Données publiques

- On entraine l'algorithme Variational AE sur données publiques

```python
# --- Variational Autoencoder ---
def variational_autoencoder():
    inputs = layers.Input(shape=(input_dim,))
    z = layers.Dense(encoding_dim, activation='relu')(inputs)
    z_mean = layers.Dense(encoding_dim//2)(z)
    z_log_var = layers.Dense(encoding_dim//2)(z)
    def sampling(args):
        mean, log_var = args
        epsilon = tf.random.normal(shape=tf.shape(mean))
        return mean + tf.exp(0.5*log_var)*epsilon
    z_samp = layers.Lambda(sampling)([z_mean, z_log_var])
    outputs = layers.Dense(input_dim, activation='linear')(z_samp)
    return models.Model(inputs, outputs)

model = variational_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 Variational 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 Variational AE sur données publique*

- Voici le résultat du model Variational AE sur données publique

```
545/545 [==============================] - 1s 1ms/step
1271/1271 [==============================] - 2s 2ms/step
=== Résultats Variational Autoencoder ===
F1-macro : 0.9226, Recall-macro : 0.9721
```

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

Données privées

- On entraine l'algorithme Variational AE sur données privées

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

inputs = layers.Input(shape=(input_dim,))
z = layers.Dense(encoding_dim, activation='relu')(inputs)
z_mean = layers.Dense(encoding_dim//2)(z)
z_log_var = layers.Dense(encoding_dim//2)(z)

def sampling(args):
    mean, log_var = args
    epsilon = tf.random.normal(shape=tf.shape(mean))
    return mean + tf.exp(0.5*log_var)*epsilon

z_samp = layers.Lambda(sampling)([z_mean, z_log_var])
outputs = layers.Dense(input_dim, activation='linear')(z_samp)
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 Variational AE sur données privées*

```
print("=== Résultats Variational 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 : suite entrainement de l'algorithme Variational AE sur données privées*

- Voici le résultat du model Variational AE sur données privées

```
11/11 [==============================] - 0s 2ms/step
25/25 [==============================] - 0s 4ms/step
=== Résultats Variational Autoencoder ===
Accuracy    : 0.9795
F1-macro    : 0.9618
Recall-macro: 0.9879
```

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