GitHub page: https://github.com/ARAlifio/use_case_test

1. Data Collection and Preprocessing

The first step involved collecting data that is representative of fraudulent and non-fraudulent transactions. This dataset typically contains several features describing each transaction, such as transaction amount, type, location, and timestamp. It also includes labels indicating whether the transaction was fraudulent or legitimate.

- **Imbalanced Dataset**: Fraud detection typically involves highly imbalanced data, where fraudulent transactions make up a very small percentage of the dataset. This imbalance can bias the model toward predicting non-fraudulent transactions. Special techniques are required to address this.
- **Handling Missing Data**: Any missing or incomplete entries were handled using imputation techniques or by dropping incomplete records, depending on the significance of the missing data.
- **Feature Engineering**: New features might have been created to enhance model performance. For instance, day and hour was extracted.

2. Model Selection: LightGBM

For this project, **LightGBM (Light Gradient Boosting Machine)** was chosen as the primary model due to its lightweight architecture and excellent performance on imbalanced datasets, making it highly suited for fraud detection.

- **Why LightGBM**: LightGBM is known for being a highly efficient implementation of gradient boosting, optimised for both speed and accuracy. Its design allows it to handle large datasets and high-dimensional feature spaces with significantly reduced memory consumption. These characteristics make it ideal for real-time fraud detection systems.
- **Dealing with Imbalanced Data**: LightGBM has built-in support for handling imbalanced datasets by allowing the use of the `is_unbalance` parameter to handle the minority (fraudulent) class. This ensures that the model remains sensitive to detecting fraud, even when legitimate transactions dominate the data.

4. Model Training

The model was trained using the training dataset, where the features were fed into the model, and the model's weights were adjusted based on the loss calculated. During training:

- **Validation Split**: A portion of the dataset was held out as a validation set to monitor the model’s performance on unseen data during training.

To improve the model’s performance, hyperparameter tuning was conducted. This process involved adjusting parameters such as the learning rate, the number of leaves, and number of boost rounds.