PRJ-MLS-045: Model Explainability with SageMaker Clarify

Certification: AWS Certified Machine Learning – Specialty
Domain: Modeling and Model Interpretation

---

1. Project Overview

This project focuses on a critical aspect of responsible machine learning: **explainability**. As models become more complex (often referred to as "black boxes"), it becomes difficult to understand *why* they make certain predictions. This lack of transparency can be a major barrier to adoption, especially in regulated industries like finance and healthcare. **Amazon SageMaker Clarify** is a feature of SageMaker that helps data scientists and ML engineers gain deeper insights into their models and data.

We will use SageMaker Clarify to run a post-training explainability analysis on a previously trained model. Specifically, we will compute **SHAP (SHapley Additive exPlanations)** values. SHAP is a game theory-based approach that explains the prediction of an instance by computing the contribution of each feature to that prediction. This allows us to understand not just which features are important globally, but how each feature influenced a specific prediction, providing local, instance-level explanations.

Key Objectives

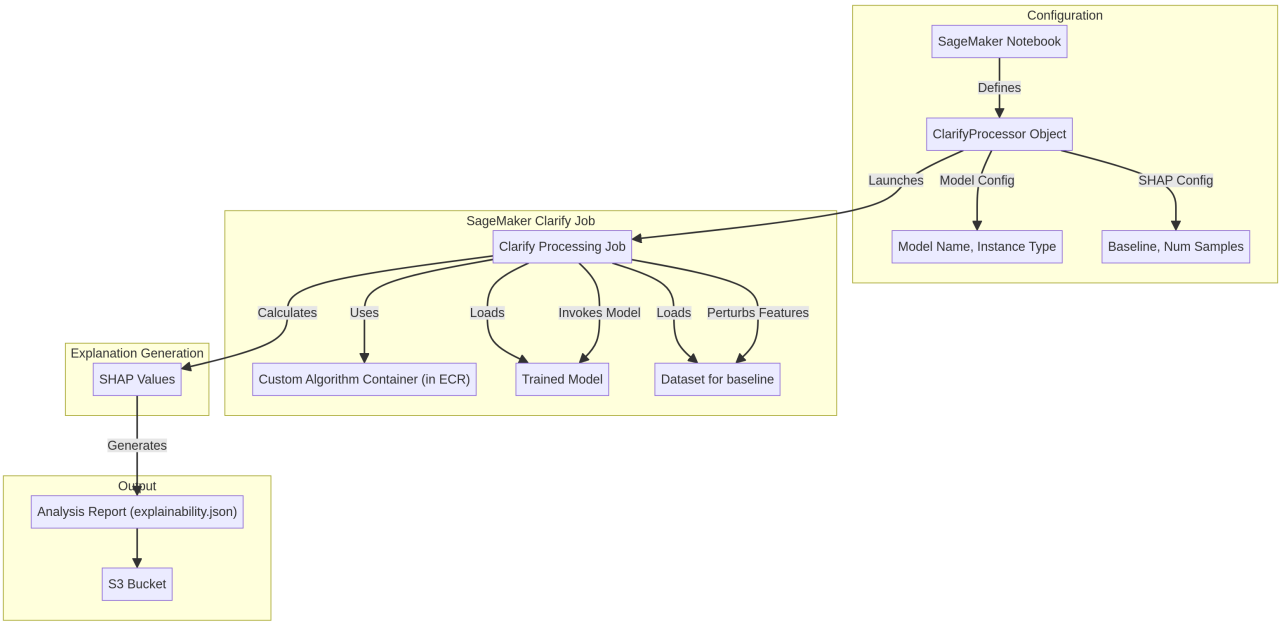
- Understand the importance of model explainability for building trust and debugging models.
- Learn about SHAP as a model-agnostic method for explaining predictions.
- Configure and launch a SageMaker Clarify processing job to analyze a trained model.
- Generate and interpret the SHAP values from the Clarify analysis report.

- Visualize feature importance both globally and for individual predictions.
- Run a Clarify job on a custom algorithm container, not just SageMaker's built-in algorithms.

---

2. Architecture

SageMaker Clarify runs as a specialized SageMaker Processing Job, which analyzes a trained model against a dataset.



Clarify Job Workflow:

1. **Configuration:**
    - In a **SageMaker Notebook**, we define a `ClarifyProcessor` object. This is a specialized processor for running Clarify jobs.
    - We configure the job with:
        - `ModelConfig`: Specifies the name of the trained model to be analyzed, the instance type to use for creating a temporary endpoint, and the content types.
        - `SHAPConfig`: Defines the parameters for the SHAP analysis, including the **baseline** (a reference dataset used for comparison) and the number of samples to use for the explanation.

2. **Clarify Job Execution:**
    - When the job is launched, SageMaker Clarify provisions the requested compute resources.
    - It automatically creates a temporary, shadow endpoint using the specified trained model.
    - It loads the dataset provided in the `SHAPConfig`.
3. **Explanation Generation:**
    - The Clarify job systematically perturbs the features of the input data (based on the baseline) and sends thousands of inference requests to the shadow endpoint.
    - By observing how the model's output changes with each perturbation, it calculates the SHAP value for each feature for each instance. This value represents the feature's contribution to pushing the model's prediction away from the baseline average.
4. **Output:**
    - The job aggregates the results and generates a detailed analysis report (e.g., `explainability.json`).
    - This report, along with visualizations, is saved to an S3 bucket.
    - The temporary endpoint and compute resources are automatically torn down.

---

3. Prerequisites

- An AWS account with administrative permissions.
- A trained model already registered in the SageMaker Model Registry or a saved model artifact in S3.
- The dataset used for training the model, stored in S3.
- A SageMaker Notebook instance.

---

4. Step-by-Step Implementation Guide

Step 4.1: Set Up the Clarify Processor

In your SageMaker Notebook:

```
import sagemaker
from sagemaker import clarify

role = sagemaker.get_execution_role()
sagemaker_session = sagemaker.Session()
bucket = sagemaker_session.default_bucket()

# Path to your dataset in S3
data_s3_uri = f"s3://{bucket}/my-dataset/data.csv"
# Path for Clarify output
clarify_output_s3_uri = f"s3://{bucket}/clarify-output"

# Name of your trained model in SageMaker
model_name = "my-trained-xgboost-model"

clarify_processor = clarify.SageMakerClarifyProcessor(
    role=role,
    instance_count=1,
    instance_type="ml.m5.xlarge",
    sagemaker_session=sagemaker_session
)
```

Step 4.2: Configure the Analysis

1. **Data Configuration:**
    - Tell Clarify where to find the data, where to save the output, the format of the data, and which column is the label.

```
data_config = clarify.DataConfig(
    s3_data_input_path=data_s3_uri,
    s3_output_path=clarify_output_s3_uri,
    label="target_column",
    headers=[...], # List of all column names in order
    dataset_type="text/csv"
)
```

2. **Model Configuration:**

   - Specify the model to be analyzed.

```
model_config = clarify.ModelConfig(
    model_name=model_name,
    instance_type="ml.c5.xlarge",
    instance_count=1,
    accept_type="text/csv",
    content_type="text/csv"
)
```

3. **SHAP Configuration:**

   - Define the baseline and the number of samples.
   - The baseline can be a single row or a file in S3 representing the average or median of your dataset.

```
shap_baseline_config = clarify.SHAPBaselineConfig(
    mime_type="text/csv",
    shap_baseline=s3_uri_to_baseline_file # S3 URI to a CSV with one
row
)


shap_config = clarify.SHAPConfig(
    baseline=shap_baseline_config,
    num_samples=100, # Number of perturbations per instance
    agg_method="mean_abs" # Global feature importance
)
```

Step 4.3: Run the Clarify Job

```
clarify_processor.run_explainability(
    data_config=data_config,
    model_config=model_config,
    explainability_config=shap_config
)
```

- This command will launch the SageMaker Clarify processing job. You can monitor its progress in the SageMaker console under **Processing jobs**.
- The job will take some time to complete as it needs to run many predictions.

Step 4.4: Analyze the Results

- Once the job is complete, the results will be in the S3 output path you specified.
- The main output is `analysis.json`.
- You can download and inspect this file, or use SageMaker Studio's built-in visualizations.
- If you ran the job from a SageMaker Studio notebook, you can often see a visualization of the feature importance directly in the notebook output.

Interpreting the Output:

- **Global Importance:** The analysis will provide a ranked list of features based on their mean absolute SHAP value. This tells you which features have the most impact on the model's predictions overall.
- **Local Explanations:** For each individual instance you explained, you will get a set of SHAP values, one for each feature. A positive SHAP value for a feature means it pushed the prediction higher (e.g., towards "customer will churn"), while a negative value means it pushed the prediction lower (e.g., towards "customer will not churn").

---

5. Explainability for Custom Algorithms

SageMaker Clarify can also explain models that are not in the SageMaker Model Registry, such as custom algorithms packaged in your own Docker container. To do this, you need to ensure your container adheres to the SageMaker inference container specification.

- **Modify `ModelConfig`:** Instead of providing a `model_name`, you provide the ECR image URI of your custom container.
- **Container Contract:** Your container must expose an `/invocations` endpoint for predictions. Clarify will load this container and send requests to it just like it would for a standard SageMaker model.

---

6. Cleanup

- The Clarify processing job and its associated shadow endpoint are automatically deleted upon completion.
- Delete the analysis results from the S3 output bucket if you no longer need them.
- Stop the SageMaker Notebook instance.

Business Context

The Problem

Organizations want to leverage AI/ML for business insights but lack the expertise to build secure, scalable ML systems. ML projects fail due to poor data quality, model drift, and lack of monitoring. Security and compliance requirements for ML systems are often overlooked.

The Solution

Production-ready ML system with automated data pipelines, model training, deployment, and monitoring. Implements MLOps best practices with version control,

A/B testing, and automated retraining. Ensures data privacy and model security throughout the ML lifecycle.

Business Value

- **Business Intelligence:** AI-driven insights improve decision-making and outcomes
- **Operational Efficiency:** Automated ML workflows reduce manual data science effort by 60%
- **Model Reliability:** Continuous monitoring detects and corrects model drift automatically
- **Compliance Assurance:** Built-in controls for data privacy and model governance

Risk Mitigation

Protects sensitive data in ML pipelines, prevents biased or inaccurate models, ensures model explainability, and maintains compliance with AI regulations.

GRC Mapping

Compliance Frameworks

- **NIST AI RMF:** Trustworthy and responsible AI
- **ISO/IEC 23894:** AI risk management
- **NIST CSF:** PR.DS-2 (Data in transit), PR.DS-5 (Data leak protection)
- **Responsible AI Principles:** Fairness, accountability, transparency, ethics

Security Controls Implemented

- Data anonymization and pseudonymization
- Model explainability and interpretability
- Bias detection and mitigation
- Model versioning and rollback
- Automated model monitoring and alerting

Audit Evidence

- Model cards documenting intended use and limitations
- Bias and fairness evaluation reports
- Model performance metrics over time
- Data processing and transformation logs

Regulatory Alignment

- **GDPR:** Article 22 (Right to explanation), Article 25 (Privacy by design)
- **AI Act (EU):** Transparency and accountability requirements
- **CCPA:** Data privacy for ML training data
- **SOC 2:** CC6.1 (Access to sensitive data)