運用實際資料集進行資料視覺化練習

簡報閱讀 範例與作業 問題討論 學習心得(完成)

重要知識點

- 了解如何使用Pandas 處理資料集，並加以視覺化效果
- 完成今日課程後你應該可以了解

```
# 導入資料集
```

數據視覺化的好處

- 當你把數據轉換成了規範的格式，也已經採用了適當的統計和分析，接下來就是展示結果的時候了，這時候數據可視化排上了用場在可視化分析中，經常會遇到多個數據分布之間的比較，分布不同，用到的表達方式也不一樣。

- 在對不同的分布數據進行比較時，通常有兩種形式，要麼突出異常值的差異，要麼突出它們各自差異的細微差別。比如，在統計過程中，不同標準的數據集會有怎樣的差別，或者，如何通過分析來改善評分功能。

- 瞭解有關資料集屬性

我們可以使用 info()或是 descript() 方法瞭解有關資料集屬性的更多資訊。

特別是行和列的數量、列名稱、它們的數據類型和空值數

導入數據集

Seaborn 在庫中附帶了幾個重要的數據集。安裝 Seaborn 後，數據集會自動下載。
您可以使用這些資料集中的任何一個進行學習。借助以下函數，您可以載入所需的數據集。

load_dataset()

- Seaborn 可以直接把 PANDAS 的 dataframe 當成資料匯入

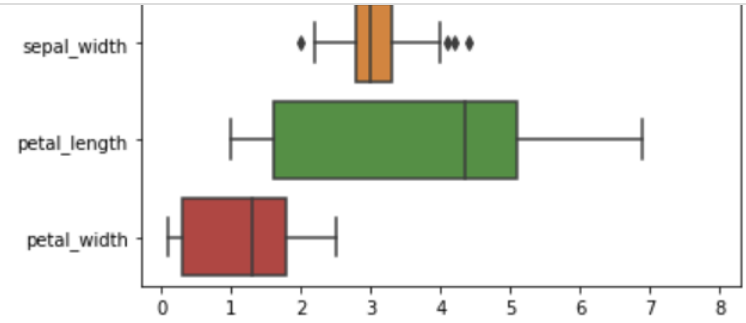
```
# 導入必要的程式庫
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt

# 取得鳶尾花資料集
df = sns.load_dataset('iris')
```

```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

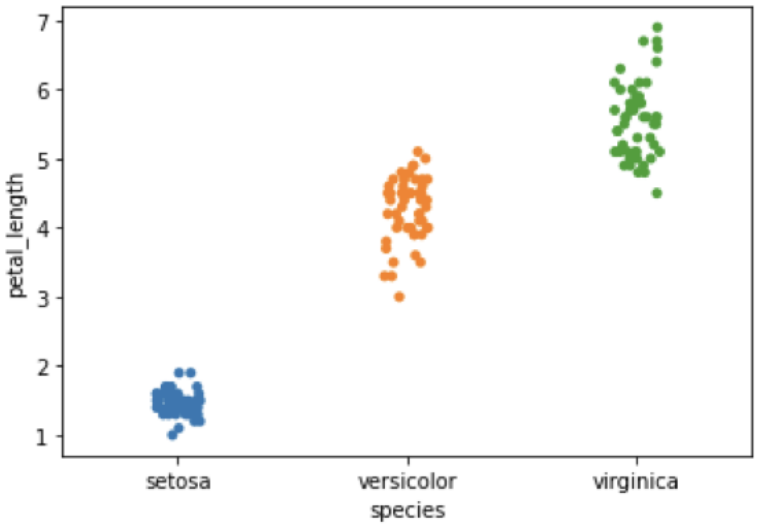
瞭解數據集

```
# 直接使用PANDAS dataframe，當作參數

# 箱形圖顯示了數據的總體分布，同時繪製了異常值的數據點。這個物理點讓它們的特定值在樣本之間容易被識別和比較。

sns.boxplot(data = df, orient = "h")
```



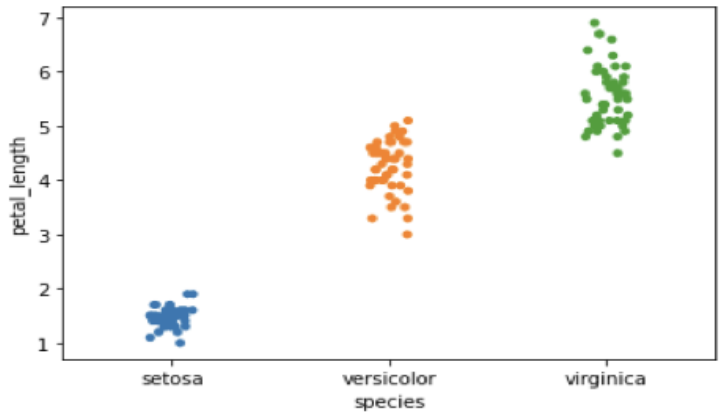
當一個或兩個正在研究的變數是分類的時，我們使用像條帶線()、swarmplot()等的圖。
查看到每個物種petal_length的差異。但是，散點圖的主要問題是散點圖上的點重疊。

```
sns.stripplot(x = "species", y = "petal_length",
data = df)
```



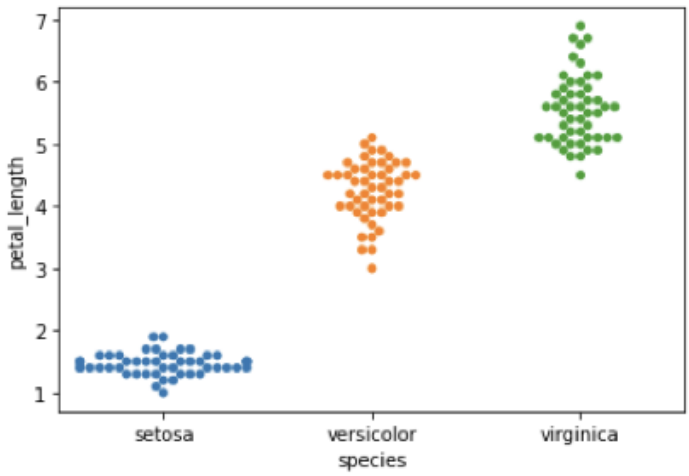
分類式的變數

上述散點圖的主要問題是散點圖上的點重疊。我們使用"抖動"參數來處理此類方案。
抖動會為數據添加一些隨機雜訊。此參數將沿分類軸調整位置。



另一個可以用作「抖動」的替代選項是函數群圖()。
此函數將散點圖的每個點都放在分類軸上，從而避免重疊點

```
sns.swarmplot(x = "species", y = "petal_length",
data = df)
```

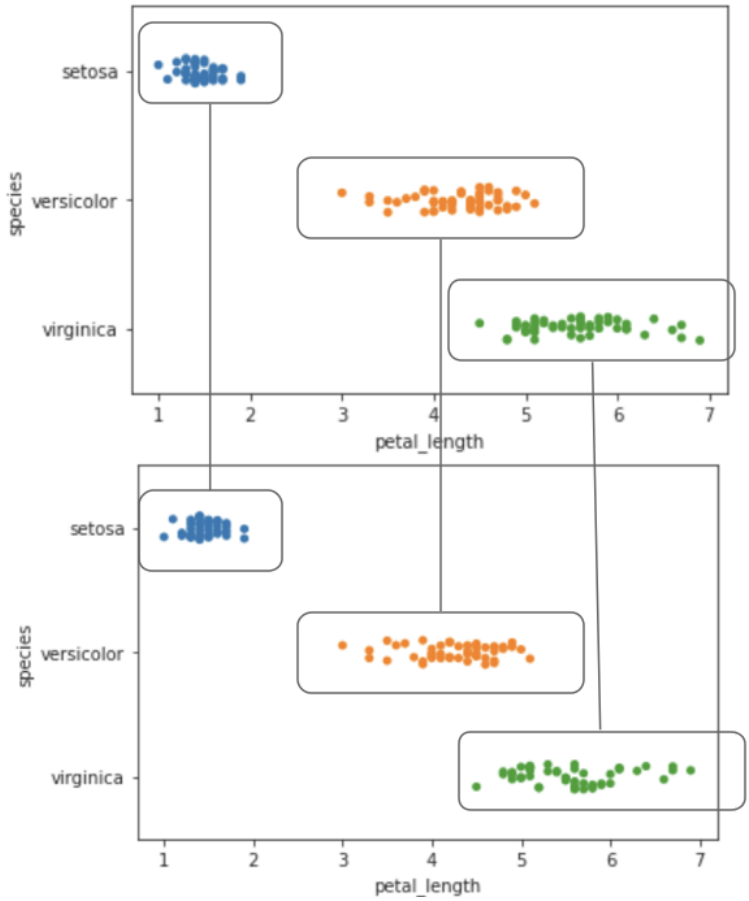


可以換個方向

```
sns.stripplot(x = "petal_length", y = "species",
data=df, jitter=True)
plt.show()
```

dodge : True / False, 若設置為True則沿著分類軸，將數據分離出來

plt.show()



圖形的密集與分布點的位置都有變化
dodge 比 jitter 更能貼近實際的分布狀況
若是要做分群，其實使用 dodge 更好觀察

知識點回顧

stripplot **參數說明**

1. x , y , hue：數據特徵，根據實際數據，x , y常用來指定x , y軸的分類名稱，
2. hue 常用來指定第二次分類的數據類別(用顏色區分)
3. data : DataFrame

講，就是讓數據分散開)

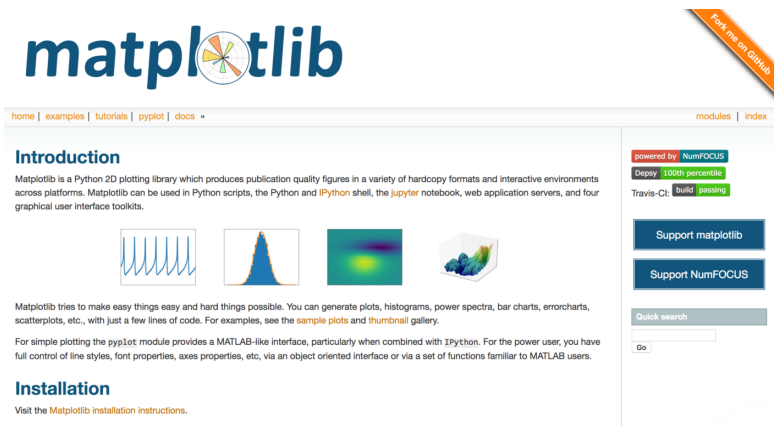
5. dodge : True / False，若設置為True則沿著分類軸，將數據分離出來成為不同色調級別的條帶，否則，每個級別的點將相互疊加
6. orient : 作用：設置圖的繪製方向(垂直或水平)
7. color : matplotlib顏色
8. palette :調色板名稱，用於對數據不同分類進行顏色區別
9. size : 設置標記大小
10. edgecolor : 設置每個點的周圍線條顏色
11. linewidth : 設置構圖的線寬度

延伸閱讀

使用 Seaborn 進行可視化

網站：[資料分析&機器學習] 第2.5講：資料視覺化(Matplotlib, Seaborn, Plotly)

- 針對 Matplotlib, Matplotlib & Pandas 帶入實例
- 針對 Seaborn, Seaborn & Pandas 帶入實例



方框

- 說明如何解決中文字在圖形得顯示問題

```
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(font="simhei")#遇到標籤需要漢字的可以在繪圖前加上這句
f, ax = plt.subplots(figsize=(10,10))
sns.heatmap(df, annot=False, ax=ax)
```

資料分布從離散型的資料轉為連續型的資料分布

《數據分析》【專有名詞】核密度估計(Kernel Density Estimation, KDE) @ 宅馬窩 :: 痞客邦 :: (pixnet.net)

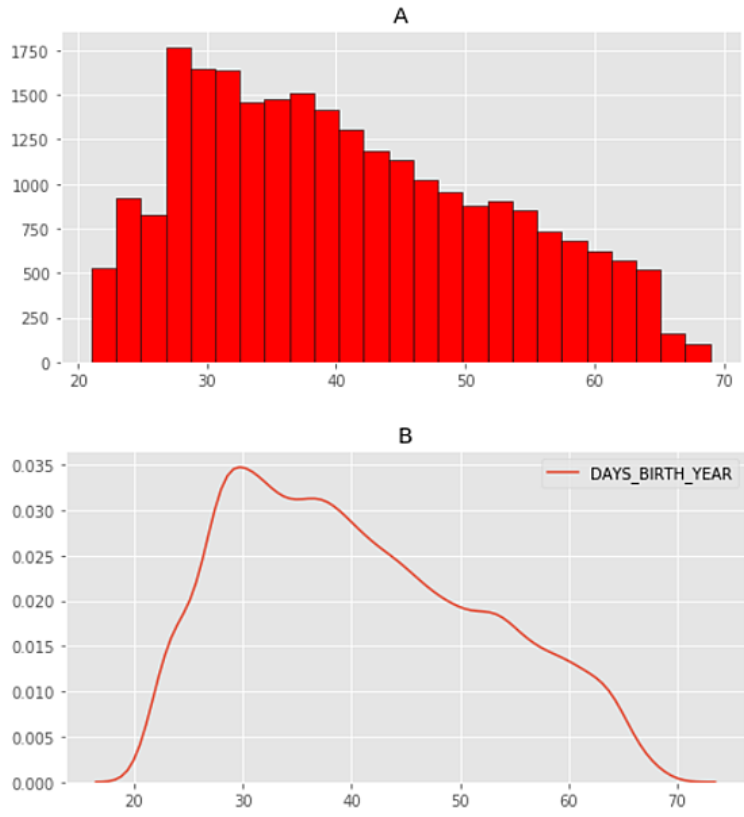
- 把直方圖轉畫成折線圖：核密度估計其實是對直方圖的一個延伸應用。把直方圖轉畫成折線圖。資料分布從離散型的資料轉為連續型的資料分布，可以觀察數據的分布與趨勢。

- KDE 核密度估計法的說明：

- Kernel Density Estimation 核密度估計法，目的是從給定的樣本中找出隨機變數的機率密度函數。密度函數就是分佈函數的一階導數。所以，我們是可以通過估計分佈函數的一階導數來估計密度函數。

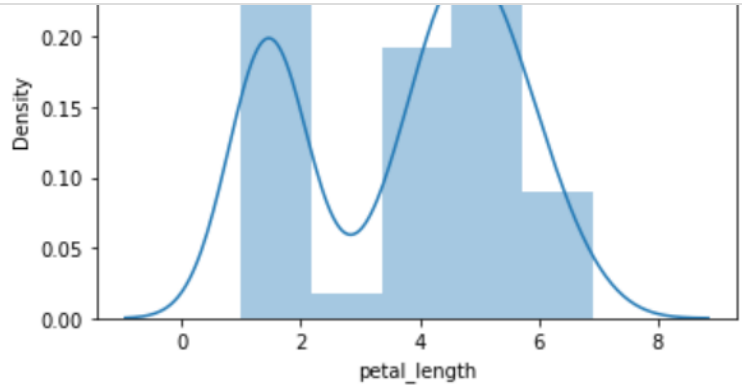
- 直方圖跟折線圖的數據點都是頂點(波峰)，兩者共通點就是圖形的覆蓋面積，從一些比

|x|,-1≤x≤1。



核密度估計(Kernel Density Estimates, KDE)

```
#載入相關套件
import pandas as pd
import seaborn as sb
from matplotlib import pyplot as plt
#載入資料集
df = sb.load_dataset('iris')
#使用displot()來顯示圖表
# "hist" 代表柱狀圖的顯示與否
sb.distplot(df['petal_length'],hist=True)
#把核密圖的圖表, 表示出來
plt.show()
# 顯示花的長度分布狀況
```



- 在處理一組數據時，通常首先需要瞭解變數的分佈方式。
- 直方圖
- 直方圖通過沿數據範圍形成條柱，然後繪製條形以顯示每個條柱中的觀測值數來表示數據的分佈。為了說明這一點，讓我們刪除密度曲線並添加一個地毯圖，該圖在每個觀測值上繪製一個小的垂直刻度。
- 繪製直方圖時，主要選擇是要使用的條柱數和放置位置。distplot() 使用簡單的規則來正確猜測預設情況下正確的數位，但嘗試更多或更少的 bin 可能會顯示資料中的其他特徵：

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats
```

```
sns.set(color_codes=True)
 x = np.random.normal(size=100)
sns.distplot(x);
sns.distplot(x, bins=20, kde=False, rug=True);
```

下一步：閱讀範例與完成作業