

Données publiques

- On fait une simulation d'injection d'anomalies sur la donnée publiques

```
# Injection d'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()
    for i in idx:
        feats = np.random.choice(len(num_cols), size=2, replace=False)
        X_[i, feats] = X_[i, feats] * np.random.uniform(6,30)
    labels = np.zeros(n)
    labels[idx] = 1
    return X_, labels

X_test_with_anom, y_test = inject_anomalies(X_test)

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

Figure 5-Erreur ! Il n'y a pas de texte répondant à ce style dans ce document.-1 : Injection anomalies simulées

On définit l'algorithme DenseMetaAE

```

# -----
# 2 Dense Autoencoder
# =====
def dense_autoencoder():
    inputs = layers.Input(shape=(input_dim,))
    e = layers.Dense(encoding_dim*2, activation='relu')(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)
    d = layers.Dense(encoding_dim*2, activation='relu')(d)
    outputs = layers.Dense(input_dim, activation='linear')(d)
    return models.Model(inputs, outputs)

# Adaptation rapide (meta-Learning)
def adapt_ae(model, X_support, lr=1e-3, steps=5):
    optimizer = tf.keras.optimizers.Adam(lr)
    X_tensor = tf.convert_to_tensor(X_support, dtype=tf.float32)
    for _ in range(steps):
        with tf.GradientTape() as tape:
            recon = model(X_tensor, training=True)
            loss = tf.reduce_mean(tf.square(recon - X_tensor))
            grads = tape.gradient(loss, model.trainable_variables)
            optimizer.apply_gradients(zip(grads, model.trainable_variables))

# =====

```

Figure 5-Erreur ! Il n'y a pas de texte répondant à ce style dans ce document.-2 : entraînement de l'algorithme DenseMetaAE sur données publique

On entraîne l'algorithme DenseMetaAE et fait le test

```

''' -----
# 3 Entraînement + Test
# =====
model = dense_autoencoder()
model.compile(optimizer='adam', loss='mse')
model.fit(X_train, X_train, epochs=30, batch_size=32, verbose=1)

support_idx = np.random.choice(X_test_with_anom.shape[0], size=10, replace=False)
X_support = X_test_with_anom[support_idx]
adapt_ae(model, X_support)

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 Dense 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.-3 : entraînement de l'algorithme DenseMetaAE sur données publique

Voici le résultat

```

12/1/12/1 [=====] - 25 ms/50
=== Résultats Dense Autoencoder ===
F1-macro : 0.9240, Recall-macro : 0.9713

```

Figure 5-Erreur ! Il n'y a pas de texte répondant à ce style dans ce document.-4 : résultat de l'algorithme DenseMetaAE sur données publique

Données privées

```

'''
# 4 Injection anomalies simulées
# =====
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)

```

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

- On fait le même processus pour les données privées

```

# 5 Dense Autoencoder
# =====
input_dim = X_train.shape[1]
encoding_dim = max(4, input_dim // 3)

def dense_meta_ae():
    inputs = layers.Input(shape=(input_dim,))
    e = layers.Dense(encoding_dim*2, activation='relu')(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)
    d = layers.Dense(encoding_dim*2, activation='relu')(d)
    outputs = layers.Dense(input_dim, activation='linear')(d)
    return models.Model(inputs, outputs)

# Adaptation rapide (meta-learning)
def adapt_ae(model, X_support, lr=1e-3, steps=5):
    optimizer = tf.keras.optimizers.Adam(learning_rate=lr)
    X_tensor = tf.convert_to_tensor(X_support, dtype=tf.float32)
    for _ in range(steps):
        with tf.GradientTape() as tape:
            recon = model(X_tensor, training=True)
            loss = tf.reduce_mean(tf.square(recon - X_tensor))
            grads = tape.gradient(loss, model.trainable_variables)
            optimizer.apply_gradients(zip(grads, model.trainable_variables))

```

Figure 5-Erreur ! Il n'y a pas de texte répondant à ce style dans ce document.-6 : entraînement de l'algorithme DenseMetaAE sur données privées

```
# =====
# 6 Entraînement + Test
# =====
model = dense_meta_ae()
model.compile(optimizer='adam', loss='mse')
model.fit(X_train, X_train, epochs=30, batch_size=32, verbose=1)

support_idx = np.random.choice(X_test_with_anom.shape[0], size=10, replace=False)
X_support = X_test_with_anom[support_idx]
adapt_ae(model, X_support)

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 Dense Meta Autoencoder (Private Data) ===")
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.-7 : entraînement de l'algorithme DenseMetaAE sur données privées

```
25/25 [=====] - 0s 2ms/step
=== Résultats Dense Meta Autoencoder (Private Data) ===
F1-macro : 0.9273, Recall-macro : 0.9759
```

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