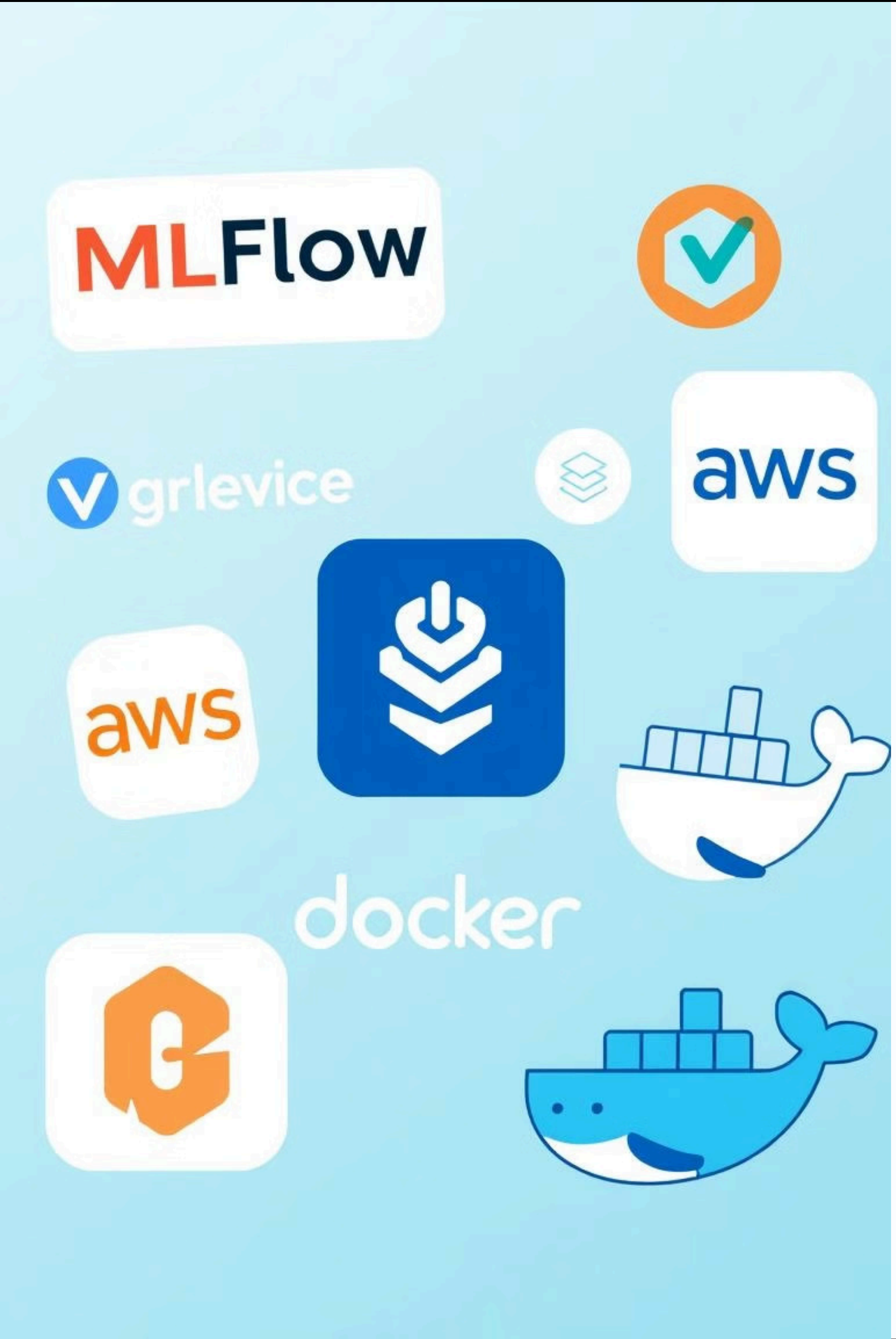
Predicting Loan Default Risk Using MLOps in Retail Banking

Personal loans are a significant source of revenue for retail banks, but they come with inherent risks of defaults. The project purpose consist to build a predictive model to estimate the probability of default for each customer based on their characteristics.

This project aims to develop an end-to-end MLOps pipeline to predict loan defaults and deploy the best-performing model on Amazon Web Services (AWS) using Streamlit.



Project Overview and Tools

MLflow

Model tracking and management

Streamlit

Interactive web app for model deployment

AWS

Cloud platform for hosting and scalability

Git & Docker

Version control and containerization for consistent deployment

ML Lifecycle: Planning

1. **Business Context**

   High default rates on personal loans threaten the bank's revenue.

2. **Success Metrics**

   AUC-ROC, Precision-Recall AUC, F1-Score, Recall, and Precision are key metrics for evaluating model performance.

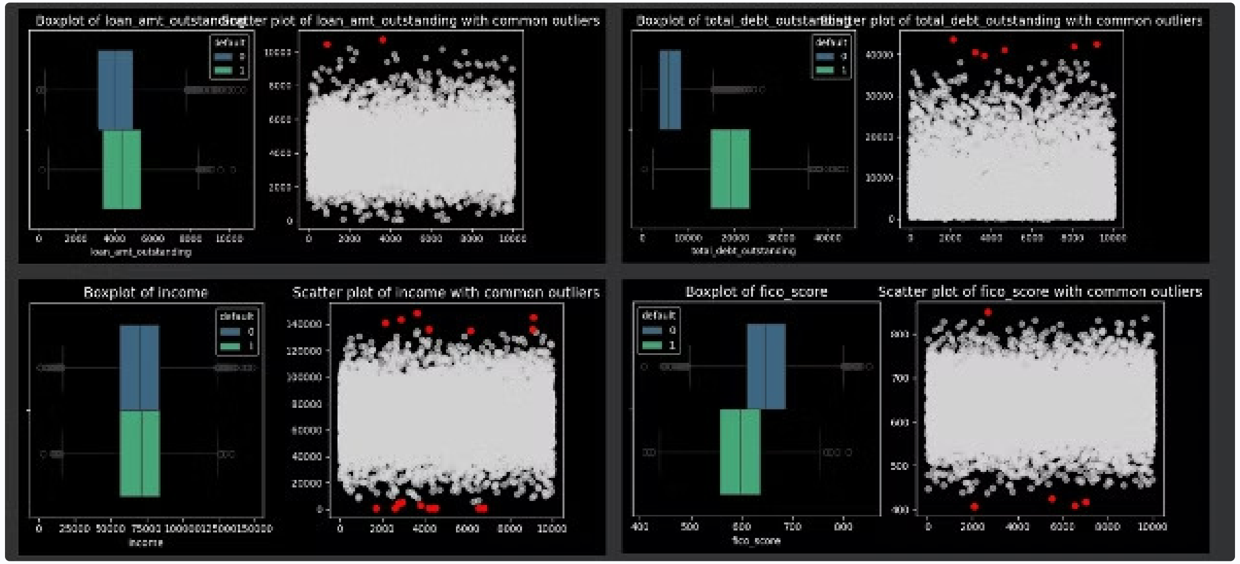
3. **MlOps Lifecycle**

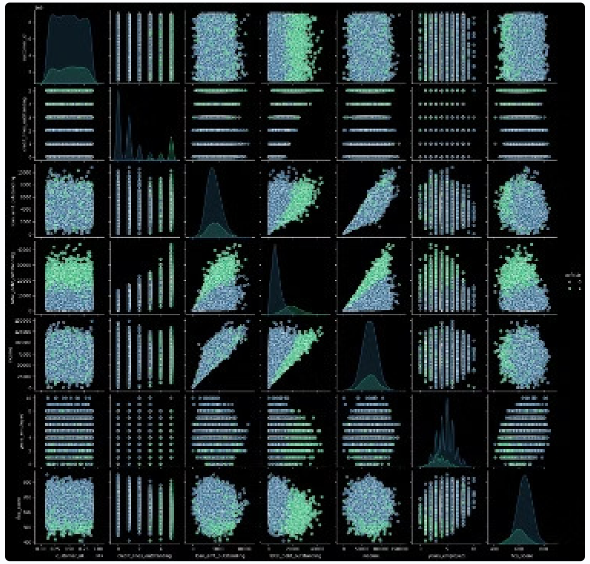
   Includes stages such as data collection and preprocessing, model development, model validation, deployment, and monitoring.

MLOps Lifecycle: Exploratory Data Analysis

EDA help understand the relationships between variables, identify missing data, detect outliers, and evaluate data quality.

- **Outliers (z-score + IQR bound adjustement) :**
  - **loan_amt_outstanding** (2.80), **total_debt_outstanding** (3.90), **income** (1.90), **fico_score** (2.00)





MLOps Lifecycle: Data Preparation and Model Engineering

1. **Data Cleaning**

   Handling missing values and removing outliers.

2. **Feature Engineering**

   Created relevant features such as Debt-to-Income Ratio.

3. **Data Splitting**

   Training, validation, testing split to avoid overfitting and ensure generalizability.

4. **Algorithm Selection**

   Random Forest Classifier, XGBoost Classifier, LightGBM Classifier.

5. **Hyperparameter Tuning**

   Used Optuna to optimize model parameters for all three classifiers.

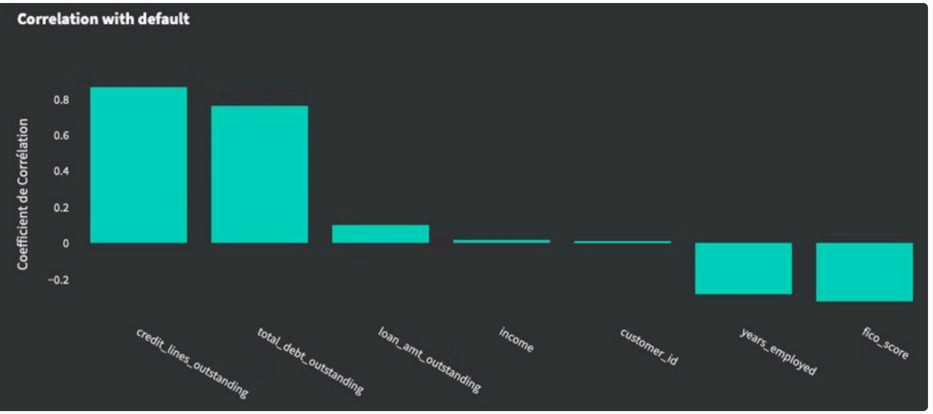
6. **Model Tracking**

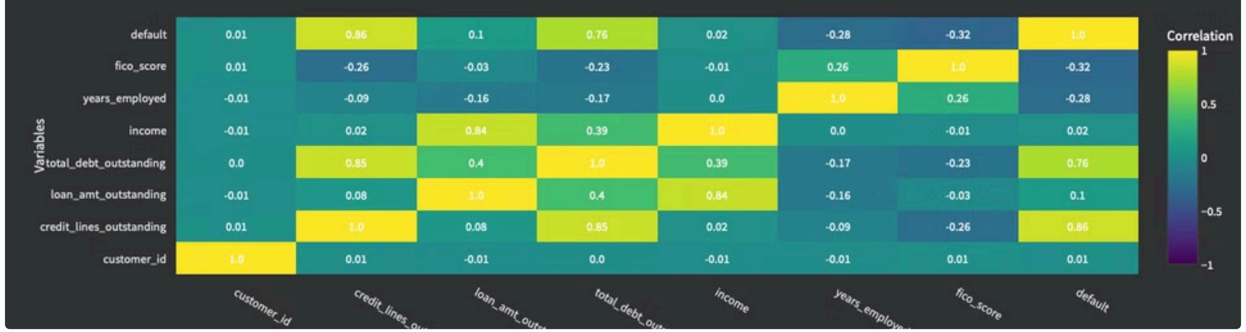
   Each experiment was tracked in MLflow, capturing metrics such as Accuracy, F1 score, Precision, Recall, and model parameters.

MLOps Lifecycle: Features engineering

Correlations with 'default'

	default
credit_lines_outstanding	0.8628
total_debt_outstanding	0.7589
loan_amt_outstanding	0.099
income	0.0163
customer_id	0.0069
years_employed	-0.2845
fico_score	-0.3245



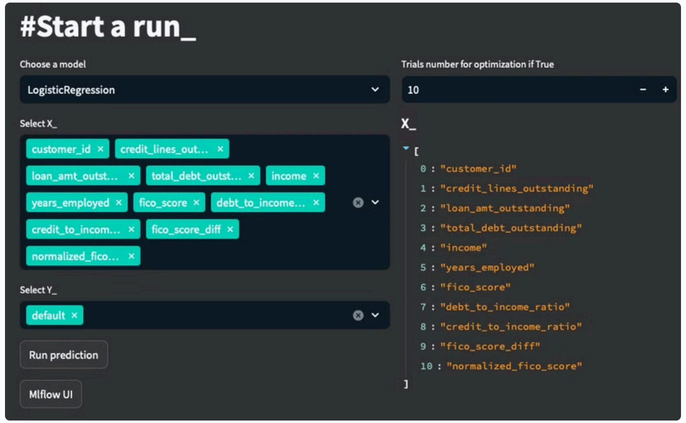



```
X_
[
  0 : "customer_id"
  1 : "credit_lines_outstanding
  2 : "loan_amt_outstanding"
  3 : "total_debt_outstanding"
  4 : "income"
  5 : "years_employed"
  6 : "fico_score"
  7 : "debt_to_income_ratio"
  8 : "credit_to_income_ratio"
  9 : "fico_score_diff"
  10 : "normalized_fico_score"
]
```

MLOps Lifecycle: Experiments with MLflow

Experiment Tracking

MLflow's experiment tracking feature helps visualize model performance over time and identify the best-performing models.



Model Management

MLflow streamlines model management, allowing for easy version control, deployment, and reproducibility of experiments.

Code Optimization

MLflow facilitates code optimization by simplifying model experimentation and allowing for better control over the model development process.

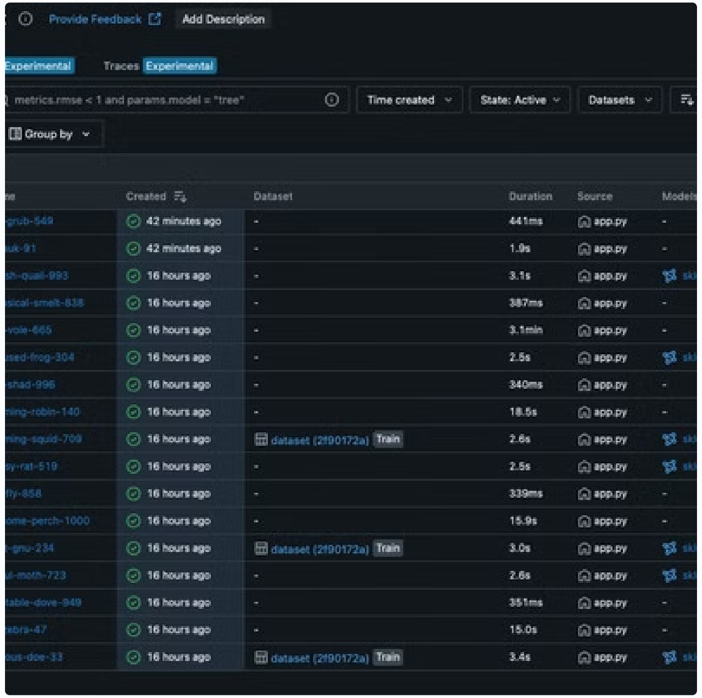
MLOps Lifecycle : optimisation with Optuna

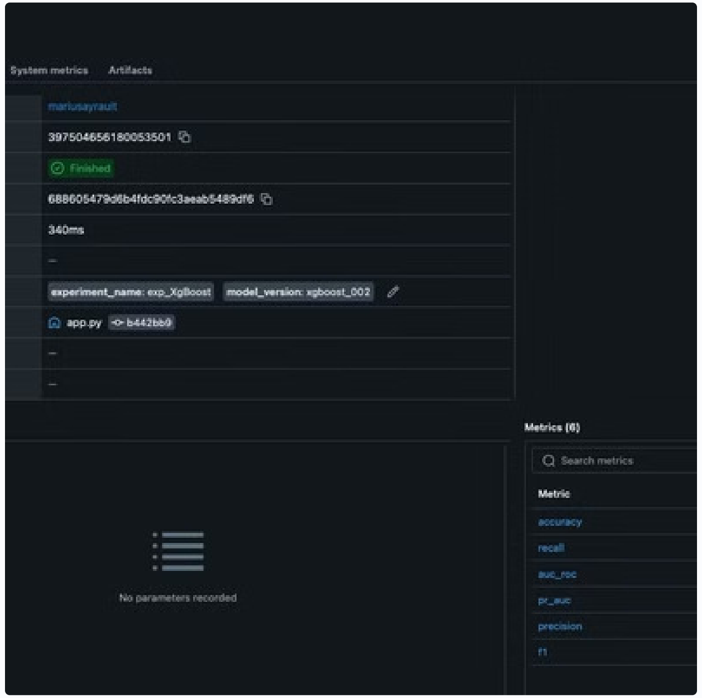
```python
def objective(self, trial, X_train, y_train, X_val, y_val):
    """
    Objective function for Optuna to optimize hyperparameters.

    :param trial: Optuna trial object.
    :param X_train: Feature matrix for training.
    :param y_train: Target vector for training.
    :param X_val: Feature matrix for validation.
    :param y_val: Target vector for validation.
    :return: Loss value to minimize.
    """
    if self.model_type == 'random_forest':
        params = {
            'n_estimators': trial.suggest_int('n_estimators', 50, 200),
            'max_depth': trial.suggest_int('max_depth', 2, 20),
            'min_samples_split': trial.suggest_int('min_samples_split', 2, 10),
            'random_state': 42
        }
    elif self.model_type == 'logistic_regression':
        params = {
            'C': trial.suggest_float('C', 1e-3, 1e2, log=True),
            'max_iter': trial.suggest_int('max_iter', 100, 1000)
        }
    elif self.model_type == 'xgboost':
        params = {
            'n_estimators': trial.suggest_int('n_estimators', 50, 200),
            'max_depth': trial.suggest_int('max_depth', 3, 15),
            'learning_rate': trial.suggest_float('learning_rate', 0.01, 0.3),
            'gamma': trial.suggest_float('gamma', 0, 1),
            'subsample': trial.suggest_float('subsample', 0.5, 1.0),
            'colsample_bytree': trial.suggest_float('colsample_bytree', 0.5, 1.0),
            'random_state': 42
        }
```

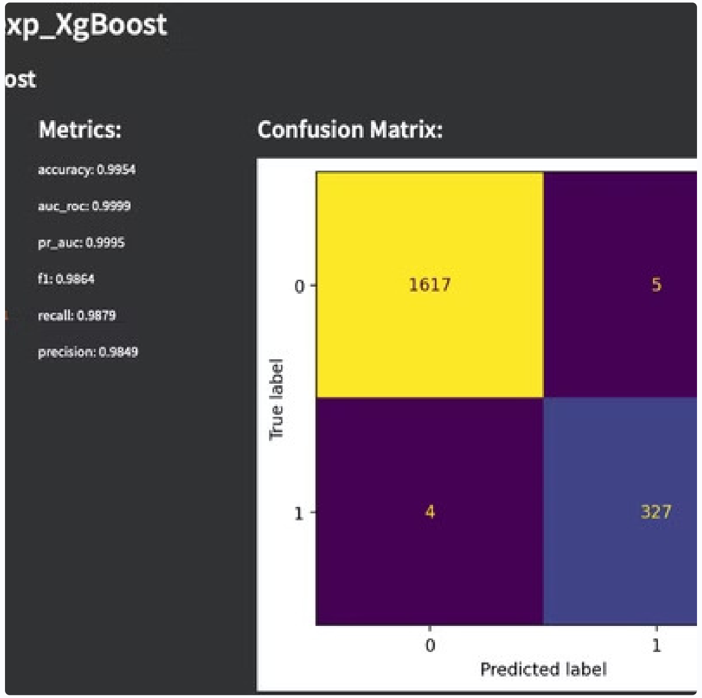
```python
self.initialize_model(params)
self.train(X_train, y_train)

y_val_pred_proba = self.model.predict_proba(X_val)[:, 1]
y_val_pred = self.model.predict(X_val)

auc_roc = roc_auc_score(y_val, y_val_pred_proba)
pr_auc = average_precision_score(y_val, y_val_pred_proba)
f1 = f1_score(y_val, y_val_pred)
recall = recall_score(y_val, y_val_pred)
precision = precision_score(y_val, y_val_pred)

composite_score = (0.4 * auc_roc) + (0.3 * pr_auc) + (0.2 * f1) + (0.05 * recall) + (0.05 * precision)

return -composite_score
```

```
[I 2024-09-09 13:49:12,889] A new study created in memory with name: no-name-c2d4e228-e97f-4607-a12c-37aed5a28cb5
[I 2024-09-09 13:49:13,659] Trial 0 finished with value: -0.9893276069823747 and parameters: {'n_estimators': 81, 'max_depth': 5, 'min_samples_split': 8}. Best is trial 0 with value: -0.9893276069823747.
[I 2024-09-09 13:49:15,056] Trial 1 finished with value: -0.9931490419596161 and parameters: {'n_estimators': 127, 'max_depth': 11, 'min_samples_split': 4}. Best is trial 1 with value: -0.9931490419596161.
[I 2024-09-09 13:49:16,610] Trial 2 finished with value: -0.9889958223822696 and parameters: {'n_estimators': 167, 'max_depth': 5, 'min_samples_split': 5}. Best is trial 1 with value: -0.9931490419596161.
[I 2024-09-09 13:49:18,599] Trial 3 finished with value: -0.9930912545056132 and parameters: {'n_estimators': 197, 'max_depth': 16, 'min_samples_split': 6}. Best is trial 1 with value: -0.9931490419596161.
[I 2024-09-09 13:49:19,848] Trial 4 finished with value: -0.992540138126755 and parameters: {'n_estimators': 119, 'max_depth': 8, 'min_samples_split': 5}. Best is trial 1 with value: -0.9931490419596161.
[I 2024-09-09 13:49:20,495] Trial 5 finished with value: -0.9900285995373841 and parameters: {'n_estimators': 65, 'max_depth': 6, 'min_samples_split': 5}. Best is trial 1 with value: -0.9931490419596161.
[I 2024-09-09 13:49:22,264] Trial 6 finished with value: -0.989448734881055 and parameters: {'n_estimators': 195, 'max_depth': 5, 'min_samples_split': 9}. Best is trial 1 with value: -0.9931490419596161.
[I 2024-09-09 13:49:24,172] Trial 7 finished with value: -0.9930910616983195 and parameters: {'n_estimators': 188, 'max_depth': 12, 'min_samples_split': 6}. Best is trial 1 with value: -0.9931490419596161.
[I 2024-09-09 13:49:25,031] Trial 8 finished with value: -0.9884832587934579 and parameters: {'n_estimators': 106, 'max_depth': 4, 'min_samples_split': 2}. Best is trial 1 with value: -0.9931490419596161.
[I 2024-09-09 13:49:26,254] Trial 9 finished with value: -0.993980076321552 and parameters: {'n_estimators': 118, 'max_depth': 14, 'min_samples_split': 9}. Best is trial 9 with value: -0.993980076321552.
```

MLOps Lifecycle: Model Evaluation







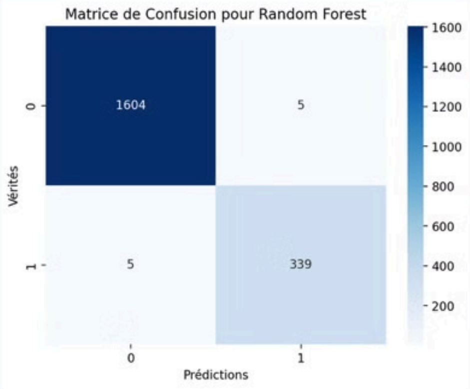
MlOps lifecycle : production models

Predict Loan Default

Random Forest_:

Classification Report pour Random Forest:

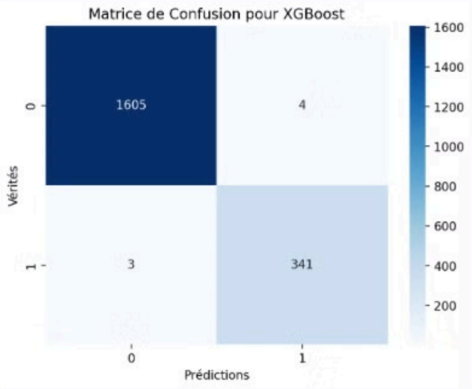
	precision	recall	f1-score	support
0.0	0.9969	0.9969	0.9969	1,609
1.0	0.9855	0.9855	0.9855	344
accuracy	0.9949	0.9949	0.9949	0.9949
macro avg	0.9912	0.9912	0.9912	1,953
weighted avg	0.9949	0.9949	0.9949	1,953



XGBoost_:

Classification Report pour XGBoost:

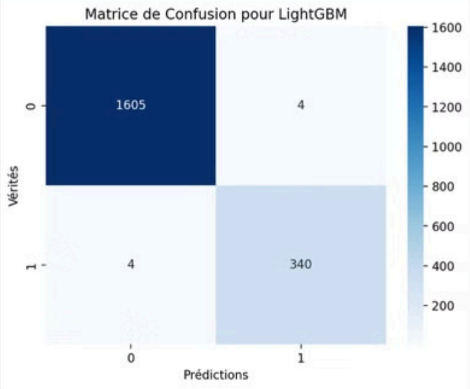
	precision	recall	f1-score	support
0.0	0.9981	0.9975	0.9978	1,609
1.0	0.9884	0.9913	0.9898	344
accuracy	0.9964	0.9964	0.9964	0.9964
macro avg	0.9933	0.9944	0.9938	1,953
weighted avg	0.9964	0.9964	0.9964	1,953



LightGBM_:

Classification Report pour LightGBM:

	precision	recall	f1-score	support
0.0	0.9975	0.9975	0.9975	1,609
1.0	0.9884	0.9884	0.9884	344
accuracy	0.9959	0.9959	0.9959	0.9959
macro avg	0.9929	0.9929	0.9929	1,953
weighted avg	0.9959	0.9959	0.9959	1,953



MLOps Lifecycle: Model Deployment

1

Containerization

Model was containerized using Docker for consistent deployment.

2

AWS Deployment

Deployed to an AWS ECR instance for scalability and reliability.

3

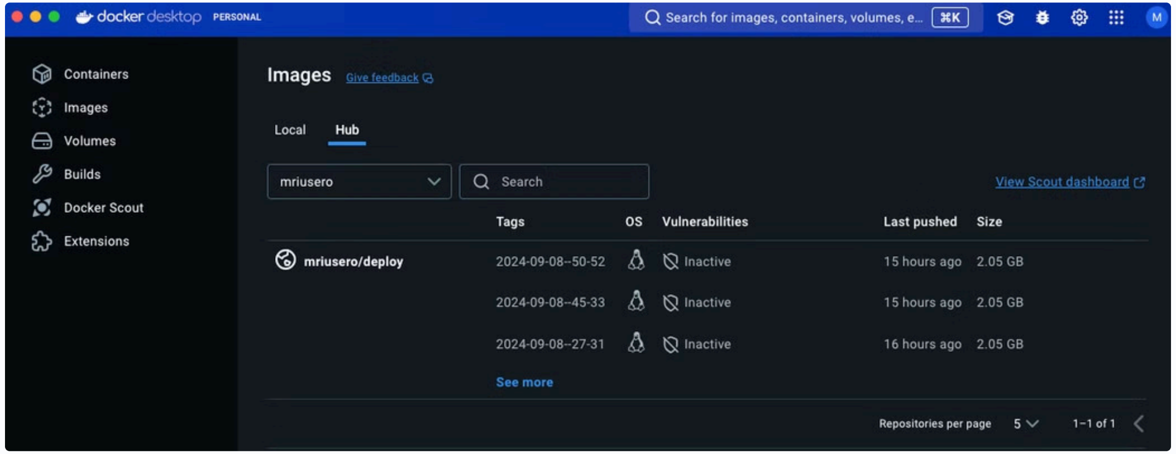
CI/CD Pipeline

Implemented using GitHub Actions to automate the deployment process.

MLOps Lifecycle: Docker Containerization

- Dependency Management with Poetry
- Docker Containerization in Docker Hub Repository

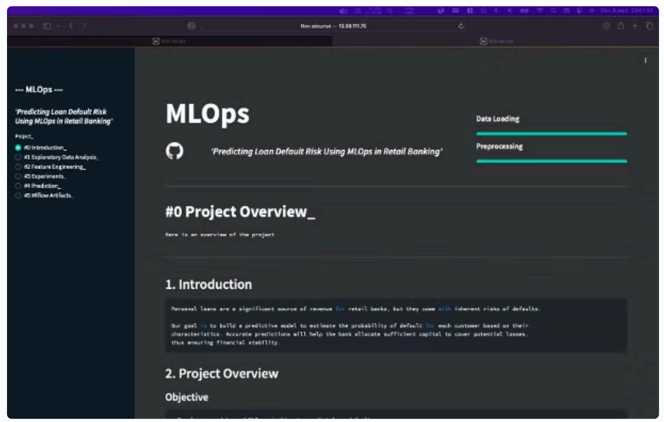
```toml
[tool.poetry]
package-mode = false
name = "projet-sda-mlops"
version = "0.1.0"
description = "1st try of poetry"
authors = ["mriusero <marius.ayrault@outlook.com>"]
license = "MIT"
readme = "README.md"

[tool.poetry.dependencies]
```



MLOps Lifecycle: AWS Deployment

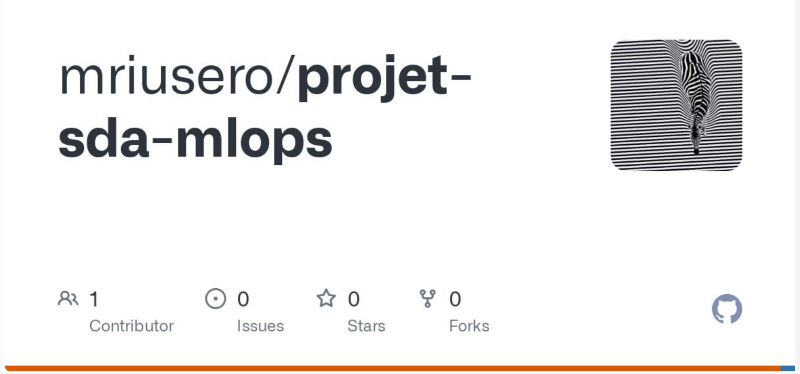
The model was deployed to an AWS Elastic Container Registry (ECR) instance. This provided scalability and reliability, allowing the model to handle increasing data volumes and user requests.

AWS ECR ensured secure storage and version control for the Docker containerized model. This facilitated efficient deployment updates and rollbacks.



MLOps Lifecycle: CI/CD Pipeline

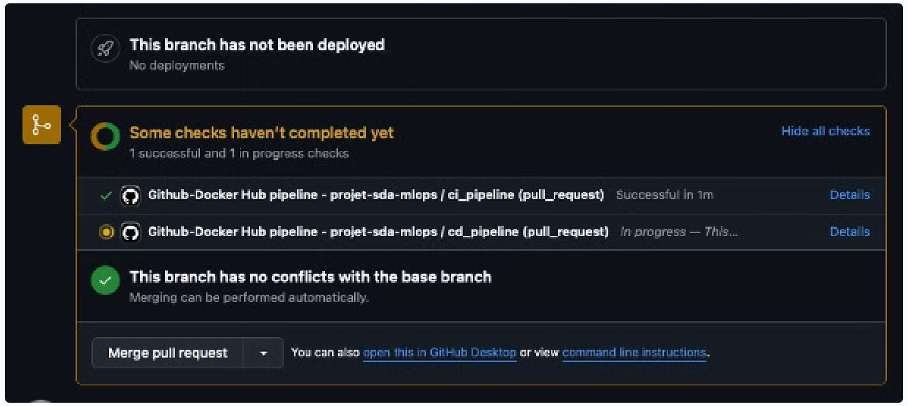
A Continuous Integration and Continuous Delivery (CI/CD) pipeline was implemented using GitHub Actions. The pipeline including code building, testing, containerization, and deployment to the AWS ECR instance.

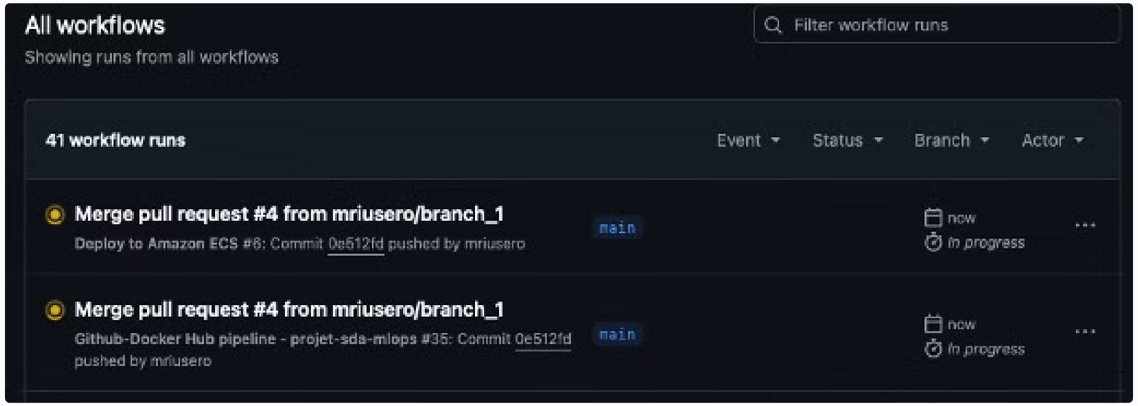


GitHub

GitHub - mriusero/projet-sda-mlops

Contribute to mriusero/projet-sda-mlops development by creating an account on GitHub.





Benefits of MLOps Implementation and Conclusion

Improved Efficiency

Automated model training, deployment, and monitoring.

Enhanced Collaboration

Better teamwork between data scientists and DevOps teams.

Faster Deployment

Reduced model deployment time from weeks to hours.

Reduced Errors

Consistent environments using Docker and CI/CD.

This project demonstrates the importance of MLOps in delivering reliable, scalable, and explainable machine learning solutions.