DTeK ESN

Machine Learning avec python

Réalisé et présenté par :

EL HAD SOILIHI

Plan

I. Introduction Machine Learning

Le machine learning, ou encore l'apprentissage automatique en français, fait partie de l'une des approches de l'intelligence artificielle.

Le machine Learning est donc une discipline scientifique centrée sur le développement, l'analyse et l'implémentation de méthodes automatisables, qui offrent la possibilité à la machine d'évoluer grâce à un processus d'apprentissage. Il se révèle particulièrement efficace lorsqu'il s'agit d'analyser de larges ensembles de données diverses et évolutives, ce que l'on nomme communément le Big Data.

Aujourd'hui toutes les entreprises collectent et stockent de grandes quantités de données. Ces bases de données, qui ne cessent d'augmenter jour après jour, sont peu exploitées, alors qu'elles cachent de connaissances décisives face au marché et à la concurrence. Pour combler ce besoin, une nouvelle industrie est en train de naître Data Science. Il existe toute une pléthore de domaines dans lesquels le machine Learning intervient, à savoir la finance, la sécurité, la médecine, l'industrie automobile et la technologie dans tout son ensemble ainsi que différents outils permettant de développer ces différents algorithmes.

Au fil des années Python est devenu un outil du quotidien pour les ingénieurs et chercheurs de toutes les disciplines scientifiques. Grâce à de très nombreuses librairies d'une grande qualité, il permet aujourd'hui d'égaler, voire de surpasser des solutions propriétaires les plus performantes du marché. Il est devenu un des outils incontournables des Data Scientists ! Nous vous proposons dans cet article de découvrir la vaste étendue de cet écosystème.

II. Apprentissage supervisé ou non supervisé

Il existe deux principaux types d'apprentissages : supervisées et non supervisées. La principale différence entre les deux types réside sur le fait que nous avons une connaissance préalable de ce que devraient être les valeurs de sortie de nos échantillons. L'algorithme d'apprentissage constitue la méthode avec laquelle le modèle statistique va se paramétrer à partir des données d'exemple. Il existe de nombreux algorithmes différents !

On choisira un type d'algorithme particulier en fonction du type de tâche que l'on souhaite accomplir et du type de données dont on dispose. Quelle est *l'entrée* de l'algorithme et quelle est *la sortie*. L'objectif ici n'est pas de rentrer dans le détail des modèles mais plutôt de donner au lecteur des éléments de compréhension sur chacun d'eux.

Par conséquent, l'objectif de l'apprentissage supervisé est d'apprendre une fonction qui, à partir d'un échantillon de données et des résultats souhaités, se rapproche le mieux de la relation entre entrée et sortie observable dans les données. En revanche, l'apprentissage non supervisé n'a pas de résultats étiquetés. Son objectif est donc de déduire la structure naturelle présente dans un ensemble de points de données

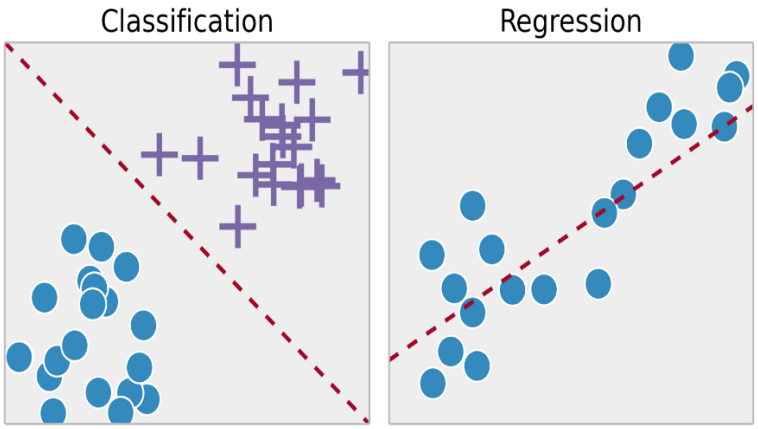
A. L'apprentissage supervisé

La majorité des apprentissages automatiques utilisent un apprentissage supervisé. L'apprentissage supervisé consiste en des variables d'entrée (x) et une variable de sortie (Y). Vous utilisez un algorithme pour apprendre la fonction de mapping de l'entrée à la sortie Y = f (X).

Le but est d'appréhender si bien la fonction de mapping que, lorsque vous avez de nouvelles données d'entrée (x), vous pouvez prédire les variables de sortie (Y) pour ces données.

L'apprentissage est dit supervisé lorsque les données qui entrent dans le processus sont déjà catégorisées et que les algorithmes doivent s'en servir pour prédire un résultat en vue de pouvoir le faire plus tard lorsque les données ne seront plus catégorisées.

La Classification ou la Régression



L'apprentissage supervisé est généralement effectué dans le contexte de la classification ou de la régression.

> Classification : Un problème de classification survient lorsque la variable de sortie est une catégorie, exemple en médecine, pour prédire si un patient à une maladie ou non.
> Régression : Un problème de régression se pose lorsque la variable de sortie est une valeur réelle, exemple prédire le prix de l'immobilier.

Voici quelques exemples populaires d'algorithmes d'apprentissage automatique supervisé :

- Arbres de décision
- K Nearest Neighbours (k les plus proches voisins)
- SVM (machine à vecteur de support)
- Régression logistique
- Naïve Bayes
- Les réseaux de neurones
- Régression linéaire

B. L’apprentissage non-supervisé ou clustering

Les données d'entrées ne sont pas annotées. Pour ce faire, le système va croiser les informations qui lui sont soumises, de manière à pouvoir rassembler dans une même classe les éléments présentant certaines similitudes. L'algorithme d'entraînement s'applique dans ce cas à trouver seul les similarités et distinctions au sein de ces données, et à regrouper ensemble celles qui partagent des caractéristiques communes.

Les algorithmes sont laissés à leurs propres mécanismes pour découvrir et présenter la structure intéressante des données. L’apprentissage non supervisé comprend deux catégories d’algorithmes, l’algorithme de regroupement ou le clustering qui consiste à séparer ou à diviser un ensemble de données en un certain nombre de groupes, de sorte que les ensembles de données appartenant aux mêmes groupes se ressemblent davantage que ceux d’autres groupes et l’association consiste à découvrir des relations intéressantes entre des variables dans de grandes bases de données. Par exemple, les personnes qui ont regardé le film le mendiant ont aussi regardé la série plus belle la vie.

Voici quelques exemples populaires d’algorithmes d’apprentissage automatique non-supervisé :

- K-means clustering
- Dimensionality Reduction (Réduction de la dimensionnalité)
- Neural networks / Deep Learning
- Principal Component Analysis (Analyse des composants principaux)
- Singular Value Decomposition (Décomposition en valeur singulière)
- Independent Component Analysis (Analyse en composantes indépendantes)
- Distribution models (Modèles de distribution)
- Hierarchical clustering (Classification hiérarchique)

C. L’apprentissage semi-supervisé

L’apprentissage semi-supervisé est généralement utilisé lorsque nous avons une petite quantité d’entrées étiquetées et beaucoup plus d’entrées non étiquetées. Par conséquent, ces problèmes se situent entre l’apprentissage supervisé et l’apprentissage non supervisé.

III. L'environnement python

Python est devenu un langage de choix pour les scientifiques, en raison de sa simplicité de mise en œuvre et de la richesse de son écosystème, notamment grâce à ses nombreuses et performantes librairies de calculs numériques bien souvent développées par les scientifiques eux-mêmes.

Il est donc recommandé d'utiliser des distributions Python qui proposent un packaging cohérent de toutes les librairies et offrent un minimum d'optimisations pour nos machines. Anaconda est la distribution la plus répandue en raison de la diversité des plateformes qu'elle supporte et d'une plus grande ouverture des outils sur lesquels elle se base.

Dans cette section, nous allons travailler étape par étape sur un petit projet de machine learning, la classification des fleurs d'iris de la base de données d'Iris de la librairie scikit-learn en utilisant l'environnement anaconda avec l'IDE spyder.

Le jeu de données comprend 150 échantillons de chacune des trois espèces d'iris (*Iris* setosa, *Iris* virginica et *Iris* versicolor). Quatre caractéristiques ont été mesurées à partir de chaque échantillon : la longueur et la largeur des sépales et des pétales, en centimètres. Sur la base de la combinaison de ces quatre variables, Fisher a élaboré un modèle d'analyse discriminante linéaire permettant de distinguer les espèces les unes des autres. Notre projet consiste à apprendre à l'ordinateur de classer automatiquement l'espèce d'une nouvelle fleur d'iris en fonction de la longueur et la largeur des sépales et des pétales en se basant sur les résultats déjà obtenus par M. Fisher.

Chaque instance (ligne) de notre data est composée de quatre attributs pour décrire une fleur d'Iris. Le jeu de données est étiqueté par le type de fleur. Ainsi pour quatre attributs décrivant une fleur d'Iris, on saura de quelle variante il s'agit. Comme on connait préalablement ce que devraient être les valeurs de sortie de notre échantillon en occurrence une variable catégorielle. On a un apprentissage supervisé de type classification, par conséquent on pourra utiliser un des algorithmes de classification vu précédemment.

Dans notre cas on utilisera un algorithme très utilisé en apprentissage qui s'appelle K-NN (**K Nearest Neighbors**) en français les K plus proches voisins. Le principe de ce modèle consiste en effet à choisir les K données les plus proches du point étudié afin d'en prédire sa classe. Notre objectif sera donc d'entrainer un modèle qui sera capable de reconnaître les trois espèces d'iris.

Premièrement nous importons les librairies nécessaires :

```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

On peut donc directement obtenir les données via un appel de fonction dans la librairie scikit-Learn.

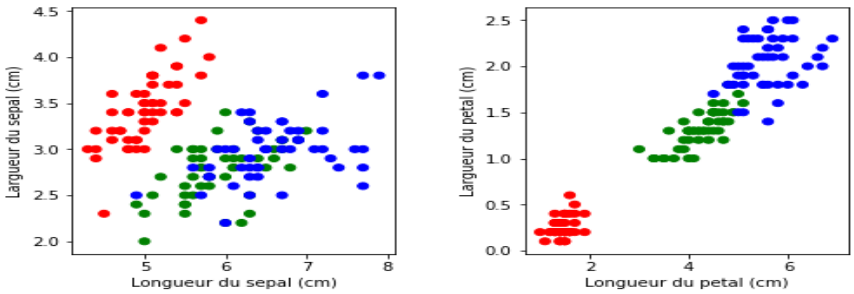
```
#chargement de jeu des données Iris
from sklearn import datasets
iris = datasets.load_iris()

#importer le jeu de données Iris dataset à l'aide du module pandas
X = pd.DataFrame(iris.data)
X.columns = ['Sepal_Length','Sepal_width','Petal_Length','Petal_width']
y = pd.DataFrame(iris.target)
y.columns = ['Targets']

```

L'objet iris contient deux entrées data et target, créons deux variables X et y pour un accès plus facile avec les labels à nos données.

Visualisation des données :



La séparation des groupes entre les longueurs et largeurs des pétales semble très nette et déterminante.

Une fois que notre dataset chargé, nous allons séparer le jeu de données en deux groupes training set (l'apprentissage) et testing set (le test).

```
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)

```

On rappelle qu'on va utiliser uniquement le training set pour entraîner notre modèle et on garde le testing set pour tester la performance de notre modèle. On a mis une répartition de 75% et 25% entre training et testing set.

On crée un classifieur 5-NN, qui prend en compte les 5 plus proches voisins pour la classification.

```
30 # Fitting K-NN to the Training set
31 from sklearn.neighbors import KNeighborsClassifier
32 classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
33 classifier.fit(X_train, y_train)
34
```

L'algorithme k-NN n'effectue aucune optimisation, mais il va juste mémoriser toutes les données en mémoire, c'est sa manière d'apprendre pour classer les nouvelles données.

Prédire les données test :

```
35 # Predicting the Test set results
36 y_pred = classifier.predict(X_test)
37
```

Et pour prédire l'espèce d'une nouvelle fleur l'algorithme va simplement chercher les **5** voisins les plus proches de ce point et trouver l'espèce constitue la majorité de ces points, afin d'en en déduire la classe du nouveau point.

Mesurer les performances de notre model :

Mesurons à présent l'erreur de notre classifieur, la méthode *accuracy_score* détermine la proportion de points correctement prédits.

```
38 # Making the Confusion Matrix
39 from sklearn.metrics import confusion_matrix
40 cm = confusion_matrix(y_test, y_pred)
41
42 #score
43 from sklearn.metrics import accuracy_score
44 accuracy_score(y_test, y_pred)
45
```

Dans notre cas, l'accuracy est à 97%, c'est-à-dire que notre algorithme était capable de bien classer 97% de nos données du testing set ce qui une erreur de 3%. On peut aussi représenter les performances de notre modèle dans une matrice de confusion.

	0	1	2
0	13	0	0
1	0	15	1
2	0	0	9

On remarque que notre algorithme a pu bien classer 37 fleurs sur 38 en fonction de leurs caractéristiques.

IV. Conclusion

L'algorithme K-NN ainsi choisi est assez simple d'un point de vu conceptuel mais illustre parfaitement les problématiques classiques qui en découlent. Cependant en pratique, cet algorithme est assez peu utilisé en classification parce qu'il est couteux en puissance de calcul**.** En effet, un modèle étant une approximation de la réalité, il repose sur un certain nombre d'hypothèses de départ pour exister. Ces hypothèses sont dépendantes du contexte (i.e. du problème considéré). Les hypothèses étant différentes pour chaque type de problème, on doit considérer différents modèles pour différents problèmes.

On aurait pu utiliser plusieurs algorithmes dans cet article pour notre jeu de données car en réalité il n'existe pas d'algorithme et modèle "ultime"**,** applicable pour tous les problèmes. Vous devez donc aborder chaque nouveau problème avec un œil neuf et veiller à tester plusieurs algorithmes afin de le résoudre, en formulant des hypothèses spécifiques à votre problème.

Quel que soit l'algorithme choisi, les étapes ne changent plus : on instancie la classe de l'algorithme, on fournit les données d'apprentissage à la méthode à la méthode *fit* pour l'apprentissage supervisé et on réalise nos prédictions avec la méthode *predict* en lui fournissant les données de test.

Par contre les difficultés dans cette discipline sont : l'acquisition des données, la répartition de données en training set et en testing set, la compréhension des notions mathématiques qui sont derrière chaque algorithme et enfin le choix des hyperparamètres.

V. Ressources

https://scikit-learn.org/stable/

https://fr.wikipedia.org/wiki/Apprentissage_automatique

https://www.udemy.com/course/machinelearning

https://machinelearningmastery.com/machine-learning-in-python-step-by-step/