Données publiques

- On entraine l'algorithme Sparse AE sur données publiques

```python
# --- Sparse Autoencoder ---
def sparse_autoencoder():
    inputs = layers.Input(shape=(input_dim,))
    e = layers.Dense(encoding_dim*2, activation='relu', activity_regularizer=tf.keras.regularizers.l1(1e-5))(inputs)
    e = layers.Dense(encoding_dim, activation='relu')(e)
    bottleneck = layers.Dense(encoding_dim//2, activation='relu')(e)
    d = layers.Dense(encoding_dim, activation='relu')(bottleneck)
    outputs = layers.Dense(input_dim, activation='linear')(d)
    return models.Model(inputs, outputs)

model = sparse_autoencoder()
model.compile(optimizer='adam', loss='mse')
model.fit(X_train, X_train, epochs=30, batch_size=32, verbose=0)

recon = model.predict(X_test_with_anom)
mse_test = np.mean(np.square(X_test_with_anom - recon), axis=1)
threshold = np.percentile(np.mean(np.square(X_train - model.predict(X_train)), axis=1), 95)
y_pred = (mse_test > threshold).astype(int)

f1_macro = f1_score(y_test, y_pred, average='macro')
recall_macro = recall_score(y_test, y_pred, average='macro')

print("=== Résultats Sparse Autoencoder ===")
print(f"F1-macro : {f1_macro:.4f}, Recall-macro : {recall_macro:.4f}")
```

*Figure 5-**Erreur ! Il n'y a pas de texte répondant à ce style dans ce document.**-1 : entrainement de l'algorithme Sparse AE sur données publique*

- *Voici le résultat du model Sparse AE sur données publique*

```
545/545 [==============================] - 1s 2ms/step
1271/1271 [==============================] - 2s 2ms/step
=== Résultats Sparse Autoencoder ===
F1-macro : 0.9246, Recall-macro : 0.9737
```

*Figure 5-**Erreur ! Il n'y a pas de texte répondant à ce style dans ce document.**-2 : résultat model Sparse AE sur données publique*

Données privées

- On entraine l'algorithme Sparse AE sur données privées

```python
# Sparse Autoencoder
# ========================
input_dim = X_train.shape[1]
encoding_dim = max(4, input_dim // 3)

inputs = layers.Input(shape=(input_dim,))
e = layers.Dense(encoding_dim*2, activation='relu', activity_regularizer=tf.keras.regularizers.l1(1e-5))(inputs)
e = layers.Dense(encoding_dim, activation='relu')(e)
bottleneck = layers.Dense(encoding_dim//2, activation='relu')(e)
d = layers.Dense(encoding_dim, activation='relu')(bottleneck)
outputs = layers.Dense(input_dim, activation='linear')(d)
model = models.Model(inputs, outputs)

model.compile(optimizer='adam', loss='mse')
model.fit(X_train, X_train, epochs=30, batch_size=32, verbose=0)

# Reconstruction
recon_test = model.predict(X_test_with_anom)
mse_test = np.mean(np.square(X_test_with_anom - recon_test), axis=1)
threshold = np.percentile(np.mean(np.square(X_train - model.predict(X_train)), axis=1), 95)
y_pred = (mse_test > threshold).astype(int)

# Metrics
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)
```

*Figure 5-**Erreur ! Il n'y a pas de texte répondant à ce style dans ce document.**-3 : entrainement de l'algorithme Sparse AE sur données privées*

```python
print("=== Résultats Sparse Autoencoder ===")
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.**-4 : entrainement de l'algorithme Sparse AE sur données privées*

- *Voici le résultat du model Sparse AE sur données privées*

```
11/11 [==============================] - 0s 2ms/step
25/25 [==============================] - 0s 2ms/step
=== Résultats Sparse Autoencoder ===
Accuracy    : 0.9648
F1-macro    : 0.9368
Recall-macro: 0.9793
```

*Figure 5-**Erreur ! Il n'y a pas de texte répondant à ce style dans ce document.**-5 : résultat de l'algorithme Sparse AE sur données privées*