

DATA WRANGLING WITH PYTHON

Encadré par:

-KAMAL IDRISI HAMZA

Réalisé par:

-Mijmij Mohcene

-ABOULHASSAN Abderrahmane

Sommaire

I- INTRODUCTION	3
II- LA QUALITÉ DES DONNÉES	4
III-OUTILS DE DATA WRANGLING	7
IV-DATA WRANGLING WITH PYTHON.....	11
V- APPLICATION	20
VI-CONCLUSION	24

I- Introduction:

Le traitement des données est le processus de suppression des erreurs et de combinaison d'ensembles de données complexes pour les rendre plus accessibles et plus faciles à analyser. En raison de l'expansion rapide de la quantité de données et de sources de données disponibles aujourd'hui, le stockage et l'organisation de grandes quantités de données à des fins d'analyse deviennent de plus en plus nécessaires.

Un processus de traitement des données, également connu sous le nom de processus de munging des données, consiste à réorganiser, transformer et cartographier les données d'une forme « brute » à une autre afin de les rendre plus utilisables et plus utiles pour une variété d'utilisations en aval, y compris l'analyse.

DATA WRANGLING peut être défini comme le processus de nettoyage, d'organisation et de transformation des données brutes dans le format souhaité que les analystes peuvent utiliser pour une prise de décision rapide. Également connu sous le nom de nettoyage de données ou de munging de données, le data wrangling permet aux entreprises de traiter des données plus complexes en moins de temps, de produire des résultats plus précis et de prendre de meilleures décisions. Les méthodes exactes varient d'un projet à l'autre en fonction de vos données et de l'objectif que vous essayez d'atteindre. De plus en plus d'organisations s'appuient de plus en plus sur des outils de traitement des données pour préparer les données pour l'analyse en aval.

II- La qualité des données

1- Définition:

Les données sont partout. Il est généré automatiquement par nos appareils mobiles, nos activités d'achat et nos déplacements physiques. Il est capté par nos compteurs électriques, nos systèmes de transport public et notre infrastructure de communication. ces données sont disponibles sous des forma différentes et avec différentes caractéristiques

DATA QUALITY est donc la mesure dans laquelle les données répondent aux attentes d'une entreprise en matière d'exactitude, de validité, d'exhaustivité et de cohérence.

En suivant la qualité des données, une entreprise peut identifier les problèmes potentiels nuisant à la qualité et s'assurer que les données partagées sont adaptées à une utilisation donnée.

Lorsque les données collectées ne répondent pas aux attentes de l'entreprise en matière d'exactitude, de validité, d'exhaustivité et de cohérence, elles peuvent avoir des impacts négatifs considérables sur le service client, la productivité des employés et les stratégies clés.

2- Importance de la qualité des données:

Des données de qualité sont essentielles pour prendre des décisions précises et éclairées. Et bien que toutes les données aient un certain niveau de « qualité », une variété de caractéristiques et de facteurs déterminent le degré de qualité des données (de haute qualité par rapport à de faible qualité). De plus, différentes caractéristiques de qualité des données seront probablement plus importantes pour les différentes parties prenantes de l'organisation.

Une liste des caractéristiques et des dimensions courantes de la qualité des données comprend :

- Accuracy
- Completeness
- Consistency
- Integrity
- Reasonability
- Timeliness
- Uniqueness/Deduplication
- Validity
- Accessibility

3- Défis liés à la qualité des données:

La mauvaise qualité des données est le premier ennemi de l'utilisation généralisée et la rentabilité de l'apprentissage automatique. Si vous souhaitez que des technologies telles que l'apprentissage automatique fonctionnent pour vous, vous devez vous concentrer sur la qualité des données. Parmi les problèmes de qualité des données les plus courants:

- **Inaccurate data.**
 - **Duplicate data.**
 - **Hidden data.**
 - **Ambiguous data.**
 - **Too much data.**
 - **Inconsistent data.**

4- Amélioration de la qualité des données:

De nombreux aspects de la qualité des données sont le produit d'un processus de data wrangling, qu'il s'agisse de rapprocher et de normaliser des unités, de clarifier la signification d'étiquettes de données obscures ou de rechercher des informations générales sur la représentativité de votre ensemble de données. Une partie de ce que cela illustre est que, dans le monde réel, garantir la qualité des données est au moins en partie le résultat de multiples processus itératifs de gestion des données. Bien que les termes de ces phases du processus de traitement des données varient. On les décrit généralement comme le nettoyage des données et l'augmentation des données.

a-Data Cleaning

En réalité, le nettoyage des données n'est pas tant une étape en soi dans le processus de data wrangling qu'une activité constante qui accompagne toutes les autres étapes, à la fois parce que la plupart des données ne sont pas propres lorsque nous les rencontrons de plus la façon dont un ensemble de données (ou une partie de celui-ci) doit être nettoyé se révèle souvent au fur et à mesure que l'on travaille. À un niveau élevé, les données propres peuvent être résumées comme étant exemptes d'erreurs ou de fautes de frappe et des valeurs manquantes ou impossibles. Bien que bon nombre d'entre eux soient au moins assez simples à reconnaître, des problèmes de données plus profonds peuvent encore persister.

b-Data Augmentation

L'augmentation d'un ensemble de données est le processus d'expansion ou d'élaboration, généralement en le connectant à d'autres ensembles de données - c'est vraiment la nature des mégadonnées au 21e siècle. En utilisant des fonctionnalités partagées entre les ensembles de données, il est possible de rassembler des données provenant de sources multiples afin d'obtenir un portrait plus complet de ce qui se passe dans le monde en comblant les lacunes, en fournissant des mesures corroborantes ou en ajoutant des données contextuelles qui nous aident à mieux évaluer l'adéquation des données. Grâce à une combinaison de recherche efficace et de traitement des données, l'augmentation des données peut nous aider à améliorer la qualité des données et à répondre à des questions bien trop nuancées pour être traitées dans un seul ensemble de données. comme le nettoyage des données, des opportunités d'augmentation des données peuvent survenir à presque n'importe quel moment du processus de traitement des données. Dans le même temps, chaque nouvel ensemble de données que nous introduisons engendrera son propre processus de traitement des données. Cela signifie que contrairement au nettoyage des données, l'augmentation des données n'a pas d'état final définitif : il y aura toujours un autre ensemble de données que nous pourrions ajouter.

III-Outils de data wrangling:

Il existe différents outils de gestion des données qui peuvent être utilisés pour collecter, importer, structurer et nettoyer les données avant qu'elles ne puissent être introduites dans les applications d'analyse et de BI. Vous pouvez utiliser des outils automatisés pour le traitement des données, où le logiciel vous permet de valider les mappages de données et d'examiner des échantillons de données à chaque étape du processus de transformation. Cela permet de détecter et de corriger rapidement les erreurs dans le mappage des données. Le nettoyage automatisé des données devient nécessaire dans les entreprises traitant des ensembles de données exceptionnellement volumineux.

1-Talend:



Les fonctionnalités de Talend incluent la possibilité d'appliquer des règles à une variété d'ensembles de données, de les enregistrer et de les partager entre les équipes. Il dispose également de processus intégrés pour des tâches telles que l'enrichissement et l'intégration, ainsi que la capacité de s'intégrer à une gamme de systèmes d'entreprise différents.

Fonctionnalités clés de Talend:

- **Intégration** : Talend permet aux entreprises de gérer n'importe quel type de données à partir d'une variété de sources de données, que ce soit dans le Cloud ou sur site.
- **Qualité des données** : Talend purifie automatiquement les données ingérées à l'aide de fonctionnalités de Machine Learning telles que la déduplication, la validation et la standardisation des données.
- **Flexible** : lors de la création de pipelines de données à partir de vos données connectées, Talend va au-delà du fournisseur ou de la plateforme. Talend vous permet d'exécuter des pipelines de données n'importe où une fois que vous les avez créés à partir de vos données ingérées.

2-Alteryx APA:



La plate-forme Alteryx APA est l'un des meilleurs outils de Data Wrangling qui fournit non seulement des outils pour le Data Wrangling, mais également pour des besoins plus généraux en matière d'analyse de données et de science des données. Si vous voulez tout au même endroit, c'est l'idéal.

Principales caractéristiques d'Alteryx:

- Collaborer et découvrir : les utilisateurs peuvent rechercher n'importe quel actif de données et coopérer avec d'autres utilisateurs non seulement pour créer de nouveaux outils d'analyse, mais également pour utiliser des modèles créés par d'autres afin d'éviter d'avoir à réinventer la roue.
- Préparer, analyser et modéliser : ce sont les trois étapes du processus. Les utilisateurs peuvent préparer leurs données et créer des modèles efficaces qui peuvent être utilisés et réutilisés pour différents ensembles de données.
- Partage d'expérience sociale/communautaire :. Alteryx encourage les utilisateurs à partager des informations. Ils ont pris quelques repères du mouvement Open Source, comme encourager la divulgation complète des informations ou des outils d'analyse produits par la communauté.

3-Altair Monarch:



Un autre des principaux outils de gestion des données, Altair Monarch, convertit des données complexes et non structurées dans un format plus lisible. Il prétend être capable d'extraire des données de n'importe quelle source, même des PDF et des rapports textuels, qui sont des formes difficiles et non structurées. Il modifie ensuite les données en fonction des règles que vous fournissez avant de les insérer directement dans votre base de données SQL. La plateforme comprend notamment un certain nombre de solutions adaptées aux exigences de reporting des secteurs de la comptabilité et de la santé. Il est extrêmement populaire dans ces domaines.

Principales caractéristiques d'Altair Monarch:

- **Intégrations** : extrayez des données de fichiers plats, de bases de données relationnelles, de systèmes OLEDB/ODBC, d'entrées Web, de modèles de données, d'espaces de travail et de sources de données multistructurées avec des intégrations de données. Exportez des données aux formats de fichier CSV, MS-Access et JSON, ainsi que des applications de création de rapports et d'analyse comme IBM Cognos Analytics, Tableau et Qlik.
- **Importation de fichiers PDF** : son moteur PDF vous permet de choisir et de modifier des tableaux à partir de fichiers PDF contenant beaucoup de texte avant de les exporter vers Data Prep Studio. Créez des grilles sur lesquelles le texte est aligné en identifiant les caractéristiques graphiques telles que les rectangles et les lignes sur les images des pages PDF produites. Extrayez tous les arrière-plans et polices, y compris les polices à espacement fixe et de forme libre, en une seule étape.
- **Recouvrement dans Excel** : pour créer des classeurs ou combiner plusieurs feuilles de calcul en une seule, extrayez des champs de données spécifiques. Pour éviter les pièges de données, réutilisez les modèles de données en supprimant les informations personnelles des modèles.

4-Trifacta:



Trifacta est une plate-forme interactive basée sur le cloud pour profiler les données et leur appliquer des modèles d'apprentissage automatique et d'analyse. Indépendamment du caractère chaotique ou complexe des ensembles de données, cet outil d'ingénierie de données tente de créer des données intelligibles. Les techniques de déduplication et de transformation linéaire permettent aux utilisateurs de supprimer les entrées en double et de remplir les cellules vides des ensembles de données.

Principales caractéristiques de Trifacta:

- Intégration au cloud : prend en charge les charges de travail de préparation dans n'importe quel environnement cloud ou hybride, permettant aux développeurs d'ingérer des données pour les disputer de n'importe où.
- Normalisation : Trifacta wrangler fournit un certain nombre de mécanismes pour détecter les modèles de données et normaliser les sorties. Les ingénieurs de données peuvent choisir de normaliser par modèle, fonction ou une combinaison des deux.
- Flux de travail facile : Trifacta utilise des flux pour organiser les tâches de préparation des données. Un flux est composé d'un ou plusieurs jeux de données ainsi que des recettes qui les accompagnent (étapes définies qui transforment les données).

IV-Data wrangling with python:

Dans ce chapitre, on discute des outils pour les données manquantes, les données en double, la manipulation de chaînes, et quelques autres transformations de données analytiques.

Python est populaire en raison de sa simplicité et de sa flexibilité, mais aussi en raison du grand nombre de bibliothèques et de frameworks que les data scientists peuvent utiliser. Voici cinq des plus utiles et des plus populaires pour le data wrangling :

- **Pandas:** One of the 'must have' tools for data wrangling, Pandas uses data structures called DataFrames, with built-in methods for grouping, filtering, and combining data.
- **NumPy:** This is primarily used for scientific computing and performing basic and advanced array operations.
- **SciPy:** This can execute advanced numerical routines, including for numerical integration, interpolation, optimization, linear algebra, and statistics.
- **Matplotlib:** A powerful data visualization library that can convert data into graphs and charts that are more user-friendly when it comes to modeling and analysis.
- **Scikit-Learn:** A popular machine learning and data modeling tool that is really useful for building regression, clustering, and classification models.

Dans cette partie on doit travailler avec la base de données data-to-clean télécharger depuis moodle dans la section de cours python pour le data science

1-Data Wrangling With Pandas

a-Data exploration:

Commençons par lire dans notre jeu de données (fichier csv) dans les pandas et afficher les noms de colonnes avec leurs types de données.

```
1-Import data:

[14] data = pd.read_csv("/content/drive/MyDrive/Pandas/datatoclean.csv")

data.dtypes

Duration      int64
Date          object
Pulse         int64
Maxpulse      int64
Calories      float64
dtype: object
```

Dans les données, nous avons les colonnes suivantes: Duration, Date, Pulse, Maxpulse et Calories.

Pour vérifier que nos données correspondent à la source, nous pouvons utiliser l'option describe dans pandas :

2-statistical data

```
data.describe()
```

	Duration	Pulse	Maxpulse	Calories
count	32.000000	32.000000	32.000000	30.000000
mean	68.437500	103.500000	128.500000	304.680000
std	70.039591	7.832933	12.998759	66.003779
min	30.000000	90.000000	101.000000	195.100000
25%	60.000000	100.000000	120.000000	250.700000
50%	60.000000	103.500000	127.500000	281.000000

Cela résume parfaitement certaines données statistiques pour toutes les colonnes numériques. Il semble que tout. Pour les données catégorielles, nous pouvons le faire en regroupant les valeurs :

```
data.shape
```

(32, 5)

```
data.groupby(by=[ 'Duration' ]).size()
```

Duration	
30	1
45	6
60	24
450	1

dtype: int64

b-Dealing with missing values:

Dans chaque dataset, il est essentiel d'évaluer les valeurs manquantes. Combien y en a-t-il? Est-ce une erreur ? Y a-t-il trop de valeurs manquantes ? Une valeur manquante a-t-elle un sens par rapport à son contexte ?

Nous pouvons résumer le total des valeurs manquantes en utilisant ce qui suit :

```
data.isna().sum()

Duration    0
Date        1
Pulse       0
Maxpulse    0
Calories    2
dtype: int64
```

Maintenant que nous avons identifié nos valeurs manquantes, nous avons quelques options. Nous pouvons les remplir avec une certaine valeur (zéro, moyenne/max/médiane par colonne, chaîne) ou les déposer par ligne.

- **drop missing value:**

```
data_2 = data.dropna(axis = 0, how = 'any')
data_2.isna().sum()

Duration    0
Date        0
Pulse       0
Maxpulse    0
Calories    0
dtype: int64
```

- **replace nan values with zero:**

```
data_2 = data.fillna(0)
data_2.isna().sum()

Duration    0
Date        0
Pulse       0
Maxpulse    0
Calories    0
dtype: int64
```

- **replace nan value with average of columns:**

```
data_2 = data.fillna(data.mean())
data_2.isna().sum()

<ipython-input-36-9df57cdab734>:1: FutureWarning: Dropping of nuisance columns in DataFrame red
data_2 = data.fillna(data.mean())
Duration    0
Date        1
Pulse       0
Maxpulse    0
Calories    0
dtype: int64
```

Si on choisit de remplacer les valeurs manquantes par la valeur moyenne de la colonne, il faut faire attention au champ dont le type n'est pas numérique. C'est le cas avec la colonne data dans notre data.

La méthode `mean()` ne peut pas s'appliquer sur la colonne Date.

```
data.mean()

<ipython-input-37-abc01cf6c622>:1: FutureWarning: Dropping of nuisance columns in DataFrame r
data.mean()
Duration      68.4375
Pulse         103.5000
Maxpulse      128.5000
Calories      304.6800
dtype: float64
```

- **Inspection des doublons:**

Pour afficher les lignes répétées, nous pouvons commencer par examiner le nombre de valeurs uniques dans chaque colonne.

```
data.nunique()

Duration      4
Date          30
Pulse         15
Maxpulse      24
Calories      26
dtype: int64
```

c-Filtering data

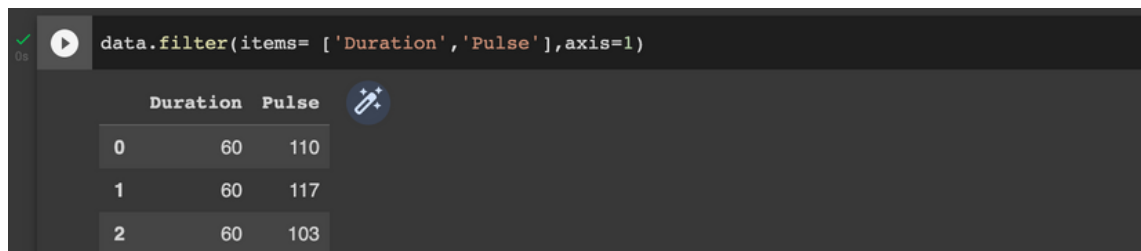
Pandas `filter()` filtre les DataFame pour les lignes et les colonnes. Le DataFrame retourné contient seulement les lignes et les colonnes qui sont spécifiées avec la fonction. Il ne met pas à jour le DataFrame existant au lieu de cela, il renvoie toujours un nouveau.

- **pandas filter() Syntax:**

```
DataFrame.filter([items=None, like=None, regex=None, axis=None])
```

- `item` : Takes list of axis labels that you wanted to filter.
- `like` : Takes axis string label that you wanted to filter
- `regex` : regular expression
- `axis` : {0 or 'index', 1 or 'columns', None}, default None. When not specified it used columns.

Allons dans notre dat frame, on doit filtrer les colonnes Duration et Pulse.

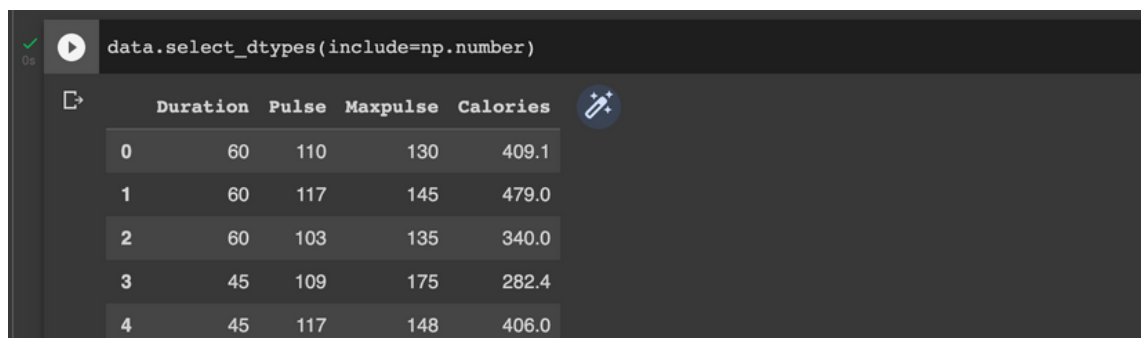


A Jupyter Notebook interface showing a code cell with the command `data.filter(items= ['Duration','Pulse'],axis=1)`. Below the code, a table displays the filtered data with columns 'Duration' and 'Pulse'.

	Duration	Pulse
0	60	110
1	60	117
2	60	103

- **Sélection de colonnes par type de données**

Nous pouvons utiliser la méthode `pandas.DataFrame.select_dtypes(include=None, exclude=None)` pour sélectionner les colonnes en fonction de leurs types de données.



A Jupyter Notebook interface showing a code cell with the command `data.select_dtypes(include=np.number)`. Below the code, a table displays the selected numerical columns: 'Duration', 'Pulse', 'Maxpulse', and 'Calories'.

	Duration	Pulse	Maxpulse	Calories
0	60	110	130	409.1
1	60	117	145	479.0
2	60	103	135	340.0
3	45	109	175	282.4
4	45	117	148	406.0

- **Les opérations de filtrage, groupage et de concaténation**

Ces opérations sont très similaires aux opérations SQL qu'il est possible d'effectuer sur une table d'une base de données.

-L'opération de filtrage

Les Dataframe Pandas permettent l'indexation booléenne, qui est un moyen assez efficace de filtrer un dataframe pour plusieurs conditions. Les conditions multiples impliquant les opérateurs `|` (pour l'opérateur ou), `&` (pour l'opérateur et), et `~` (pour l'opération non) peuvent être regroupées à l'aide de parenthèses `()`.

Exemple:

-On doit créer une data set depuis notre data , qui contient les colonnes dont la valeur de la donnée Pulse est supérieure à 100

```
[16] resultat = data[(data['Pulse']>100)]
```

resultat

	Duration	Date	Pulse	Maxpulse	Calories
0	60	'2020/12/01'	110	130	409.1
1	60	'2020/12/02'	117	145	479.0
2	60	'2020/12/03'	103	135	340.0

-L'opération de groupage

La fonction groupby() est l'une des fonctions les plus utiles lorsqu'il s'agit de traiter des dataframes volumineux dans Pandas. Une opération groupby consiste généralement à diviser la dataframe, à appliquer une fonction et à combiner les résultats.

Exemple: Regroupons les données du dataframe par Duration. On peut applique d'autres fonctions (min, max, mean....).

```
data2=data.groupby('Duration')
for Duration , donnee in data2:
    print("=====")
    print("Duration:",Duration)
    print("===data===")

    print(donnee)
    print()
```

=====
Duration: 30
===data===

	Duration	Date	Pulse	Maxpulse	Calories
8	30	'2020/12/09'	109	133	195.1

=====
Duration: 45
===data===

	Duration	Date	Pulse	Maxpulse	Calories
3	45	'2020/12/04'	109	175	282.4
4	45	'2020/12/05'	117	148	406.0
18	45	'2020/12/18'	90	112	NaN
20	45	'2020/12/20'	97	125	243.0
22	45	NaN	100	119	282.0

```
[16] resultat = data[(data['Pulse']>100)]
```

```
data2=data.groupby('Duration').mean()
data2
```

	Pulse	Maxpulse	Calories
Duration			
30	109.000	133.000000	195.10000
45	103.000	135.166667	291.88000
60	103.375	126.416667	314.46087
450	104.000	134.000000	253.30000

-L'opération de concaténation:

On peut concaténer des dataframes le long de l'axe des lignes ou des colonnes. Supposons qu'on a plusieurs dataframe comportant les mêmes champs et on veut les combiner en un seul le long de l'axe des lignes. Ou encore, si on a des champs supplémentaires pour les données actuelles qu'on a ajoutées, on peut les concaténer le long de l'axe des colonnes

Exemple:

1-dataframes comportant les mêmes champs et on veut les combiner en un seul le long de l'axe des lignes.

```
[39] df1 = data[(data['Pulse']<100)]
[40] df2 = data[(data['Pulse']>=100)]
[41] data2 = pd.concat([df1, df2])

data2.head()
```

	Duration	Date	Pulse	Maxpulse	Calories
9	60	'2020/12/10'	98	124	269.0
15	60	'2020/12/15'	98	123	275.0
16	60	'2020/12/16'	98	120	215.2
18	45	'2020/12/18'	90	112	NaN

2- champs supplémentaires pour les données actuelles qu'on a ajoutées,

```
print("====df1: ")
print(df1.head(4))
print("====df2")
print(df2.head(4))
print("==== Concaténation horizontale de df1 et df2")
print(data2.head(4))
print("====")

====df1:
  Duration  Pulse
0        60   110
1        60   117
2        60   103
3        45   109
====df2
  Maxpulse
0       130
1       145
2       135
3       175
==== Concaténation horizontale de df1 et df2
  Duration  Pulse  Maxpulse
0        60   110       130
1        60   117       145
2        60   103       135
3        45   109       175
=====
```

2- Data wrangling using sklearn:

Le package `sklearn.preprocessing` fournit plusieurs fonctions utilitaires et classes de transformateurs communes pour transformer les vecteurs de caractéristiques brutes en une représentation plus adaptée aux estimateurs en aval.

En général, les algorithmes d'apprentissage bénéficient de la standardisation de l'ensemble de données. Si certaines valeurs aberrantes sont présentes dans l'ensemble, des échelles ou des transformateurs robustes sont plus appropriés. Les comportements des différents scalers, transformateurs et normalisateurs sur un jeu de données contenant des valeurs aberrantes marginales sont mis en évidence dans Comparer l'effet de différents scalers sur les données avec des valeurs aberrantes.

a- Standardisation

La normalisation des ensembles de données est une exigence commune pour de nombreux estimateurs d'apprentissage automatique implémentés dans `scikit-learn` ; ils peuvent mal se comporter si les caractéristiques individuelles ne ressemblent pas plus ou moins à des données standard normalement distribuées : gaussiennes avec une moyenne nulle et une variance unitaire.

Exemple:

```
[21] from sklearn import preprocessing
import numpy as np

[60] X_train = np.array([[ 1., -1.,  2.],[ 2.,  0.,  0.],[ 0.,  1., -1.]])
      scaler = preprocessing.StandardScaler().fit(X_train)

[63] X_scaled = scaler.transform(X_train)
      X_scaled

array([[ 0.          , -1.22474487,  1.33630621],
       [ 1.22474487,  0.          , -0.26726124],
       [-1.22474487,  1.22474487, -1.06904497]])

le moyen: 4.9343245538895844e-17
la variance : 1.0
```

b- MinMaxScaler

Une normalisation alternative consiste à mettre à l'échelle les entités pour qu'elles se situent entre une valeur minimale et maximale donnée, souvent entre zéro et un, ou de sorte que la valeur maximale absolue de chaque entité soit mise à l'échelle en fonction de la taille de l'unité.

Example:

```
[21] from sklearn import preprocessing
      import numpy as np

[32] X_train = np.array([[ 1., -1.,  2.],
                        [ 2.,  0.,  0.],
                        [ 0.,  1., -1.]])

[53] scaler = preprocessing.StandardScaler().fit(X_train)

[57] min_max_scaler = preprocessing.MinMaxScaler()
      X_train_minmax = min_max_scaler.fit_transform(X_train)

X_train_minmax
array([[0.5       , 0.       , 1.       ],
       [1.       , 0.5     , 0.33333333],
       [0.       , 1.       , 0.       ]])
```

V- APPLICATION:

1- Import data

Vous pouvez trouver le "Jeu de données automobiles" à partir du lien suivant :

-<https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/CognitiveClass/DA0101EN/auto.csv>

```
df.head()
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	

2- work with missing data

a- Identify missing data:

```
missing_data = df.isnull()
missing_data
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke
0	False	True	False	False	False	False	False	False	False	False	...	False	False	False	False
1	False	True	False	False	False	False	False	False	False	False	...	False	False	False	False
2	False	True	False	False	False	False	False	False	False	False	...	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False
...

-Count missing values in each column.

```
for column in missing_data.columns.values.tolist():
    print(column)
    print (missing_data[column].value_counts())
    print("")
```

- "normalized-losses": 41 missing data.
- "stroke": 4 missing data.
- "bore": 4 missing data.
- "horsepower": 2 missing data.
- "peak-rpm": 2 missing data.

b-Deal with missing data:

Dans notre ensemble de données, aucune des colonnes n'est suffisamment vide pour être entièrement supprimée. Nous avons une certaine liberté dans le choix de la méthode de remplacement des données ;

```
2-Deal with missing data:
Replace by mean.

[29] avg_norm_loss = df["normalized-losses"].astype("float").mean(axis=0)
      print("Average of normalized-losses:", avg_norm_loss)
      df["normalized-losses"].replace(np.nan, avg_norm_loss, inplace=True)

Average of normalized-losses: 122.0
```

c- Correct data format

Les variables numériques doivent avoir le type 'float' ou 'int', et les variables avec des chaînes telles que les catégories doivent avoir le type 'object'. Par exemple, les variables 'bore' et 'stroke' sont des valeurs numériques qui décrivent les moteurs, nous devrions donc nous attendre à ce qu'elles soient du type 'float' ou 'int' ; cependant, ils sont affichés en tant que type 'objet'.

```
3-correct data format:

df.dtypes
```

symboling	int64
normalized-losses	object
make	object
fuel-type	object
aspiration	object
num-of-doors	object
body-style	object
drive-wheels	object
engine-location	object
wheel-base	float64
length	float64
width	float64
height	float64
curb-weight	int64
engine-type	object
num-of-cylinders	object
engine-size	int64
fuel-system	object
bore	object
stroke	object
compression-ratio	float64
horsepower	object
peak-rpm	object
city-mpg	int64
highway-mpg	int64
price	object
dtype:	object

Convertissez les types de données au format approprié :

```
Convert data types to proper format:

df[["bore", "stroke"]] = df[["bore", "stroke"]].astype("float")
df[["normalized-losses"]] = df[["normalized-losses"]].astype("int")
df[["price"]] = df[["price"]].astype("float")
df[["peak-rpm"]] = df[["peak-rpm"]].astype("float")
```

3-Data Standardization

Dans notre ensemble de données, les colonnes de consommation de carburant "ville-mpg" et "autoroute-mpg" sont représentées par l'unité mpg (miles par gallon). Supposons que nous développons une application dans un pays qui accepte la consommation de carburant avec la norme L/100km

```
# Convert mpg to L/100km by mathematical operation (235 divided by mpg)
df['city-L/100km'] = 235/df['city-mpg']

# check your transformed data
df.head()
```

body-style	drive-wheels	engine-location	wheel-base	...	fuel-system	bore	stroke	compression-ratio	horsepower	peak-rpm	city-mpg	highway-mpg	price	city-L/100km
convertible	rwd	front	88.6	...	mpfi	3.47	2.68	9.0	111	5000.0	21	27	13495.0	11.190476
convertible	rwd	front	88.6	...	mpfi	3.47	2.68	9.0	111	5000.0	21	27	16500.0	11.190476

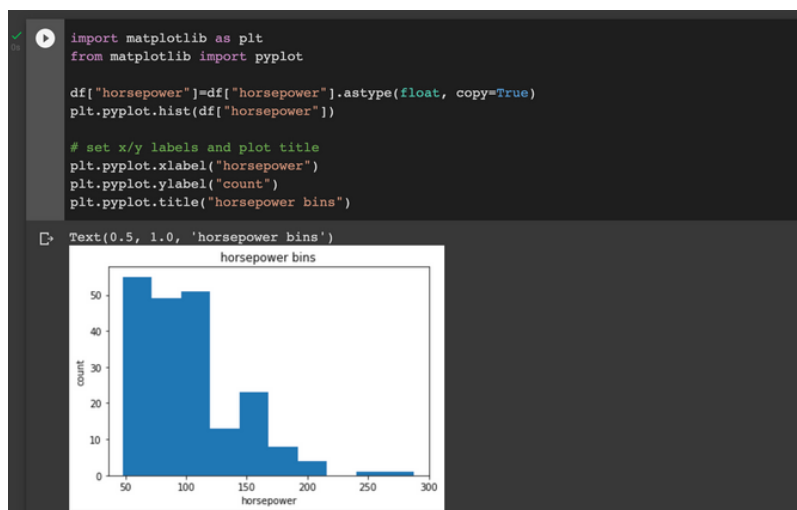
4-Data Normalization:

To demonstrate normalization, let's say we want to scale the columns "length", "width" and "height"

```
[40] df['length'] = df['length']/df['length'].max()
      df['width'] = df['width']/df['width'].max()
      df['height'] = df['height']/df['height'].max()
```

5-Binning

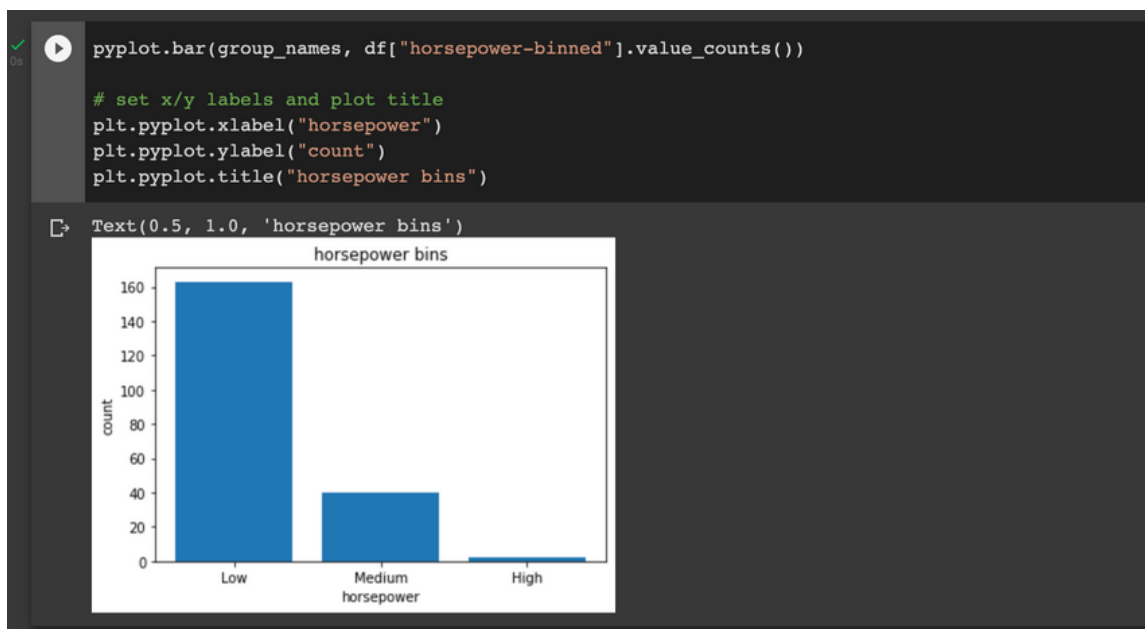
Le binning est un processus de transformation de variables numériques continues en bacs catégoriels discrets, pour une analyse groupée.



Nous aimerions 3 bacs de bande passante de taille égale, nous utilisons donc l'espace linspace de numpy (start_value, end_value, numbers_generated function).

```
group_names = ['Low', 'Medium', 'High']
bins = np.linspace(min(df["horsepower"]), max(df["horsepower"]), 4)
df['horsepower-binned'] = pd.cut(df['horsepower'], bins, labels=group_names, include_lowest=True)
df[['horsepower', 'horsepower-binned']].head(20)
```

	horsepower	horsepower-binned
0	111.0	Low
1	111.0	Low
2	154.0	Medium
3	102.0	Low
4	115.0	Low
5	110.0	Low
6	110.0	Low



Conclusion:

Une préparation efficace des données peut améliorer considérablement la productivité en vous permettant de passer plus de temps à analyser les données et moins de temps à les préparer pour l'analyse. Nous avons exploré un certain nombre d'outils dans ce projet.

Dans un premier temps on a vu les différents outils de data wrangling, des logiciels opens source, et des bibliothèques sur python dans Pandas sont la plus utilisés dans les projets de data science. Deuxièmement on a appliqué les différentes techniques sur une dataset téléchargée depuis Kaggle.

Conclusion, ce projet a été une occasion pour améliorer notre compétence en data science d'une manière générale, et en data preprocessing d'une manière particulière.