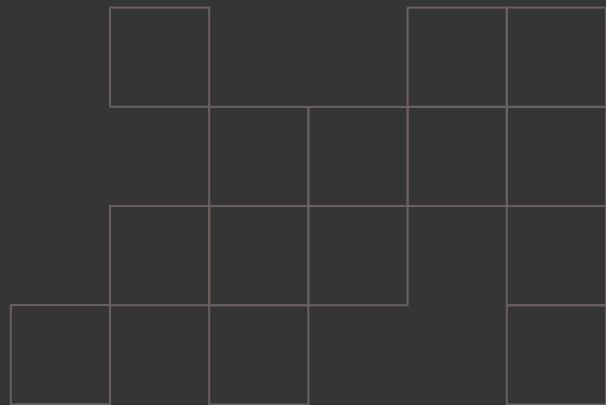


By Brandyn Ewanek 9216750

# From Model to Production

An Automated MLOps Pipeline for  
Fraud Detection

---



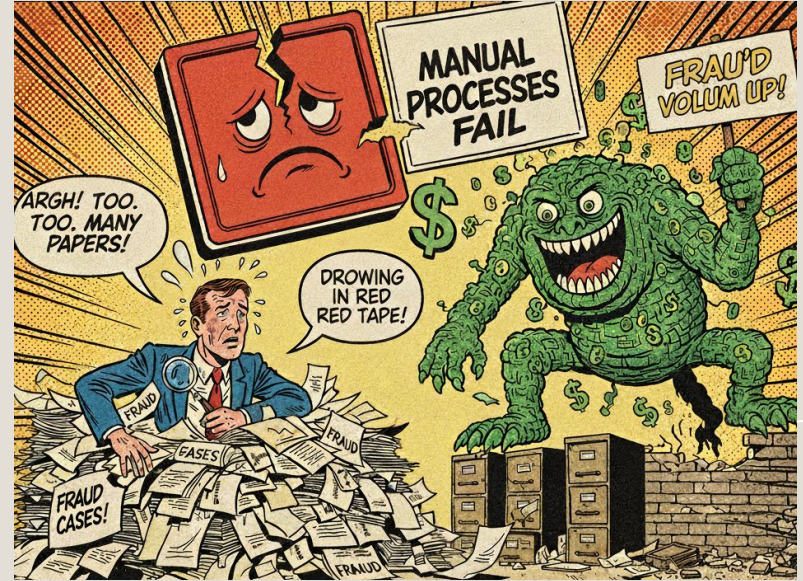
# The Fraud Problem

## Problem:

The government agency's manual processes to detection fraud are failing because they are not able to manage the volume.

## Concerns:

- Distributions of certain features have changed over time. (**Data Drift**)
- Believe that relation between features and fraud have changed ( **Concept Drift**)
- Need a system to always be live, no down time.
- Wants to ensure high rate of detection of fraud cases



# The Data: Anonymized & Imbalanced

- **Source:** Kaggle's Credit Card Fraud Dataset.
- **Scale:** 284,807 transactions over two days, with 31 columns.
- **Anonymization:** To protect citizen privacy, the 28 primary features (V1-V28) are Principal Components (PCA), meaning the original data is hidden. This was a key "government agency" requirement.
- **Features:** We are left with 30 continuous features (the anonymized V1-V28, plus Time and Amount) and one binary target.
- **Target:** Class is our binary target: 1 for Fraud, 0 for Normal.
- **Key Challenge: *Extreme Class Imbalance*:** The dataset is highly unbalanced. Only 492 transactions are fraudulent (0.172%), which makes detection very difficult.

#	Column	Non-Null Count		Dtype
0	Time	284807	non-null	float64
1	V1	284807	non-null	float64
2	V2	284807	non-null	float64
3	V3	284807	non-null	float64
4	V4	284807	non-null	float64
5	V5	284807	non-null	float64
6	V6	284807	non-null	float64
7	V7	284807	non-null	float64
8	V8	284807	non-null	float64
9	V9	284807	non-null	float64
10	V10	284807	non-null	float64
11	V11	284807	non-null	float64
12	V12	284807	non-null	float64
13	V13	284807	non-null	float64
14	V14	284807	non-null	float64
15	V15	284807	non-null	float64
16	V16	284807	non-null	float64
17	V17	284807	non-null	float64
18	V18	284807	non-null	float64
19	V19	284807	non-null	float64
20	V20	284807	non-null	float64
21	V21	284807	non-null	float64
22	V22	284807	non-null	float64
23	V23	284807	non-null	float64
24	V24	284807	non-null	float64
25	V25	284807	non-null	float64
26	V26	284807	non-null	float64
27	V27	284807	non-null	float64
28	V28	284807	non-null	float64
29	Amount	284807	non-null	float64
30	Class	284807	non-null	int64

# Conceptual Architecture

## **S3 Data Lake**

Stores all raw data, processed files, and model artifacts, acting as the pipeline's single source of truth .

## **EventBridge Trigger**

The "Scheduled Trigger". This rule automatically runs the SageMaker Pipeline on the first of every "month" to ensure regular, planned retraining.

## **SageMaker Pipeline**

The core MLOps "factory". It automates our Python scripts (preprocess, train, evaluate) into a single, orchestrated workflow to build a new model .

## **Model Registry**

Acts as the central "model library". It versions each newly trained model and tracks its "Approved" or "Rejected" status based on performance metrics .

## **CI/CD Trigger**

The "Continuous Deployment" (CD) trigger. An EventBridge rule detects when a new model is set to "Approved" in the registry, which then triggers an AWS Lambda function to deploy it

## **SageMaker Endpoint**

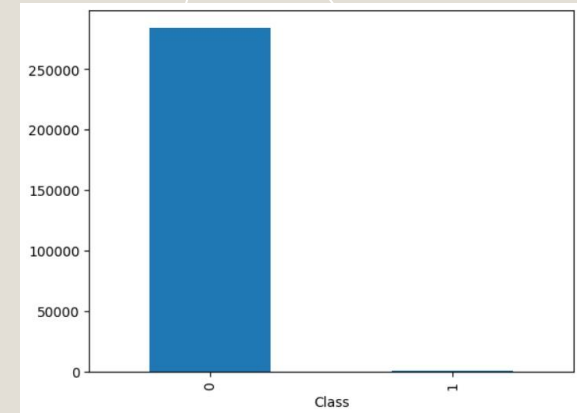
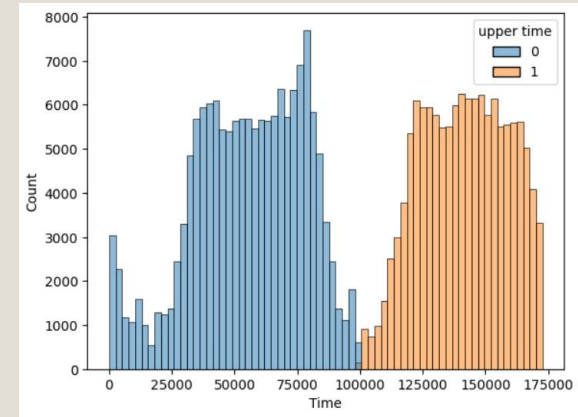
The production RESTful API. The agency's application sends new data here and receives a live fraud probability score in real-time .

## **Model Monitor**

The "Drift Trigger" . It captures live API data, detects data drift by comparing it to the training baseline , and fires an alarm to trigger a new pipeline run, completing the MLOps loop.

# Preprocessing

- **Binning Time** - This feature is bimodal, which potentially represents two distinct relationships with target. The choice was made to create a 'upper time' binary feature.
- **Train Test Split** - Advance models have an ability to 'map' detailed patterns between features however some of the patterns will be random non-repeating. To ensure our model is ready for production we will reserve 30% for testing, simulating real world performance.
- **Smote** - A common problem with Fraud predictions is the large imbalance. To fix this we will be upsampling class '1'. So that our model doesn't only learn to predict class '0'.
- **Standardize** - To ensure our model doesn't let the scale of different features impact its selection of features potentially skew the model we will standardize the features. Using standard scaling but taking mean and std from train data.



# Modeling Initial Experiments

	Train Recall	Test Recall	Train Accuracy	Test Accuracy	Train Precision	Test Precision
Random Forest	1.000000	0.846939	1.000000	0.999544	1.000000	0.213075
Gradient Boosting	0.983130	0.897959	0.988769	0.994119	0.994344	0.213075
Logistic Regression	0.966511	0.908163	0.979097	0.991345	0.991467	0.155323

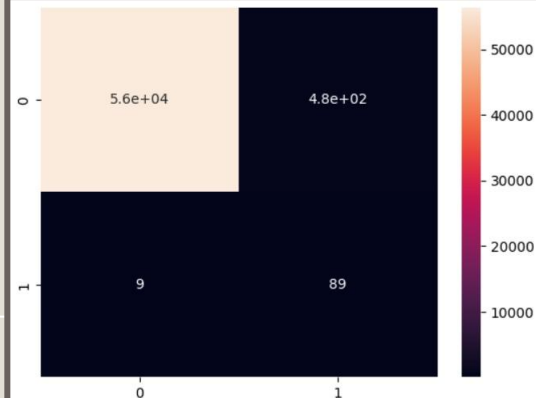
In the initial phase of model development we used the default setting to build the initial pipeline and in future we will further enhance models.

Tested **RandomForest**, **GradientBoosting**, and **LogisticBagging** models. Recall represents the % of the target class the model was able to find in the dataset, precision represents that total % target class predictions correction.

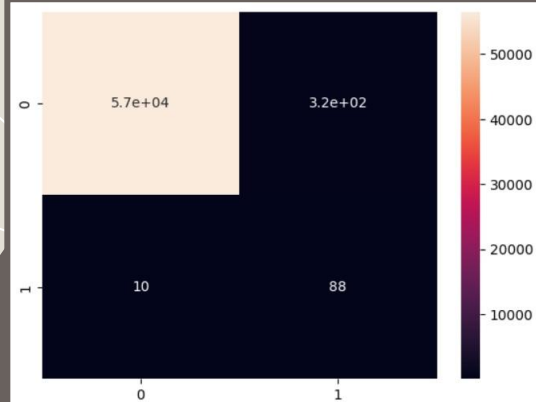
For this system **recall** will be the metric of interest. This is because we are trying to detect as many fraud cases as possible and the over prediction of class 0 will have a margin cost to it.

With type I errors being low real world Cost Logistic Regression inside Bagging Ensemble will be our model of choice.

## Logistic Regression



## Gradient Boosting



# Complete Modeling Workflow

## Phase 1: Event

- The workflow begins when a new monthly data file is uploaded to a specific folder in your Amazon S3 bucket

## Phase 2: MLOPs Pipeline

- Amazon SageMaker Pipeline
- Preprocesses the data, store train and test data.
- Trains new model with new data.
- Compares New Model with Old.
- Load best model to model registry

## Phase 3: Continuous Pipeline

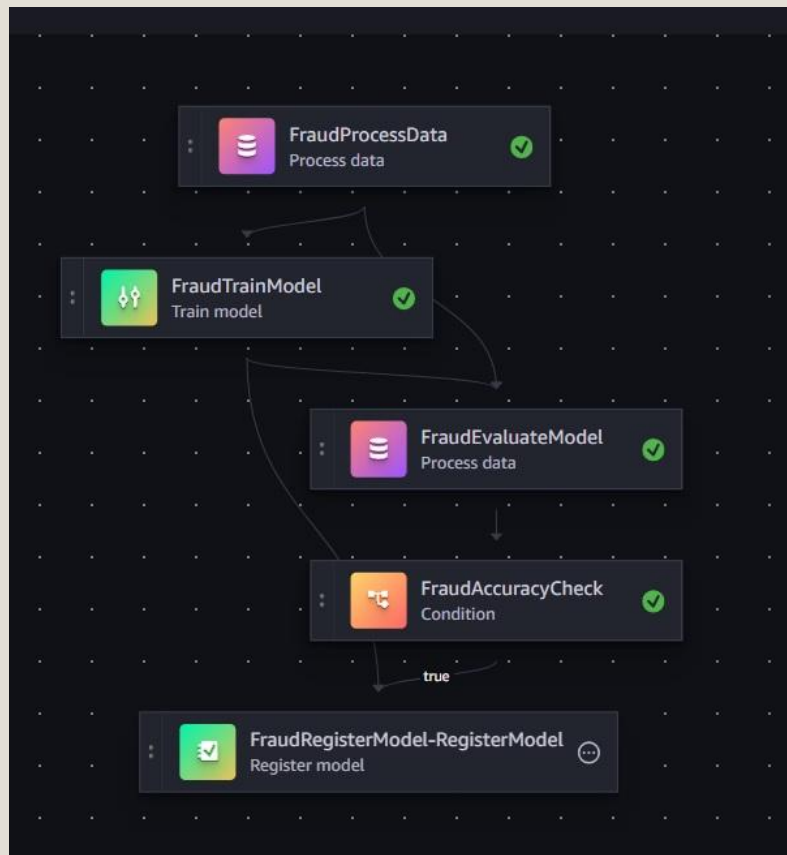
- Model Registry - sees new model approved
- Triggers Event Bridge to run Lambda function.
- Lambda function deploys new model to endpoint.

## Phase 4: Production and Monitoring

- SageMaker Endpoint (the API) is now serving live predictions
- Model Monitor compares new data with old.
  - If drift detected triggers Pipeline.

# The Automated MLOps Pipeline Steps

1. **Data Processing** ([FraudProcessData](#)) Automates feature engineering and splits the raw [creditcard.csv](#) into training and testing datasets using a SageMaker Processing Job.
2. **Model Training** ([FraudTrainModel](#)) Trains a Logistic Regression model on the processed data, automatically applying SMOTE to fix the class imbalance issues.
3. **Model Evaluation** ([FraudEvaluateModel](#)) Runs the trained model against the unseen test set to generate a "report card" of metrics (Recall, F1-Score) saved as a JSON file.
4. **Quality Gate** ([FraudAccuracyCheck](#)) A conditional logic step that checks the evaluation report. It only registers the model if the F1-Score is above your threshold (0.60), ensuring bad models never reach production.



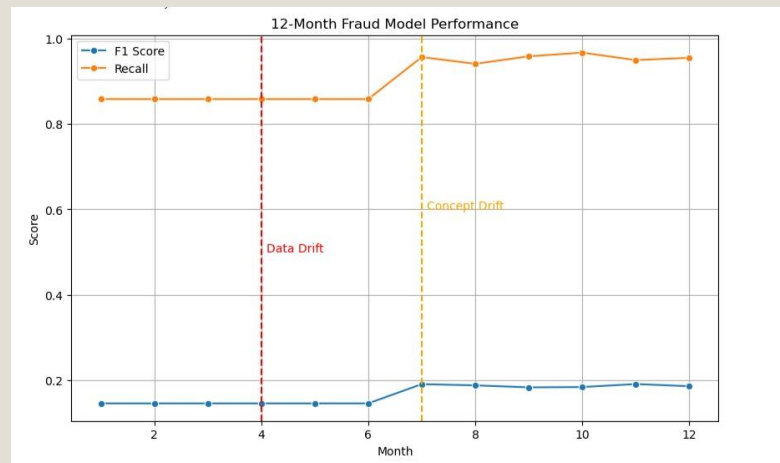
# Detecting Drift 12 Month Simulation

## Pipeline Steps

- **Data Processing:** Automated feature selection and train/test splitting.
- **Model Training:** Trained Logistic Regression using SMOTE to fix class imbalance.
- **Evaluation:** Calculated performance metrics (Recall, F1-Score) on unseen data.
- **Quality Gate:** Conditional logic that only registers models if Recall > 0.60 and better than previous model.

## Drift

- **Months 1 – 3** baseline,
- **Months 4 to 6** We progressively shifted the mean of feature V1 by adding a small drift factor (0.1, 0.2, 0.3).
- **Months 7 to 12** We introduced a new rule: High-value transactions (top 5% by amount) that were previously "Normal" were flipped to "Fraud" at a rate of 10%.



# Key Challenges & Solutions (Permissions vs. Environment)

## Challenge 1: Root User vs. IAM User Conflict

- Symptom: The Unknown **IAM PrincipalArn: ...:root error** when trying to create a project.
- Solution: The system was trying to use my Root login. I had to **create a new browser profile** (or use Incognito) to force a clean login as my brandyn IAM user.

## Challenge 2: The "Invalid Security Token" Error

- Symptom: The red error bar (Failed to fetch git connections...) and the **JupyterLab "gray screen"**.
- Solution: This looked like a permission error but was a **browser cookie policy**. As confirmed by support, SageMaker requires third-party cookies to connect the console to the notebook domain. The fix was to "Allow cookies" for the SageMaker site in my Incognito browser.

## Challenge 3: "Hanging Cell" & Incomplete Startup

- Symptom: The notebook would open, but a simple **!pip install would take 20+ minutes and fail**. Running a cell (like import pandas) would hang indefinitely, and the "stop" button wouldn't work.
- Solution: After confirming all permissions were correct, AWS Support helped identify the root cause: **the SageMaker Domain was in VPC only mode without the necessary VPC endpoints**.

## Challenge 4: "Dependency Hell" Breaking S3 Access

- Symptom: The notebook failed to load data with a cryptic **NoneType can't be used in 'await' expression error**, even though IAM permissions were perfect.
- Root Cause: **Installing awscli triggered an automatic upgrade of botocore (to v1.41.3)**, which broke the aiobotocore library that Pandas uses for S3 connections.
- Solution: We identified the version conflict and modified the environment setup to **exclude awscli** (which is pre-installed), restoring the compatible dependency chain.

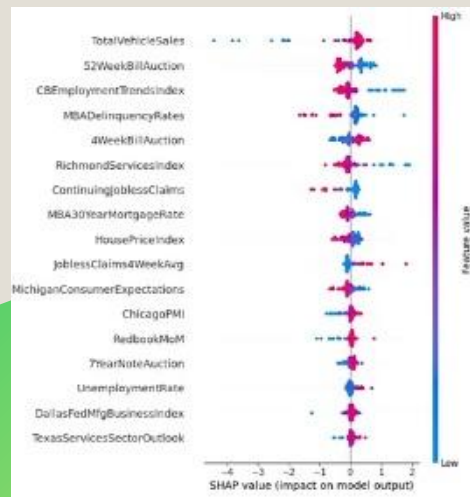
# Next Steps

## Further model experimentation

- We spent little time on this to focus on establishing a working CI/CD pipeline. With this complete we can return to the model and further enhance its predictive ability further increase the agency's profit margins.

## Deep Analysis of Deep Cases

- Using **Amazon Athena** or Sagemaker Studio we could employ a deep analysis of fraud cases potentially identify valuable information that could be used to prevent the cases before hitting our AWS endpoint.
- **Shaply values** would be valuable in determining the effect the features have on target. Learning that when feature V2 is high we see high chances of fraud could be valuable to the fraud prevention team.



# Appendix.

Coding Submission Folder.

