP we plan to suggest is a comprehensive presentation that outlines the key components of our data processing pipeline, from infrastructure setup to the final dataset. This aims to provide better insights on our approach to managing and processing data to produce a quality dataset. This presentation will demonstrate our codebase encompassing our infrastructure design, code, and the data pipeline. Additionally, the presentation will also include the results of our data pipeline which is a prototype dataset with roughly around 30 features that have been cleaned, synthesised and enriched.

## **2. Prototype Overview**

### **2.1 Prototype concept**

The prototype consists of a detailed presentation that highlights two critical components:

1. **Codebase**: A comprehensive explanation of our GitHub repository, showcasing the infrastructure design, data processing scripts, and automation workflows that constitute the data pipeline.
2. **Final Dataset:** An initial version of the enriched dataset containing around 30 key features essential for effective fraud detection in insurance claims.

### 

### **2.2 Key features**

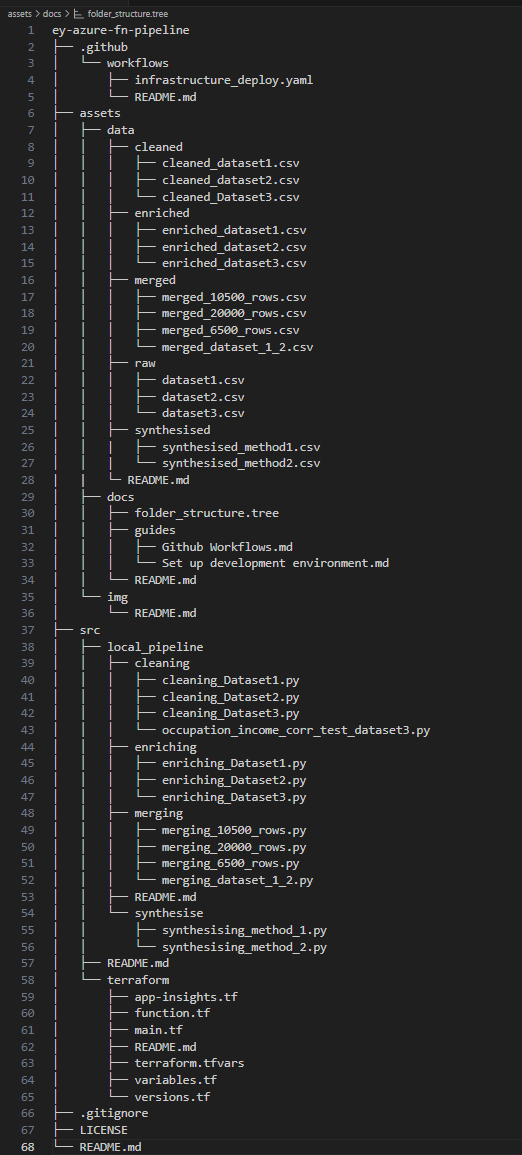
1. **Infrastructure as Code:** Utilising Terraform to define and manage Azure infrastructure, ensuring reproducibility and manageability.
2. **Automated Deployment:** Implementing GitHub Actions scripts for seamless deployment of infrastructure and Azure Functions to the cloud.
3. **Data Loading:** Ensuring efficient and secure data transfer with PowerShell and Azure Data Lake Gen2 REST API to ingest raw data.
4. **Data Processing Pipeline:** The pipeline includes scripts for data cleaning (such as renaming and removing columns), synthesising, enriching, and merging data from multiple sources. At this stage, the code is not yet deployed on Azure. Instead, we are developing and testing it locally, simulating the pipeline that will eventually be deployed on the Azure platform. All scripts are version-controlled on GitHub and are currently being executed manually.
5. **Final Dataset:** A dataset with essential columns that will be used by the AI team to train their initial prototype.

## **3. Current project Codebase & Dataset**

### **3.1 Codebase**

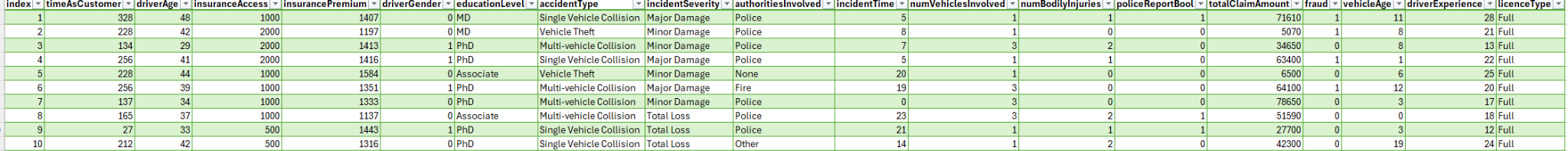
The most up to date version of the codebase can always be found at this link: <https://github.com/AlanDataPortfolio/ey-azure-fn-pipeline>.

The folder structure cannot be shown with a screenshot so here is the best way to visualise the current codebase.



*Figure 1: Current Codebase*

### **3.2 Dataset**

*Figure 2: Final Dataset*

The dataset is in a csv format and has 20, 876 rows of data.

## **Sponsor meeting, feedback and response to feedback**

## **Prototype Feedback Adjustment**

**Analysis, Design + Testing Documentation for DATA SCIENCE projects**

# **1. Feature engineering:**

# *(deciding what you are looking for and how to go about it)*

## 

## **1.1 Trends and ranges**

Identifying the trends and ranges is crucial for feature engineering as The patterns will help to better understand the dataset. For our case, when detecting fraudulent insurance claims, it is important to look at any anomalies in the data. For example, if the ‘totalClaimAmount’ is high and the ‘incidentSeverity’ is minor, it is most likely to be a fraudulent claim. The ‘totalClaimAmount’ column itself can be a good indicator of a fraudulent claim. If the amount is too high and is significantly more than the average claim amount, that claim will be flagged for a fraudulent claim.

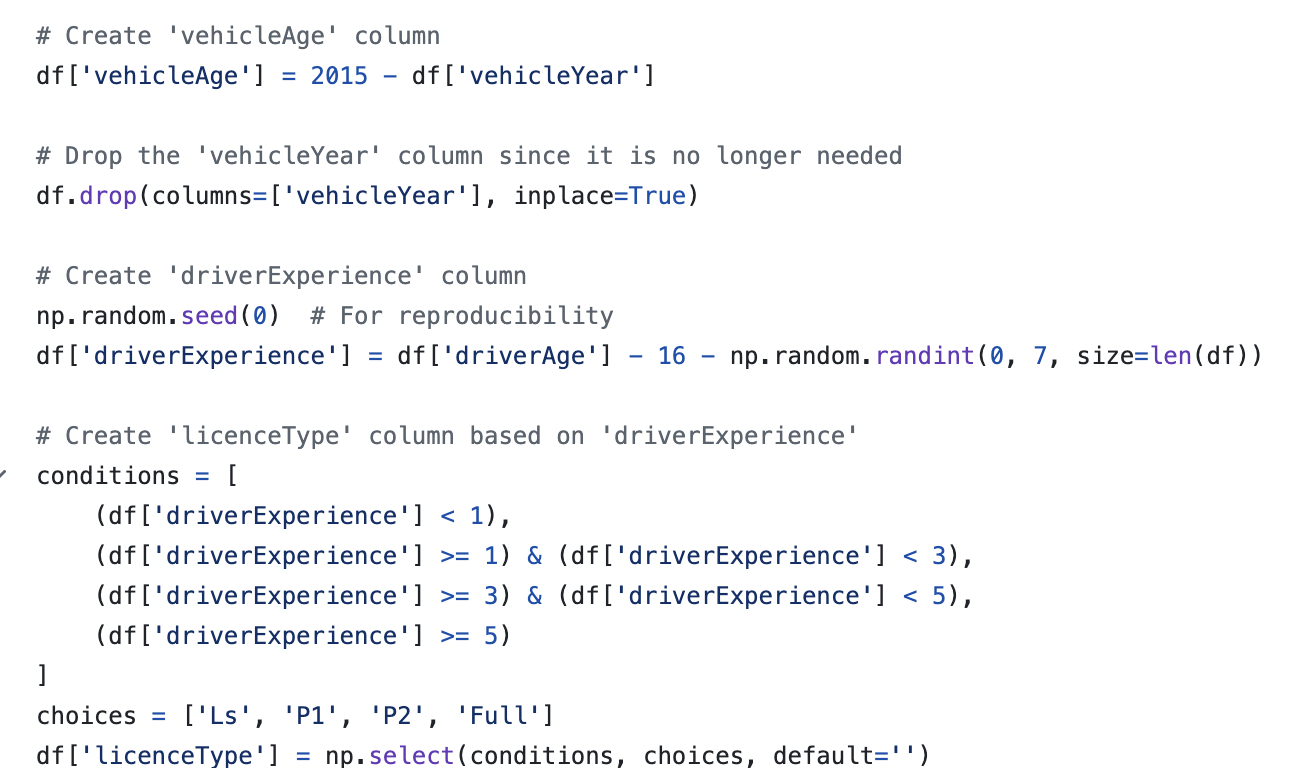
Columns such as ‘driverAge’, ‘driverExperience’, and ‘licenseType’ can give us information about the driver. These columns can also provide insights on the driver behaviour. For example, younger drivers, who are less experienced, are more likely to be driving rough and engage in speeding. Whereas, middle-aged drivers are more experienced and are more likely to drive safely. Therefore, these columns can help to identify the type of driver and help in interpreting whether the claim is fraudulent or not.

## **1.2 Data characteristics and pipeline**

As the dataset is created with the intention to use it to train an AI model that will detect fraudulent claims, fraud indicators will need to be considered. Therefore, we would need a fraud column and columns that can help explain a fraudulent claim. Information relating to the incident and the customer details will also be needed to determine a false claim.

Our data pipeline consists of cleaning, enriching, synthesising and merging to get the combined dataset. Our dataset 1 is the dataset which we are using as a base dataset. In data cleaning, the columns were renamed to match camel casing. All three datasets had the same column names and followed camel casing. For dataset 1, the missing values were handled. For dataset 2, ‘EducationLevel’ was filled in assuming the lowest degree which is ‘High School’. For dataset 3, the missing columns were filled in using various imputation methods such as KNN, Random Forest, GridSearchCV, etc.

In enrichment, ‘True’ ‘False’, ‘YES’ ‘NO’, ‘MALE’ ‘FEMALE’ were all converted to numeric 0’s and 1’s. Here, the categorical variables that could have been converted have been converted to numerical 0’s and 1’s to make it more understandable to the model. Here, the unwanted columns were also dropped in all three datasets. Some columns were generated such as the ‘vehicleAge’ column. As our dataset was ~~based in~~ 2015, we subtracted 2015 from the ‘vehicleYear’ column. The ‘driverExperience’ column was also generated using the ‘driverAge’ column. The ‘driverAge’ is subtracted by 16 as it is the minimum driving age and is further subtracted by a random number from 0 to 7 as we want some variability. The ‘licenseType’ column is also generated based on the ‘driverExperience’ column. If the ‘driverExperience’ is less than 1 year, it was an ‘L’ licence, if the ‘driverExperience’ is more than 1 year and less than 3 years, it was a ‘P1’ licence, if the ‘driverExperience’ is more than 3 years and less than 5 years, it was a ‘P2’ licence and finally if the ‘driverExperience’ is more than 5 years, it was a ‘Full’ licence. The code to generate these columns is provided below:

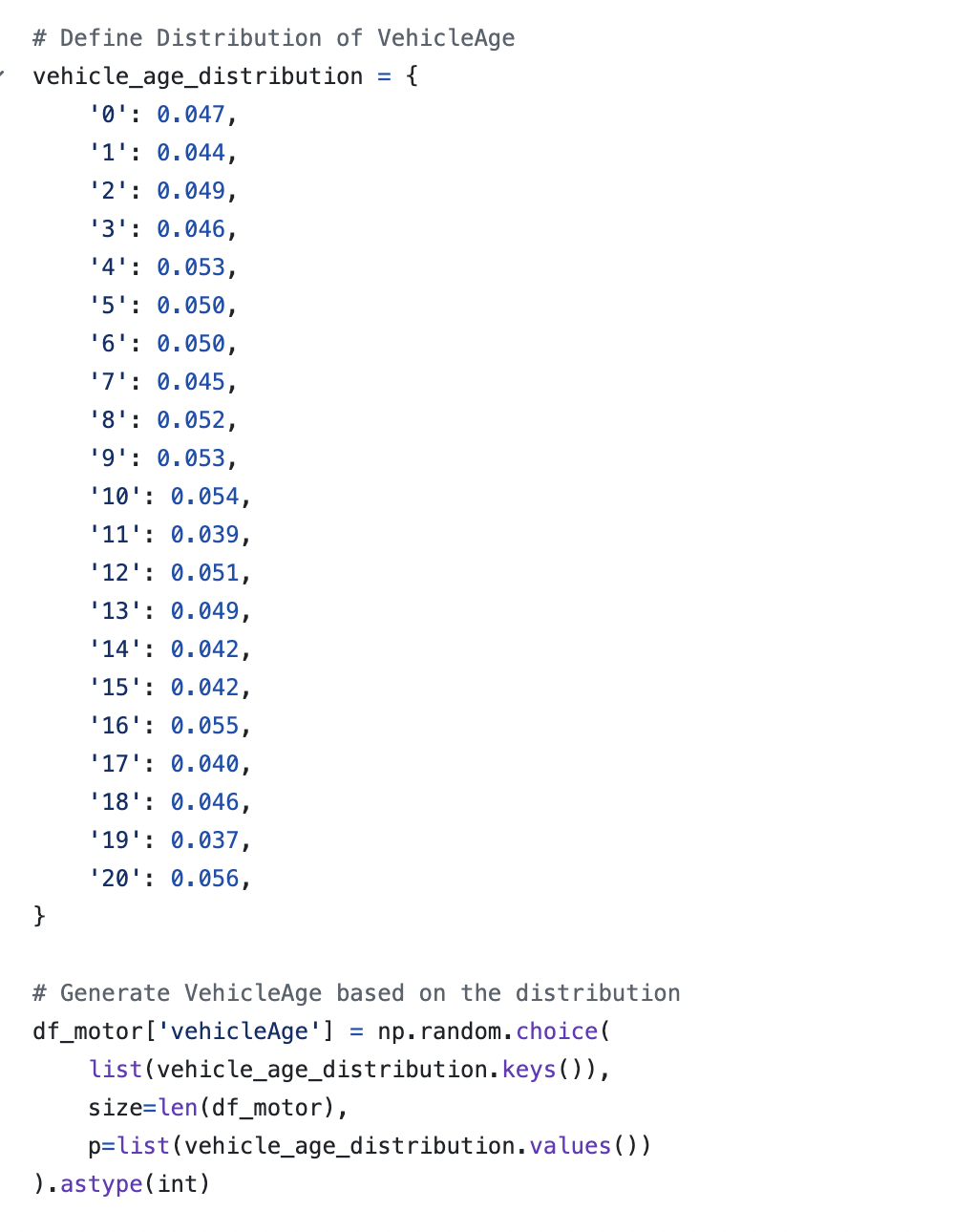


*Figure 3: Code to generate licence type*

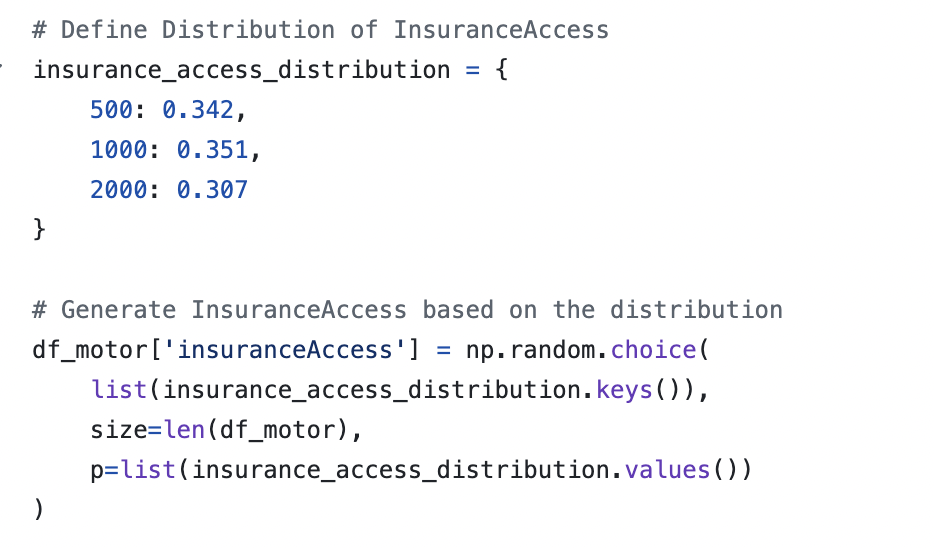
For dataset 2, a few columns had to be generated based on dataset 1 as dataset 2 did not have those columns. The ‘AccidentType’ column had to be enriched based on the distribution of that column in dataset 1. Using that distribution, ‘AccidentType’ was then randomised and generated. For ‘NumVehiclesInvolved’, the same method was used to generate it. If the ‘AccidentType’ was ‘Multi-Vehicle Collision’, the column was randomly generated based on the normal distribution of ‘Multi-Vehicle Collision’ in dataset 1. Otherwise, the column returned 1 as the rest of the accident types were involving a single vehicle. The ‘VehicleAge’ column was also generated based on the distribution of it in dataset 1. The weights were assigned to the different ages and were randomly generated once again. A similar process was also used to generate the ‘InsuranceAccess’ column. The code for the columns generated is provided below:



*Figure 4: Code to generate the Accident Type and Number of vehicles involved column*



*Figure 5: Code to generate the Vehicle Age column*



*Figure 6: Code to generate the Insurance Access column*

In all of the datasets, Fraud was converted to 1 and Non-Fraud was converted to 0. The same processes are used to generate the features in dataset 3. The 2 datasets were then merged and based on the merged dataset, 2 synthesization methods were used to generate 8000 rows of data.

## **1.3 Feature definition**

For ‘driverExperience’, we can name a feature called ‘experienceCategory’ which will classify the driver experience into different categories. This will help to determine the information about the experience of the driver. Ranges for this feature can be: <3 years = ‘Low Experience’, 3-6 years = ‘Moderate Experience’, >6 years = ‘High Experience’.

The same is partially done for ‘licenseType’. A feature called ‘licenseCategory’ can be named which will classify the licence type of the driver. In our case, if the ‘driverExperience’ was less than 1 year, then the licence was a Learners or ‘L’. If the ‘driverExperience’ was more than 1 year and less than 3, then the licence was a Provisional P1 or ‘P1’. If the ‘driverExperience’ was more than 3 years and less than 5, then the licence was a Provisional P2 or ‘P2’. If the ‘driverExperience’ was more than 5 years, then the licence was a Full License or ‘Full’.

The ages for the driver can also be categorised into a feature called ‘driverAgeCategory’. The ranges for that category can be: <25 years = ‘Young’, 25-50 years = ‘Middle-Aged’, >50 years = ‘Old’.

The same can be done for ‘vehicleAge’. A new feature called ‘vehicleAgeCategory’ can be created. The ranges for that category can be: <3 years = ‘New’, 3-10 years = ‘Moderate’, >10 years = ‘Old’.

A feature called ‘timeCategory’ can be created to describe the time the incident took place. The range for that category can be, 0-6 = ‘Late Night’, >6-12 = ‘Morning’, >12-18 = ‘Afternoon’, >18-24 = ‘Night’.

Similarly, a lot of features can be created following this process. However, these are some of the important ones.

# **2. Solution Architecture:**

# *(choice of macro architecture / pipeline)*

### 

## **2.1 Current architecture**

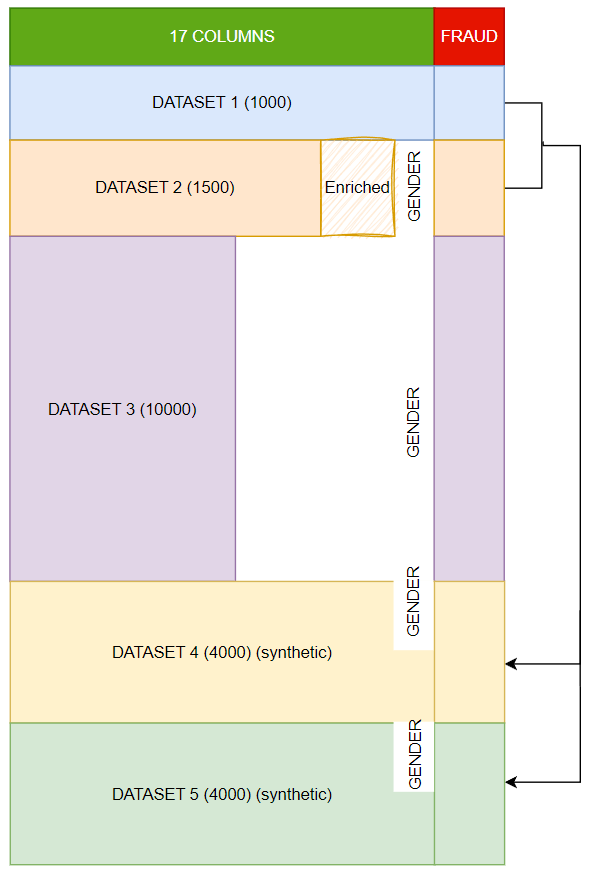
### 

*Figure 7: Current Architecture Diagram*

There are 3 raw data sources from Kaggle in csv format. Since these 3 sources are from different providers, they do not contain the same column and differ in quality. Naturally, the first task is to clean them which include renaming a similar column to a pre-defined name and filling in missing values. There are quite a few imputation methods that we have developed, some are simple such as using median and mean for numerical data, and mode for some categorical data. We also used algorithms like Support Vector Regression or Random Forest Classifier to impute other columns.

Enriching is a data engineering process that allows us to create more meaningful columns, however, we are also short of data. The data collected is not enough for the AI team to train their model so we have to synthesise more data. Between enriching and synthesising, we chose to enrich the data first because we would not want to create other columns from synthesised data as that would significantly reduce the reliability of our final dataset. Rather, we want to first enrich the data, then synthesise because our synthesising method might be able to learn the features and pattern of the enriched data and will give a better reliable dataset.

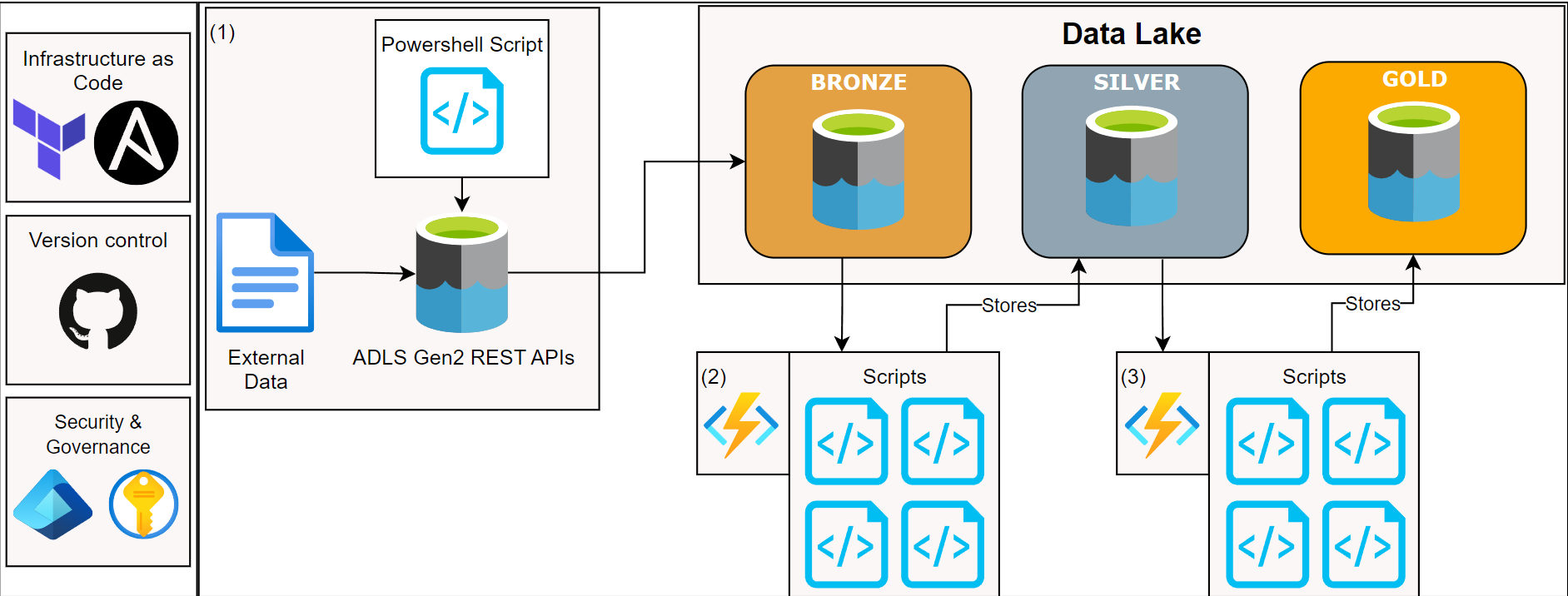
Dataset 3 has many missing columns from what we need to be able to enrich it. So we only enrich dataset 1 and 2. Which is then merged and synthesised using 2 different methods.



*Figure 8: Final Dataset Diagram*

The final dataset is a combination of enriched dataset 1 and 2, synthesised dataset 4 and 5 from 2 methods and dataset 3.

## **2.2 Cloud architecture (not yet deployed)**



*Figure 9: Cloud Architecture Diagram*

The order of the pipeline is similar to our current architecture. We have implemented the pipeline, though not automated, is able to run on our local machine. The only difference is that we will use Terraform as our Infrastructure as Code. Moreover, all data will be stored in Azure Data Lake Gen2 which will be partitioned into 3 layers. The Bronze layer will only hold our raw data. The Silver layer will hold our data in between its transformation, for example, after being cleaned, after being merged, after being synthesised and even after being merged. Then the chosen dataset will be pushed to the gold layer. Note, these layers are an abstraction which allows us to easily manage our data process. We also make use of Azure Functions which also run python scripts and it will all be automated whenever we make changes to our scripts.

# **3. Algorithms / models methods:**

# *(detail what is in each part of the solution architecture, including models used and initial conditions / config settings)*

Cleaning the dataset is a crucial step in the project as the quality of the dataset directly affects the accuracy and reliability of the outputs from the AI model. This project with NRMA involves collaborating with Team 13 to create an AI solution to detect fraudulent claims. Our team is responsible for delivering a dataset to Team 13 that is clean, complete and comprehensive so it can be used to train the AI model to make accurate predictions and generalise the data properly generating reliable predictions. To do this the raw dataset contains missing and inconsistent values using machine learning models to impute and standardise the values.

The main method of addressing the missing values was predictive supervised models due to their ability to make accurate predictions in relation to patterns in the data. Compared to other imputation methods like mean or mode, predictive supervised models are able to use the underlying relationships between features resulting in the imputed values being more context-aware. As the dataset contain both numerical and categorical data appropriate models were selected to handle each type of data

* **Numerical data:** we decided to use K-Nearest Neighbors (KNN) and Random Forest Regressor to predict missing numerical values. KNN is able to capture local relationships between data points, thus effective for data that can be predicted based on similar observations in the dataset. While Random Forest Regressor is able to capture complex relationships in the dataset and is robust against overfitting, this is effective for data that is more prone to overfitting and has more complex relationships.
* **Categorical data:** we decided to use Random Forest Classifier to predict the missing categorical values. While operating similarly to Random Forest Regressor, Classifier is used to assign labels to an inputs providing a categorical output.

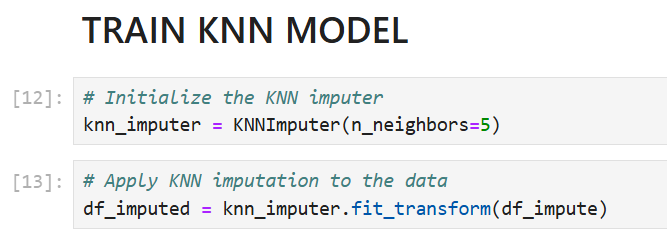
Through the combination of these predictive models, the dataset was clean so that the integrity and patterns of the original dataset is preserved and enables accurate training for the AI model.

Missing values:

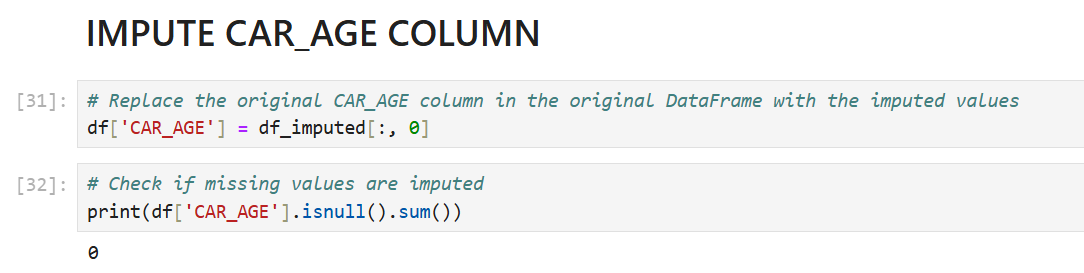
* Dataset 1: police\_report\_available, authorities\_contacted
* Dataset 2: address\_line2 (8505), city (54), customer\_education\_level (529), authorities\_contacted (1945), vendor\_id (3245)
* Dataset 3: age (7), yoj (548), income (570), home\_val (575), occupation (665), car\_age (639)

## **3.1. KNN Model**

KNN was used to impute the missing numerical values in the dataset. This is done by finding similar data on other columns and imputing the missing value based on the average of its nearest neighbour. KNN was used on columns like age, income, home\_val and car\_age in dataset 3 to impute the missing values from that column.



*Figure 10: Training the KNN model*



*Figure 11: Impute values into the ‘CAR\_AGE’ column*

### **3.1.1. Configuration Settings**

* n\_neighbours = 5: balances computations accuracy and complexity

### **3.1.2. Reasons for Model selection**

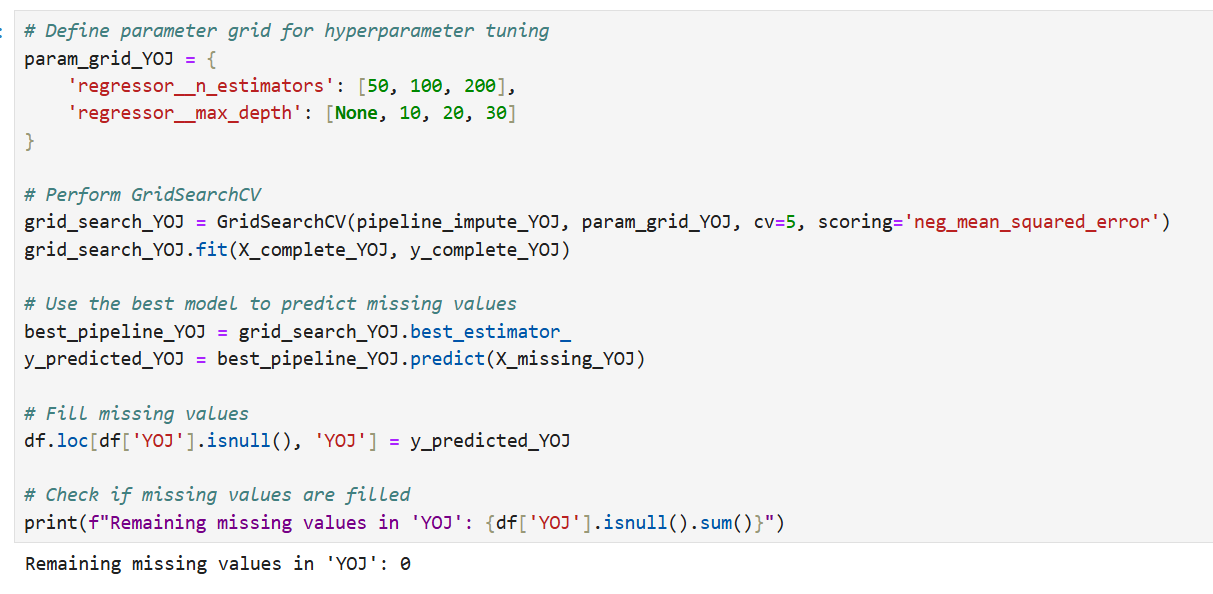
* KNN is highly intuitive and simple to implement. The idea of using neighbors for predictions is easy to understand and visualize, making it a user-friendly model, especially for imputation tasks.
* KNN is versatile and works effectively with both numerical and categorical data, making it suitable for a wide range of imputation tasks across different types of datasets.
* KNN is non-parametric, meaning it does not make assumptions about the underlying data distribution, making it flexible in handling complex and varied data patterns.
* For imputation, KNN easily fills in missing values based on the average or most common value of similar data points (neighbors), without the need for complex pre-processing or assumptions about data structure.
* While KNN can be computationally intensive on very large datasets due to its reliance on distance calculations, its simplicity makes it efficient on moderately-sized datasets.
* KNN inherently captures non-linear relationships since it relies on proximity in feature space rather than any functional mapping, which gives it an edge over linear models for data with non-linear dependencies.
* Key hyperparameters like the number of neighbors (k) can be easily tuned to optimize model performance, improving accuracy in various settings.

## **3.2. Random Forest Models**

* **Random Forest Regressor:** is used to predict continuous values in regression problems by combining the predictions of numerous decision trees to improve accuracy and prevent overfitting from occuring. This is mainly used on numerical columns like yoj that do not have a high accuracy with using KNN to impute due to issues such as overfitting or non-linear relationships.
* **Random Forest Classifier:** is also used to handle missing values in a similar manner to Random Forest Regressor however, is used in classification tasks in a manner that handles class imbalance and non-linear relationships.

### **3.2.1. Configuration Settings**

* GridSearchCV: hyperparameter tuning used to improve accuracy
* RandomisedSearchCV: hyperparameter tuning used to improve accuracy
* n\_estimators = 100: number of trees was 100, which provided a good balance between both computation and performance time
* Max\_depth = none: allows the tree to fully grow increasing the ability of the model to capture complex data patterns



*Figure 12: Using GridSearchCV and Random Forest Regressor to impute YOJ column*

****

*Figure 13: Using RandomisedSearchCV and Random Forest Classfier to impute Occupation column*

### **3.2.2. Reasons for Model selection**

* Though more complex than KNN, Random Forest is still easy to interpret. Each tree in the forest is a simple decision tree, and the final prediction is an average (for regression) or a majority vote (for classification), which is relatively straightforward.
* Random Forest handles both numerical and categorical data effortlessly. It is particularly useful in datasets with mixed types of features, as it can split on categorical variables without needing to encode them.
* Like KNN, Random Forest is non-parametric and does not assume any underlying distribution in the data. This flexibility allows it to model complex, non-linear relationships between features and the target variable effectively.
* Random Forest can handle missing data internally by building multiple decision trees that rely on different feature subsets. It does not require pre-imputation of missing values, which simplifies the preprocessing pipeline.
* Random Forest is highly scalable and can handle large datasets efficiently. It works well with both high-dimensional data and large sample sizes due to its ensemble approach, which reduces variance and prevents overfitting.
* Random Forest is excellent for capturing non-linear relationships in data. Each decision tree can split non-linearly at different thresholds, making Random Forest a robust model for tasks where non-linear interactions are important.
* Random Forest offers several hyperparameters (e.g., number of trees, maximum depth, minimum samples per split) that can be tuned to optimize performance. This flexibility allows it to adjust to various datasets and improve predictive accuracy.
* For classification tasks, Random Forest can deal with class imbalances effectively through techniques like adjusting class weights or oversampling the minority class during training. This makes it a strong choice when the dataset has skewed class distributions.
* Since Random Forest is an ensemble of decision trees, it significantly reduces the risk of overfitting compared to individual decision trees. It builds trees on random subsets of features and data, which improves generalization.

## **3.3. Why were these models chosen over other approaches?**

3.3.1. Predictive Models

* Linear regression: is a model that predicts missing values through linear relationships between target and independent variables, however, are not able to capture complex interactions in the data.
* Decision Trees: however, are prone to overfitting on small datasets like dataset 1 and dataset 2 and may not perform effectively with non-linear relationships.

3.3.2. Supervised Learning Models

* Neural networks: are able to predict missing values through the analysis of complex patterns in the dataset however, are computationally expensive and require significant tuning which, furthermore, they require large datasets for accurate training.
* Gradient Boosting: However, this is also computationally expensive and require significant tuning while also having a slower performance.

3.3.3. Unsupervised Learning Models

* K-means: has the potential to group similar data points and impute missing values based on the mode or mean of that cluster however, it method mainly focuses on clustering hence not ideal for getting accurate prediction of missing values
* Principal Component Analysis: can be used to reduce dimensionality and use the dataset’s principal components to fill in missing values however, this method will result in simple imputations failing to capture patterns in the data.

3.3.4. Classifiers

* Logistic Regression: can be used to predict missing categorical values through modelling binary outcomes however, due to this it is not suited for imputing missing values.
* Naive Bayes: uses probability distributions to inpute missing values

3.3.5. Chosen Models

* KNN: we chose to use KNN for our numerical columns as local relationships between columns were a key part of imputing missing values. Furthermore, as the datasets were smaller in nature there were not any complex patterns to be analysed so using KNN provided faster performance and was less time consuming to meet our deadlines.
* Random Forest: we chose Random Forest models for both categorical and numerical columns as it is able to capture complex relationships leading to better accuracy and suitable to diverse datasets. Furthermore, the ability to handle overfitting and class imbalance allowe for better imputed values and a dataset with better quality values for training,

# **4. Detailed Data descriptions**

*Data being used, data being generated, data being stored, as well as any summaries and/or reports.*

## **4.1 Original Raw Datasets**

The raw datasets used in this project consist of three datasets sourced from Kaggle, each containing historical data on automobile insurance claims. These datasets were selected primarily because they include a column that explicitly indicates whether a claim was fraudulent, which is essential for training machine learning and AI models for fraud detection.

The original datasets are in a raw, unclean state, with many missing or non-standardized values. A thorough cleaning and enrichment process was applied to prepare them for analysis.

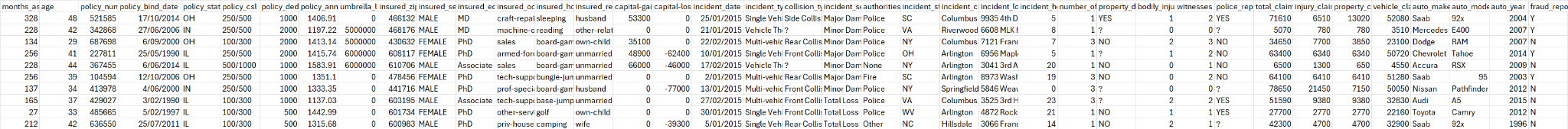
* Dataset 1: This dataset, containing 1,000 rows, is well-structured and widely available across various websites. It is frequently used in insurance fraud analysis due to its quality and completeness. The dataset was sourced from Kaggle, as it is an open-source platform. The dataset is hosted on GitHub at: [GitHub Link](https://github.com/mwitiderrick/insurancedata/blob/master/insurance_claims.csv), and the Kaggle source is available here: [Kaggle Link](https://www.kaggle.com/datasets/sumansuhag/insurance-dataset-csv).
* Dataset 2: This dataset is also quite well-structured and originally contains 10,000 rows but includes data on various types of fraud, not limited to automobile insurance. After filtering for automobile fraud, approximately 1,500 relevant rows were extracted. This dataset offers a broader range of fraud types but required focused extraction for the target use case of automobile insurance fraud detection. The dataset can be accessed on Kaggle: [Kaggle Link](https://www.kaggle.com/datasets/mastmustu/insurance-claims-fraud-data?select=insurance_data.csv).
* Dataset 3: This dataset, also containing 10,000 rows, is the most challenging due to its high level of missing, blank, and non-standardized values. It required significant cleaning efforts to make it usable for analysis. Despite its issues, it provides valuable data for automobile insurance fraud detection once processed. The dataset can be found on Kaggle: [Kaggle Link](https://www.kaggle.com/datasets/xiaomengsun/car-insurance-claim-data).

These datasets form the foundation for the final dataset to be used by the fraud detection model. Providing this historical claims data allows the machine learning algorithms to learn patterns and anomalies indicative of fraudulent claims. The presence of a fraud indicator in each dataset ensures that they are particularly well-suited for supervised learning in fraud detection.

## 

## **4.2. Columns in Raw Datasets**

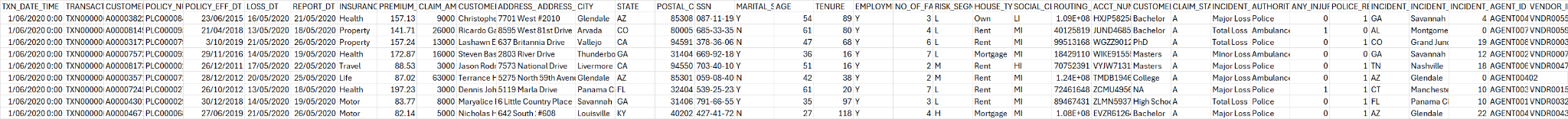
### **Raw Dataset 1:**



*Figure 14: Raw Dataset 1*

1. months\_as\_customer: Number of months the individual has been a customer (Numerical).
2. age: Age of the customer (Numerical).
3. policy\_number: Unique identifier for the insurance policy (Categorical).
4. policy\_bind\_date: The date when the policy was created/bound (Date).
5. policy\_state: The state where the policy was issued (Categorical).
6. policy\_csl: Combined Single Limit for the policy (Categorical).
7. policy\_deductable: The amount of deductible in the policy (Numerical).
8. policy\_annual\_premium: The annual premium paid for the insurance policy (Numerical).
9. umbrella\_limit: The umbrella limit in the policy (Numerical).
10. insured\_zip: Zip code of the insured person (Categorical).
11. insured\_sex: Gender of the insured person (Categorical).
12. insured\_education\_level: Education level of the insured person (Categorical).
13. insured\_occupation: Occupation of the insured person (Categorical).
14. insured\_hobbies: Hobbies of the insured person (Categorical).
15. insured\_relationship: Relationship status of the insured person (Categorical).
16. capital-gains: Capital gains of the insured person (Numerical).
17. capital-loss: Capital loss of the insured person (Numerical).
18. incident\_date: Date of the incident (Date).
19. incident\_type: Type of incident (Categorical).
20. collision\_type: Type of collision (Categorical).
21. incident\_severity: Severity of the incident (Categorical).
22. authorities\_contacted: Whether authorities were contacted (Categorical).
23. incident\_state: State where the incident occurred (Categorical).
24. incident\_city: City where the incident occurred (Categorical).
25. incident\_location: Specific location of the incident (Categorical).
26. incident\_hour\_of\_the\_day: Hour of the day when the incident occurred (Numerical).
27. number\_of\_vehicles\_involved: Number of vehicles involved in the incident (Numerical).
28. property\_damage: Whether property damage occurred (Categorical).
29. bodily\_injuries: Number of bodily injuries (Numerical).
30. witnesses: Number of witnesses to the incident (Numerical).
31. police\_report\_available: Whether a police report is available (Boolean).
32. total\_claim\_amount: The total amount claimed (Numerical).
33. injury\_claim: Claim amount for injuries (Numerical).
34. property\_claim: Claim amount for property damage (Numerical).
35. vehicle\_claim: Claim amount for vehicle damage (Numerical).
36. auto\_make: Make of the vehicle involved (Categorical).
37. auto\_model: Model of the vehicle involved (Categorical).
38. auto\_year: Year of the vehicle involved (Numerical).
39. fraud\_reported: Whether fraud was reported (Boolean).

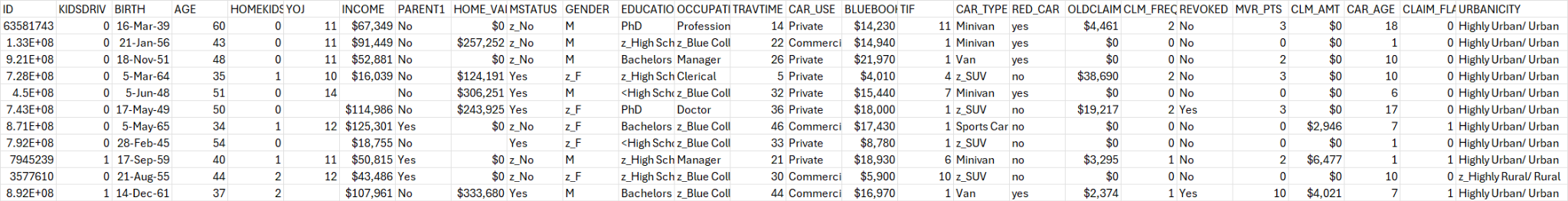
### **Raw Dataset 2:**



*Figure 15: Raw Dataset 2*

1. TXN\_DATE\_TIME: The date and time of the transaction (Date/Time).
2. TRANSACTION\_ID: Unique identifier for the transaction (Categorical).
3. CUSTOMER\_ID: Unique identifier for the customer (Categorical).
4. POLICY\_NUMBER: Unique identifier for the policy (Categorical).
5. POLICY\_EFF\_DT: The policy effective date (Date).
6. LOSS\_DT: Date when the loss occurred (Date).
7. REPORT\_DT: Date when the incident was reported (Date).
8. INSURANCE\_TYPE: Type of insurance (Categorical).
9. PREMIUM\_AMOUNT: Premium amount for the insurance policy (Numerical).
10. CLAIM\_AMOUNT: Amount claimed (Numerical).
11. CUSTOMER\_NAME: Name of the customer (Categorical).
12. ADDRESS\_LINE1: Address line 1 of the customer (Categorical).
13. ADDRESS\_LINE2: Address line 2 of the customer (Categorical).
14. CITY: City of the customer (Categorical).
15. STATE: State of the customer (Categorical).
16. POSTAL\_CODE: Postal code of the customer (Categorical).
17. SSN: Social Security Number (Categorical).
18. MARITAL\_STATUS: Marital status of the customer (Categorical).
19. AGE: Age of the customer (Numerical).
20. TENURE: Number of years the customer has been with the insurance company (Numerical).
21. EMPLOYMENT\_STATUS: Employment status of the customer (Categorical).
22. NO\_OF\_FAMILY\_MEMBERS: Number of family members (Numerical).
23. RISK\_SEGMENTATION: Risk category assigned to the customer (Categorical).
24. HOUSE\_TYPE: Type of house the customer owns or rents (Categorical).
25. SOCIAL\_CLASS: Social class of the customer (Categorical).
26. ROUTING\_NUMBER: Customer’s bank routing number (Categorical).
27. ACCT\_NUMBER: Customer’s bank account number (Categorical).
28. CUSTOMER\_EDUCATION\_LEVEL: Education level of the customer (Categorical).
29. CLAIM\_STATUS: Status of the claim (Categorical).
30. INCIDENT\_SEVERITY: Severity of the incident (Categorical).
31. AUTHORITY\_CONTACTED: Whether the authorities were contacted (Boolean).
32. ANY\_INJURY: Whether any injury was reported (Boolean).
33. POLICE\_REPORT\_AVAILABLE: Whether a police report is available (Boolean).
34. INCIDENT\_STATE: State where the incident occurred (Categorical).
35. INCIDENT\_CITY: City where the incident occurred (Categorical).
36. INCIDENT\_HOUR\_OF\_THE\_DAY: Hour of the day when the incident occurred (Numerical).
37. AGENT\_ID: Unique identifier for the insurance agent (Categorical).
38. VENDOR\_ID: Unique identifier for the vendor (Categorical).

### **Raw Dataset 3:**



*Figure 16: Raw Dataset 3*

1. ID: Unique identifier for the customer (Categorical).
2. KIDSDRIV: Number of kids driving in the household (Numerical).
3. BIRTH: Birth date of the customer (Date).
4. AGE: Age of the customer (Numerical).
5. HOMEKIDS: Number of kids living in the household (Numerical).
6. YOJ: Years on the job (Numerical).
7. INCOME: Income of the customer (Numerical, Currency).
8. PARENT1: Whether the customer is a single parent (Categorical).
9. HOME\_VAL: Value of the home (Numerical, Currency).
10. MSTATUS: Marital status (Categorical).
11. GENDER: Gender of the customer (Categorical).
12. EDUCATION: Education level of the customer (Categorical).
13. OCCUPATION: Occupation of the customer (Categorical).
14. TRAVTIME: Travel time to work (Numerical).
15. CAR\_USE: Whether the car is used for private or commercial purposes (Categorical).
16. BLUEBOOK: Value of the car (Numerical, Currency).
17. TIF: Time in force of the insurance policy (Numerical).
18. CAR\_TYPE: Type of car (Categorical).
19. RED\_CAR: Whether the car is red (Boolean).
20. OLDCLAIM: Amount claimed in prior incidents (Numerical, Currency).
21. CLM\_FREQ: Frequency of claims (Numerical).
22. REVOKED: Whether the driver's licence has been revoked (Boolean).
23. MVR\_PTS: Motor Vehicle Record points (Numerical).
24. CLM\_AMT: Amount claimed in the current incident (Numerical, Currency).
25. CAR\_AGE: Age of the car (Numerical).
26. CLAIM\_FLAG: Whether a claim was filed (Boolean).
27. URBANICITY: Urban/rural classification (Categorical).

## **4.3. Cleaning & Enriching**

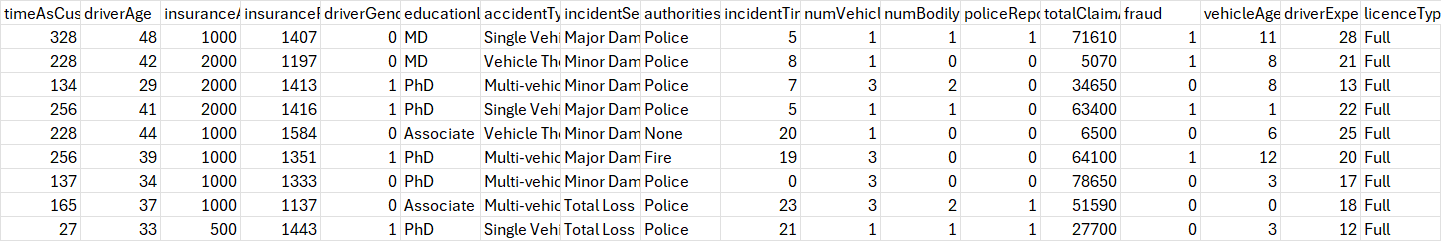
### **Dataset 1 & Dataset 2**

After the cleaning and enrichment process, Dataset 1 (1,000 rows) and Dataset 2 (approximately 1,500 rows) were refined by selecting 16 key columns and generating 2 additional columns using the original data. These newly created columns, Driver Experience and Licence Type, were calculated using straightforward mathematical formulas.

* Driver Experience: This column is derived from the driver's age, using the formula:  
  driver experience = driver age - 16 + R,  
  where R is a random variable between 0 and 6. The subtraction of 16 accounts for the minimum age to obtain a licence, while the random variable represents the variability in the time drivers might wait before getting their licence.
* Licence Type: The licence type is determined by a simple set of conditional statements based on the driver’s experience:
  + Less than 1 year of experience: Ls
  + 1 to 2 years of experience: P1
  + 2 to 4 years of experience: P2
  + More than 4 years of experience: Full

For Dataset 2, no gender data was available, so the driverGender column remains blank for that dataset, ensuring consistency without introducing any erroneous or bias inducing data.

### **Cleaned & Enriched Dataset 1:**

*Figure 17: Cleaned and Enriched Dataset 1*

1. timeAsCustomer: Number of months the individual has been a customer (Numerical).
2. driverAge: Age of the driver (Numerical).
3. insuranceAccess: The level of access to insurance (Numerical).
4. insurancePremium: The premium amount for the insurance policy (Numerical).
5. driverGender: Gender of the driver (Boolean: 0 = Male, 1 = Female).
6. educationLevel: Education level of the driver (Categorical).
7. accidentType: Type of accident (Categorical).
8. incidentSeverity: Severity of the incident (Categorical).
9. authoritiesInvolved: Whether authorities were involved (Categorical).
10. incidentTime: Time of the incident (Numerical).
11. numVehiclesInvolved: Number of vehicles involved in the accident (Numerical).
12. numBodilyInjuries: Number of bodily injuries (Numerical).
13. policeReportBool: Whether a police report was available (Boolean: 0 = No, 1 = Yes).
14. totalClaimAmount: The total claim amount (Numerical).
15. fraud: Whether the claim was fraudulent (Boolean: 0 = No, 1 = Yes).
16. vehicleAge: Age of the vehicle (Numerical).
17. driverExperience: Number of years of driving experience (Numerical).
18. licenceType: Type of licence held by the driver (Categorical).

### **Cleaned & Enriched Dataset 2:**

*Figure 18: Cleaned and Enriched Dataset 2*

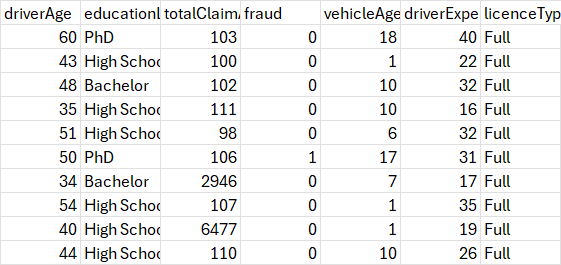
1. timeAsCustomer: Number of months the individual has been a customer (Numerical).
2. driverAge: Age of the driver (Numerical).
3. insuranceAccess: The level of access to insurance (Numerical).
4. insurancePremium: The premium amount for the insurance policy (Numerical).
5. driverGender: Gender of the driver (Boolean: 0 = Male, 1 = Female, empty & not populated).
6. educationLevel: Education level of the driver (Categorical).
7. accidentType: Type of accident (Categorical).
8. incidentSeverity: Severity of the incident (Categorical).
9. authoritiesInvolved: Whether authorities were involved (Categorical).
10. incidentTime: Time of the incident (Numerical).
11. numVehiclesInvolved: Number of vehicles involved in the accident (Numerical).
12. numBodilyInjuries: Number of bodily injuries (Numerical).
13. policeReportBool: Whether a police report was available (Boolean: 0 = No, 1 = Yes).
14. totalClaimAmount: The total claim amount (Numerical).
15. fraud: Whether the claim was fraudulent (Boolean: 0 = No, 1 = Yes).
16. vehicleAge: Age of the vehicle (Numerical).
17. driverExperience: Number of years of driving experience (Numerical).
18. licenceType: Type of licence held by the driver (Categorical).

### **Dataset 3**

Dataset 3, which is the largest dataset (10000) rows, was missing many of the columns present in the other datasets. Instead of synthesising all the columns it was missing, it was decided to just get the columns that match those and leave the rest empty. This was to prevent such a large dataset becoming biassed and skewed from data from smaller datasets. It increased variation and variety of data, and would assist in helping the AI model learn what to do if not all rows of data were provided.

Thus, Dataset 3’s selected columns are (included LicenceType and DriverExperience which were enriched using same formula as previous datasets):

### **Cleaned & Enriched Dataset 3:**



*Figure 19: Cleaned and Enriched Dataset 3*

1. driverAge: Age of the driver (Numerical).
2. educationLevel: Education level of the driver (Categorical).
3. totalClaimAmount: The total claim amount (Numerical).
4. fraud: Whether the claim was fraudulent (Boolean: 0 = No, 1 = Yes).
5. vehicleAge: Age of the vehicle (Numerical).
6. driverExperience: Number of years of driving experience (Numerical).
7. licenceType: Type of licence held by the driver (Categorical).

**4.4. Generating Data:**

The dataset consists of 4,000 rows of synthetic data generated using Method 1 and an additional 4,000 rows generated using Method 2. Method 1 involved using statistical remapping and random sampling techniques to synthesise data based on trends in the original dataset. Method 2 leveraged advanced machine learning techniques, such as Generative Adversarial Networks (GANs), to create more realistic synthetic data by modelling the underlying patterns of the original dataset.

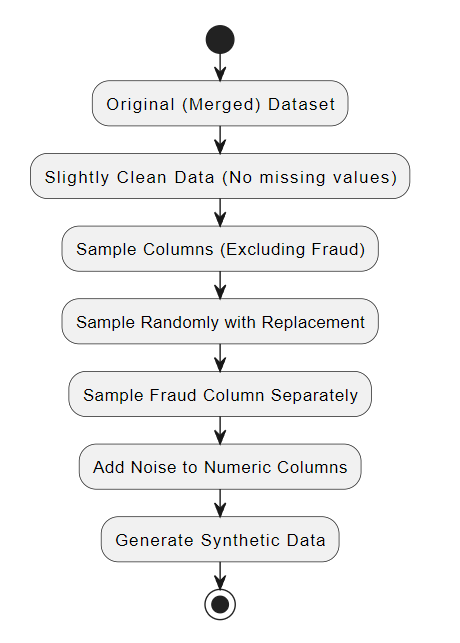
This dual approach allowed us to create a more diverse and comprehensive dataset for analysis and machine learning tasks.

### **Method 1:**

Synthetic Data Method 1 generates additional data by sampling from the original dataset, maintaining the same distribution and characteristics of the real data. This method is used to expand the dataset, especially when more data is needed for training machine learning models, but without introducing random or irrelevant data.

The synthetic data is generated by first cleaning any missing or blank values (such as replacing blanks in authoritiesInvolved with "none"). Then, we sample the original dataset to create new rows. The important point here is that the fraud column is kept reliable and unchanged. While other columns are sampled and noise is added to numeric fields to make the data varied, the fraud column is handled separately to ensure that its distribution remains consistent with the original data.

For the fraud column, we directly sample from the original fraud values and then add them to the synthetic data without any modification or randomization. This ensures that the synthetic data reflects the same patterns of fraudulent and non-fraudulent cases found in the original dataset, maintaining its integrity and reliability.

By preserving the fraud column’s original structure and distribution, this method ensures that synthetic data can be confidently used for analysis or machine learning without introducing artificial patterns into the fraud labels.

Steps:

1. Original Dataset: The raw data with columns like driverAge, accidentType, fraud, etc.
2. Clean Data: Any missing values (like in authoritiesInvolved) are replaced with "none".
3. Sample Columns (Excluding Fraud): Randomly sample all columns except fraud.
4. Sample Fraud Column Separately: The fraud column is sampled independently.
5. Add Noise to Numeric Columns: Add small noise to numeric columns to vary values.
6. Generate Synthetic Data: Final synthetic dataset output.

### 

### **Method 2:**

Synthetic Data Method 2 uses a more advanced approach compared to Method 1. It leverages a generative model called CTGAN (Conditional Tabular GAN), which is designed to generate highly realistic synthetic data by learning the underlying patterns and relationships in the original dataset. This method is particularly effective when dealing with complex datasets that include both categorical and continuous variables.

The process begins by loading the original dataset and defining its metadata. The CTGANSynthesizer is then used to train a model on the original data. This model learns the data's distribution and structure over several epochs (1,800 in this case), ensuring that the synthetic data it generates closely mimics the patterns in the original dataset.

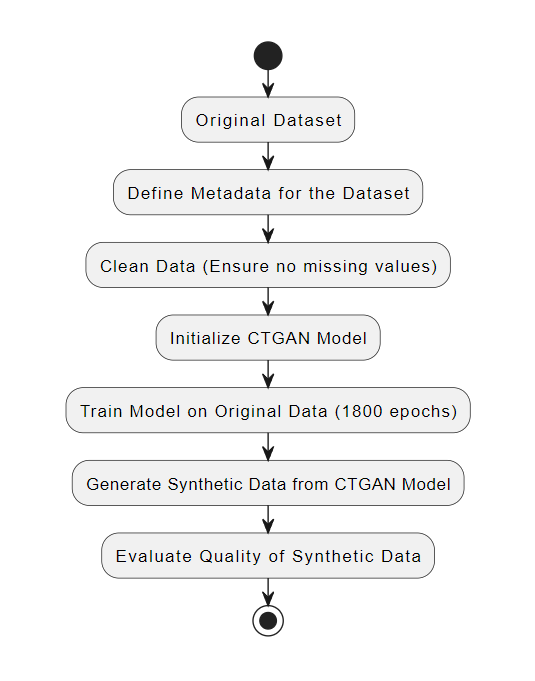
One of the key strengths of Method 2 is that it not only captures basic distributions, but it also models the relationships between different columns. This means that the generated data retains important dependencies between variables, making it more robust for tasks like machine learning or data analysis.

Fraud Column Reliability:

Similar to Method 1, the fraud column in Method 2 is kept reliable. Since the CTGAN model learns from the actual data distribution, it maintains the integrity of the fraud column and generates fraud labels in line with the original dataset. Unlike random sampling, this method ensures that the synthetic data generated by the GAN reflects the real-world proportions and relationships between fraudulent and non-fraudulent cases.

Once the synthetic data is generated, it is evaluated for quality using a tool that compares it to the real data. This ensures that the synthetic data is both accurate and usable for further analysis. Finally, the data is saved, with the index reset to start from 1, ensuring a clean output.

In summary, Synthetic Data Method 2 provides an advanced and reliable approach for generating synthetic data by using CTGAN. It captures the complex relationships in the dataset while ensuring that the fraud column remains accurate and trustworthy.

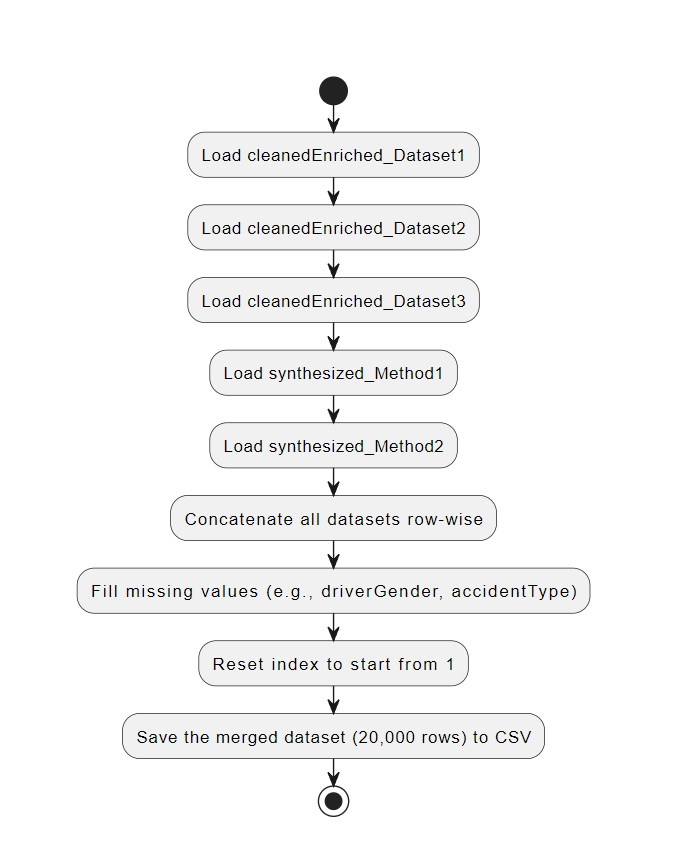
Steps:

1. Original Dataset: Load the original dataset with columns such as driverAge, accidentType, and fraud.
2. Define Metadata: The structure and relationships of the columns are defined in the metadata, which helps guide the model's learning.
3. Clean Data: Ensure that missing values are handled, for example, replacing empty values in authoritiesInvolved with "none".
4. Initialize CTGAN Model: The CTGAN model is initialised with hyperparameters such as the number of training epochs.
5. Train Model on Original Data: The model is trained on the original dataset, learning the underlying patterns and relationships.
6. Generate Synthetic Data: After training, the model generates new synthetic data that mirrors the original data in terms of structure and distribution.
7. Evaluate Quality: The generated synthetic data is evaluated against the original data to ensure quality and reliability, particularly for sensitive columns like fraud.

### **Merging Datasets:**

The dataset merging process involves combining three cleaned and enriched datasets with two synthesised datasets into a single dataset containing ~ 20,500 rows. This process ensures the data is consolidated, structured, and ready for further analysis or machine learning tasks.

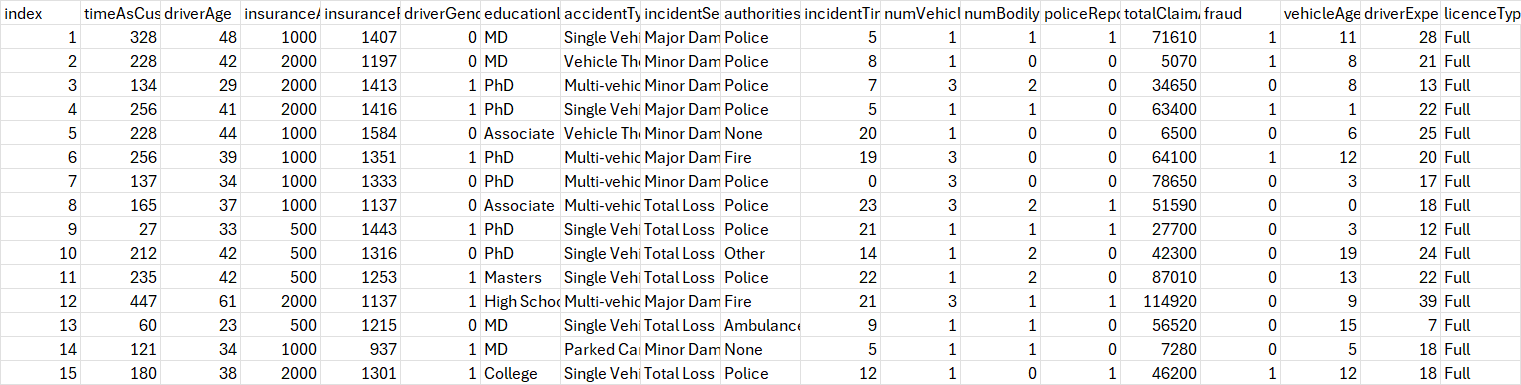
Steps in the Dataset Merger:

1. Loading the Datasets:
   * Three cleaned and enriched datasets (cleanedEnriched\_Dataset1, cleanedEnriched\_Dataset2, and cleanedEnriched\_Dataset3) are loaded. These datasets have undergone data cleaning and enrichment processes to ensure completeness and quality.
   * Two synthesized datasets (Synthesized\_Method1 and Synthesized\_Method2) are also loaded, representing synthetic data generated through advanced methods.
2. Concatenating the Datasets:
   * The datasets are concatenated row-wise, combining rows from all five datasets into one. This ensures that the merged dataset includes a complete set of data points from all the cleaned, enriched, and synthesised sources.
3. Filling Missing Values:
   * Columns such as driverGender, accidentType, incidentSeverity, and authoritiesInvolved are checked for missing or blank values. Any blanks are filled with "none" to ensure that the dataset is fully populated. This is critical for ensuring the integrity of the dataset when performing further analysis.
4. Resetting the Index:
   * A new index is generated for the merged dataset, starting from 1. This provides a unique identifier for each row in the dataset, ensuring that it is well-structured and organised.
5. Saving the Merged Dataset:
   * The final merged dataset, containing 20,000 rows, is saved to a CSV file. This dataset is now ready for further use, such as in machine learning models or data analysis.

Steps in the Diagram:

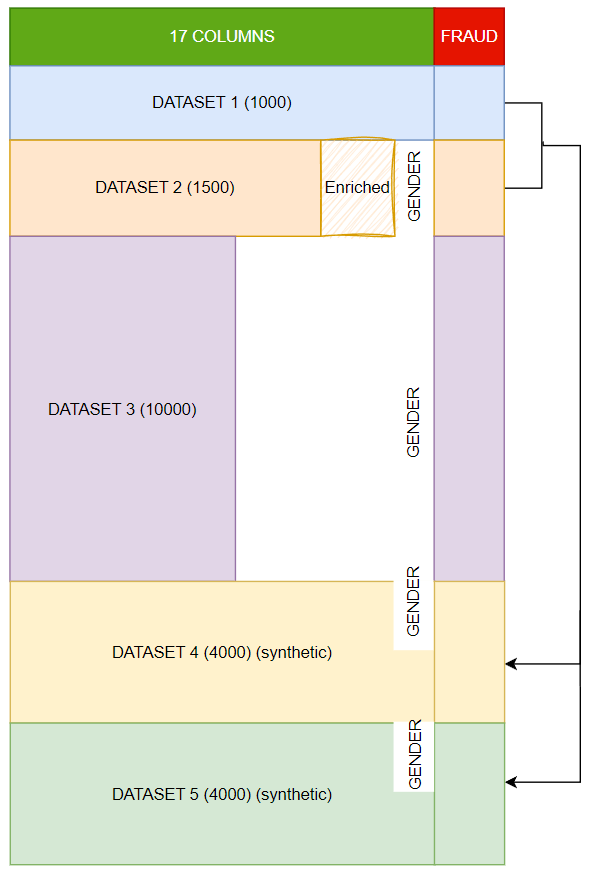
1. Load cleanedEnriched\_Dataset1: The first cleaned and enriched dataset is loaded.
2. Load cleanedEnriched\_Dataset2: The second cleaned and enriched dataset is loaded.
3. Load cleanedEnriched\_Dataset3: The third cleaned and enriched dataset is loaded.
4. Load synthesized\_Method1: The first synthesised dataset is loaded.
5. Load synthesized\_Method2: The second synthesised dataset is loaded.
6. Concatenate Datasets: The datasets are merged row-wise, appending rows from all datasets into one.
7. Fill Missing Values: Missing values, particularly in columns like driverGender, are filled with "none".
8. Reset Index: A new index is generated, starting from 1.
9. Save the Merged Dataset: The merged dataset, containing 20,000 rows, is saved to a CSV file for further use.

Merged Dataset (~20500 rows):



*Figure 20: Merged Dataset*

1. index: Row index for tracking purposes (Numerical).
2. timeAsCustomer: Number of months the individual has been a customer (Numerical).
3. driverAge: Age of the driver (Numerical).
4. insuranceAccess: The level of access to insurance (Numerical).
5. insurancePremium: The premium amount for the insurance policy (Numerical).
6. driverGender: Gender of the driver (Boolean: 0 = Male, 1 = Female).
7. educationLevel: Education level of the driver (Categorical).
8. accidentType: Type of accident (Categorical).
9. incidentSeverity: Severity of the incident (Categorical).
10. authoritiesInvolved: Whether authorities were involved (Categorical).
11. incidentTime: Time of the incident (Numerical).
12. numVehiclesInvolved: Number of vehicles involved in the accident (Numerical).
13. numBodilyInjuries: Number of bodily injuries (Numerical).
14. policeReportBool: Whether a police report was available (Boolean: 0 = No, 1 = Yes).
15. totalClaimAmount: The total claim amount (Numerical).
16. fraud: Whether fraud was reported (Boolean: 0 = No, 1 = Yes).
17. vehicleAge: Age of the vehicle (Numerical).
18. driverExperience: Number of years of driving experience (Numerical).
19. licenceType: Type of licence held by the driver (Categorical).



*Figure 21: Structure of resulting dataset after merging:*

**TEST SPECIFICATION**

## 

# **1. Model evaluation**

*(how will any data / outputs be compared / tested / evaluated for correctness and accuracy?)*

### 

## **1.1. Model Evaluation and Comparison**

**Data Cleaning, Synthesisation, and Imputation:**

To ensure the accuracy and reliability of our data, a comprehensive evaluation was conducted on the datasets, including comparisons between original, cleaned, and synthesised data. This process was crucial for validating the integrity and consistency of the datasets used in our analysis.

##### 

### **1.1.1. Correlation Matrix and ANOVA Analysis:**

**Correlation Matrix Comparison:** We conducted correlation matrix comparisons on our datasets before and after cleaning, as well as between the cleaned data and synthesised data. The correlation matrix highlights the strength of linear relationships between numerical features. This allowed us to identify any significant changes caused by the data cleaning and enhancement processes. By ensuring that the relationships between key variables remained consistent across datasets, we confirmed the quality of our preprocessing steps.

**ANOVA Tests:** ANOVA was used to assess the differences in means among groups within the datasets. We performed ANOVA tests to evaluate whether the data distributions in the cleaned data sets differed significantly from those in the original dataset and to compare distributions between cleaned/enhanced data and synthesised data. This statistical test helped verify that preprocessing steps did not introduce biases or significant changes in data distribution.

**Consistency Check:** In both synthesised datasets (generated using different methods), the total number of fraud cases is counted and compared. Both methods resulted in fraud counts around 400, aligning closely with the original data. This consistency in fraud counts suggests that both synthetic methods accurately represent the underlying distribution of fraud cases. Validation through Distribution: Given the similarity in fraud counts across the synthesised datasets, it is highly unlikely that such results are coincidental. Instead, this indicates that the models are correctly capturing the underlying fraud patterns present in the original data, validating the reliability of the synthetic generation methods.

##### 

### **1.1.2. Validation of Synthesised Data:**

**Fraud Count Consistency:** Both synthetic datasets generated approximately 400 instances of fraud, providing strong validation for the synthesis process. Given the rarity of fraud cases, it is improbable that both synthesised datasets would have this coincidence unless the process successfully replicated the underlying distribution patterns of the real data. This consistency was used as an additional checkpoint to ensure that the synthetic data reflected the real-world occurrences correctly.

##### 

### **1.1.3. Imputation Techniques:**

Missing data presents a challenge that can significantly impact the quality and reliability of model predictions. We explored multiple imputation methods for both numerical and categorical columns. Given the complexity of our dataset and the need for more advanced handling of missing values, we decided to use K-Nearest Neighbors (KNN), Support Vector Regression (SVR), and Random Forest models for imputation. We used several key metrics to evaluate the accuracy and effectiveness of our imputation methods for both numerical and categorical columns.

##### 

**For Numerical Columns:**

The primary metrics used for evaluating imputation in numerical columns were:

* **Mean Squared Error (MSE):** MSE measures the average squared difference between the imputed values and the actual values (where known). Our target MSE was for it to be as low as possible ideally <0.10
* **Mean Absolute Error (MAE):** MAE is similar to MSE but focuses on the absolute differences between the imputed values and the actual values, making it less sensitive to large errors.Our target MSE was for it to be as low as possible ideally <0.05.
* **R-squared (R²):** R² measures how well the imputed values explain the variance in the dataset. Our target for R2 was to be as close to 1 ideally >0.80.

These metrics provided insight into how well the imputed values aligned with the true values in the dataset, allowing us to optimise and refine our approach.

##### 

**For Categorical Columns:**

For categorical columns, accuracy metrics are more appropriate for evaluating model performance. We focused on:

* **Accuracy:** Measures the percentage of correctly imputed categories compared to the total number of categories.Our aim for imputation accuracy was to >75%
* **Consistency Checks**: For instance, checking if the imputed values for columns like "Occupation" and "Education Level" logically fit the other related columns (e.g., if a person with a PhD is categorised under an occupation like "Clerical" might indicate poor imputation).

#### 

## **1.2. How we compared and evaluated Models and decided what to choose**

### In our project, we used several models for both imputation and classification, including K-Nearest Neighbors (KNN), Support Vector Regression (SVR), and Random Forest. While the choice of model is essential, the performance evaluation metrics we used to assess these models played an even more significant role in guiding our final decisions. These evaluation techniques allowed us to assess the models' ability to accurately impute missing data and classify fraudulent and non-fraudulent claims. While also comparing and contrasting the models objectively, ensuring we selected the most suitable approach for each specific task.

### **1.2.1. R² (Coefficient of Determination)**

The **R²** metric is particularly useful for continuous variables. It measures how well the model explains the variability in the target variable. An R² closer to 1 indicates that the model captures most of the variability, while an R² closer to 0 means the model explains little of the variance.

Impact on model choice: For tasks like imputing continuous columns, we relied heavily on R² to understand how well the model was capturing the underlying patterns. A low R² suggested the model was underperforming, prompting us to try different models or adjust the features being used. On the other hand, a higher R² indicated the model was a good fit for the data, leading us to keep and refine those models.

For the YOJ (Years on Job) column, which is continuous data, R² became the most critical. The first model we applied gave us an R² score of 42%, indicating that the model explained only 42% of the variance in the target variable. This was relatively low, and it informed us that the model wasn't capturing the complexity of the data. Therefore we used it to refine the model. With such a low R², we explored other models (KNN, SVR, and Random Forest) to determine if we could achieve better performance. For continuous data like YOJ, a higher R² is crucial because it shows how well the model is capturing the real-world variations in the data. After hyperparameter tuning, we managed to improve the R² score , justifying our selection of the final model.

### **1.2.2. Accuracy**

Accuracy is a key metric for evaluating classification tasks. It simply measures the percentage of correctly predicted outcomes out of the total predictions made. While accuracy is often the most intuitive metric, it can sometimes be misleading, especially when dealing with imbalanced datasets.

Impact on model choice: In our tasks, accuracy helped us identify models that performed well overall. However, we also recognized its limitations, especially in cases where certain categories or classes were underrepresented. For example, if only a small portion of the data represented a specific outcome (e.g., fraudulent claims), a model might have high accuracy while still failing to predict the minority class effectively. As a result, we supplemented accuracy with other metrics (like precision and recall) to ensure the model was suitable for all parts of the data.

When imputing the 'Occupation' column, accuracy was used to evaluate how well each model predicted missing values. Initial attempts with one model resulted in low accuracy scores, indicating that it struggled to capture the relationships in the data. By analysing the accuracy, we identified that tweaking the hyperparameters and switching models improved performance.

For the 'Education Level' column, the accuracy score again played a crucial role. When accuracy was low, we either adjusted the model or added more features to improve predictions. The feedback from accuracy scores ensured that we did not settle on a model unless it provided reasonable confidence in the imputed values. While accuracy was a useful initial measure, we avoided making decisions solely based on it, as it didn’t always capture the nuances in our dataset.

#### 

### **1.2.3. Mean Squared Error (MSE) and Mean Absolute Error (MAE)**

Both MSE and MAE are fundamental metrics for evaluating the performance of models, especially for tasks like imputing missing values or handling continuous data. These metrics help quantify the prediction errors, guiding us toward selecting models that minimise these errors.

MSE: Measures the average squared difference between actual and predicted values. Because it squares the differences, it penalises larger errors more heavily. This makes MSE especially useful when outliers need to be handled with care. Impact on model choice: In cases where larger errors are undesirable, we prioritised models with lower MSE scores, knowing they would provide more reliable predictions, particularly for continuous data.

MAE: Represents the average of the absolute differences between actual and predicted values. Unlike MSE, it doesn’t overly penalise larger errors, making it more robust to outliers. Impact on model choice: When MSE indicated that certain models were too sensitive to outliers, we referred to MAE to see if a model could still offer stable performance. Models with consistently low MAE scores were prioritised for their robustness in handling diverse data points. By comparing MSE and MAE, we balanced error sensitivity and the need for stability, selecting models that provided the best trade-off between the two metrics.

For the YOJ (Years on Job) column MSE and MAE were used for Error Minimisation. Alongside R², we used MSE and MAE to measure the precision of our predictions. The models that minimised these errors were retained, while those with high MSE or MAE were discarded. Random Forest, in particular, performed well in minimising these errors, making it a suitable choice for YOJ imputation. R² was a crucial indicator for assessing models’ ability to generalise and provide accurate predictions for continuous data, influencing which models we selected for such tasks.

### 

## **1.3. Overall**

Ultimately, the evaluation metrics like MSE, MAE, accuracy, and R² guided our decision-making far more than the models themselves. These metrics provided detailed insights into the strengths and weaknesses of each model, allowing us to choose models that performed best for specific tasks.

Instead of focusing solely on which model to use, we focused on how these evaluation techniques informed us about model performance. This approach ensured that we selected models based on how well they fit the data and how effectively they minimised errors or maximised predictive power, rather than relying purely on the theoretical strengths of each model.

# **2. Performance evaluation results**

*(What are the results of any tests run so far and what are the future planned tests for future iterations?)*

It is possible to know that the synthesising process is reliable and accurate because both synthesising methods and processes were different and carried out independently. However, their output results were similar, indicating that both need to have been correct to have gotten similar outputs. I.e. it couldn’t have been a coincidence.

An indicator of their similar output is the occurrence of fraud in the fraud column. 466 occurrences of fraud in method 1, and 417 occurrences in method 2. The total rows synthesised by each method is 4000, so in 4000 rows being this close to each other certainly means that both methods of synthesising are valid and reliable.

