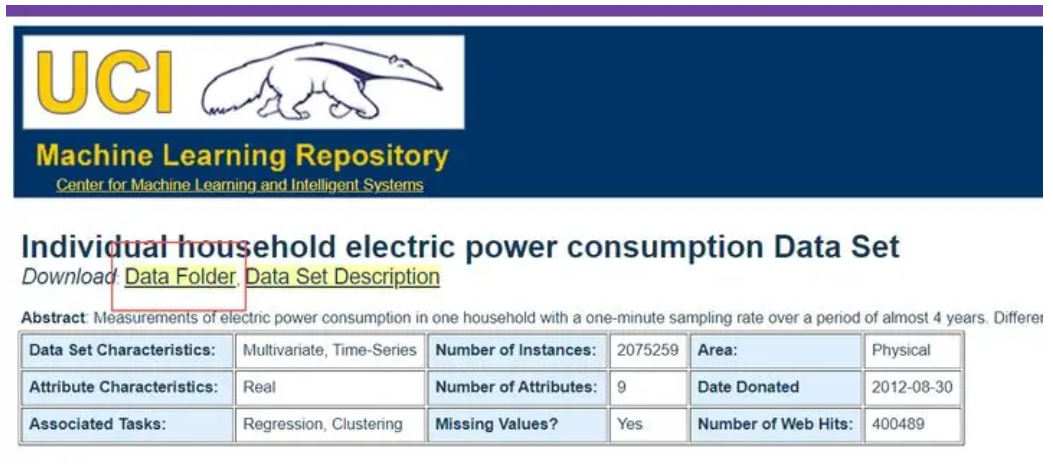
項目：【線性回歸】以法國璽鎮4年內的家庭用電情況分析

<https://archive.ics.uci.edu/ml/datasets/individual+household+electric+power+consumption>



點擊上面的【Data Folder】即可下載數据集

1、瞭解數据和數据集：

詳見具體網頁分析內容

2、導入相關庫、讀取數据：

首先先導入常用的pip庫：

%matplotlib inline  #這行是為了方便jupyter畫圖，非jupyter環境可以刪除這句

import matplotlib.pyplot as plt  #如不在python內畫圖可以不用導入該庫，此行展示僅供參考import numpy as np

from sklearn.linear\_model import LinearRegression

import pandas as pd  #個人比較喜歡用modin庫，據說會快一些，代碼如下：import modin.pandas as pd

import sqldf #如果不太熟悉pandas語法的朋友可以導入這個庫，可以用這個pip庫來實現通過SQL語法更改dataframe

讀取數据要用到這個指令：

df = pd.read\_csv('household\_power\_consumption.txt',sep=';')  #該代碼不會自動拆分txt，不加分隔符號（sep=';'）必出問題，如果報錯，請在括弧內添加Low memory=False

3、特徵處理

特徵處理可以說是機器學習最重要的一步，這裡我們對特徵的處理主要是作【數据清洗】：

1、空值處理

2、數据型態型轉換

3、單位轉換

4、特徵縮放

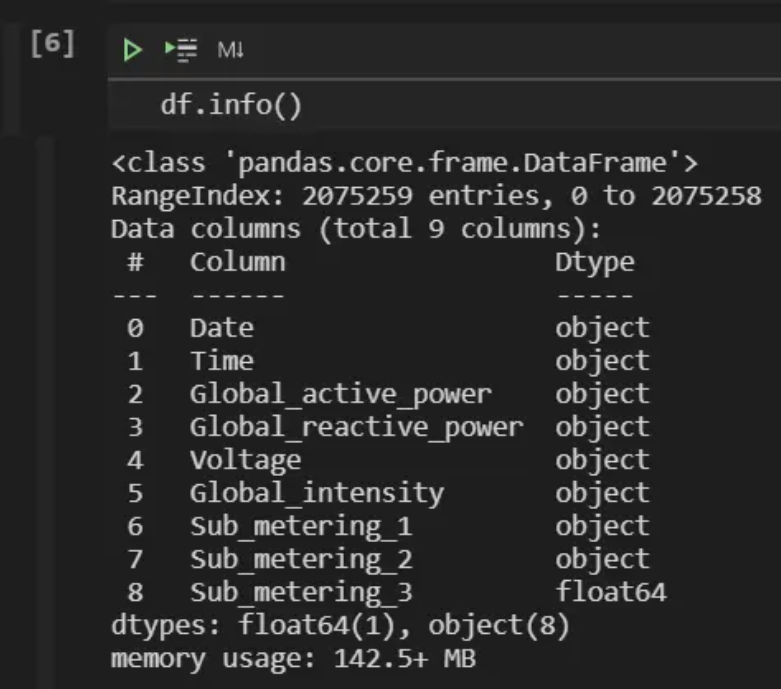
5、異常值處理

……

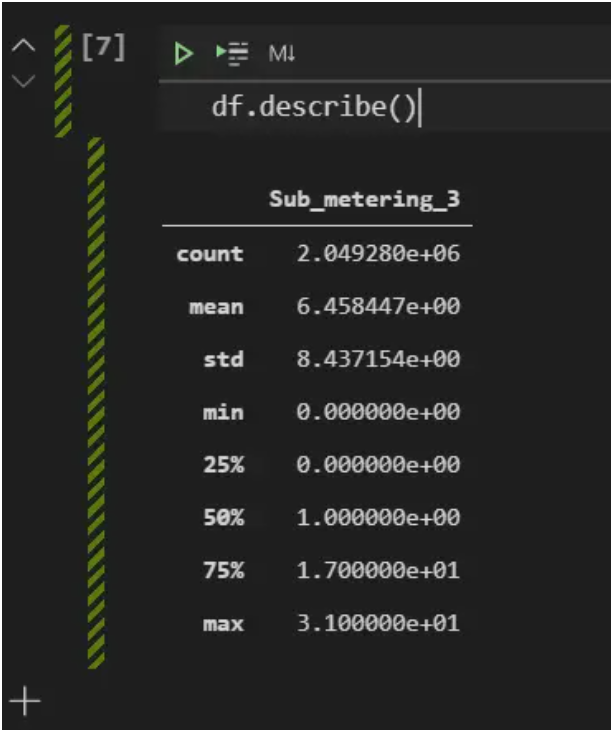
並非所有的數据都要經歷以上N個步驟，我們要根據數据的實際情況進行取捨。

首先看看整個數据長啥樣，最常用的指令無非就是下面2個：

df.info() #查看數據概況



df.describe() #對數据型態特徵進行描述統計

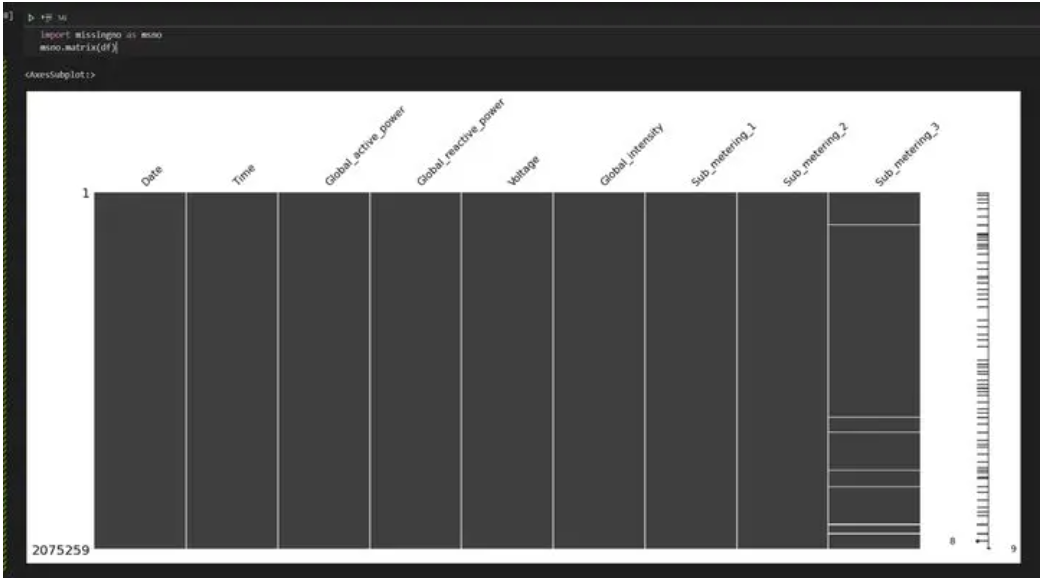


注意，【object】代表“物件”這一數据型態型，然而根據我們第一部分對【數据描述】的閱讀，除了Date和Time代表時間以外，其他的數据型態型可判別應該可能是數据型態。

但另外我們可以看到，在【df.describe()】運行之後卻只有一欄結果，也就是說：除了Sub\_metering\_3是真正數据型態特徵以外，其他的都不是數据型態特徵之數据型態型。這裡就能夠確定【數据型態轉換】是我們一定要做的數据清洗步驟了，不過在這之前，我們先進行【空值處理】，這裡推薦一段缺失值視覺化代碼：

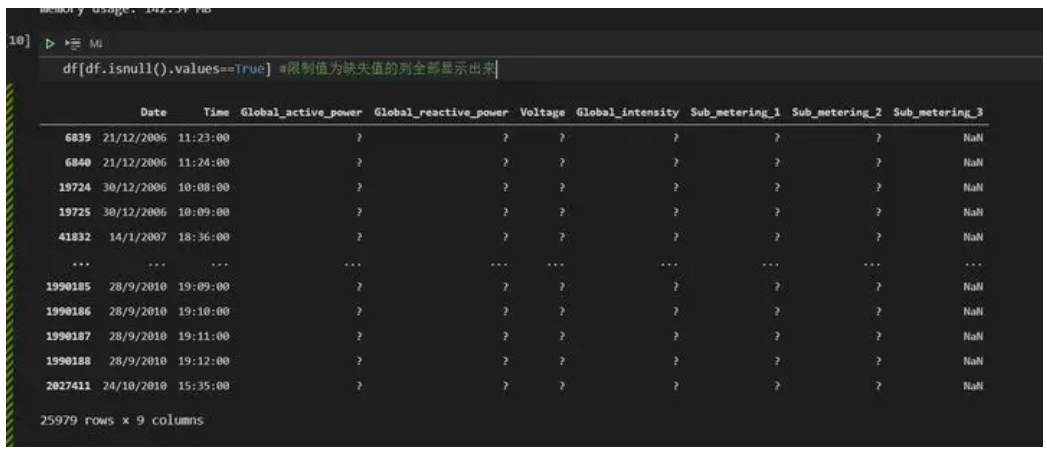
import missingno as msno

msno.matrix(df)



特徵Sub\_metering\_3的空行就是缺失值，這裡我們來看看這些缺失值長啥樣，根據缺失值的實際情況再決定處理方案：

df[df.isnull().values==True]  #限制值為缺失值的列全部顯示出來



可以看出，特徵Sub\_metering\_3中缺失值所對應的列有很大一部分都是“？”，正是因為非法字元的存在，因此許多特徵的數据型態型才並非【數据描述】中提到的【數值型】，而全部都是object。

我們這裡將異常字元“？”全部替換為缺失值nan然後再統一刪除

#將？替換成缺失值

df = df.replace(to\_replace='?',value = np.nan)

#丟棄帶有缺失值的數据

df = df.dropna()

刪除掉缺失值和異常值之後，我們再對所有的特徵進行【數值型別轉換】：

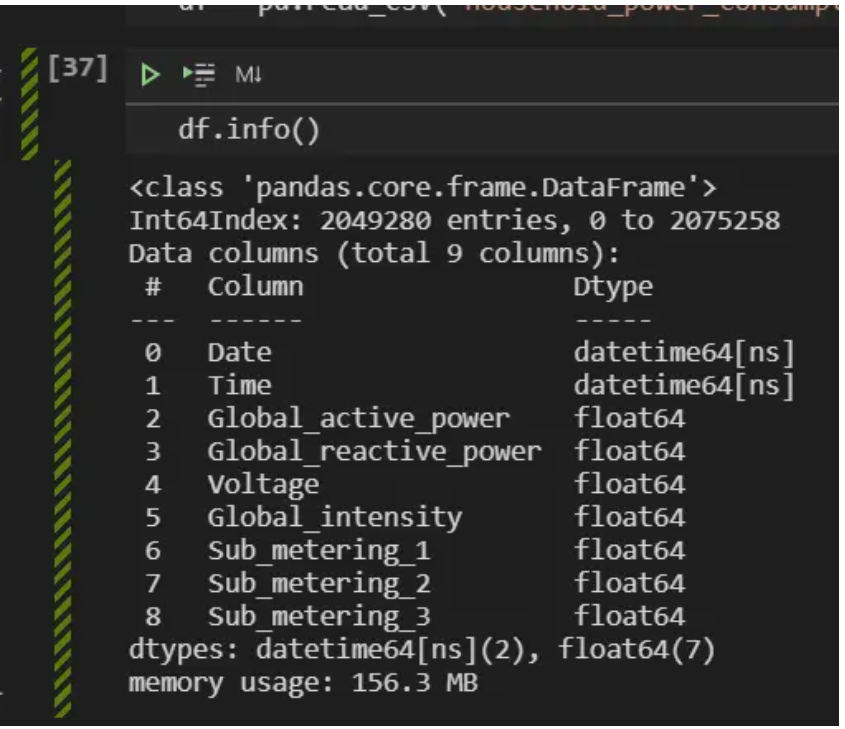
df=df.apply(pd.to\_numeric, errors='ignore')  #首先將能轉換成數值類型的全部特徵轉換為數据型態型

執行完上述代碼後，再執行：

df['Time'] = pd.to\_datetime(df['Time'],format="%H:%M:%S")

df['Date'] = pd.to\_datetime(df['Date'],format="%d/%m/%Y")  #將2個時間特徵轉換成對應的時間格式

最後我們再來看看現在的數据型態：



到現在您已經完成了一半的工作！接下來的部分可能要求一點點機器學習基礎，您可以查閱周志華老師的《機器學習》第二章 2.1 經驗誤差與過擬合(<https://www.jianshu.com/p/c6a701b87e45>) 來補足一些基本概念。

4、劃分訓練集和測試集

接下來讓我們給模型要驗證的X變數和Y變數賦值：

X = df[['Sub\_metering\_1','Sub\_metering\_2','Sub\_metering\_3']]

Y = df['Global\_active\_power']

賦值完成後，劃分訓練集和測試集，方便對模型進行檢驗：

from sklearn.model\_selection import train\_test\_split

X\_train,X\_test,Y\_train,Y\_test = train\_test\_split(X, Y, test\_size=0.3, random\_state=10)  # 測試集為30%

5、特徵值**標準化**

在原始的數据中，各變數的範圍大不相同。對於某些機器學習的演算法，若沒有做過標準化，目標函數會無法適當的運作。

舉例來說，多數的分類器利用兩點間的距離計算兩點的差異，若其中一個特徵具有非常廣的範圍，那兩點間的差異就會被該特徵左右，因此，我們這裡要對模型進行標準化。

from sklearn.preprocessing import StandardScaler  #通過去除均值並縮放到單位方差來標準化特徵

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)  # 必須先fit才能使用標準化，因此我們這裡對模型訓練並轉換

X\_test = scaler.transform(X\_test)  # 直接使用在模型構建數据上進行一個數据標準化操作，對剩餘的數据（testData）使用同樣的均值、方差、最大最小值等指標進行轉換transform(testData)，從而保證train、test處理方式相同:

6、模型訓練

可能有很多朋友會認為模型構建是最難的，其實不然，**前輩們已經為我們準備好了非常多完善的封裝函數，只要學會不同函數的應用場景和參數調整，並調用即可**，這裡用到的是最經典的線性回歸：

# 模型訓練（線性模型）

lr = LinearRegression()

lr.fit(X\_train, Y\_train) # 訓練模型

y\_predict = lr.predict(X\_test)  # 預測

7、模型檢驗

接下來就是見證奇跡的時刻了！我們已經構建好了模型，可以在這一步查看模型的整體效果如何，這裡面的一些概念需要一些 統計學《計量經濟學》(<https://baike.baidu.com/item/%E8%AE%A1%E9%87%8F%E7%BB%8F%E6%B5%8E%E5%AD%A6/80374?fr=kg_general>)基礎，這方面您可自行補足

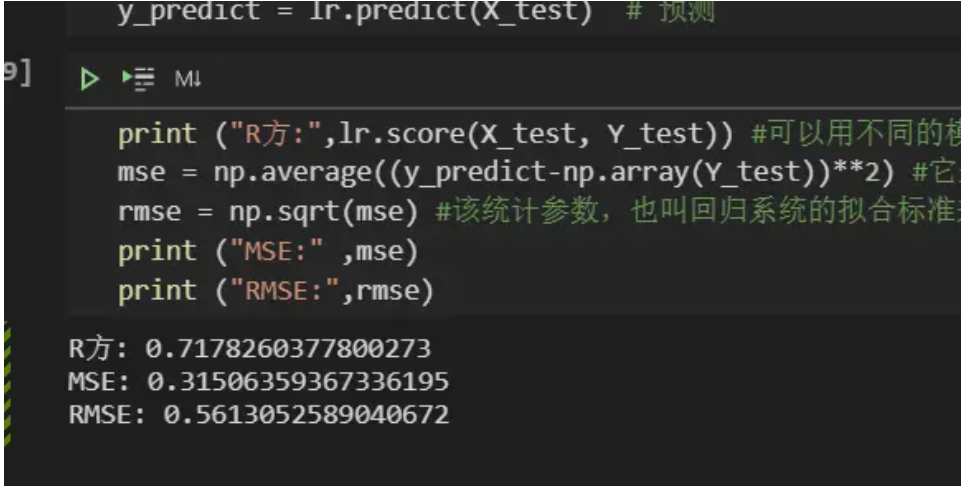
print ("R方:",lr.score(X\_test, Y\_test))  #可以用不同的模型來輸出R方，R方的含義是，預測值解釋了y變數的方差的多大比例，該統計參數是預測數据和原始數据對應點誤差的平方和的均值

mse = np.average((y\_predict-np.array(Y\_test))\*\*2)  #它是“誤差”的平方的期望值，誤差就是估計值與被估計量的差

rmse = np.sqrt(mse)  #統計參數，也叫回歸系統的擬合標準差，是MSE的平方根

print ("MSE:" ,mse)

print ("RMSE:",rmse)



R方有0.717，從統計學意義上來講還是非常不錯的，如果有一些基礎的同學，可以通過以下的代碼計算調整後的R方：

y = Y\_test

yhat = y\_predict

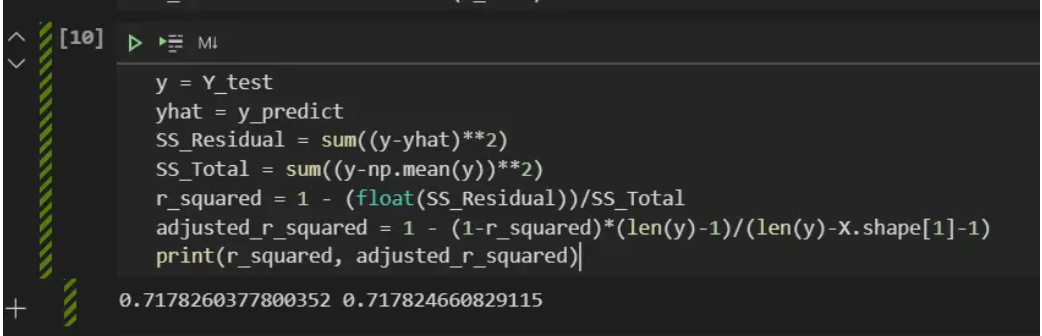
SS\_Residual = sum((y-yhat)\*\*2)

SS\_Total = sum((y-np.mean(y))\*\*2)

r\_squared = 1 - (float(SS\_Residual))/SS\_Total

adjusted\_r\_squared = 1 - (1-r\_squared)\*(len(y)-1)/(len(y)-X.shape[1]-1)

print(r\_squared, adjusted\_r\_squared)



以上就是機器學習專案之學習流程，省思一下再將代碼歸納整理！