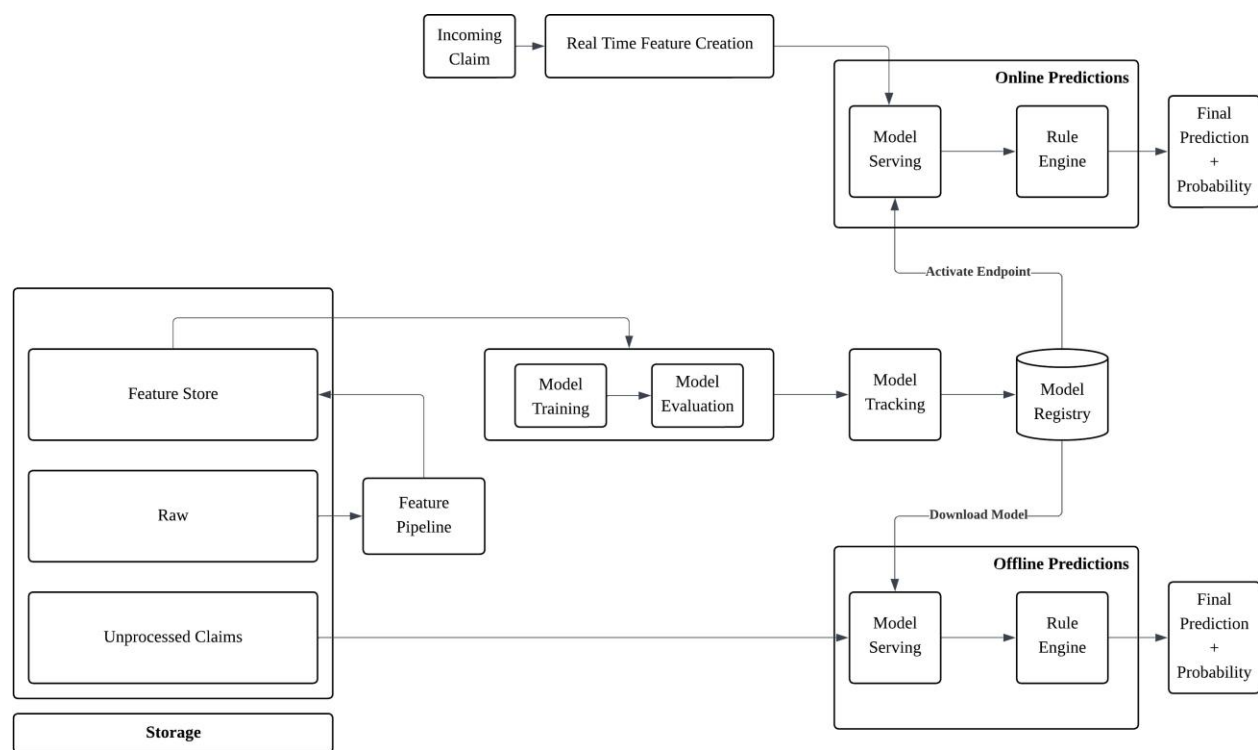


# Fraud Claims Detection – AI/ML Component

## Introduction

This document contains a high-level overview of the AI/ML Component for the fraud claims detection solution built on Microsoft Fabric. The objective of this module is to add an intelligence layer on top of a rule-based approach for detecting fraudulent claims.

## Solution Overview



## Data and features

The data in the raw layer consists of the following tables -

- Claims
- Policies
- Customer details
- Customer payment info
- Products and details

## Feature Engineering Pipeline

Raw data by itself is not sufficient for machine learning models to learn patterns within the data. We undergo a feature engineering process to create and modify the available raw data and transform them into meaningful features which would give the model a richer input to identify fraudulent patterns.

Categories of features to extract/transform -

- Claims profile (Customer profile regarding previous claims)
- Temporal features (Claim timing, policy changes/expirations)
- Behavioral Features (Behavior of customers)
- Geographic (Analyzing fraud claims in surrounding areas)
- Policy Details

Features are transformed as batches in offline mode or in real time and loaded into a Feature Store (Lakehouse) for future model training. This would prevent us from spending extra compute extracting features from the dataset each time model training happens.

## Model Selection

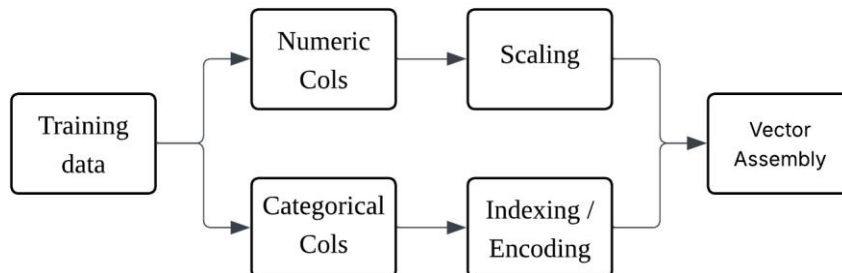
By analyzing an existing/historic dataset with claims being labeled as fraudulent, we could take a supervised learning approach. The task at hand is binary classification.

Model options -

Model Name	Description	Significance in use case ?
Random Forest	Ensemble of decision trees	Among more accurate classifiers due to performance in noisy data.
Logistic Regression	Linear classifier	Understanding feature importance, interpretable (regulatory /auditing)
Gradient Boosting	Ensemble learning technique that builds models sequentially, where each new model corrects the errors of the previous ones	Capture nonlinear relationships well. Tune to handle class imbalance

Insurance fraud datasets are highly imbalanced where the majority class is, legitimate claims made. Used class weighting or SMOTE (Synthetic Minority Oversampling Technique) to balance the dataset before training.

## Training Pipeline



Data is prepared for model training. Categorical cols are index/encoded and numerical columns are scaled (Optional, if needed.). All the features are converted into one vector format to begin training.

## Metrics

When evaluating models, we look at a few statistics to compare and determine which model to promote to production. These metrics also help us identify model drift which happens over time.

Metric name	Description	Why its important
Precision	The fraction of transactions flagged as fraud that were actually fraud	Efficiency: Don't waste time reviewing good claims
Recall	The fraction of all actual fraud cases that the model correctly caught	Risk Mitigation: Prevent fraud slipping through
F1 Score	Creates single score to balance precision and recall	Model Comparison: Accuracy is misleading when data set is imbalanced. (i.e: fraud data)
AUC –ROC	Measures the model's ability to distinguish between fraud and non-fraud across all thresholds.	General Discrimination.

## Model Serving

Fabric provides the ability for us to register ML models in the model registry and deploy the chosen model as an endpoint to use it in real time predictions.

### Endpoints

This ML model can serve real-time predictions from API endpoints. [Learn more](#)

#### Default endpoint for this model

Calls to this endpoint will use the model's default version to make predictions.

#### Model endpoint URL

<https://api.fabric.microsoft.com/v1/workspaces/5b2b0ee-5c15-4a01-bf6e-402b499db172>

#### Default version \*

Select a model version to use as the default

- ☒ Version 7
- ☐ Version 6
- ☐ Version 5
- ☐ Version 4
- ☐ Version 3
- ☐ Version 2
- ☐ Version 1

For offline predictions – Load the model to memory

```
1 import mlflow
2 from synapse.ml.predict import MLFlowTransformer
3
4
5 model = MLFlowTransformer(
6     inputCols=["incident_hour", "incident_type", "incident_s",
7     outputCol="predictions",
8     modelName="Model_Vehicles",
9     modelVersion=7
10 )
```