

## Données publiques

Dans chaque model nous allons :

- Séparer les données en train et test

```
X_train, X_test = train_test_split(X_all, test_size=0.3, random_state=42)
```

*Figure 5- Erreur ! Il n'y a pas de texte répondant à ce style dans ce document.-1 : séparation des données public en train et test*

- Faire une Injection des anomalies simulées

```
def inject_anomalies_targeted(X, df_original, frac=0.12, seed=42):
    np.random.seed(seed)
    n = X.shape[0]
    n_anom = max(1, int(n * frac))
    idx = np.random.choice(n, size=n_anom, replace=False)
    X_ = X.copy()
    n_num = len(num_cols)
    for i in idx:
        if n_num > 0:
            feats = np.random.choice(n_num, size=max(1, n_num//2), replace=False)
            X_[i, feats] = X_[i, feats] * (np.random.uniform(6, 30)) + np.random.uniform(10, 300)
    labels = np.zeros(n, dtype=int)
    labels[idx] = 1
    return X_, labels

X_test_with_anom, y_test = inject_anomalies_targeted(X_test, df, frac=0.15)
print("Test size:", X_test.shape[0], "Anomalies injected:", y_test.sum())
```

*Figure 5-Erreur ! Il n'y a pas de texte répondant à ce style dans ce document.-2 : Injection anomalies simulées*

- Faire entrainer l'algorithme Isolation Forest sur les données publiques

```

iso = IsolationForest(n_estimators=200, contamination=y_test.mean() if y_test.mean()>0 else 0.1, random_state=42)
iso.fit(X_train)
scores_iso = -iso.decision_function(X_test_with_anom)

cont = max(0.01, min(0.5, y_test.mean()))
th = np.percentile(scores_iso, 100*(1-cont))
y_pred_iso = (scores_iso >= th).astype(int)

f1_iso_macro = f1_score(y_test, y_pred_iso, average='macro', zero_division=0)
recall_iso_macro = recall_score(y_test, y_pred_iso, average='macro', zero_division=0)
print("\n--- IsolationForest (macro) ---")
print("F1 macro:", f1_iso_macro)
print("Recall macro:", recall_iso_macro)
print(classification_report(y_test, y_pred_iso, zero_division=0))
print("ConfusionMatrix (iso):\n", confusion_matrix(y_test, y_pred_iso))

```

*Figure 5-Erreur ! Il n'y a pas de texte répondant à ce style dans ce document.-3 : entraînement de l'algorithme Isolation Forest sur données publique*

- Pour enfin avoir le résultat (l'accuracy)

```

--- IsolationForest (macro) ---
F1 macro: 0.9126840969089549
Recall macro: 0.912814168910292
      precision    recall   f1-score   support
          0       0.97     0.97     0.97    14810
          1       0.85     0.85     0.85     2613

      accuracy                           0.96    17423
      macro avg       0.91     0.91     0.91    17423
  weighted avg       0.96     0.96     0.96    17423

ConfusionMatrix (iso):
[[14421  389]
 [ 387 2226]]

```

*Figure 5-Erreur ! Il n'y a pas de texte répondant à ce style dans ce document.-4 : résultat model Isolation sur données public*

Données privées

Dans chaque model nous allons :

- Séparer les données en train et test

```
# Split train/test
# =====
X_train, X_test = train_test_split(X_all, test_size=0.3, random_state=42)
```

*Figure 5-Erreur ! Il n'y a pas de texte répondant à ce style dans ce document.-5 : séparation des données privées train et test*

- Faire une Injection des anomalies simulées

```
# Injection anomalies
# =====
def inject_anomalies(X, frac=0.15, seed=42):
    np.random.seed(seed)
    n = X.shape[0]
    n_anom = int(n*frac)
    idx = np.random.choice(n, n_anom, replace=False)
    X_ = X.copy()
    n_num = len(num_cols)
    for i in idx:
        feats = np.random.choice(n_num, size=max(1,n_num//2), replace=False)
        X_[i, feats] = X_[i, feats] * np.random.uniform(6,30) + np.random.uniform(10,300)
    labels = np.zeros(n)
    labels[idx] = 1
    return X_, labels

X_test_with_anom, y_test = inject_anomalies(X_test, frac=0.15)
```

*Figure 5-Erreur ! Il n'y a pas de texte répondant à ce style dans ce document.-6 : Injection anomalies simulées*

- Faire entrainer l'algorithme Isolation Forest sur les données privées

```

# Isolation Forest
# =====
iso = IsolationForest(
    n_estimators=200,
    contamination=y_test.mean() if y_test.mean()>0 else 0.1,
    random_state=42
)
iso.fit(X_train)
y_pred = iso.predict(X_test_with_anom)
y_pred = np.where(y_pred == 1, 0, 1)

# =====
# Scores
# =====
f1 = f1_score(y_test, y_pred, average='macro')
recall = recall_score(y_test, y_pred, average='macro')
acc = accuracy_score(y_test, y_pred)

print("== Résultats Isolation Forest ==")
print(f"Accuracy : {acc:.4f}")
print(f"F1-macro : {f1:.4f}")
print(f"Recall-macro: {recall:.4f}")

```

*Figure 5-Erreur ! Il n'y a pas de texte répondant à ce style dans ce document.-7 : entraînement de l'algorithme Isolation Forest sur données privées*

- Pour enfin avoir le résultat (l'accuracy)

```

== Résultats Isolation Forest ==
Accuracy : 0.8387
F1-macro : 0.6997
Recall-macro: 0.7113

```

*Figure 5-Erreur ! Il n'y a pas de texte répondant à ce style dans ce document.-8 : résultat model Isolation sur données privées*