département
Informatique

Université Claude Bernard Lyon 1

Année universitaire	**2025-2026**		
Filière	**Informatique**	**Année**	**M2 DS**
Module	**Machine Learning**		
Enseignants	**Haytham Elghazel et Ichrak Ennaceur**		
Intitulé :	**Atelier : Détection d'anomalies avec Python**		
Contenu	• **Détection d'anomalies : outliers/nouveautés** • **Approches non supervisées : Isolation Forest, LOF** • **Approches supervisées pour la détection d'anomalie** • **Apprentissage sur des données déséquilibrées (Imbalanced Learning)** • **Évaluation en présence de données déséquilibrées**		
Rendu	**Rendu sur Moodle : Un code factorisé, bien structuré et commenté**		

Dans cet atelier pratique, vous allez expérimenter des algorithmes de traitement de données pour répondre à différents problèmes liés à la détection d'anomalies avec le langage **Python**.

Pour lancer le notebook Python, il faut taper la commande **jupyter notebook** dans votre dossier de travail. Une fenêtre va se lancer dans votre navigateur pour ouvrir l'application Jupyter. Créer un nouveau notebook Python et taper le code suivant dans une nouvelle cellule :

```
import numpy as np
np.set_printoptions(threshold=10000, suppress = True)
import pandas as pd
import warnings
import matplotlib.pyplot as plt
warnings.filterwarnings('ignore')
```

La détection d'anomalies (dite aussi détection d'outliers ou détection de nouveauté dans certains cas) est une tâche de l'apprentissage automatique qui consiste à déceler dans les données, les instances (individus) ayant un comportement différent (inhabituel) des autres instances de la base dites normales. Dans le cas de la détection des fraudes par exemple, toute dépense très étrange par carte de crédit est suspecte. Les variations possibles sont si nombreuses et les exemples de formation si rares, qu'il n'est pas possible de savoir à quoi ressemble une activité frauduleuse ou un incident. L'approche de la détection des anomalies consiste simplement à apprendre à quoi ressemble l'activité normale (à l'aide d'un historique de transactions supposées non-frauduleuses) et d'identifier tout ce qui est très différent. Différents algorithmes ont été proposées dans la littérature dont certaines sont supervisées et d'autres non supervisées. Pour plus de détails considérant ces approches, vous pouvez vous référer sur les diapositives du cours sur *moodle* ou sur l'aide de *scikit-learn*.

Scikit-learn propose différentes approches pour la détection d'anomalies (outliers et nouveautés). Nous nous intéressons ici à plusieurs approches (voir le cours sur Moodle) non supervisées et supervisées. Nous allons appliquer cette approche sur trois jeux de données **"mouse.txt"** (https://elki-project.github.io/datasets/) et le fichier **"creditcard.csv"** du challenge Kaggle (https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud) et la **Kddcup99** pour la détection d'intrusions dans les réseaux (accessibles sur *sklearn* pour la version complète corrigée, ou à partir du fichier moodle pour la version réduite ou sur directement sur kaggle également).

NB :

Avant de commencer, le code devra être **structuré, bien commenté et factorisé en fonctions modulaires** (visualisation, détection d'anomalies, évaluation, etc.) afin de pouvoir appeler directement ces fonctions pour chaque méthode testée. Toutes les fonctions seront **regroupées dans un fichier utils.py**.

Ajoutez également un bloc pour **fixer le random_state** afin d'assurer la reproductibilité.

Le TP doit être organisé de manière à :

- Appeler les fonctions depuis un **notebook principal** pour exécuter les expérimentations et générer les résultats/visualisations.
- Avoir une **structure claire, réutilisable et facilement extensible**, permettant de tester différentes méthodes et paramètres sans duplication de code.

1. Sur la base de données Mouse

Ce fichier contient 500 instances décrites par deux variables $\mathbf{x_1}$ et $\mathbf{x_2}$, représentant des points de la tête de Mickey Mouse. Les 10 dernières instances du fichier sont aberrantes (*outliers*).

- Télécharger ce jeu de données et analyser le.
- Donner une représentation graphique des données.
- Appliquer la technique **Isolation Forest** pour détecter les outliers dans ce jeu de données.
- Appliquer la technique **Local Outlier Factor** pour détecter les outliers dans ce jeu de données.
- Modifier la représentation graphique précédente pour visualiser les données aberrantes avec chacune de ces deux approches.
- Afficher **l'histogramme des scores d'anomalie** pour chacune des deux approches (à l'aide de ***seaborn*** ou ***matplotlib***)
- Pour chacune de ces deux approches, proposer deux méthodologies non supervisée (***clustering*** et ***écart interquartile* IQR**) permettant de mieux choisir le seuil de décision (anomalie ou non).
- Comparer les résultats obtenus numériquement et visuellement.
- Proposer une méthodologie pour la **détection de nouveautés** utilisant l'approche **Local Outlier Factor**[1] et analyser visuellement les résultats obtenus.

2. Sur le jeu de données des cartes de crédits et de détection d'intrusions dans les réseaux

L'objectif au début de cette partie est de **détecter les fraudes dans les transactions de cartes bancaires**. Pour plus d'informations sur ce jeu de données, vous pouvez visiter le lien suivant (https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud).

Ce jeu de données est très déséquilibré avec seulement 0.172% de fraudes (492 fraudes sur 284 807 transactions). Si vous rencontrez des problèmes de performance, vous pouvez utiliser un **échantillon aléatoire stratifié** de ce jeu de données. Sinon, vous pouvez travailler sur la totalité des données.

a) Préparation des données

Préparer ce jeu de données (ne pas utiliser la variable Time).

b) Méthodologie de détection d'outliers

Proposer une méthodologie pour **la détection d'outliers** en comparant plusieurs approches entre elles :

- L'approche EasyEnsemble disponible sur *imblearn*
- Les deux approches Isolation Forest et Local Outlier Factor.
- **Deux approches supervisées classiques**, combinées à des techniques de gestion du déséquilibre des classes :
  - *Undersampling* (ex. : **Tomek Links**, disponible dans *imblearn*),
  - *Oversampling* (ex. : **SMOTE**, disponible dans *imblearn*),

[1] https://scikit-learn.org/stable/modules/outlier_detection.html#outlier-detection

- et *balancing* (rééquilibrage global).
- **L'approche EasyEnsemble** (disponible dans imblearn).
- **Un modèle XGBoost**.
- **Deux approches non supervisées** : **Isolation Forest** et **Local Outlier Factor (LOF)**.

Points à prendre en compte dans votre méthodologie :

- Utiliser un **protocole de comparaison rigoureux** :
  - soit basé sur un **échantillonnage aléatoire stratifié**,
  - soit sur une **validation croisée stratifiée** (si les performances computationnelles le permettent).
- **Évaluer les performances** à l'aide de métriques appropriées (voir le cours sur Moodle) :
  - Matrice de confusion,
  - *F1-score* ou *Balanced Accuracy*,
  - *ROC AUC* ou *Average Precision Score*.
- **Normaliser les données** si nécessaire : StandardScaler, MinMaxScaler, ou RobustScaler. Bien choisir la bonne méthodologie.
- **Encoder les variables catégorielles** si besoin : OneHotEncoder ou OrdinalEncoder.
- **Afficher les courbes de performance** :
  - *ROC Curve* et/ou *Precision-Recall Curve*,
  - Indiquer le *Recall@Precision ≥ X*.
- **Optimiser les hyperparamètres** :
  - Pour Isolation Forest → nombre d'arbres (*n_estimators*),
  - Pour LOF → nombre de voisins (*n_neighbors*),
  - ainsi que le **seuil de décision optimal**.
- **Factoriser votre code** en fonctions modulaires pour automatiser la comparaison entre méthodes.

c) Application sur le jeu de données KDDCup99

Appliquer la même méthodologie à un second jeu de données : la **détection d'intrusions dans les réseaux** (KDDCup99).

3. Bonus (pour aller plus loin) :

- **Explicabilité** : proposer une approche d'interprétation des résultats d'**Isolation Forest sur** les **données Credit Card**, permettant d'identifier les **variables les plus importantes** dans la détection d'anomalies (via la bibliothèque **SHAP**).
- **Robustesse** : évaluer la robustesse des différentes approches par **injection d'anomalies synthétiques**, et **afficher graphiquement les scores de performance** en fonction du **taux d'anomalies injectées**.
- **Détection de nouveautés** : proposer une **méthodologie basée sur un *autoencodeur simple*** pour la **détection de nouveautés** sur les **données Mouse**.