



UNIVERSITÉ SIDI MOHAMED BEN ABDLLAH
FACULTÉ DES SCIENCES DHAR EL MAHRAZ DE FÈS



DÉPARTEMENT D'INFORMATIQUE



DATA MINING

Tabular Dataset Pre-Processing

Présenter par :
ABIBOU SOUKAYNA

Supervisé par :
RIFFI JAMAL

Année universitaire : 2020/2021

Filière MIDVI

I- Introduction :

Le traitement des données est au cœur du Machine Learning. Vos **outils d'apprentissage automatique sont aussi bons que la qualité de vos données**. Ce projet traite des différentes étapes du **nettoyage des données**. Vos données doivent passer par quelques étapes avant de les transmettre à des modèles d'apprentissage automatique

Étapes impliquées dans le prétraitement des données:

1. Importation des bibliothèques requises
2. Importer l'ensemble de données
3. Traitement des données manquantes.
4. Codage des données catégoriques.
5. Fractionnement de l'ensemble de données en ensemble de test et ensemble d'apprentissage.
6. Mise à l'échelle des fonctionnalités.

Étape 1: importation des bibliothèques requises

Pour suivre, vous devrez télécharger cet ensemble de données: [Data.csv](#)

Chaque fois que nous créons un nouveau modèle, nous aurons besoin d'importer Numpy et Pandas. Numpy est une bibliothèque qui contient des fonctions mathématiques et est utilisée pour le calcul scientifique tandis que Pandas est utilisé pour importer et gérer les ensembles de données.

```
# -*- coding: utf-8 -*-
"""
Created on Mon May 17 22:59:48 2021

@author: Lenovo
"""
import pandas as pd
import numpy as np
```

Ici, nous importons les pandas et la bibliothèque Numpy et assignons respectivement un raccourci « pd » et « np ».

Étape 2: importation du data

Les ensembles de données sont disponibles au format .csv.

Un fichier CSV stocke des données tabulaires en texte brut.

Chaque ligne du fichier est un enregistrement de données.

Nous utilisons la méthode Read csv de la bibliothèque pandas pour lire un fichier CSV local en tant que **dataframe**.

Nous allons créer une matrice d'entités dans notre data (X) et créer un vecteur dépendant (Y) avec leurs observations respectives. Pour lire les colonnes, nous utiliserons iloc de pandas (utilisé pour fixer les index de sélection) qui prend deux paramètres – [sélection de ligne, sélection de colonne].

```
dataset = pd.read_csv('Data.csv')
X = dataset.iloc[:, :-1].values
Y = dataset.iloc[:, 3].values
```

Étape 3: Traitement des données manquantes

Les données que nous obtenons sont rarement homogènes. Parfois, des données peuvent être manquantes et doivent être traitées pour ne pas réduire les performances de notre modèle d'apprentissage automatique.

Pour ce faire, nous devons remplacer les données manquantes par la moyenne ou la médiane de la colonne entière. Pour cela, nous utiliserons la bibliothèque `sklearn.preprocessing` qui contient une classe appelée `Imputer` qui nous aidera à prendre soin de nos données manquantes.

```
from sklearn.preprocessing import Imputer
imputer = Imputer(missing_values = "NaN", strategy = "mean", axis = 0)
```

Notre nom d'objet est `imputer`. La classe `Imputer` peut prendre des paramètres tels que:

1. `Missing_values` : c'est l'espace réservé pour les valeurs manquantes. Toutes les occurrences de `missing_values` seront imputées. Nous pouvons lui donner un entier ou « NaN » pour qu'il trouve les valeurs manquantes.
2. `Stratégie` : C'est la stratégie d'imputation – Si « moyenne », alors remplacez les valeurs manquantes en utilisant la moyenne le long de l'axe (colonne). D'autres stratégies incluent « médiane » et « la plus fréquente ».
3. `Axe` : Il peut être attribué 0 ou 1, 0 pour imputer le long des colonnes et 1 pour imputer le long des lignes.

Nous adaptons maintenant l'objet `imputer` à nos données.

```
imputer = imputer.fit(X[:, 1:3])
```

Remplacez maintenant les valeurs manquantes par la moyenne de la colonne en utilisant la méthode de transformation.

```
X[:, 1:3] = imputer.transform(X[:, 1:3])
```

Etape 4 : Codage des données catégoriques

Toute variable qui n'est pas quantitative est catégorique. Les exemples incluent la couleur des cheveux, le sexe, le domaine d'études, l'université fréquentée, l'affiliation politique, le statut de l'infection par la maladie.

Mais pourquoi encoder?

Nous ne pouvons pas utiliser des valeurs telles que «Homme» et «Femme» dans les équations mathématiques du modèle, nous devons donc encoder ces variables en nombres.

Pour ce faire, nous importons la classe «`LabelEncoder`» de la bibliothèque «`sklearn.preprocessing`» et créons un objet `labelencoder_X` de la classe `LabelEncoder`. Ensuite, nous utilisons la méthode `fit_transform` sur les fonctionnalités catégorielles.

Après l'encodage, il est nécessaire de faire la distinction entre les variables de la même colonne, pour cela, nous utiliserons la classe OneHotEncoder de la bibliothèque sklearn.preprocessing

Encodage One-Hot :

Un encodage à chaud transforme les fonctionnalités catégorielles en un format qui fonctionne mieux avec les algorithmes de classification et de régression.

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
labelencoder_X = LabelEncoder()
X[:, 0] = labelencoder_X.fit_transform(X[:, 0])
onehotencoder = OneHotEncoder(categorical_features = [0])
X = onehotencoder.fit_transform(X).toarray()
labelencoder_Y = LabelEncoder()
Y = labelencoder_Y.fit_transform(Y)
```

Étape 5: Division de l'ensemble de données en ensemble d'apprentissage et ensemble de test

Maintenant, nous divisons nos données en deux ensembles, l'un pour entraîner notre modèle appelé **ensemble d'apprentissage** et l'autre pour tester les performances de notre modèle appelé **ensemble de test**. Le partage est généralement de 80/20. Pour ce faire, nous importons la méthode « train_test_split » de la bibliothèque « sklearn.model_selection ».

```
from sklearn.model_selection import train_test_split
```

Maintenant, pour construire nos ensembles de formation et de test, nous allons créer 4 ensembles.

```
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, random_state = 0)
```

Étape 6: mise à l'échelle des fonctionnalités

La plupart des algorithmes d'apprentissage automatique utilisent la **distance euclidienne** entre deux points de données dans leurs calculs. Pour cette raison, **les entités de grande magnitude pèseront plus** dans les calculs de distance **que les entités de faible magnitude**. Pour éviter cette normalisation des fonctionnalités ou la normalisation du score Z est utilisée. Cela se fait en utilisant la classe «StandardScaler» de «sklearn.preprocessing».

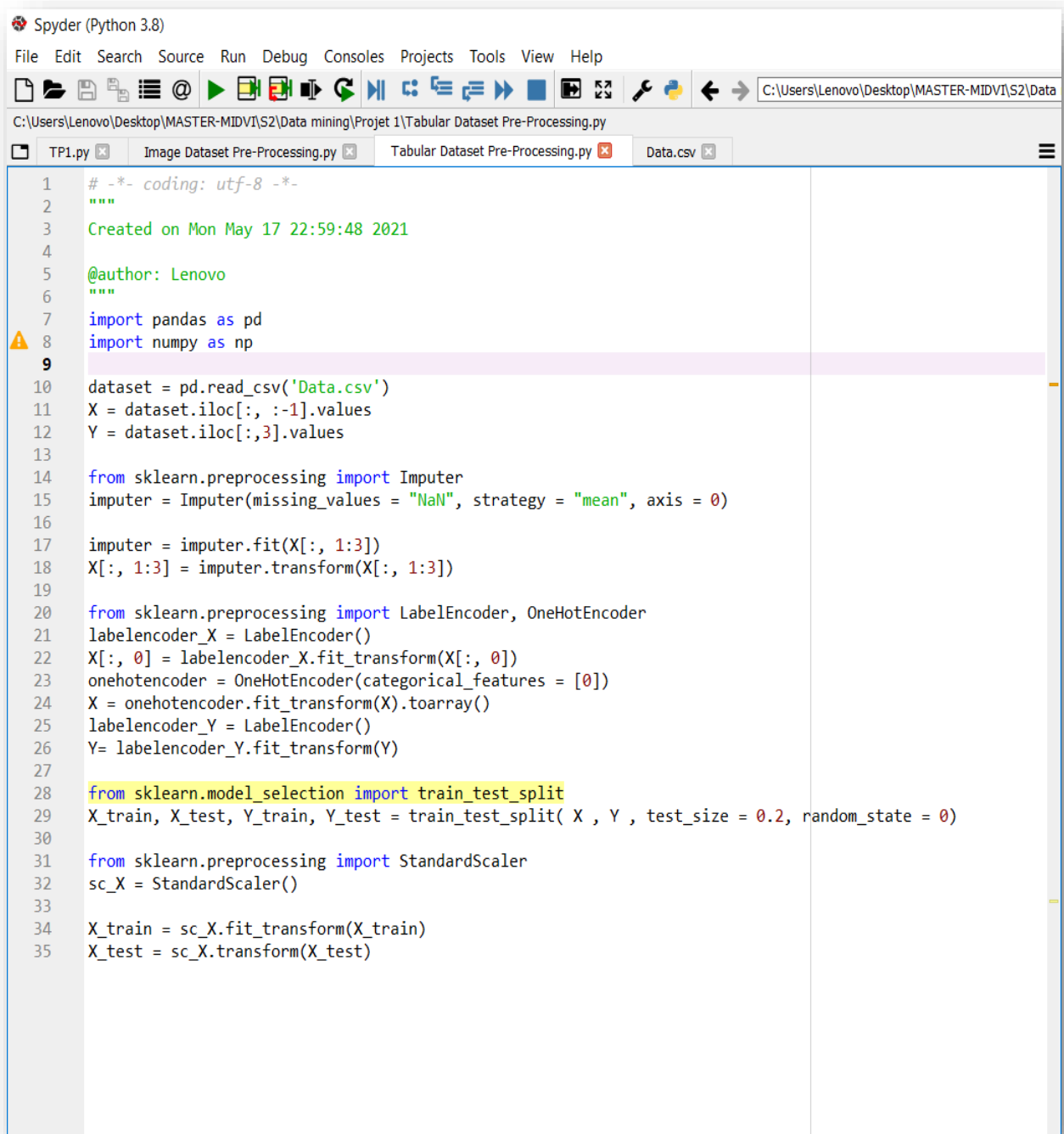
```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
```

De plus, nous transformerons notre ensemble X_test tandis que nous aurons besoin d'ajuster et de transformer notre ensemble X_train.

La fonction de transformation transformera toutes les données à une même échelle normalisée.

```
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
```

• Code :



The image shows the Spyder Python IDE interface. The top menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Projects, Tools, View, and Help. The toolbar contains icons for file operations, running, and debugging. The current file path is C:\Users\Lenovo\Desktop\MASTER-MIDVI\S2\Data mining\Projet 1\Tabular Dataset Pre-Processing.py. The script is named 'Tabular Dataset Pre-Processing.py' and is open in the editor. The code is as follows:

```
1  # -*- coding: utf-8 -*-
2  """
3  Created on Mon May 17 22:59:48 2021
4
5  @author: Lenovo
6  """
7  import pandas as pd
8  import numpy as np
9
10 dataset = pd.read_csv('Data.csv')
11 X = dataset.iloc[:, :-1].values
12 Y = dataset.iloc[:, 3].values
13
14 from sklearn.preprocessing import Imputer
15 imputer = Imputer(missing_values = "NaN", strategy = "mean", axis = 0)
16
17 imputer = imputer.fit(X[:, 1:3])
18 X[:, 1:3] = imputer.transform(X[:, 1:3])
19
20 from sklearn.preprocessing import LabelEncoder, OneHotEncoder
21 labelencoder_X = LabelEncoder()
22 X[:, 0] = labelencoder_X.fit_transform(X[:, 0])
23 onehotencoder = OneHotEncoder(categorical_features = [0])
24 X = onehotencoder.fit_transform(X).toarray()
25 labelencoder_Y = LabelEncoder()
26 Y = labelencoder_Y.fit_transform(Y)
27
28 from sklearn.model_selection import train_test_split
29 X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state = 0)
30
31 from sklearn.preprocessing import StandardScaler
32 sc_X = StandardScaler()
33
34 X_train = sc_X.fit_transform(X_train)
35 X_test = sc_X.transform(X_test)
```