Cheatsheet วิธีใช้ และเทคนิคใน Pandas (Python) ฉบับสมบูรณ์

Perth N November 5, 2017

Pandas ถือเป็นเครื่องมือหลักในการทำ Data Wrangling บน Python และสามารถนำไปใช้ประโยชน์คู่กับ Package อื่น เช่น เอาไปเตรียมข้อมูลก่อนทำ Model ใน SKLearn ได้ด้วย

และเนื่องจากหน้านี้เป็น Cheatsheet รวมเทคนิคเยอะแยะมากมาย ถ้าไล่อ่านกันอาจจะหายาก ผมเลยเตรียมสารบัญมาให้ด้านล่างนี้ครับ

Contents [hide

Pandas คืออะไร?

Pandas เป็น Library ใน Python ที่ทำให้เราเล่นกับข้อมูลได้ง่ายขึ้น เหมาะมากสำหรับทำ Data Cleaning / Wrangling ครับผม

วิธีการใช้งาน Pandas คือ โหลดไฟล์ข้อมูล เช่น CSV เข้าไป แล้วเราจะได้ข้อมูลในรูปแบบตาราง (DataFrame) ที่แบ่งข้อมูลตามแถวและคอลัมน์ หรือเหมือน Excel ที่เราใช้กันนั่นเอง

3	20140502T000000	420k	3	2.25	2000	8030	1.0	0	0	4	1000	1000	1963	NaN
4	20140502T000000	550k	4	2.50	1940	10500	1.0	0	0	4	1140	800	1976	NaN

ตัวอย่าง DataFrame ของ Pandas เป็นตารางเหมือน Excel เลยครับ

ป.ล. Pandas ไม่เกี่ยวกับหมีแพนด้านะฮะ จริง ๆ แล้วมาจากคำว่า PANel DAta ซึ่งหมายถึงข้อมูลที่มีหลายมิตินั่นเอง

เทคนิคการใช้ Pandas

อย่างที่**แอดมินเคยเล่า** ว่าการทำ Data Wrangling เป็นงานที่ค่อนข้างถึกครับ วันนี้เลยรวบรวมโค้ดต่าง ๆ ในการใช้ Pandas มาให้ ซึ่งน่าจะครอบคลุมการใช้งานประมาณ 80 – 90% แล้วครับผม

โค้ดบางส่วนมาจากคลาส Data Wrangling ที่แอดมินเรียน และจากเว็บไซต์ **MyCheatSheet** ครับ

วิธีเช็ค Version Pandas

โค้ดนี้เหมือนไม่สำคัญ แต่จริง ๆ แล้วสำคัญมากเวลาเราอ่าน Documentation ครับ เพราะถ้าเกิดมีอะไรพัง เราจะเทียบได้ว่า Pandas ของเราเป็นเวอร์ชั่นตามใน Documentation มั้ย

```
1.  print ("Pandas version",pandas.__version__)
```

วิธีการโหลดไฟล์ CSV (Import)

จุดเริ่มต้นของการทำ Data Exploration & Analysis ใน Pandas ก็คือการโหลดไฟล์ข้อมูลแบบ CSV มาใช้งานนั่นเองครับ

```
csvdf = pd.read_csv('data.csv',encoding = "ISO-8859-1")
# Print head and tail
csvdf.head()
csvdf.tail()
```

วิธีเช็คจำนวนแถว และจำนวนคอลัมน์

ใน Pandas มีฟังชั่นสำหรับนับจำนวนแถว และจำนวนคอลัมน์แบบง่าย ๆ โดยใช้

```
csvdf.shape
```

วิธีสุ่มข้อมูลสำหรับเช็ค (Sample)

ปกติเราเช็คข้อมูลว่าถูกต้องมั้ยด้วย head กับ tail ซึ่งเป็นการเช็คจากด้านบนหรือด้านล่าง อีกวิธีที่น่าสนใจ คือ เช็คแบบสุ่มข้อมูลขึ้นมานั่นเองครับ ทำได้ง่าย ๆ โดยใช้

```
csvdf.sample()
```

วิธีเช็คข้อมูลหาความผิดปกติใน DataFrame เบื้องต้น

หลังจากโหลดข้อมูลมาแล้ว เราอยากรู้ว่าข้อมูลมีกี่แถว, Missing value เท่าไหร่, แต่ละคอลัมน์เป็น Data Type อะไรบ้าง ก็รันคำสั่งนี้ได้เลย มีประโยชน์มากครับ

```
df.info()
```

```
bathrooms        4600 non-null float64
sqft_living      4600 non-null int64
sqft_lot         4600 non-null int64
floors           4600 non-null float64
waterfront       4600 non-null int64
view             4600 non-null int64
condition        4600 non-null int64
sqft_above       4600 non-null int64
sqft_basement    4600 non-null int64
yr_built         4600 non-null int64
yr_renovated     224 non-null datetime64[ns]
street           4600 non-null object
city             4600 non-null object
statezip         4600 non-null object
country          4600 non-null object
state            4600 non-null object
zipcode          4600 non-null object
dtypes: datetime64[ns](1), float64(2), int64(10), object(7)
memory usage: 718.8+ KB
```

df.info() จะแสดงสรุปข้อมูลมาให้

นอกจากนั้นยังมีคำสั่ง df.dtypes (ไม่มีวงเล็บ) สำหรับดู Data Type แต่ละคอลัมน์อย่างเดียว

วิธีแปลงประเภทข้อมูล (Data Type) ใน Data Frame

บางครั้งประเภทข้อมูลของคอลัมน์เป็น String แต่เราต้องการ Integer หรือเราต้องการ Date เราสามารถแปลงข้อมูลได้ง่าย ๆ ดังนี้เลยครับ

```
1.  df['hour'] = pd.to_numeric(df['hour']) # แปลงเป็น Numeric
2.  df['hour'] = df['hour'].astype('int') # อีกวิธีในการแปลงค่า สามารถใช้วิธีนี้แปลงเป็น float ได้
```

วิธีเช็ค Summary ของแต่ละคอลัมน์ (count, min, max, mean)

ถ้าเราอยากรู้ Distribution คร่าว ๆ ของแต่ละคอลัมน์ว่าเป็นอย่างไร สามารถใช้คำสั่ง describe() ได้

บางครั้งเราไม่ต้องการรู้ Summary ของทั้งคอลัมน์ แต่อยากให้แสดงผลเฉพาะในคอลัมน์นั้น ๆ ครับ ซึ่งมีประโยชน์มากเวลาเราทำ Data Analysis แล้วอยากรู้ว่าบางกลุ่มมีอะไรผิดปกติหรือเปล่า

```
test = df.groupby(['Gender'])
test.describe()
```

วิธีสร้าง DataFrame ใหม่

วิธีสร้างแบบง่ายที่สุด ถ้าต้องการข้อมูลหลายรูปแบบ เราสามารถใช้ Dictionary แบบนี้เลยครับ

```
dataframe = pandas.DataFrame({
    'C1': pandas.date_range('20170101', periods=4),
    'C2' : [10,20,30,40],
    'C3': pandas.Categorical(['A','B','C','D']),
    'C4': 1})
```

แต่ถ้าเราต้องการแค่เป็นแบบตัวเลขทั่วไป ใช้ Numpy แบบนี้ได้เลย

```
array = numpy.array([(1,2,3), (4,5,6),(7,8,9)])
dataframe = pandas.DataFrame(array,columns=['C1','C2','C3'])
```

วิธีเลือกหลายคอลัมน์จาก DataFrame

ปกติถ้าเราต้องการเลือกแค่ 1 Column ก็เขียนแบบนี้ได้เลย

```
df['C1']
```

แต่ถ้าต้องการเลือกหลายคอลัมน์ ให้ทำแบบนี้

```
df[['C1','C2']]
```

วิธีเลือกคอลัมน์ตามเงื่อนไขที่ต้องการ

```
dataframe2 = dataframe.loc[:,(dataframe>50).any()]
dataframe3 = dataframe.loc[:,(dataframe>50).all()]
```

เราสามารถใช้หาคอลัมน์ที่มี Missing Values หรือหาคอลัมน์ที่ไม่มี Missing Values เลยก็ได้

```
dataframe2 = dataframe.loc[:,dataframe.isnull().any()]
dataframe3 = dataframe.loc[:,dataframe.notnull().all()]
```

วิธีเลือกแถวตามเงื่อนไขที่ต้องการ

```
dataframe[dataframe['C1']>50] # เงื่อนไขแบบง่าย ๆ
dataframe2 = dataframe.loc[dataframe.C1.isin([1,2,3])] # เงื่อนไขแบบซับซ้อน
```

ถ้ามีหลายเงื่อนไขเราสามารถใช้ & (and) หรือ | (or) ได้

```
dataframe[(dataframe['C1']>50) & ((dataframe['C2']<25) | (dataframe['C2']>75))]
```

หรือใช้ Query เป็นเงื่อนไขได้ด้วย มีประโยชน์มากเวลาเรามีเงื่อนไขแปลก ๆ ไม่ต้องเขียนลูปขึ้นมาเองเลยครับ

```
dataframe2 = dataframe.query('C1 > C2')
```

วิธีเพิ่มคอลัมน์ใหม่

สามารถเพิ่มคอลัมน์ใหม่ได้ 2 แบบ คือ

1. เพิ่มโดยอิงจากคอลัมน์เดิม (เช่น เอาคอลัมน์เดิม + 10 หรือ เอาคอลัมน์ A – คอลัมน์ B มีประโยชน์มากตอนทำ Feature Engineering)
2. เพิ่มคอลัมน์โดยตั้งค่า Fix ไปเลยสำหรับทุกแถว ส่วนใหญ่จะใช้วิธีนี้เวลาเราอยากได้ค่าอะไรแปลก ๆ ที่ต้องเขียนลูปเพื่อใส่ค่า ก็สร้างคอลัมน์แบบ Fix ค่าก่อน แล้วต่อด้วยลูป

```
df['new'] = dataframe['old'] + 10 # use old values
```

การใช้งานคำสั่ง Transpose (ทำให้แถวกลายเป็น Vector) ทำได้ง่ายมากๆ แค่

```
dataframe.T
```

การต่อ DataFrame

การต่อ Data Frame คือการเอา Data Set 2 ชุดมาต่อกันในแถวตั้งหรือแนวนอน สำหรับการต่อแบบปะติดไปเลย

มี 2 คำสั่งที่เหมือนกัน คือ concat กับ append แต่ให้ใช้ concat ไปเลย เพราะ append เป็นคำสั่งที่ไม่ Memory Efficient

```
pd.concat([df1,df2], axis=1) # รวมกัน 2 คอลัมน์ (axis = 0 คือแถว, axis = 1 คือคอลัมน์)
pd.concat([df1,df2,df3)] # รวมมากกว่า 2 คอลัมน์ก็ได้
pd.concat(…, ignore_index=True) # รวมเสร็จแล้ว reset index ให้ด้วย ควรใช้ไม่งั้นจะเจอ row ID ซ้ำกันตอนรวมร่าง
pd.concat(…, join='inner') # รวมร่างเฉพาะคอลัมน์ที่ df1 กับ df2 มีทั้งคู่
pd.concat(…, keys=['source1', 'source2']) # เพิ่มคอลัมน์เข้าไปด้วยเพื่อระบุว่า Row แต่ละอันมาจาก Data Frame อันไหน
pd.concat(…, join_axes=[df2.index]) # เลือกรวมร่างเฉพาะ row index ที่เรากำหนดได้
```

การต่อ DataFrame แบบ Join

ถ้าต้องการต่อ DataFrame แบบ Advance หน่อย เราก็สามารถ Join DataFrame ได้เหมือน Join Table ใครเขียน SQL มาก่อนน่าจะถนัดเลย

```
pd.merge(df1, df2, left_on="col1", right_on="col2", how="inner")
```

เราสามารถเปลี่ยนตรง how=”inner” เป็น “outer”, “left”, “right” เพื่อเปลี่ยนเป็น Outer Join, Left Join, Right Join ได้อีกด้วย

การหาค่า Mean, Sum, Max (Aggregate) แบบทั้ง DataFrame

การ Aggregate แบบตามกลุ่มที่ต้องการ

บางทีเราอยาก Aggregate ข้อมูลตามการจัดกลุ่มในคอลัมน์อื่น เช่น เราอยากได้รายจ่ายทั้งหมดของแต่ละคน (ต้อง aggregate sum ของคอลัมน์รายจ่าย โดยแบ่งกลุ่มตามคอลัมน์ User ID) ใช้แบบนี้

```
1.  aggregate = dataframe.groupby('C1').sum()
```

การรัน Function เดียวกันทุกแถว หรือทุกคอลัมน์

เวลาเราอยากรันคำสั่งอะไรสักอย่างสำหรับทุกแถว หรือทุกคอลัมน์ เราสามารถเขียนได้แบบนี้

```
1.  # sum for columns
2.  sum_columns = dataframe[['C1','C2']].apply(sum,axis=0)
3.  # sum for rows
4.  sum_rows = dataframe[['C1','C2']].apply(sum,axis=1)
```

เหมือนกับฟังก์ชั่น apply() ใน R นั่นเอง

รันคำสั่งที่เขียนเองกับทุกแถวใน 1 คอลัมน์

ถ้าต้องการรันคำสั่ง (Function) ที่เขียนเอง สำหรับทุกแถวในคอลัมน์อันใดอันหนึ่ง ใช้แบบนี้ได้

```
1.  dataframe['C1'] = dataframe['C1'].map(lambda x: x-100)
```

รันคำสั่งที่เขียนเองกับทุกค่า

ถ้าต้องการรันคำสั่งที่เขียนเองกับทุกค่าใน DataFrame ใช้โค้ดนี้

```
1.  function_result = dataframe.applymap(lambda x: x*10)
```

Find Correlation & Covariance

เวลาเราอยากรู้ว่าค่าต่าง ๆ ใน Data Set เรา Correlate กันมั้ย

```
dataframe.corr() # Correlation
dataframe.cov() # Covariance
```

แต่ค่าที่ออกมาเป็นตัวเลขอาจจะดูยากนิดนึง เราสามารถพลอตสวย ๆ ด้วย Seaborn ได้ครับ สามารถใช้โค้ดด้านล่างนี้ได้เลย

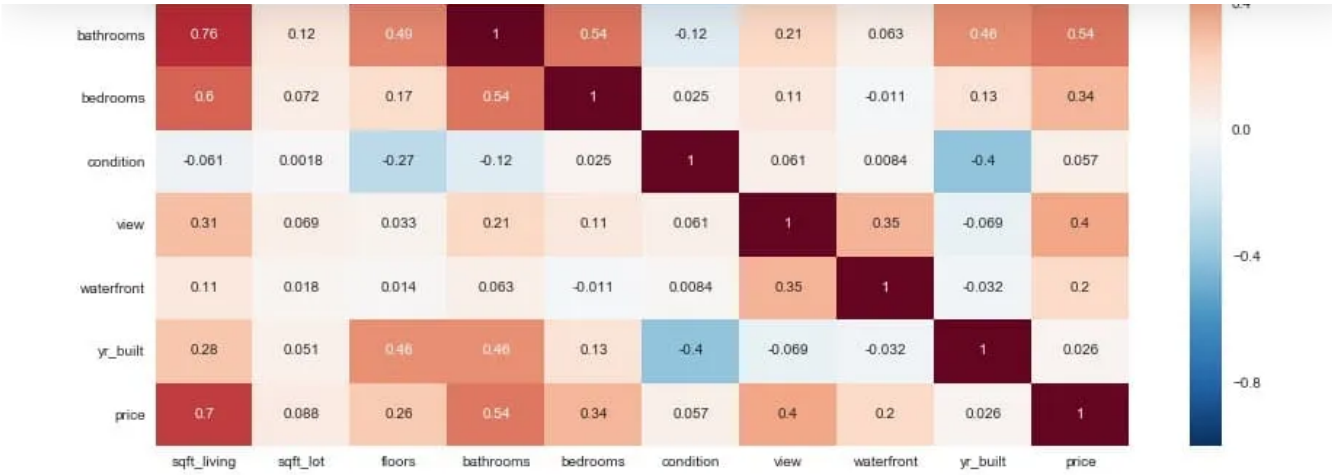
```
import seaborn as sns

corr = modeldf.corr()

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(15, 8))

# Generate a custom diverging colormap
cmap = sns.diverging_palette(10, 10, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, annot=True)
```



Correlation Plot สวย ๆ ด้วย Seaborn

คำนวณ Cross Tabulation

Cross Tabulation มีประโยชน์มากเวลาเราอยากรู้ว่ามี Data ที่ตรงกับกรุ๊ป A ของคอลัมน์ 1 และกรุ๊ป B ของคอลัมน์ 2 เท่าไหร่ เช่น มีนักเรียนผู้ชาย (คอลัมน์ gender) กี่คนในมัธยมปลาย (คอลัมน์ education) แบบนี้เป็นต้น

หรือถ้าใครใช้ PivotTable ใน Excel มาก่อน ก็เหมือนกันเลยครับ

```
1.  aggregate = pandas.crosstab(dataframe.C1, dataframe.C2)
```

วิธีหาค่า Unique ในแต่ละคอลัมน์

คำสั่งนี้มีประโยชน์มาก เอาไว้ใช้เช็คว่าแต่ละคอลัมน์มีค่าแปลก ๆ มั้ย

ตัวอย่างการใช้งานก็คือ เราอยากรู้ว่า มีบ้านไหนที่มีจำนวนห้องนอนแปลก ๆ มั้ย (เช่น 50 ห้องนอน หรือ -5 ห้องนอน) ก็หาค่า unique จากคอลัมน์ “bedrooms”

```
1.  dataframe['C1'].unique()
```

คืออะไร)

```
dataframe.duplicated() # หาอันที่เหมือนกันทุกคอลัมน์
dataframe.duplicated('C1') # หาอันที่ซ้ำกันเฉพาะคอลัมน์ C1
dataframe.duplicated(['C1', 'C2']) # หาอันที่ซ้ำกันเฉพาะคอลัมน์ C1 และ C2
```

ปกติแล้วถ้ามีไอเทมซ้ำ คำสั่งนี้จะไม่แสดงไอเทมแรกในกลุ่มที่ซ้ำ (เช่น ถ้า C1=5 มี 2 แถว มันจะแสดงเฉพาะแถวที่ 2) เราสามารถใส่ Argument **keep=False** เข้าไปเพื่อบังคับให้มันแสดงทุกแถวได้

นอกจากนั้นเรายังสามารถนับจำนวนแถวที่ Duplicate และลบทิ้งได้ด้วย

วิธีการนับจำนวน Duplicate

```
len(df[ df.duplicated(['A', 'B', 'C'], keep = False) ])
```

วิธีการลบ Duplicate

เอาไว้ใช้ตอนเราเจอว่าทุกคอลัมน์ซ้ำกันหมดเลย ซึ่งเป็นเคสที่บอกว่าข้อมูลน่าจะซ้ำ ลบออกได้ (ขึ้นอยู่กับข้อมูลด้วยนะครับ บางข้อมูลอาจจะไม่ได้แปลว่าซ้ำแล้วลบได้):

```
# Remove the duplicates
df.drop_duplicates(['A', 'B', 'C'], inplace=True)

# Reset dataframe index after drop_duplicates.
df.reset_index(drop=True, inplace=True)

len(df)
```

สำหรับโค้ดข้างบน จะเห็นว่าเราต้อง reset index หลังลบ duplicate ด้วยนะครับ

วิธีการลบแถว และลบคอลัมน์

ลบคอลัมน์สามารถทำได้แบบนี้

นท์ DataFrame)

```
dataframe = dataframe.drop(5, axis=0)
dataframe.reset_index(drop=True) # Reset index
```

ลบแถวแล้วอย่าลืมเช็คด้วยว่าที่ลบไปถูกต้องมั้ย และหลังจากลบแถวต้อง Reset Index ด้วย

วิธีการลบแถวที่มี Missing Value

ข้อควรระวัง: การที่อยู่ ๆ เราลบแถวที่มี Missing Value ทิ้งไปเลยอาจจะไม่ใช่วิธีที่ดีที่สุดในการทำ Data Analysis เสมอไปนะครับ บางเคสการ Impute (คำนวณหาค่าไปใส่) จะดีกว่าครับ

```
dataframe2 = dataframe.dropna(axis=0)
```

วิธีแทนค่า Missing Value ด้วยค่าเฉลี่ย (Mean Imputation)

วิธีหนึ่งในการแทนค่าที่หายไป คือการทำสิ่งที่เรียกว่า Mean Imputation หรือหาค่าเฉลี่ยของคอลัมน์นั้น แล้วเอามาแทนค่าที่หายไปนั่นเองครับ

ข้อดีของการทำ Mean Imputation คือ สามารถทำได้ง่าย แต่ก็ต้องระวังเรื่องข้อเสีย เช่น ทำแบบนี้จะเป็นการไม่สนใจความสัมพันธ์ระหว่างตัวแปร ทำให้เกิด Bias สูง ควรใช้เฉพาะเวลา Missing Value ไม่เยอะเท่านั้นครับ

สามารถรันโค้ดด้านล่างเพื่อทำ Mean Imputation ได้ง่าย ๆ เลย

```
import numpy as np
meanAge = np.mean(df.Age) # Get mean value
df.Age = df.Age.fillna(meanAge) # Fill missing values with mean
```

การลูปข้อมูลแต่ละคอลัมน์ และแต่ละแถว

```
for col_idx,data in dataframe.iteritems():
    print ("column:",col_idx)
    print ("column data:")
    print (data,"\n")
```

การลูปข้อมูลแต่ละแถว

```
for col_idx,data in dataframe.iterrows():
    print ("row:",col_idx)
    print ("row data:")
    print (data,"\n")
```

วิธีเปลี่ยน DataFrame จากแบบ Wide เป็น Long (Melt)

การ Melt Data มีประโยชน์มากเวลาเราต้องการเอาข้อมูลไปพลอต Data Visualization หรือเราต้องการ Aggregate ครับ

```
dataframe2 = dataframe.melt()
```

วิธีการเปลี่ยนชื่อคอลัมน์ (Rename)

บางทีเราต้องการเปลี่ยนชื่อเพื่อให้สั้นลง ให้พิมพ์สะดวกขึ้น สามารถทำได้ดังนี้

```
dataframe.rename(columns={'old':'new'},inplace=True)
```

วิธีการใส่คำนำหน้าคอลัมน์ (Prefix)

อันนี้มีประโยชน์มากตอนเรามีข้อมูลหลาย ๆ ชุด และต้องการ Merge โดยอยากให้ชื่อคอลัมน์ไม่ซ้ำกัน

```
thisdata = thisdata.add_prefix('data_')
```

วิธีการแทนค่าใน DataFrame

```
dataframe2 = dataframe.replace(1, -100)
```

เราสามารถ Replace หลายค่าพร้อมกันได้ด้วยครับ และสามารถกำหนด Column ที่ต้องการให้แทนค่าได้ด้วย

```
df['city'].replace({
        'BKK':'Bangkok',
        'BNK':'Bangkok'
    }, inplace=True)
```

วิธีการ Export DataFrame เป็นไฟล์ CSV

หลังจากที่เราจัดการ Data เรียบร้อยแล้ว ก็สามารถ Export เป็น CSV เอาไปใช้ต่อกับโปรแกรมอื่น หรืองานส่วนอื่น ๆ ได้ (แอดทำบ่อยเพราะบางทีต้องสลับ Python <-> R รัว ๆ)

```
dataframe.to_csv('dataframe.csv')
```

สรุปการใช้งาน Pandas

จากเทคนิคต่าง ๆ ด้านบน จะเห็นว่า Pandas มีฟังก์ชั่นให้เราทำงานได้ง่ายขึ้นมาก ๆ ครับ หวังว่าบทความนี้จะมีประโยชน์กับท่านที่กำลังหัดใช้งาน Pandas กันอยู่นะครับ :)

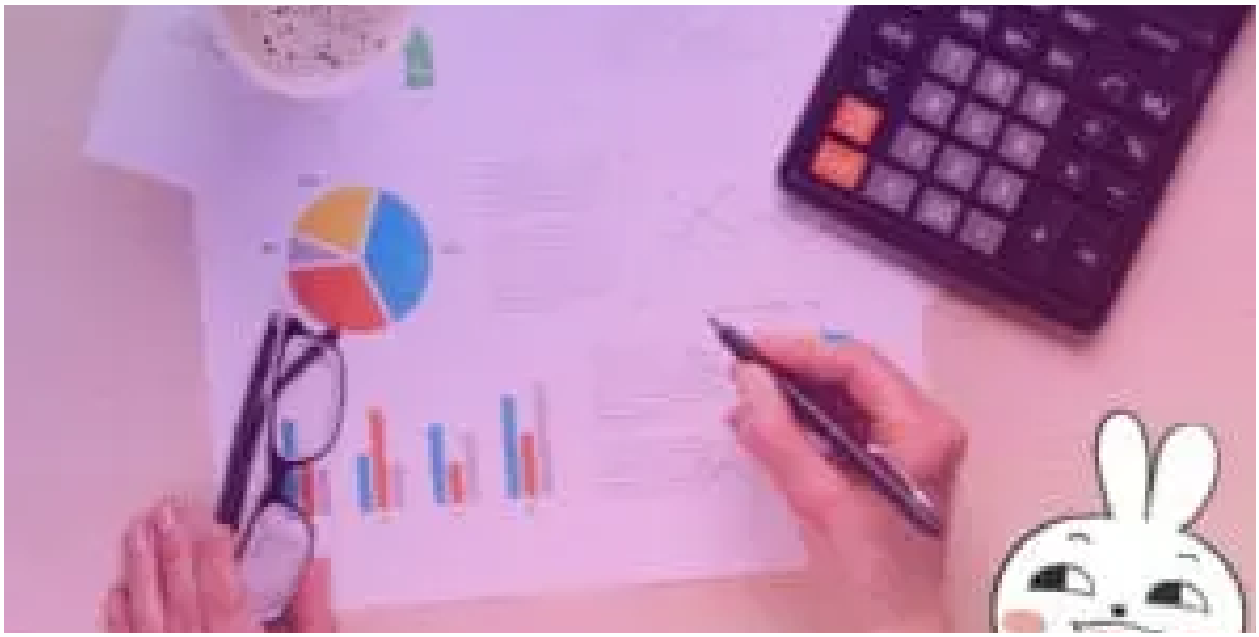
หมวดหมู่: Data Science # แท็ก: Data Cleaning, Data Science, Data Wrangling, Pandas, Python

บทความอื่น ๆ ที่เกี่ยวข้อง

ไม่เก่ง Data Science ก็ทำ Machine Learning ได้!? มารู้จักกับ Cloud Computing และบริการเด็ด

เรียนปริญญาโท Data Science ดีมั้ย? มาดูข้อดี-ข้อเสียกัน

เจาะลึกแนวคิดการนำข้อมูลไปสร้างโอกาสใหม่ทางธุรกิจ ในช่วง COVID19 จากพี่โบ๊ท CEO Builk



3 คำถาม Business Analyst vs Data Scientist ช่วยตัดสินใจ เลือกเรียน/ทำงานสายไหนดี

สัมภาษณ์แอดบอยด์ Data Scientist ที่ SCB เจ้าของเพจ BigData RPG

PREVIOUS NEXT

Perth N

Data Scientist ที่ทำเว็บไซต์หน้าตาดีคนนึง ที่มีความสนใจด้าน Marketing / Growth Hacking อย่างมาก ชอบเกี่ยวกับ Data Science การนำข้อมูลมาใช้ประโยชน์ทางธุรกิจเป็นพิเศษ

บทความอื่น ๆ จากผู้เขียน

Join the Conversation

3 Comments

Pingback:

Python Workshop เรียนจริง ลงมือทำจริง สอนโดยอาจารย์จุฬา - Data Science เข้าใจง่าย ๆ กับ DataTH

Sittidech Lobyam

May 29, 2019 at 4:00 pm

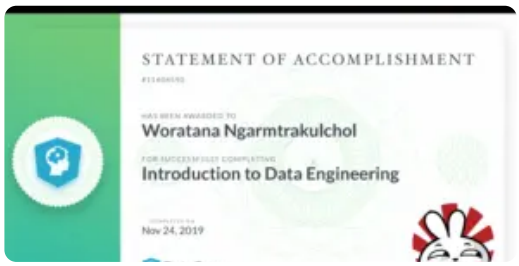
AttributeError: Module ‘pandas’ has no attribute ‘Excelwriter’
แก้ไขอย่างไรครับ ถ้าเกิด alarm ตัวนี้ครับ

Perth N

June 8, 2019 at 5:59 am

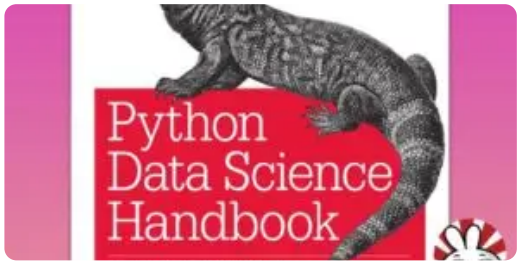
ลองเช็ค Version ของ Pandas ดูครับผม คิดว่าน่าจะเป็นเวอร์ชั่นเก่าอยู่ แนะนำให้ลองลบแล้วติดตั้งใหม่ครับจะได้เวอร์ชั่นล่าสุด

บทความแนะนำ



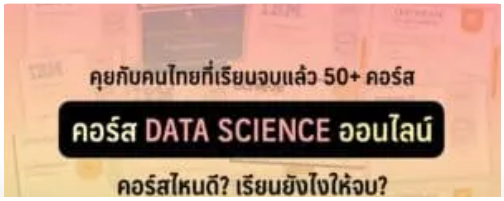
สรุปคอร์ส Intro to Data Engineering จาก DataCamp เข้าใจง่าย ครบ เหมือนเรียนเอง

แนะนำหนังสือ 6 เล่มสำหรับเริ่มหัดทำ Data Science



แจกฟรี: หนังสือ Python for Data Science สอนพื้นฐานครบ!

[แจกวีดิโอภาษาไทย] สอนเขียน R Programming เข้าใจง่าย ๆ ใน 20 นาที



อยากเป็น Data Scientist คุณเองก็เริ่มได้! คุยกับคนไทยที่เรียนคอร์สออนไลน์จบแล้วกว่า 50 คอร์ส