Fiche TP N°4

Prétraitement de données avec Python

Le but de ce TP est de mettre en œuvre les concepts de base de prétraitement de données à l'aide de Python. Par conséquent, l'ensemble des techniques de prétraitement de données étudiées et appliquées avec WEKA, dans la série 2, seront mises en œuvre avec Python.

Pour rappel, le prétraitement fait référence aux transformations appliquées à nos données avant de les transmettre à un modèle. Le prétraitement de données est une technique utilisée pour convertir les données brutes en un ensemble de données propres.

1. **Jeux de données utilisé :** L'objectif de l'ensemble de données est de prédire de manière diagnostique si un patient est diabétique ou non, sur la base de certaines mesures incluses dans l'ensemble de données. Ce dernier contient plusieurs variables prédictives médicales et d'une variable cible. Les variables prédictives comprennent le nombre de grossesses que la patiente a eues, son IMC, son taux d'insuline, son âge, etc. Le lien vers le dataset est le suivant : https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database

2. **Étapes de prétraitement des données :** le prétraitement des données s'effectue en suivants les étapes suivantes :
   - Importation des bibliothèques nécessaires :

```
# importing libraries
import pandas as pd
import scipy
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import OrdinalEncoder
import seaborn as sns
from sklearn.ensemble import ExtraTreesClassifier
import matplotlib.pyplot as plt
```

   - Importation de l'ensemble de données

```
# Load the dataset
df = pd.read_csv('C:/Users/LENOVO/diabetes.csv')
df
```

Comme vous pouvez le voir dans les informations, le jeu de données comporte 9 attributs et 768 instances.

1. **Encodage des données catégorielles**

Les variables catégorielles doivent être encodées numériquement avant d'être utilisées dans un algorithme de Machine Learning. Les techniques les plus répandues sont l'encodage one-hot (transformation en variables binaires) ou l'encodage ordinal (association d'un nombre à chaque modalité). Pour inspecter les données catégorielles, on utilise **df.select_dtypes()** pour que Pandas identifie automatiquement les colonnes de types catégorie.

```
# Détection des colonnes catégorielles
cat_columns = df.select_dtypes(include=["category"]).columns
print('Colonnes catégorielles :', cat_columns)
```

On encode les colonnes catégorielles identifiées à l'étape 1 en utilisant **OrdinalEncoder ()** de scikit-learn, qui assigne un entier à chaque category. On affiche uniquement les 5 premières lignes des données encodées pour avoir rapidement un aperçu des données encodées, sans tout afficher.

```
# Encodage
encoder = OrdinalEncoder()
encoded_data = encoder.fit_transform(Data[cat_columns])
print("Données encodées :", encoded_data[:5])
```

2. Séparer la classe est les caractéristiques

Il est essentiel de séparer les caractéristiques des données, qui servent de variables d'entrée au modèle, de la classe à prédire qui est la cible. On nomme X le jeu de données contenant les caractéristiques (attributs), et y le vecteur des classes. Cette séparation est nécessaire pour l'application des algorithmes d'apprentissage supervisé.

```
X = df.drop(columns =['Outcome'])
Y = df.Outcome
```

3. Normalisation

MinMaxScaler met à l'échelle les données de sorte que chaque caractéristique se situe dans l'intervalle [0, 1]. Cette méthode fonctionne bien lorsque les caractéristiques ont des échelles différentes et que l'algorithme utilisé est sensible à l'échelle des caractéristiques, comme les k-voisins les plus proches (KNN) ou les réseaux neuronaux.

Rééchelonnez vos données à l'aide de scikit-learn en utilisant l'échelle MinMax.

```
# initialising the MinMaxScaler
scaler = MinMaxScaler(feature_range=(0, 1))

# learning the statistical parameters for each of the data and transformi
rescaledX = scaler.fit_transform(X)
rescaledX[:5]
```

4. Standardisation

La standardisation est une technique utile pour transformer des attributs avec une distribution gaussienne et des moyennes et écarts types différents en une distribution gaussienne standard avec une moyenne de 0 et un écart type de 1.

Nous pouvons standardiser les données en utilisant scikit-learn avec la classe StandardScaler.

Cette méthode fonctionne bien lorsque les caractéristiques ont une distribution normale ou lorsque l'algorithme utilisé n'est pas sensible à l'échelle des caractéristiques.

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler().fit(X)
rescaledX = scaler.transform(X)
rescaledX[:5]
```

5. Valeurs manquantes

Pandas considère que None et NaN (Not a Number) sont essentiellement les éléments indiquant des valeurs manquantes ou nulles. Il existe plusieurs fonctions utiles pour détecter, supprimer et remplacer les valeurs manquantes par des valeurs raisonnables comme la moyenne et la médiane dans Pandas DataFrame : isnull(), notnull(), dropna(), fillna(), replace(), interpolate().

Le code ci-dessous, permet d'identifier les valeurs manquantes dans le dataset :

```
# using isnull() function
X.isna()
```

Selon le résultat, est ce que le dataset contient des valeurs manquantes ?

Dans le code suivant, les valeurs manquantes sont remplacées par la valeur médiane de cette colonne.

```
X = X.fillna(X.median())
X
```

6. Valeurs aberrantes

Les valeurs aberrantes, ou outliers en anglais, sont des points de données qui s'éloignent anormalement du reste des données. La détection des outliers peut se faire de manière graphique à l'aide de **boxplots**, ou algorithmique avec des métriques statistiques comme le **z-score**, le **IQR**, etc. Le code ci-dessous, permet de dessiner un boxplot pour identifier les colonnes du dataset qui contiennent des valeurs aberrantes

```
fig, a = plt.subplots()
a.boxplot(X)

a.set_title("Boxplot")
a.set_ylabel("Valeurs")
```

Ou bien, identifier les colonnes une par une, avec le code suivant :

```
sns.boxplot(data=X,x=X['Pregnancies'])
```

Selon le boxplot, quels sont les attributs qui contiennent des valeurs aberrantes ?

Dans le traitement, on peut supprimer complètement les outliers s'ils résultent certainement d'erreurs. Sinon, réduire leur impact sur les statistiques en plafonnant les valeurs extrêmes (**capping**), ou bien les remplacer par des valeurs raisonnables comme la moyenne et la médiane. Certains algorithmes de machine learning sont aussi robustes aux outliers.

7. Sélection des attributs

La sélection d'attributs, appelée aussi sélection de caractéristiques, est une étape de prétraitement des données visant à ne conserver que les attributs les plus pertinents. Elle permet de réduire la dimensionnalité des données pour atténuer le risque de sur-apprentissage et améliorer les performances des modèles.

En Python, on peut effectuer cette sélection d'attributs :

```
#Feature selection, with extratressclassifier
model = ExtraTreesClassifier()
model.fit(X,Y)
print(model.feature_importances_)
feat_importances = pd.Series(model.feature_importances_, index=X.columns)
feat_importances.nlargest(12).plot(kind='barh')
plt.show()
```

8. Diviser les données en données de test et d'entraînement

La division des données en jeux d'entraînement et de test est une étape critique en ML. Elle consiste à séparer les données disponibles en deux parties : les données d'entraînement pour entraîner un modèle, et les données de test pour évaluer ses performances. Un split classique est 80% données d'entraînement, 20% données de test.

Les instructions suivantes permettent d'effectuer cette division.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test=train_test_split(X, Y, test_size=0.2)
```