**Phase 5**

**Problem Statement:**

Develop a machine learning model to detect and prevent fraudulent credit card transactions accurately and efficiently.

**Key Objectives:**

1. Identify and flag potentially fraudulent credit card transactions in real-time or near real-time.
2. Minimize false positives to avoid inconveniencing legitimate cardholders while maximizing the detection of actual fraud.
3. Ensure the security and confidentiality of cardholder information and transaction data during the detection process.
4. Continuously update and improve the fraud detection system to adapt to evolving fraud tactics and patterns.

**Thinking Process**

1. Detecting credit card fraud involves a complex thinking process that combines data analysis, machine learning, and a deep understanding of fraudulent behaviors. Here's a step-by-step thinking process for credit card fraud detection:
2. Data Collection:
   1. Gather historical credit card transaction data, including details like transaction amount, merchant, date and time, and other relevant information.
   2. Ensure the dataset contains labeled examples, indicating whether each transaction is legitimate or fraudulent.
3. Data Preprocessing:
   1. Clean and preprocess the data, handling missing values, outliers, and inconsistencies.
   2. Normalize or scale features to ensure they have consistent scales and are suitable for modeling.
4. Data Exploration:
   1. Perform exploratory data analysis (EDA) to gain insights into the dataset.
   2. Examine the distribution of legitimate and fraudulent transactions to understand the data's class imbalance.
   3. Identify any patterns or trends in the data that may indicate fraudulent activity.
5. Feature Engineering:
   1. Create new features or modify existing ones to capture relevant information for fraud detection.
   2. Feature selection may be necessary to reduce dimensionality and improve model performance.
6. Model Selection:
   1. Choose an appropriate machine learning algorithm or model for fraud detection. Common choices include:
      1. Logistic Regression
      2. Decision Trees
      3. Random Forests
      4. Gradient Boosting
      5. Support Vector Machines
      6. Neural Networks
      7. Anomaly Detection Models (e.g., Isolation Forest, One-Class SVM)
7. Data Split:
   1. Split the dataset into training, validation, and testing sets. Ensure the class imbalance is preserved in each split.
8. Model Training:
   1. Train the selected model using the training data.
   2. Hyperparameter tuning and cross-validation can help optimize the model's performance.
9. Model Evaluation:
   1. Evaluate the model's performance on the validation set using appropriate metrics, including:
      1. Accuracy
      2. Precision
      3. Recall (Sensitivity)
      4. F1-Score
      5. Area Under the ROC Curve (AUC-ROC)
   2. Choose the evaluation metric that best aligns with the fraud detection objectives (e.g., minimizing false positives or maximizing true positives).
10. Model Testing:
    1. Assess the model's performance on the separate testing dataset to simulate real-world performance.
    2. Calculate and report the same evaluation metrics used in validation.
11. Imbalanced Data Handling:
    1. Address class imbalance through techniques such as oversampling the minority class (fraudulent transactions), undersampling the majority class (legitimate transactions), or using hybrid methods like SMOTE (Synthetic Minority Over-sampling Technique).
12. Real-Time Processing:
    1. Implement a real-time or near real-time processing pipeline for credit card transactions.
    2. Ensure the model can make quick decisions without significant delays.
13. Continuous Learning and Monitoring:
    1. Continuously update the model to adapt to new fraud patterns and techniques.
    2. Monitor the system's performance and retrain the model as necessary.
14. Threshold Selection:
    1. Set a decision threshold for the model to classify transactions as fraudulent or legitimate.
    2. Adjust the threshold to balance between minimizing false positives and maximizing true positives based on business needs and risk tolerance.
15. Deployment:
    1. Deploy the model into a production environment where it can process incoming credit card transactions in real time.
16. Fraud Alerts and Actions:
    1. Implement an alert system to notify relevant parties (cardholders, financial institutions, or fraud departments) when potential fraud is detected.
    2. Define a set of actions to take when fraud is suspected, such as blocking the card, notifying the cardholder, and initiating investigations.
17. Documentation and Reporting:
    1. Document the entire process, including data preprocessing steps, model architecture, and performance metrics.
    2. Create regular reports on the model's effectiveness in fraud detection.
18. The thinking process for credit card fraud detection requires a combination of data analysis, model selection, evaluation, and continuous improvement to stay ahead of evolving fraudulent activities while minimizing false positives and ensuring the security of cardholders' financial transactions.

**Dataset Used**

The dataset contains transactions made by credit cards in September 2013 by European cardholders.This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.

It contains only numerical input variables which are the result of a PCA transformation. Unfortunately, due to confidentiality issues, we cannot provide the original features and more background information about the data. Features V1, V2, … V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the transaction Amount, this feature can be used for example-dependant cost-sensitive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

### **Data Preprocessing Steps:**

1. Data Cleaning:
   1. Handle missing values: Identify and fill missing data or remove records with missing values.
   2. Outlier detection and treatment: Identify and address extreme values that may distort the model.
2. Feature Selection and Engineering:
   1. Select relevant features: Choose the features that are most likely to capture fraudulent patterns.
   2. Create new features if necessary to enhance the model's ability to detect anomalies.
3. Normalization or Scaling:
   1. Scale numerical features to ensure they have consistent scales, typically between 0 and 1.
   2. Common techniques include Min-Max scaling or Z-score standardization.
4. Data Split:
   1. Split the dataset into training, validation, and testing sets. Ensure that the class imbalance is maintained in each split.
5. Class Imbalance Handling:
   1. Deal with the class imbalance problem, as fraudulent transactions are typically a small fraction of the dataset. Common techniques include:
      1. Oversampling the minority class (fraudulent transactions).
      2. Undersampling the majority class (legitimate transactions).
      3. Using synthetic data generation methods like SMOTE (Synthetic Minority Over-sampling Technique).

### **Model Training Process:**

1. Select the Anomaly Detection Algorithm:
   1. Choose an appropriate anomaly detection algorithm. Common choices include:
      1. Isolation Forest: A tree-based algorithm that isolates anomalies efficiently.
      2. One-Class SVM: A support vector machine approach that identifies anomalies as points that deviate from the majority of data.
      3. Autoencoders: A neural network-based approach for dimensionality reduction and anomaly detection.
      4. Mahalanobis Distance: Measures the distance between data points and the centroid.
2. Model Training:
   1. Train the selected anomaly detection model on the preprocessed training data.
   2. The model learns the normal behavior of legitimate transactions during training.
3. Validation:
   1. Evaluate the model's performance on the validation dataset to set appropriate hyperparameters.
   2. Choose the model's sensitivity or threshold that balances between false positives and true positives based on the business requirements.
4. Testing:
   1. Assess the model's performance on the separate testing dataset to evaluate its real-world effectiveness.
5. Model Interpretation:
   1. Interpret the model's results to identify the anomalies detected and their associated features.
6. Continuous Monitoring and Updating:
   1. Implement a system to continuously update and retrain the model to adapt to evolving fraud patterns.
   2. Monitor the model's performance and retrain it as necessary.
7. Alerting and Action:
   1. Set up an alerting system to notify relevant parties when anomalies are detected.
   2. Define a set of actions to take when fraud is suspected, such as blocking the card, notifying the cardholder, and initiating investigations.
8. It's important to note that anomaly detection models may require careful parameter tuning and careful consideration of the trade-offs between false positives and false negatives, depending on the specific use case. Additionally, the choice of algorithm and preprocessing steps may vary based on the characteristics of the dataset and the performance requirements of the fraud detection system.

Machine Learning Alorithm Used:

A support vector machine is another effective technique for detecting anomalies. A SVM is typically associated with supervised learning, but there are extensions (OneClassCVM, for instance) that can be used to identify anomalies as an unsupervised problems (in which training data are not labeled). The algorithm learns a soft boundary in order to cluster the normal data instances using the training set, and then, using the testing instance, it tunes itself to identify the abnormalities that fall outside the learned region. Depending on the use case, the output of an anomaly detector could be numeric scalar values for filtering on domain-specific thresholds or textual labels (such as binary/multi labels). In this jupyter notebook we are going to take the credit card fraud detection as the case study for understanding this concept in detail using the following Anomaly Detection Techniques namely

Isolation Forest Anomaly Detection Algorithm Density-Based Anomaly Detection (Local Outlier Factor)Algorithm Support Vector Machine Anomaly Detection Algorithm Credit Card Fraud Detection Problem Statement: The Credit Card Fraud Detection Problem includes modeling past credit card transactions with the knowledge of the ones that turned out to be fraud. This model is then used to identify whether a new transaction is fraudulent or not. Our aim here is to detect 100 % of the fraudulent transactions while minimizing the incorrect fraud classifications.

Evaluation Metrics:

When evaluating a credit card fraud detection system using anomaly detection techniques, it's important to use metrics that are suitable for imbalanced datasets and prioritize the system's ability to detect fraudulent transactions while keeping false positives in check. Here are some key evaluation metrics for credit card fraud detection using anomaly detection:

1. \*\*True Positive (TP):\*\* The number of actual fraudulent transactions correctly identified as fraudulent by the model.

2. \*\*True Negative (TN):\*\* The number of actual legitimate transactions correctly identified as legitimate by the model.

3. \*\*False Positive (FP):\*\* The number of actual legitimate transactions incorrectly identified as fraudulent by the model (Type I error).

4. \*\*False Negative (FN):\*\* The number of actual fraudulent transactions incorrectly identified as legitimate by the model (Type II error).

These basic metrics can be used to calculate the following evaluation metrics:

5. \*\*Accuracy:\*\* The overall accuracy of the model in classifying transactions, calculated as (TP + TN) / (TP + TN + FP + FN). However, accuracy is not always the most informative metric for imbalanced datasets.

6. \*\*Precision (Positive Predictive Value):\*\* The ability of the model to correctly identify fraudulent transactions out of all transactions it classified as fraudulent, calculated as TP / (TP + FP). A high precision means a low false positive rate.

7. \*\*Recall (Sensitivity or True Positive Rate):\*\* The ability of the model to correctly identify fraudulent transactions out of all actual fraudulent transactions, calculated as TP / (TP + FN). A high recall means a low false negative rate.

8. \*\*F1-Score:\*\* The harmonic mean of precision and recall, which provides a balanced measure of a model's performance, calculated as 2 \* (Precision \* Recall) / (Precision + Recall).

9. \*\*Area Under the Receiver Operating Characteristic (ROC-AUC):\*\* The area under the Receiver Operating Characteristic curve, which measures the model's ability to distinguish between legitimate and fraudulent transactions. A higher ROC-AUC indicates better discrimination performance.

10. \*\*Area Under the Precision-Recall (PR-AUC):\*\* The area under the Precision-Recall curve, which focuses on the trade-off between precision and recall. PR-AUC is particularly useful when dealing with imbalanced datasets.

11. \*\*F2-Score:\*\* An alternative to F1-Score that places more emphasis on recall, calculated as (1 + 2^2) \* (Precision \* Recall) / (2^2 \* Precision + Recall). This metric can be useful when you want to prioritize recall over precision.

12. \*\*Specificity (True Negative Rate):\*\* The ability of the model to correctly identify legitimate transactions out of all actual legitimate transactions, calculated as TN / (TN + FP).

When evaluating a credit card fraud detection system, it's essential to choose the metrics that align with the specific goals and constraints of your application. In most cases, you would want to prioritize recall to minimize false negatives (missing fraud cases) while keeping precision in check to avoid excessive false positives (legitimate transactions flagged as fraud). The choice of the threshold for anomaly detection can significantly impact these metrics, so it's essential to fine-tune it according to your business requirements and risk tolerance.

CODE FILE REFERENCE

https://github.com/Jonal003/Jonal123.git