# Rapport

Le code fourni effectue la détection de fraudes par carte de crédit en utilisant deux modèles différents : un modèle de forêts aléatoires (Random Forest) et un modèle de réseaux de neurones (MLPClassifier). Voici un compte rendu de chaque partie du code :

Partie 1 : Utilisation d'un modèle de forêts aléatoires

1. Suppression des échantillons avec des valeurs manquantes : Les échantillons avec des valeurs manquantes sont supprimés à la fois des données d'entraînement et des données de test en utilisant la méthode **‘dropna()**’. Les variables**’ X\_train\_dropna’** et**’ X\_test\_dropna’** contiennent les caractéristiques (X) après suppression des valeurs manquantes, et les variables **‘y\_train\_dropna**’ et **‘y\_test\_dropna**’ contiennent les variables cibles (y) correspondantes.
2. Construction du modèle de forêts aléatoires : Un objet ‘**RandomForestClassifie’r** est créé pour représenter le modèle de forêts aléatoires.
3. Entraînement du modèle : Le modèle de forêts aléatoires est entraîné sur les données d'entraînement sans valeurs manquantes à l'aide de la méthode **‘fit()’**. Les variables **‘X\_train\_dropna’** et**’ y\_train\_dropna**’ sont utilisées pour l'entraînement.
4. Prédiction des classes pour l'ensemble de test : Le modèle est utilisé pour prédire les classes des échantillons de l'ensemble de test sans valeurs manquantes en utilisant la méthode **‘predict()’**. Les prédictions sont stockées dans la variable **‘y\_pred\_rf**’.
5. Évaluation du modèle : La matrice de confusion et le rapport de classification sont affichés à l'aide des fonctions **‘confusion\_matrix()**’ et **‘classification\_report()’** respectivement. Ces mesures fournissent des informations sur les performances du modèle, y compris la précision, le rappel et le score F1 pour chaque classe.

Partie 2 : Utilisation d'un modèle de réseaux de neurones

1. Charger le jeu de données : Le jeu de données est chargé à partir d'un fichier CSV à l'aide de la fonction read\_csv() de pandas.
2. Suppression des échantillons avec des valeurs manquantes : Les échantillons avec des valeurs manquantes sont supprimés des données en utilisant la méthode dropna(). Les variables X\_dropna et y\_dropna contiennent les caractéristiques et les variables cibles respectivement après suppression des valeurs manquantes.
3. Diviser les données en ensembles d'entraînement et de test : Les données sont divisées en ensembles d'entraînement et de test à l'aide de la fonction train\_test\_split().
4. Construction du modèle de réseaux de neurones : Un objet MLPClassifier est créé pour représenter le modèle de réseaux de neurones avec une architecture de deux couches cachées de 64 neurones chacune.
5. Entraînement du modèle : Le modèle de réseaux de neurones est entraîné sur l'ensemble d'entraînement en utilisant la méthode fit(). Les variables X\_train et y\_train sont utilisées pour l'entraînement.
6. Prédiction des classes pour l'ensemble de test : Le modèle est