資料分析與學習基石

Homework 1 First visit in Kaggle data

基本資料集描述

資料集名稱：Data Scientist Salary

資料集介紹：

Data Scientist Salary 是由一位名叫 Nikhil Bhathi 的人使用網頁爬蟲套件“Selenium”從 www.glassdoor.com 網頁中擷取與 2021 Data Scientists 相關的工作海報資訊， 每個工作擁有以下這些資料欄位：1. Job title, 2.Salary Estimate, 3. Job Description, 4. Rating, 5. Company, 6. Location, 7. Headquarters, 8. Size, 9. Company Founded Date, 10. Type of Ownership, 11. Industry, 12. Sector, 13. Revenue, 14. Competitors. 要注意的一點是，並不是每個工作都有相對應的資訊，所以作者 Nikhil Bhathi 在處理這些沒有資訊的工作欄位時，一率填上 “-1”。

資料欄位：

Job title	Salary	Job Description	Rating	Company	Location	Headquarters

Size	Company Founded Date	Type of Ownership	Industry	Sector	Revenue	Competitors

Job title: 工作名稱

Salary: 薪資範圍

Job Description: 此工作公司所需要的特質 、工作預期實況

Rating: 對此工作評價，點數 -1.0~5.0

Company: 公司名稱

Location: 工作地點

Headquarters: 公司總部地點

Size: 公司員工數量範圍

Founded: 公司成立時間

Type of ownership: 公司的所有權 “private, public, government owned”

Industry: 公司類別 eg. IT industry, pharmaceutical industry

Sector: 公司的界別

Revenue: 公司的總收入

Competitors: 公司的競爭對手

資料特性與分析

根據作者整理出來的資料欄位，資料分析後最主要的產出肯定是資料科學家薪水與公司、地緣之間的視覺化關聯，以規模較大的角度來審視，最有可能且最有價值的例子就是找到

「 國家、地方近期受益於資料科學而賺錢的產業鏈 」 ；

另一方面個體取向，「 人們、社會新鮮人能夠透過統整後的資料特性擴大視野看到其他資料科學相關工作，或是了解原本已感興趣的工作是否適合自己 」。

「 是否有前景、徵招有能力的資料科學家遞補進產業鏈 」

這就是這份資料集的特性及分析後能得到的價值。

Code 方法與比較 1:

套件 Package:

```python
#pandas library use to load dataset and also manipulate tabular data
import pandas as pd
#numpy library use to do array operations and also to do calculations
import numpy as np
#matplot library use to plot different graphs
import matplotlib.pyplot as plt
#seaborn library use to plot differentplots
import seaborn as sns
# figure size in inches
from matplotlib import rcParams
#importing stats model to add intercept and to implement OLS
import statsmodels.api as sm
#importing sklearn for split data
from sklearn.model_selection import train_test_split
from statsmodels.tools.eval_measures import rmse
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
# importing random lib for replace age values
import random
```

Python

此 code 是為了運用現有資訊（Job title, location, etc），也就是資料欄位，來預測平均薪水有多少，為了讓建立的模型

更有普遍性，作者將操作分成兩大部分，一為資料前處理（刪去不必要的欄位、更正欄位等），再來才是建立模型的code。

載入、觀察資料集 Load & Observation

這邊使用 pandas 套件讀入表格、建立 DataFrame

pd.read_cv() 讀入 .csv

head() 則是顯示 DataFrame

```python
# Here we are importing dataset by using pandas
df= pd.read_csv("/kaggle/input/data-scientist-salary-us-glassdoor/data_cleaned_2021.csv")
```

Python

```python
salary=df #creating new data frame
```

Python

```python
pd.options.display.max_columns=None
```

Python

```python
# Here we are checking data is loaded or not and also checking top 5 observations
salary.head()
```

Python

	index	Job Title	Salary Estimate	Job Description	Rating	Company Name	Location	Headquarters	Size	Founded	Type of ownership	Indu
0	0	Data Scientist	$53K-$91K (Glassdoor est.)	Data Scientist\nLocation: Albuquerque, NM\nEdu...	3.8	Tecolote Research\n3.8	Albuquerque, NM	Goleta, CA	501 - 1000	1973	Company - Private	Aeros & De
1	1	Healthcare Data Scientist	$63K-$112K (Glassdoor est.)	What You Will Do:\n\nI. General Summary\n\nThe...	3.4	University of Maryland Medical System\n3.4	Linthicum, MD	Baltimore, MD	10000+	1984	Other Organization	H Servi Hos
2	2	Data Scientist	$80K-$90K (Glassdoor est.)	KnowBe4, Inc. is a high growth information sec...	4.8	KnowBe4\n4.8	Clearwater, FL	Clearwater, FL	501 - 1000	2010	Company - Private	Se Ser
3	3	Data Scientist	$56K-$97K (Glassdoor est.)	*Organization and Job ID**\nJob ID: 310709\n\n...	3.8	PNNL\n3.8	Richland, WA	Richland, WA	1001 - 5000	1965	Government	E
4	4	Data Scientist	$86K-$143K (Glassdoor est.)	Data Scientist\nAffinity Solutions / Marketing...	2.9	Affinity Solutions\n2.9	New York, NY	New York, NY	51 - 200	1998	Company - Private	Adver Mark

將 “-1” 和 value -1 用NaN取代

```python
salary=salary.replace(['-1'],[np.nan])
```

Python

```python
salary=salary.replace([-1],[np.nan])
```

Python

處理null (None, NaN)的欄位

找出要處理的欄位

```python
salary.isnull().sum()
```

✓ 0.2s Python

Output exceeds the size limit. Open the full output data in a text editor

```
Job Title              0
Salary Estimate        0
Job Description        0
Rating                11
Company Name           0
Location               0
Headquarters           1
Size                   0
Founded               50
Type of ownership      0
Industry              10
Sector                10
Revenue                0
Competitors          460
Hourly                 0
Employer provided      0
Lower Salary           0
```

補值（物件或字串）將NaN, None用該欄位出現頻率最高的值填補

mode() 會回傳出現頻率最多次的欄位，若進行補值，則要使用indexing (I.e. mode ( ) [0])

```
salary["Competitors"] = salary["Competitors"].fillna(salary["Competitors"].mode()[0])
```
✓ 0.3s Python

```
salary["Industry"] = salary["Industry"].fillna(salary["Industry"].mode()[0])
```
✓ 0.5s Python

```
salary["Sector"] = salary["Sector"].fillna(salary["Sector"].mode()[0])
```
✓ 0.2s Python

```
salary["Headquarters"] = salary["Headquarters"].fillna(salary["Headquarters"].mode()[0])
```
✓ 0.2s Python

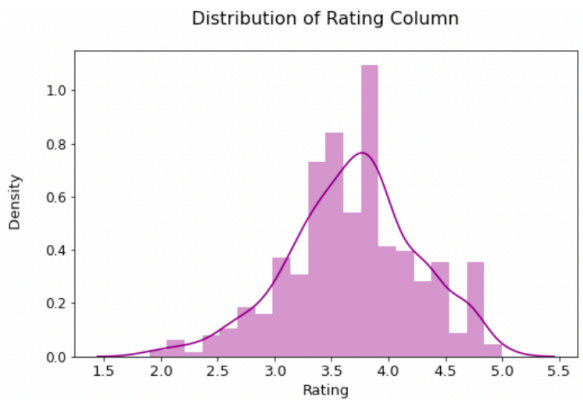
3. 補值（數值）

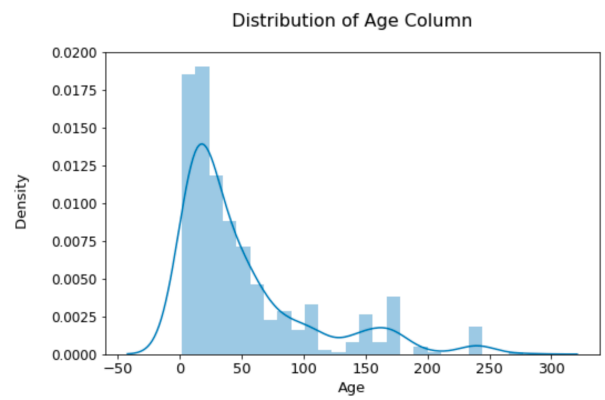
為了不影響原欄位代表之特性，補值前考慮數值 Density Distribution

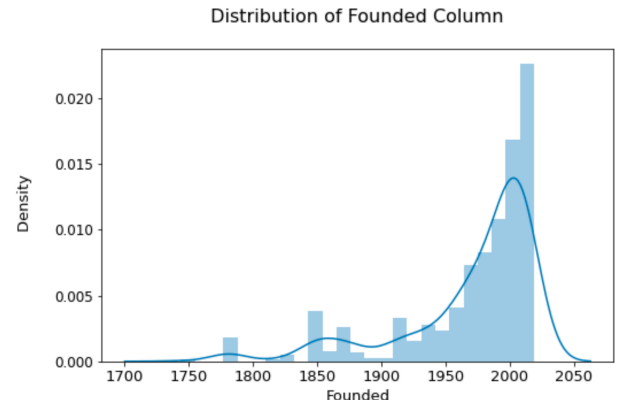
```
# Lets look at how the 'Rating' column is distributed:

plt.figure(figsize=(8,5))
plt.title('\n Distribution of Rating Column \n', size=16, color='black')
plt.xlabel('\n Rating \n', fontsize=13, color='black')
plt.ylabel('\n Density\n', fontsize=13, color='black')
plt.xticks(fontsize=13)
plt.yticks(fontsize=12)
sns.distplot(salary.Rating,color="purple")
plt.show()
```
✓ 0.3s Python

```
# Lets look at how the 'Age' column is distributed:

plt.figure(figsize=(8,5))
plt.title('\n Distribution of Age Column \n', size=16, color='black')
plt.xlabel('\n Age \n', fontsize=13, color='black')
plt.ylabel('\n Density\n', fontsize=13, color='black')
plt.xticks(fontsize=13)
plt.yticks(fontsize=12)
sns.distplot(salary.Age)
plt.show()
```
✓ 0.1s Python

```
# Lets look at how the 'Founded' column is distributed:

plt.figure(figsize=(8,5))
plt.title('\n Distribution of Founded Column \n', size=16, color='black')
plt.xlabel('\n Founded \n', fontsize=13, color='black')
plt.ylabel('\n Density\n', fontsize=13, color='black')
plt.xticks(fontsize=13)
plt.yticks(fontsize=12)
sns.distplot(salary.Founded)
plt.show()
```
✓ 0.1s Python







分佈像 Normal Distribution ，則使用 Normal Distribution Mean

分佈若為 Skewed Normal Distribution，則使用 Distribution Median

```
salary["Rating"] = salary["Rating"].fillna(salary["Rating"].mean())
```
Python

```
salary["Age"] = salary["Age"].fillna(salary["Age"].median())
```
Python

```
salary["Founded"].median()
```
✓ 0.2s Python

```
1992.0
```

```
salary["Founded"] = salary["Founded"].fillna(salary["Founded"].median())
```
Python

最後確認是否遺漏

```
salary.isnull().sum()
```
✓ 0.2s Python

```
Output exceeds the size limit. Open the full output data in a text editor
Job Title          0
Salary Estimate    0
Job Description    0
Rating             0
Company Name       0
Location           0
Headquarters       0
Size               0
Founded            0
Type of ownership  0
Industry           0
Sector             0
Revenue            0
Competitors        0
Hourly             0
```

去除多餘的字串資訊和去除相異性

字串簡潔化 （i.e. : Rating 參雜在公司名稱）

```
salary["Company Name"]
```

✓ 0.3s Python

```
0                  Tecolote Research\n3.8
1      University of Maryland Medical System\n3.4
2                        KnowBe4\n4.8
3                          PNNL\n3.8
4                  Affinity Solutions\n2.9
                   ...
737                         GSK\n3.9
738                   Eventbrite\n4.4
739          Software Engineering Institute\n2.6
740                     Numeric, LLC\n3.2
741           Riverside Research Institute\n3.6
Name: Company Name, Length: 742, dtype: object
```

```
salary["Company Name"]=salary["Company Name"].str[:-4] #"company Name column conatin name with rating also so we are removing
```

✓ 0.4s Python

value_counts ( ) 計算出”compare” 欄位的元素統計值 True ? False ?

```
salary['compare'] = (salary['Company Name'] == salary['company_txt'])
salary['compare']
```

Python

+ Code + Markdown

```
salary['compare'].value_counts()# checking both column values are same or not but we get to know that 11 are different values
```

✓ 0.3s Python

```
True    731
False    11
Name: compare, dtype: int64
```

loc [ ] 方法能定位指定的列位及欄位項

因此將有所不同的欄位項進行更正

```
salary.loc[[176,266], 'company_txt'] = "CA-One Tech Cloud" #As name gets reduced so we will change name of above index
```

✓ 0.3s Python

```
salary.loc[[176,266], 'Company Name'] = "CA-One Tech Cloud"
```

✓ 0.4s Python

```
salary.loc[[370,404,551,609,641,730], 'Company Name'] = "Kronos Bio"
```

✓ 0.3s Python

```
salary.loc[[370,404,551,609,641,730], 'company_txt'] = "Kronos Bio"
# salary.loc[[370]].columns
```

✓ 0.4s Python

```python
salary.loc[[300], 'company_txt'] = "Muso"
salary.loc[[300], 'Company Name'] = "Muso"
```

✓ 0.2s Python

+ Code + Markdown

```python
salary.loc[[456], 'company_txt'] = "ALIN"
salary.loc[[456], 'Company Name'] = "ALIN"
```

✓ 0.3s Python

```python
salary.loc[[581], 'company_txt'] = "Monte Rosa Therapeutic"
salary.loc[[581], 'Company Name'] = "Monte Rosa Therapeutic"
```

✓ 0.3s Python

最後確認

```python
salary['compare'] = (salary['Company Name'] == salary['company_txt'])
```

✓ 0.3s Python

```python
salary['compare'].value_counts()
```

✓ 0.3s Python

```
True   742
Name: compare, dtype: int64
```

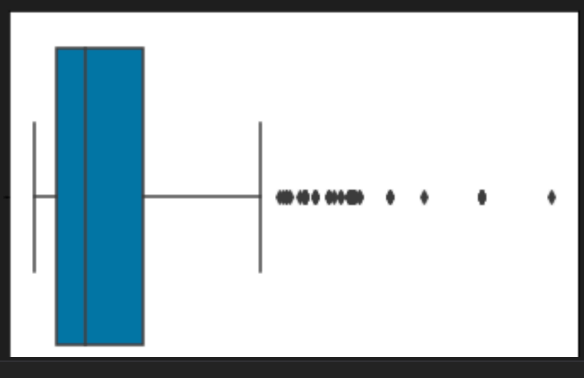
去除年齡Outlier

quantile ( ) 可以計算出特定%之累積值

```python
sns.boxplot(x="Age",data=salary)
```

✓ 0.7s Python

```
<AxesSubplot:xlabel='Age'>
```



```python
Q1=salary["Age"].quantile(0.25)
Q3=salary["Age"].quantile(0.75)
print(Q1, Q3)
# IQR=Q3-Q1
# print(IQR)
```

✓ 0.4s Python

```
14.0 60.0
```

為了讓模型不會受離群值影響，所以將離群值往累計25%及75%間移動

```
salary.loc[salary['Age'] < 14 , 'Age'] = random.randint(14,60)
```
✓ 0.4s Python

```
salary.loc[salary['Age'] > 60 , 'Age'] = random.randint(14,60)
```
✓ 0.3s Python

關係探討移除不必要的值（降維）

Degree and Seniority

```
salary.seniority_by_title.value_counts()
```
✓ 0.3s Python

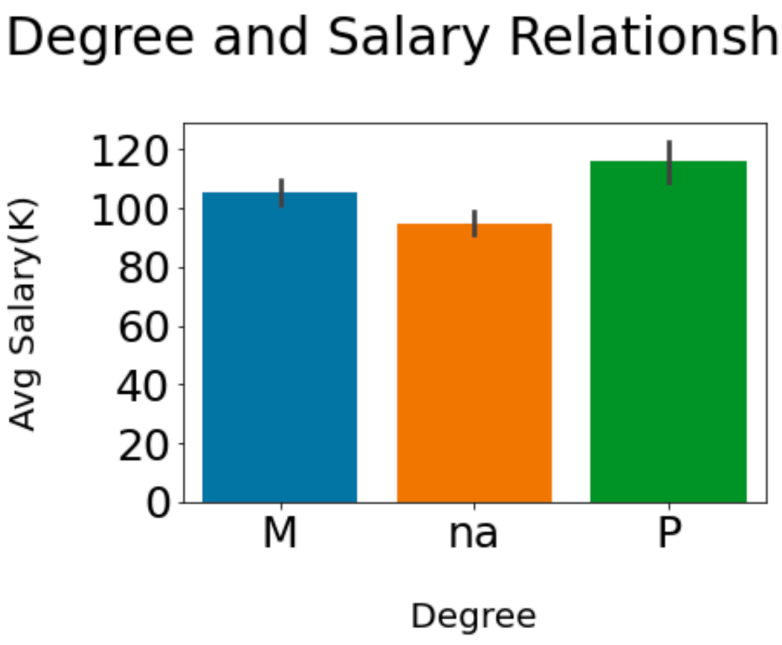
```
na   519
sr   220
jr     3
Name: seniority_by_title, dtype: int64
```

```
salary.Degree.value_counts()
```
✓ 0.3s Python

```
na   383
M    252
P    107
Name: Degree, dtype: int64
```

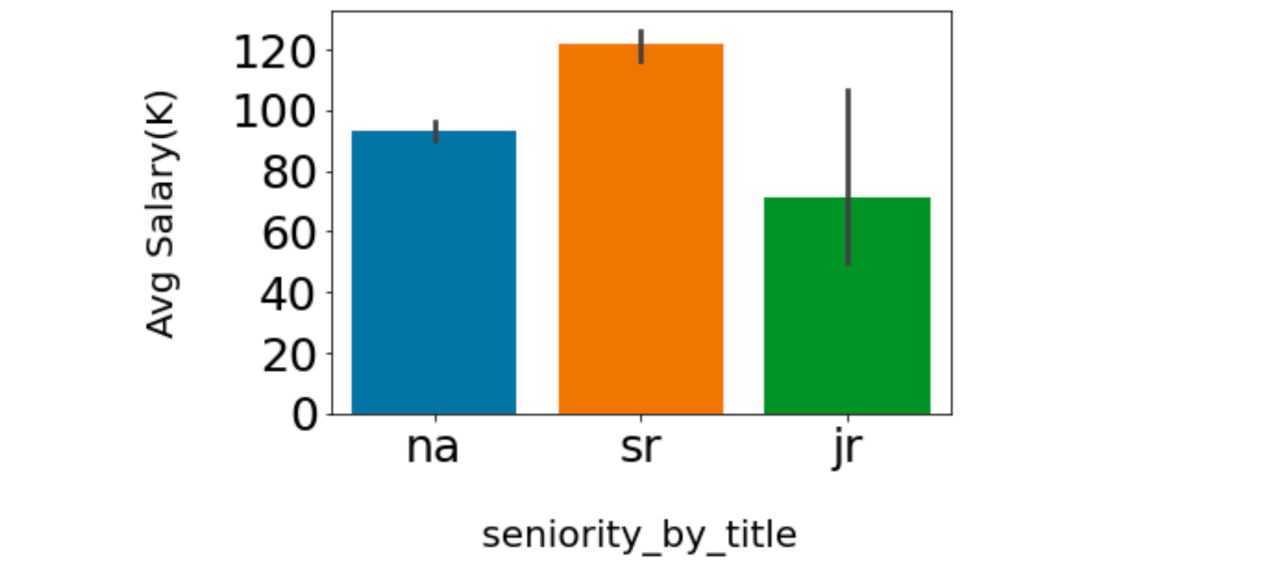
Degree versus Salary

```
# We check relation between degree and avg salary
sns.barplot(y="Avg Salary(K)",x="Degree",data=salary)
#Beautifying the plot

plt.title('\n Degree and Salary Relationship Plot \n', size=30, color='black')
plt.xticks(fontsize=25)
plt.yticks(fontsize=25)
plt.xlabel('\n Degree \n', fontsize=20, color='black')
plt.ylabel('\n Avg Salary(K) \n', fontsize=20, color='black')
plt.show()
```
✓ 0.1s Python



Degree 欄位與薪水有所關聯，不能刪

Seniority versus Salary



Seniority 欄位與薪水有所關聯，不能刪

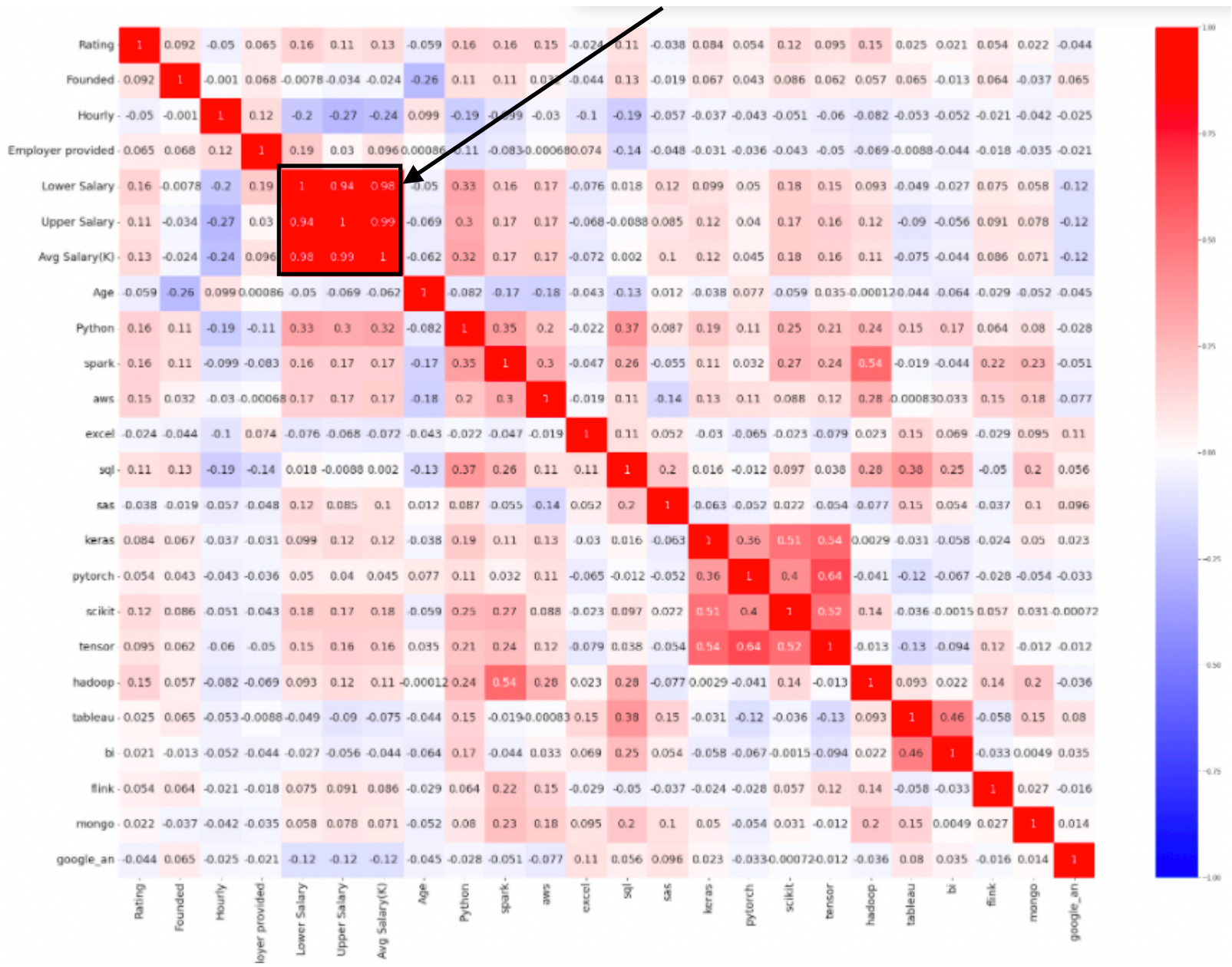
Correlation 觀察

```
correlation=salary.corr()
```

Python

```
# set the plot size
plt.figure(figsize = (35,25))
# plot the heat map
# correlation: give the correlation matrix
# cmap: color code used for plotting
# annot_kws: sets the font size of the annotation
# annot: prints the correlation values in the chart
# vmax: gives a maximum range of values for the chart
# vmin: gives a minimum range of values for the chart
sns.heatmap(correlation,annot = True, vmax = 1.0, vmin = -1.0, cmap = 'bwr', annot_kws = {"size": 18})
# set the size of x and y axes labels using 'fontsize'
plt.xticks(fontsize = 18)
plt.yticks(fontsize = 18)
plt.show()
```

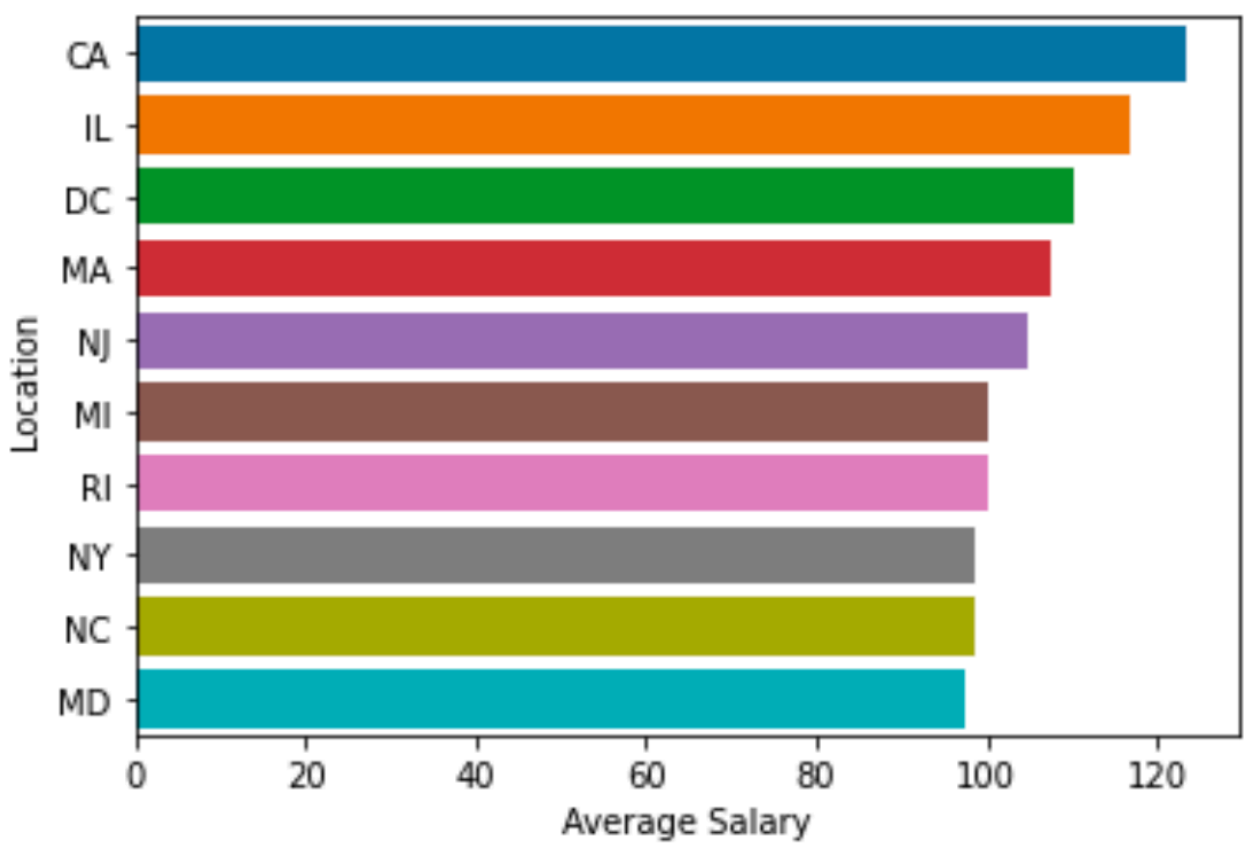
可以觀察出upper, lower, average salary有很大的關聯，為了減少運算量，因此建模型時會使用其中一項，在此是使用Average Salary



接下來code與前雷同，為了整齊所以省略

Job location versus Salary

觀察發現工作的地方是與薪水有所關聯，不能刪



Education versus Salary

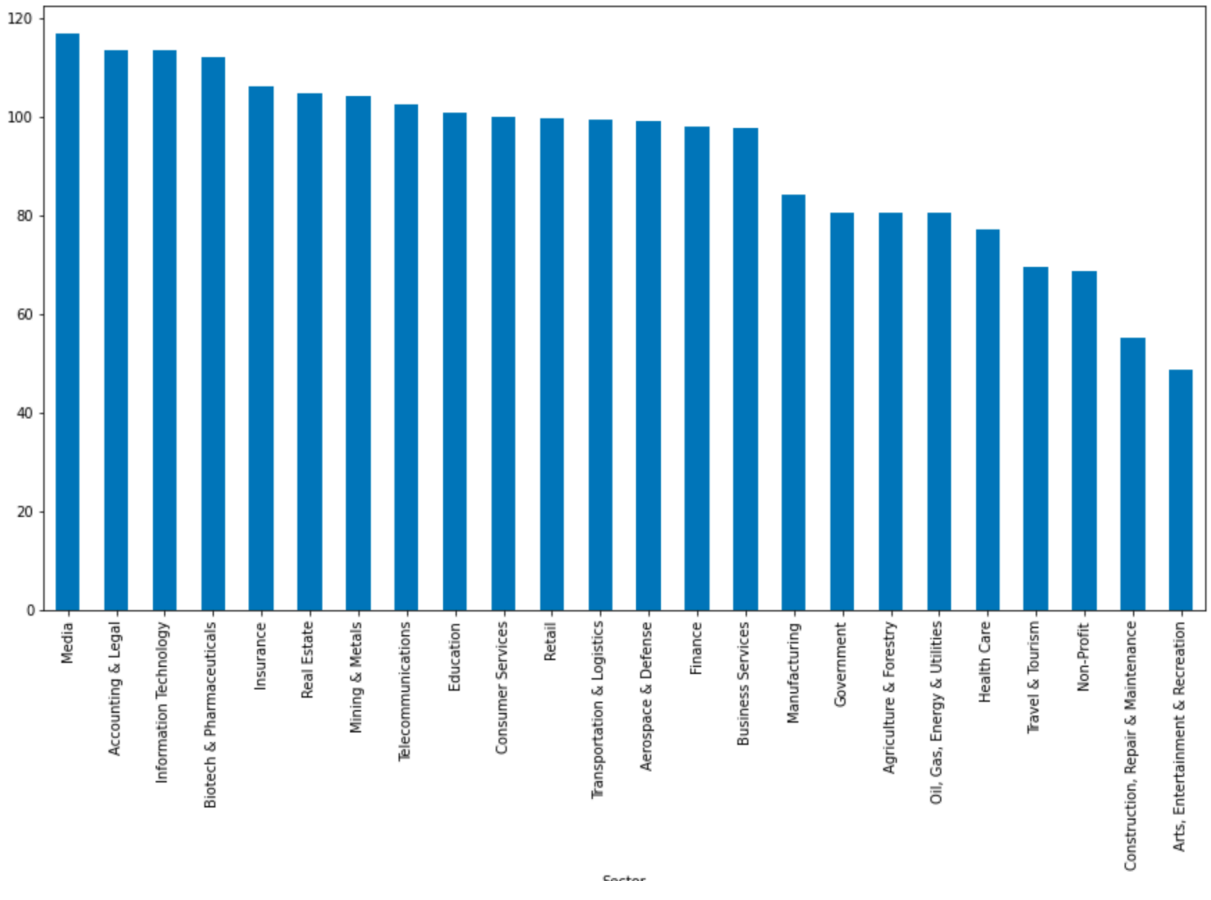
觀察發現PHD普遍擁有更高的薪水

```
pd.pivot_table(salary, index = 'job_title_sim', columns = "Degree", values = 'Avg Salary(K)', aggfunc = ['mean','count']).rename(columns={"mean":"Avg Salary(K)"})
```
✓ 0.4s

	Avg Salary(K)			count		
Degree	**M**	**P**	**na**	**M**	**P**	**na**
job_title_sim						
Data scientist project manager	83.800000	NaN	68.409091	5.0	NaN	11.0
analyst	61.333333	NaN	67.858108	27.0	NaN	74.0
data analitics	82.500000	NaN	61.300000	3.0	NaN	5.0
data engineer	103.714286	132.250000	105.469512	35.0	2.0	82.0
data modeler	63.166667	NaN	99.000000	3.0	NaN	2.0
data scientist	117.847826	129.578947	115.613139	138.0	38.0	137.0
director	NaN	150.666667	101.000000	NaN	3.0	2.0
machine learning engineer	125.000000	134.200000	118.950000	2.0	10.0	10.0
na	92.000000	NaN	91.083333	4.0	NaN	6.0
other scientist	102.771429	100.509259	66.064815	35.0	54.0	54.0

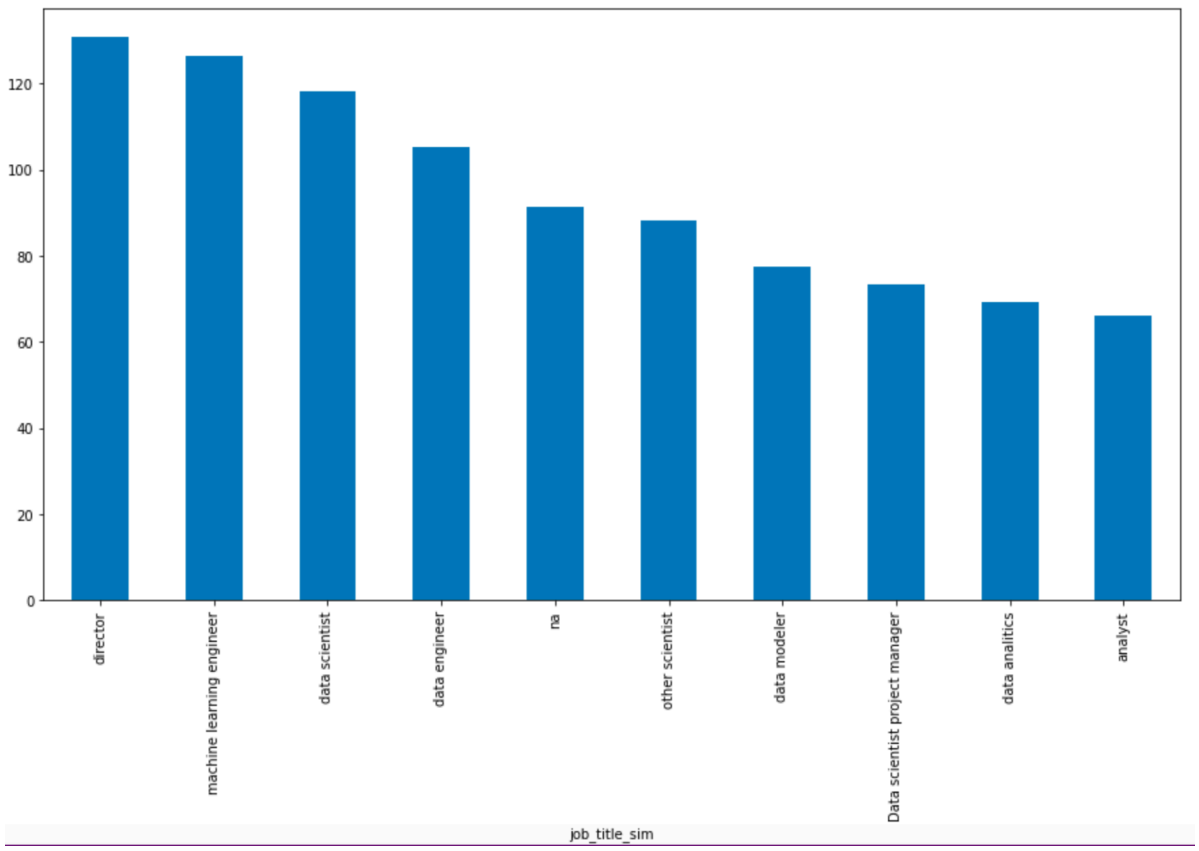
Sector versus Salary

觀察對於資料科學家薪水，行業之間有所差距，不能刪此欄位



Job title versus Salary

觀察資料科學家薪水職位薪水一表有所傾向，不能刪此欄位



欄位重塑

去除不相關的欄位，留下與薪水有關係的欄位

這裡 inplace = True 代表會對此 DataFrame做修改

```
salary.drop("Job Description",axis=1,inplace=True)
```
✓ 0.3s Python

```
salary.drop("Headquarters",axis=1,inplace=True)# I dropped Headquarter column as I think it not reauired for salary prediction
```
✓ 0.4s Python

```
salary.drop("Competitors",axis=1,inplace=True)# I dropped Competitors column as I think it not reauired for salary prediction
```
✓ 0.3s Python

```
salary.drop("Job Title",axis=1,inplace=True)# I dropped Job Title column as we have another column in which they have segregate
#  Job title in main columns
```
✓ 0.3s Python

```
salary.drop("Salary Estimate",axis=1,inplace=True)# I dropped Salary Estimate column as we have another column "Avg Salary(k)"
```
✓ 0.3s Python

```
salary.drop("Industry",axis=1,inplace=True) # As Industry gets divide into sector so I am dropping Industry column
```
✓ 0.3s Python

```
salary.drop("Hourly",axis=1,inplace=True)
salary.drop("Employer provided",axis=1,inplace=True)# as both column not required for salary prediction so we will drop them
```
✓ 0.3s Python

```
salary.drop("Lower Salary",axis=1,inplace=True)
salary.drop("Upper Salary",axis=1,inplace=True)#As Lower salary and upper salary are correlated and it doesn't take into consideration while predict wo we will drop
```
```
salary.drop("Company Name",axis=1,inplace=True)
```
✓ 0.4s Python

建模型 (事前準備)

將有用的資訊彙整成新的DataFrame

將 DataFrame 分成「 數值 」以及「 類別 」

```
df_numeric = salary.select_dtypes(include=np.number)
```
✓ 0.4s Python

```
df_numeric.head()
```
✓ 0.4s Python

	Rating	Founded	Avg Salary(K)	Age	Python	spark	aws	excel	sql	sas	keras	pytorch	scikit	tensor	hadoop	tableau	bi	flink	mongo	google_an
0	3.8	1973.0	72.0	48.0	1	0	0	1	0	1	0	0	0	0	0	1	1	0	0	0
1	3.4	1984.0	87.5	37.0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	4.8	2010.0	85.0	42.0	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0
3	3.8	1965.0	76.5	56.0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	2.9	1998.0	114.5	23.0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0

```
df_categoric = salary.select_dtypes(include = object)
```
✓ 0.5s Python

```
df_categoric.head()
```
✓ 0.4s Python

	Size	Type of ownership	Sector	Revenue	Job Location	job_title_sim	seniority_by_title	Degree
0	501 - 1000	Company - Private	Aerospace & Defense	$50 to $100 million (USD)	NM	data scientist	na	M
1	10000+	Other Organization	Health Care	$2 to $5 billion (USD)	MD	data scientist	na	M
2	501 - 1000	Company - Private	Business Services	$100 to $500 million (USD)	FL	data scientist	na	M
3	1001 - 5000	Government	Oil, Gas, Energy & Utilities	$500 million to $1 billion (USD)	WA	data scientist	na	na
4	51 - 200	Company - Private	Business Services	Unknown / Non-Applicable	NY	data scientist	na	na

為了做回歸 (Regression)，要先將資料類別做 Dummy Encode

Dummy encode the catergorical variables

```
dummy_encoded_variables = pd.get_dummies(df_categoric, drop_first = True)
```
✓ 0.5s Python

```
dummy_encoded_variables.head()
```
✓ 0.5s Python

	Size_10000+	Size_1001 - 5000	Size_201 - 500	Size_5001 - 10000	Size_501 - 1000	Size_51 - 200	Size_unknown	Type of ownership_Company - Private	Type of ownership_Company - Public	Type of ownership_Government	Type of ownership_Hospital
0	0	0	0	0	1	0	0	1	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	1	0	0	1	0	0	0
3	0	1	0	0	0	0	0	0	0	1	0
4	0	0	0	0	0	1	0	1	0	0	0

```
# concatenate the numerical and dummy encoded categorical variables column-wise
df_dummy = pd.concat([df_numeric, dummy_encoded_variables], axis=1)

# display data with dummy variables
df_dummy.head()
```
✓ 0.6s Python

	Rating	Founded	Avg Salary(K)	Age	Python	spark	aws	excel	sql	sas	keras	pytorch	scikit	tensor	hadoop	tableau	bi	flink	mongo	google_an	Size_10000
0	3.8	1973.0	72.0	48.0	1	0	0	1	0	1	0	0	0	0	0	1	1	0	0	0	
1	3.4	1984.0	87.5	37.0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	4.8	2010.0	85.0	42.0	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	
3	3.8	1965.0	76.5	56.0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	2.9	1998.0	114.5	23.0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	

```python
# add the intercept column using 'add_constant()'
df_dummy = sm.add_constant(df_dummy)

# separate the independent and dependent variables
# drop(): drops the specified columns
X = df_dummy.drop(["Avg Salary(K)"], axis = 1)

# extract the target variable from the data set
y = df_dummy[["Avg Salary(K)"]]
```

✓ 0.1s Python

建模型（OLS）Ordinary Least Squares

呼叫模型架構 sm.OLS ( )

```python
# split data into train data and test data
# what proportion of data should be included in test data is passed using 'test_size'
# set 'random_state' to get the same data each time the code is executed
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 1)

# check the dimensions of the train & test subset for
# print dimension of predictors train set
print("The shape of X_train is:",X_train.shape)

# print dimension of predictors test set
print("The shape of X_test is:",X_test.shape)

# print dimension of target train set
print("The shape of y_train is:",y_train.shape)

# print dimension of target test set
print("The shape of y_test is:",y_test.shape)
```

✓ 0.4s Python

```python
# build a full model using OLS()
# consider the log of sales price as the target variable
# use fit() to fit the model on train data
linreg_full = sm.OLS(y_train, X_train).fit()

# print the summary output
print(linreg_full.summary())
```

✓ 0.1s Python

Output exceeds the size limit. Open the full output data in a text editor

OLS Regression Results

Dep. Variable:	Avg Salary(K)	R-squared:	0.666
Model:	OLS	Adj. R-squared:	0.584
Method:	Least Squares	F-statistic:	8.096
Date:	Tue, 08 Mar 2022	Prob (F-statistic):	5.68e-62
Time:	12:04:25	Log-Likelihood:	-2665.6
No. Observations:	593	AIC:	5567.
Df Residuals:	475	BIC:	6085.
Df Model:	117		
Covariance Type:	nonrobust		

	Model	RMSE	R-Squared	Adj. R-Squared
0	Linreg full model	28.255571	0.666017	0.583752

Testing & Evaluation (RMSE method)

```
actual_salary = y_test["Avg Salary(K)"]
actual_salary
```
✓ 0.9s Python

```
# calculate rmse using rmse()
linreg_full_rmse = rmse(actual_salary,linreg_full_predictions )

# calculate R-squared using rsquared
linreg_full_rsquared = linreg_full.rsquared

# calculate Adjusted R-Squared using rsquared_adj
linreg_full_rsquared_adj = linreg_full.rsquared_adj
```
✓ 0.7s Python

```
# create the result table for all accuracy scores
# accuracy measures considered for model comparision are RMSE, R-squared value and Adjusted R-squared value
# create a list of column names
cols = ['Model', 'RMSE', 'R-Squared', 'Adj. R-Squared']

# create a empty dataframe of the colums
# columns: specifies the columns to be selected
result_tabulation = pd.DataFrame(columns = cols)

# compile the required information
linreg_full_metrics = pd.Series({'Model': "Linreg full model ",
                 'RMSE':linreg_full_rmse,
                 'R-Squared': linreg_full_rsquared,
                 'Adj. R-Squared': linreg_full_rsquared_adj
                })

# append our result table using append()
# ignore_index=True: does not use the index labels
# python can only append a Series if ignore_index=True or if the Series has a name
result_tabulation = result_tabulation.append(linreg_full_metrics, ignore_index = True)

# print the result table
result_tabulation
```

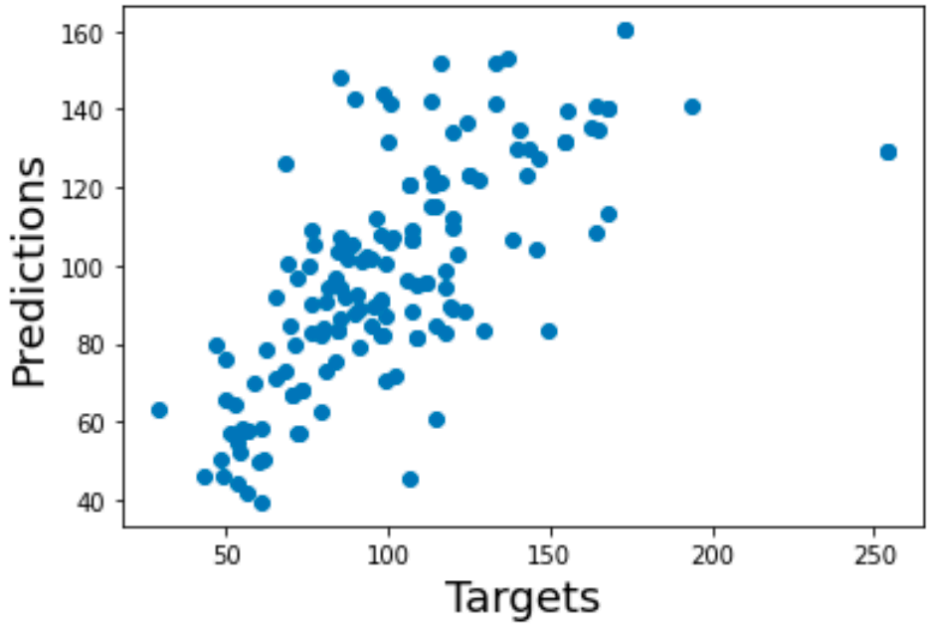
(SGD) Stochastic Gradient Descent

呼叫架構 SGDRegressor ( )

```
# build the model
SGD_model = SGDRegressor(loss="squared_loss",alpha = 0.1, max_iter = 3000) #Instantiation

# fit the model
SGD_model.fit(X1_train, y1_train) #Training
```

```
# predict the values
y1_pred_SGD  = SGD_model.predict(X1_test)
```
✓ 0.1s Python

```
plt.scatter(y1_test,y1_pred_SGD)
plt.xlabel('Targets' ,size = 18)
plt.ylabel('Predictions',size = 18)
plt.show()
```
✓ 0.1s Python



```python
# compile the required information
linreg_full_metrics = pd.Series({'Model': "Linear regression with SGD ",
                 'RMSE':rmse_SGD,
                 'R-Squared': r_squared_SGD,
                 'Adj. R-Squared': Adj_r_squared_SGD
                })

# append our result table using append()
# ignore_index=True: does not use the index labels
# python can only append a Series if ignore_index=True or if the Series has a name
result_tabulation = result_tabulation.append(linreg_full_metrics, ignore_index = True)

# print the result table
result_tabulation
```

✓ 0.9s Python

	Model	RMSE	R-Squared	Adj. R-Squared
0	Linreg full model	28.255571	0.666017	0.583752
1	Linear regression with SGD	26.768678	0.650966	0.563154

結果探討

可以發現 OLS 以及 SGD Regression 兩種預測結果的R-Squared差距不大，代表還有些許的資料沒辦法被機器所解釋，究其原因可以追溯至作者在做資料前處理時的大量補值，不過能初步理解是如何分析如此多欄的資料模型，我認為這樣的預測結果是可以接受的。

Code 方法與比較 2:

載入、觀察資料集 Load & Observation

一樣使用pandas套件讀入資料

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import re
import seaborn as sns

ds_data = pd.read_csv('data_cleaned_2021.csv')
```

```python
ds_data.drop(columns = ['index'], inplace = True)
```

✓ 0.2s Python

相比前一份code，這份code要做出的結果並不是預測薪水的值，而是將整體dataframe作系統式整理，提供視覺化的種種關係圖；因此資料前處理著重於去除無代表性的欄位，而不是補值，進而不影響原資料的特性。

處理重複值

duplicated ( )可以找出欄位重複值，並顯示數目

配合 drop_duplicates ( ) 可以 移除重複值

```
if ds_data.duplicated().sum() == 0:
    print("There are no duplicated rows in this dataset")
else:
    print("There are",ds_data.duplicated().sum(),"in this dataset")
```

```
ds_data.drop_duplicates(inplace = True)
print("The dataset contains",ds_data.shape[0],"row, and",ds_data.shape[1],"columns")
```

✓ 0.3s Python

The dataset contains 467 row, and 41 columns

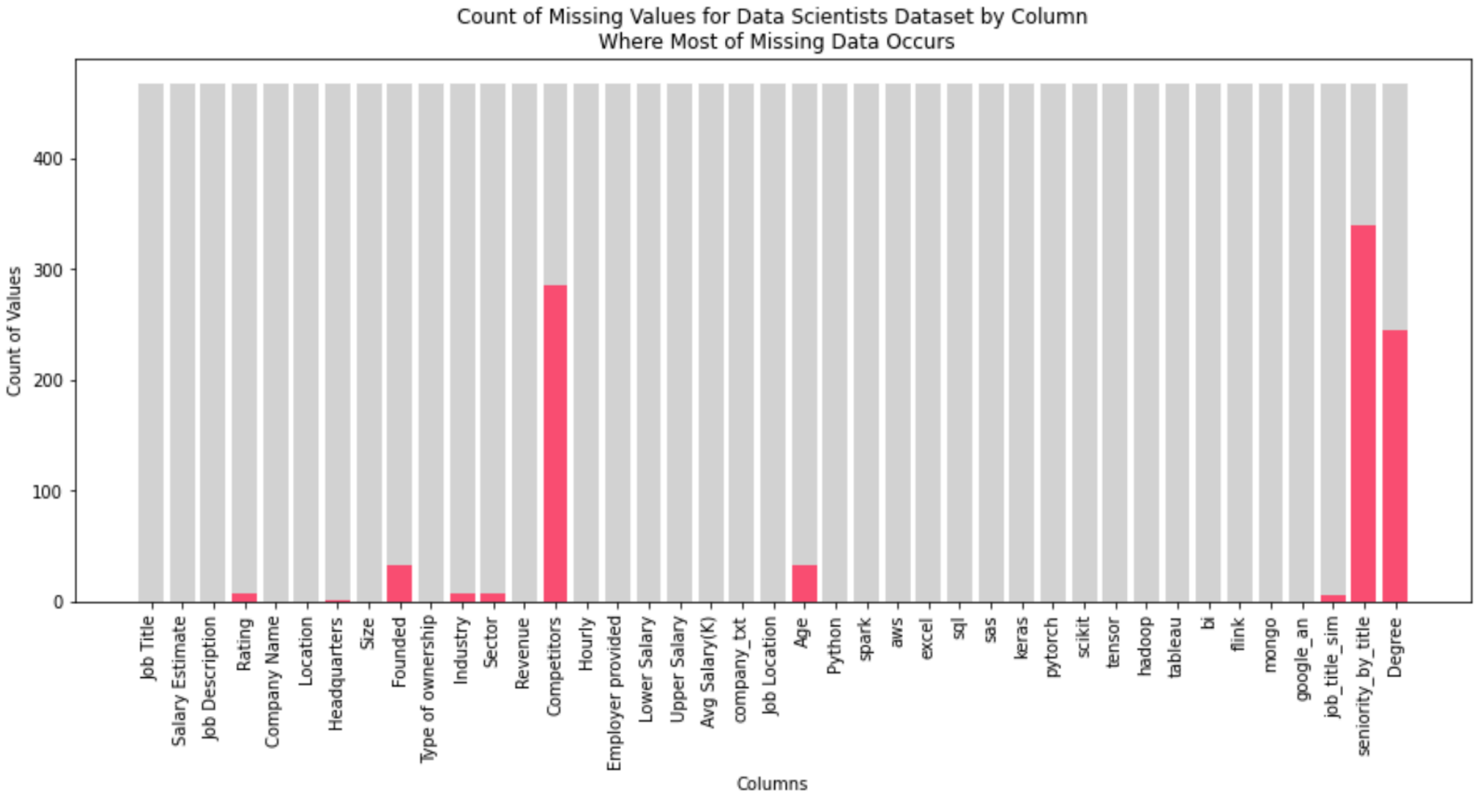
處理 “-1” “NaN” “Null”

replace ( )可以用特定值取代特定值

```
ds_data.replace("-1", np.nan, inplace=True)
ds_data.replace(-1, np.nan, inplace=True)
ds_data.replace('na', np.nan, inplace=True)
```

```
null_values = ds_data.isnull().sum()
not_null = ds_data.notnull().sum()

plt.figure(figsize=(15,6))
ax = plt.subplot()
plt.bar(range(len(ds_data.columns)),null_values,color='crimson', alpha=.7)
plt.bar(range(len(ds_data.columns)),not_null,col  color: Any  ',bottom=null_values)
ax.set_xticks(range(len(ds_data.columns)))
ax.set_xticklabels(ds_data.columns,rotation=90)
plt.xlabel('Columns')
plt.ylabel('Count of Values')
plt.title('Count of Missing Values for Data Scientists Dataset by Column\n Where Most of Missing Data Occurs')
plt.show()
```

✓ 0.2s Python



由此可以看到各欄位遺失值的情況

刪除過多遺失資料的欄位

```
ds_data = ds_data.drop(columns=['seniority_by_title','job_title_sim','Degree'])
```

✓ 0.3s Python

轉換”Founded”單位（節省計算時間）

將浮點數轉成至今的年數

Errors = ‘ignore’ 代表會略過NaN的情況

```
ds_data['Founded'] = pd.to_datetime(ds_data['Founded'], format='%Y', errors='ignore')
```

✓ 0.3s Python

欄位重塑

Job Title 觀察

1. Check Unique

```
print("There are",len(ds_data['Job Title'].unique()),"unique entries under Job Title column")
```

✓ 0.2s Python

There are 264 unique entries under Job Title column

2. 分欄位，將“Job title” 變成 “Rank/Seniority” 和 “Profession”

3. 轉成小寫

4. 作者在此擷取一些他研究裡常出現的職業

5. 使用正規表示

6. 補值（Imputer）

```
import re

ds_data['Job Title'] = ds_data['Job Title'].str.lower()

ds_data['Profession'] = ds_data['Job Title'].str.extract('(machine learning engineer|data engineer|software engineer|modeler|administrator|research scienti
ds_data['Profession'] = ds_data['Profession'].fillna('Other')
ds_data['Profession'] = ds_data['Profession'].apply(lambda x: re.sub(',[a-z ]+','',x))
ds_data['Profession'] = ds_data['Profession'].str.title()
```

✓ 0.3s Python

Drop Salary Estimate

已經有 lower Salary 和 Upper Salary, 所以去除 “Salary Estimate”

```
ds_data.drop(columns=['Salary Estimate'], inplace = True)
```

✓ 0.2s Python

Drop Company Name

已經有 “ company_txt ”，所以去除有參雜rating的 “Company Name”

```
ds_data.drop(columns=['Company Name'], inplace = True)
ds_data.rename(columns={"company_txt": "Company Name"}, inplace = True)
```

✓ 0.2s Python

使用正則表示法來處理字串，並將它們分成適合的欄位

```
ds_data['Location'] = ds_data['Location'].apply(lambda x: re.sub(', [A-Z]+', "", x))
ds_data.rename(columns = {'Location': 'Location (City)', 'Job Location': 'Location (State)'}, inplace = True)
```

✓ 0.2s Python

```
ds_data['Headquarters'] = ds_data['Headquarters'].fillna('Not Available')
ds_data['Headquarters (State)'] = ds_data['Headquarters'].str.extract(', ([a-zA-Z]+)')
ds_data['Headquarters'] = ds_data['Headquarters'].apply(lambda x: re.sub(', [A-Z]{2}', "", x))
ds_data.rename(columns = {'Headquarters': 'Headquarters (City)'}, inplace = True)
ds_data['Headquarters (State)'] = ds_data['Headquarters (State)'].fillna('Not Available')
ds_data['Headquarters (State)'] = ds_data['Headquarters (State)'].apply(lambda x: re.sub('[A-Z]{2}', 'United States', x))
```

✓ 0.3s Python

經由觀察 “Size” 和 “Revenue” 欄位形式, 作者將他們轉成 categorical 形式

```
ds_data['Size'] = pd.Categorical(ds_data['Size'], ['1 - 50 ', '51 - 200 ', '201 - 500 ', '501 - 1000 ','1001 - 5000 ', '5001 - 10000 ', '10000+ '], orderec
ds_data['Revenue'] = pd.Categorical(ds_data['Revenue'], ['Less than $1 million (USD)', '$1 to $5 million (USD)', '$5 to $10 million (USD)', '$10 to $25 mil
```

✓ 0.2s Python

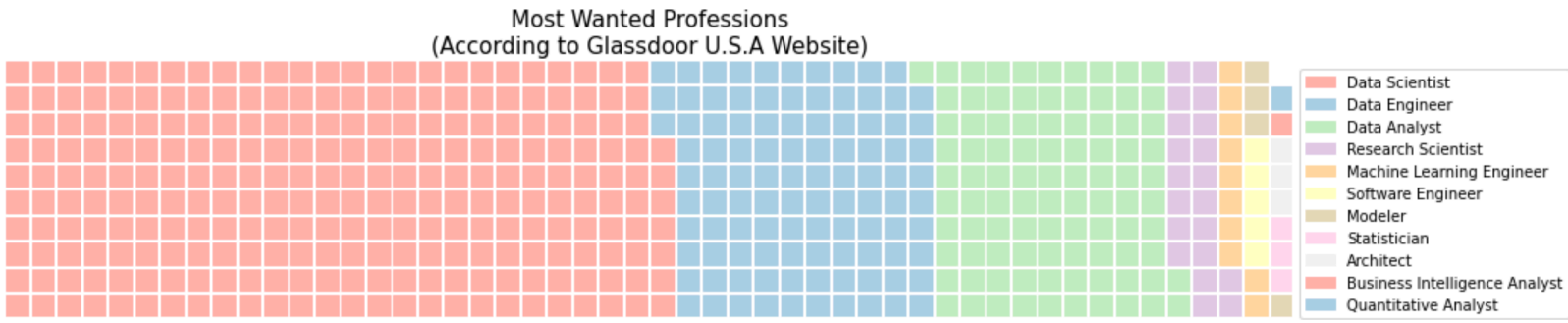
視覺化表示觀察結果 & 統整

需求職業比較圖

```
from pywaffle import Waffle

profession_vis = ds_data[ds_data['Profession'] != 'Other']['Profession'].value_counts().to_dict()

plt.figure(
    FigureClass=Waffle,
    rows=10,
    columns=50,
    values=profession_vis,
    legend={'loc': 'upper left', 'bbox_to_anchor': (1, 1)},
    figsize=(15,8),
    cmap_name="Pastel1"
)

plt.title("Most Wanted Professions\n(According to Glassdoor U.S.A Website)",fontsize=15)
plt.show()
```

✓ 0.3s Python

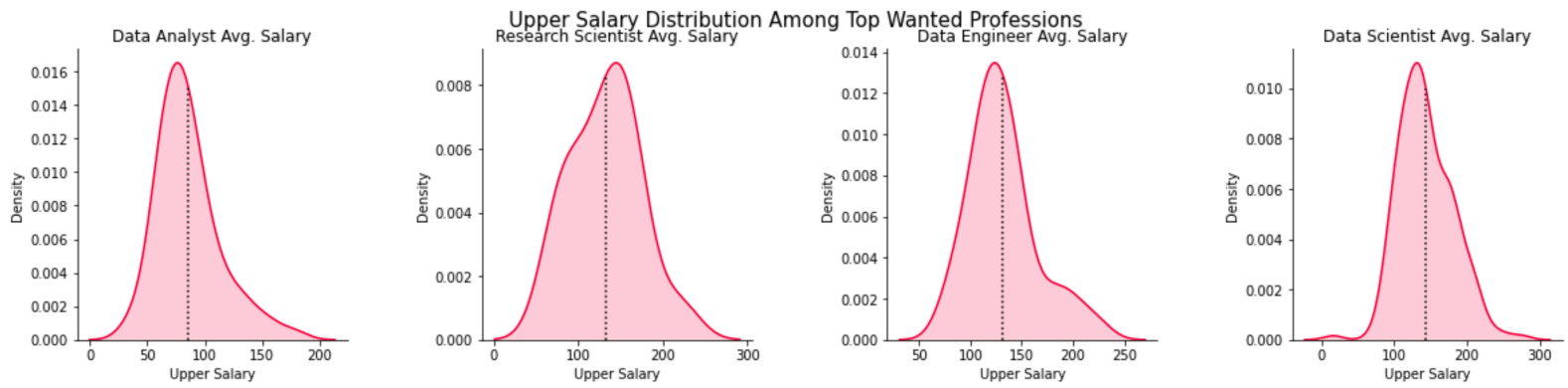
從此圖就可以看出最業界最想要哪種職業人才



再來抓出前四名的職業薪水分佈圖

```
data_science = ds_data[ds_data['Profession'] == 'Data Scientist']['Upper Salary']
data_engineer = ds_data[ds_data['Profession'] == 'Data Engineer']['Upper Salary']
data_analyst = ds_data[ds_data['Profession'] == 'Data Analyst']['Upper Salary']
research_scientist = ds_data[ds_data['Profession'] == 'Research Scientist']['Upper Salary']

plt.figure(figsize=(20,4))

ax1 = plt.subplot(1,4,1)
ax1 = sns.kdeplot(data_analyst, color='crimson', shade=False)
kdeline = ax1.lines[0]
mean = data_analyst.mean()
xs = kdeline.get_xdata()
ys = kdeline.get_ydata()
height = np.interp(mean, xs, ys)
ax1.vlines(mean, 0, height, color='black', ls=':')
ax1.fill_between(xs, 0, ys, facecolor='crimson', alpha=0.2)
plt.title('Data Analyst Avg. Salary')
for s in ['top', 'right']:
    ax1.spines[s].set_visible(False)

ax2 = plt.subplot(1,4,2)
ax2 = sns.kdeplot(research_scientist, color='crimson', shade=False)
kdeline = ax2.lines[0]
mean = research_scientist.mean()
xs = kdeline.get_xdata()
ys = kdeline.get_ydata()
height = np.interp(mean, xs, ys)
ax2.vlines(mean, 0, height, color='black', ls=':')
ax2.fill_between(xs, 0, ys, facecolor='crimson', alpha=0.2)
plt.title('Research Scientist Avg. Salary')

for s in ['top', 'right']:
    ax2.spines[s].set_visible(False)

ax3 = plt.subplot(1,4,3)
ax3 = sns.kdeplot(data_engineer, color='crimson', shade=False)
kdeline = ax3.lines[0]
mean = data_engineer.mean()
xs = kdeline.get_xdata()
ys = kdeline.get_ydata()
height = np.interp(mean, xs, ys)
ax3.vlines(mean, 0, height, color='black', ls=':')
ax3.fill_between(xs, 0, ys, facecolor='crimson', alpha=0.2)
plt.title('Data Engineer Avg. Salary')

for s in ['top', 'right']:
    ax3.spines[s].set_visible(False)

ax4 = plt.subplot(1,4,4)
ax4 = sns.kdeplot(data_science, color='crimson', shade=False)
kdeline = ax4.lines[0]
mean = data_science.mean()
xs = kdeline.get_xdata()
ys = kdeline.get_ydata()
height = np.interp(mean, xs, ys)
ax4.vlines(mean, 0, height, color='black', ls=':')
ax4.fill_between(xs, 0, ys, facecolor='crimson', alpha=0.2)
plt.title('Data Scientist Avg. Salary')

for s in ['top', 'right']:
    ax4.spines[s].set_visible(False)

plt.subplots_adjust(wspace=0.5)
plt.suptitle('Upper Salary Distribution Among Top Wanted Professions',fontsize=15)
plt.show()
```

結果呈現 （職業與薪水之間的關係）



前四大需求職業之行業比較圖

```
data_science_i = ds_data[ds_data['Profession'] == 'Data Scientist']['Industry'].value_counts().head(5)
data_engineer_i = ds_data[ds_data['Profession'] == 'Data Engineer']['Industry'].value_counts().head(5)
data_analyst_i = ds_data[ds_data['Profession'] == 'Data Analyst']['Industry'].value_counts().head(5)
research_scientist_i = ds_data[ds_data['Profession'] == 'Research Scientist']['Industry'].value_counts().head(5)

plt.figure(figsize=(25,5))

ax1 = plt.subplot(1,4,1)
colors = ['crimson','crimson','lightpink','lightpink','lightpink']
plt.bar(range(len(data_science_i)), data_science_i, color=colors)
ax1.set_xticks(range(len(data_science_i)))
ax1.set_yticks([5,10,15,20,25])
ax1.set_xticklabels(data_science_i.index, rotation=80)
plt.title("Data Scientists")
plt.ylabel("Number of Open Positions")

for s in ['top', 'right']:
    ax1.spines[s].set_visible(False)

ax2 = plt.subplot(1,4,2)
colors = ['crimson','crimson','lightpink','lightpink','lightpink']
plt.bar(range(len(data_engineer_i)), data_engineer_i, color=colors)
ax2.set_xticks(range(len(data_engineer_i)))
ax2.set_yticks([5,10,15,20,25])
ax2.set_xticklabels(data_engineer_i.index, rotation=80)
plt.title("Data Engineers")
plt.ylabel("Number of Open Positions")
```

```
for s in ['top', 'right']:
    ax2.spines[s].set_visible(False)

ax3 = plt.subplot(1,4,3)
colors = ['crimson','crimson','lightpink','lightpink','lightpink']
plt.bar(range(len(data_analyst_i)), data_analyst_i, color=colors)
ax3.set_xticks(range(len(data_analyst_i)))
ax3.set_yticks([5,10,15,20,25])
ax3.set_xticklabels(data_analyst_i.index, rotation=80)
plt.title("Data Analysts")
plt.ylabel("Number of Open Positions")

for s in ['top', 'right']:
    ax3.spines[s].set_visible(False)

ax4 = plt.subplot(1,4,4)
colors = ['crimson','crimson','lightpink','lightpink','lightpink']
plt.bar(range(len(research_scientist_i)), research_scientist_i, color=colors)
ax4.set_xticks(range(len(research_scientist_i)))
ax4.set_yticks([5,10,15,20,25])
ax4.set_xticklabels(research_scientist_i.index, rotation=80)
plt.title("Research Scientists")
plt.ylabel("Number of Open Positions")

for s in ['top', 'right']:
    ax4.spines[s].set_visible(False)

plt.subplots_adjust(wspace=0.5)
plt.suptitle("Top Recruiting Industries per Profession", fontsize=15)
plt.show()
```

✓ 0.2s Python

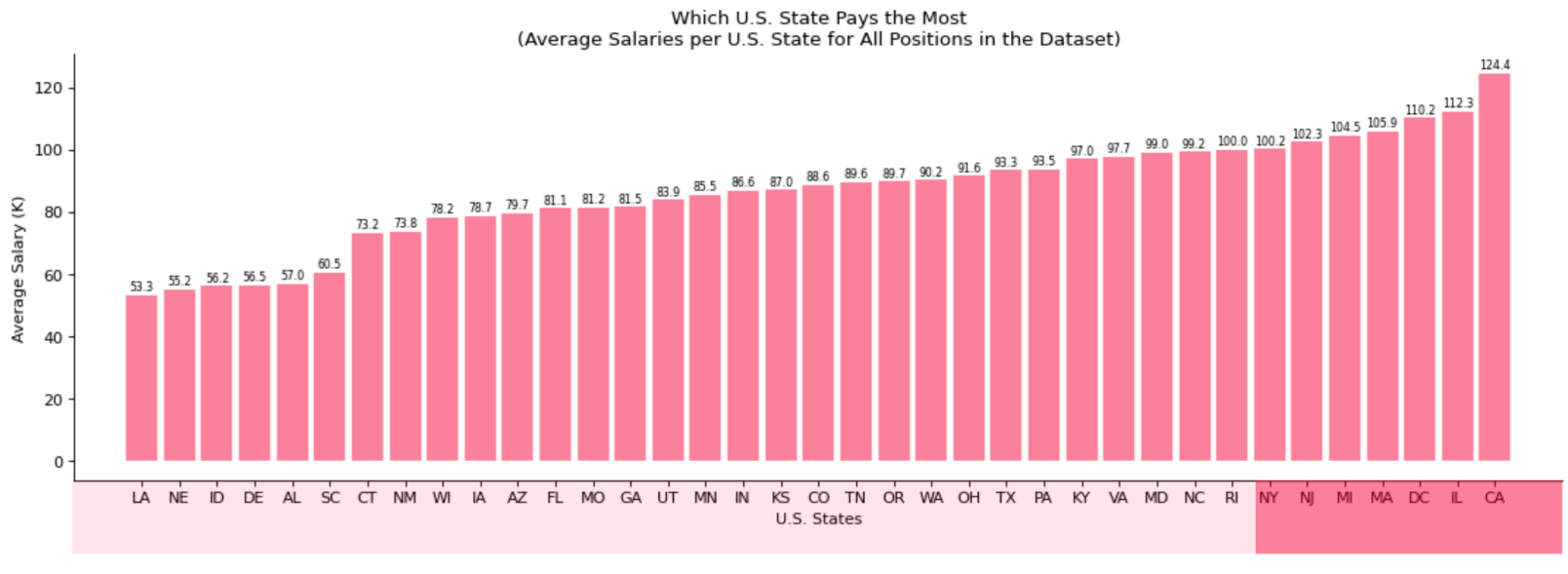
結果呈現，此分佈圖提供人明確的求職選擇路

地緣與薪水的比較圖

```python
df = ds_data[['Location (State)', 'Avg Salary(K)']].groupby('Location (State)').apply(lambda x: x.mean())
df.sort_values('Avg Salary(K)', inplace=True)
df.reset_index(inplace=True)

import matplotlib.patches as patches

fig, ax = plt.subplots(figsize=(17,5), facecolor='white', dpi= 80)
ax.vlines(x=df.index, ymin=0, ymax=df['Avg Salary(K)'], color='crimson', alpha=0.5, linewidth=20)

for i, salary in enumerate(df['Avg Salary(K)']):
    ax.text(i, salary+1.5, round(salary, 1), horizontalalignment='center', fontsize=7)

ax.set_xticks(range(len(df)))
ax.set_xticklabels(list(df['Location (State)']))
plt.title("Which U.S. State Pays the Most\n(Average Salaries per U.S. State for All Positions in the Dataset)")
plt.xlabel('U.S. States')
plt.ylabel('Average Salary (K)')

for s in ['top', 'right']:
    ax.spines[s].set_visible(False)

p1 = patches.Rectangle((.74, -0.005), width=.16, height=.13, alpha=.5, facecolor='crimson', transform=fig.transFigure)
p2 = patches.Rectangle((.124, -0.005), width=.617, height=.13, alpha=.1, facecolor='crimson', transform=fig.transFigure)
fig.add_artist(p1)
fig.add_artist(p2)
plt.show()
```

✓ 0.3s Python

結果呈現，提供人們對於工作地區以及薪水上量化的取捨

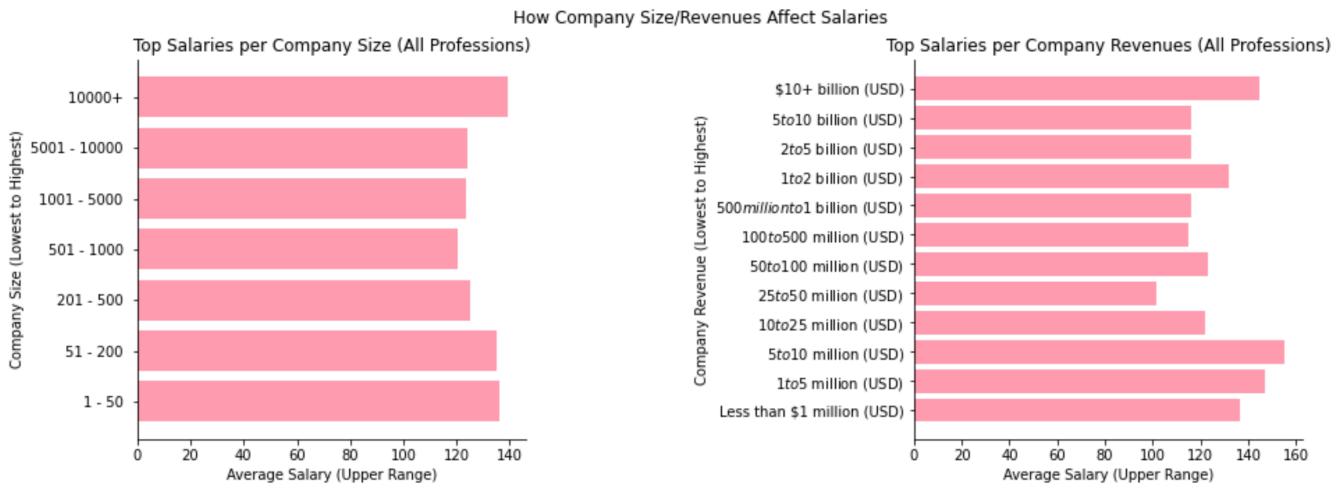


公司規模、收入與薪水量化圖

```
size_salary = ds_data.groupby('Size')['Upper Salary'].mean()
revenue_salary = ds_data.groupby('Revenue')['Upper Salary'].mean()

plt.figure(figsize=(15,5))

ax1 = plt.subplot(1,2,1)
plt.barh(range(len(size_salary)),size_salary, color='crimson', alpha=0.4)
plt.xlabel('Average Salary (Upper Range)')
plt.ylabel('Company Size (Lowest to Highest)')
ax1.set_yticks(range(len(size_salary)))
ax1.set_yticklabels(size_salary.index)
plt.title('Top Salaries per Company Size (All Professions)')

for s in ['top', 'right']:
    ax1.spines[s].set_visible(False)

ax2 = plt.subplot(1,2,2)
plt.barh(range(len(revenue_salary)),revenue_salary, color='crimson', alpha=0.4)
plt.xlabel('Average Salary (Upper Range)')
plt.ylabel('Company Revenue (Lowest to Highest)')
ax2.set_yticks(range(len(revenue_salary)))
ax2.set_yticklabels(revenue_salary.index)
plt.title("Top Salaries per Company Revenues (All Professions)")

for s in ['top', 'right']:
    ax2.spines[s].set_visible(False)

plt.subplots_adjust(wspace=1)
plt.suptitle('How Company Size/Revenues Affect Salaries')
plt.show()
```

結果呈現，公司大小以及總收入與配發薪水天花板並沒有絕對的正關係

超大規模公司、小規模公司擁有好的薪水發放，然而中型規模的公司卻因為某種因素使薪水天花板顯得較矮



地區與工作機會關係量化圖

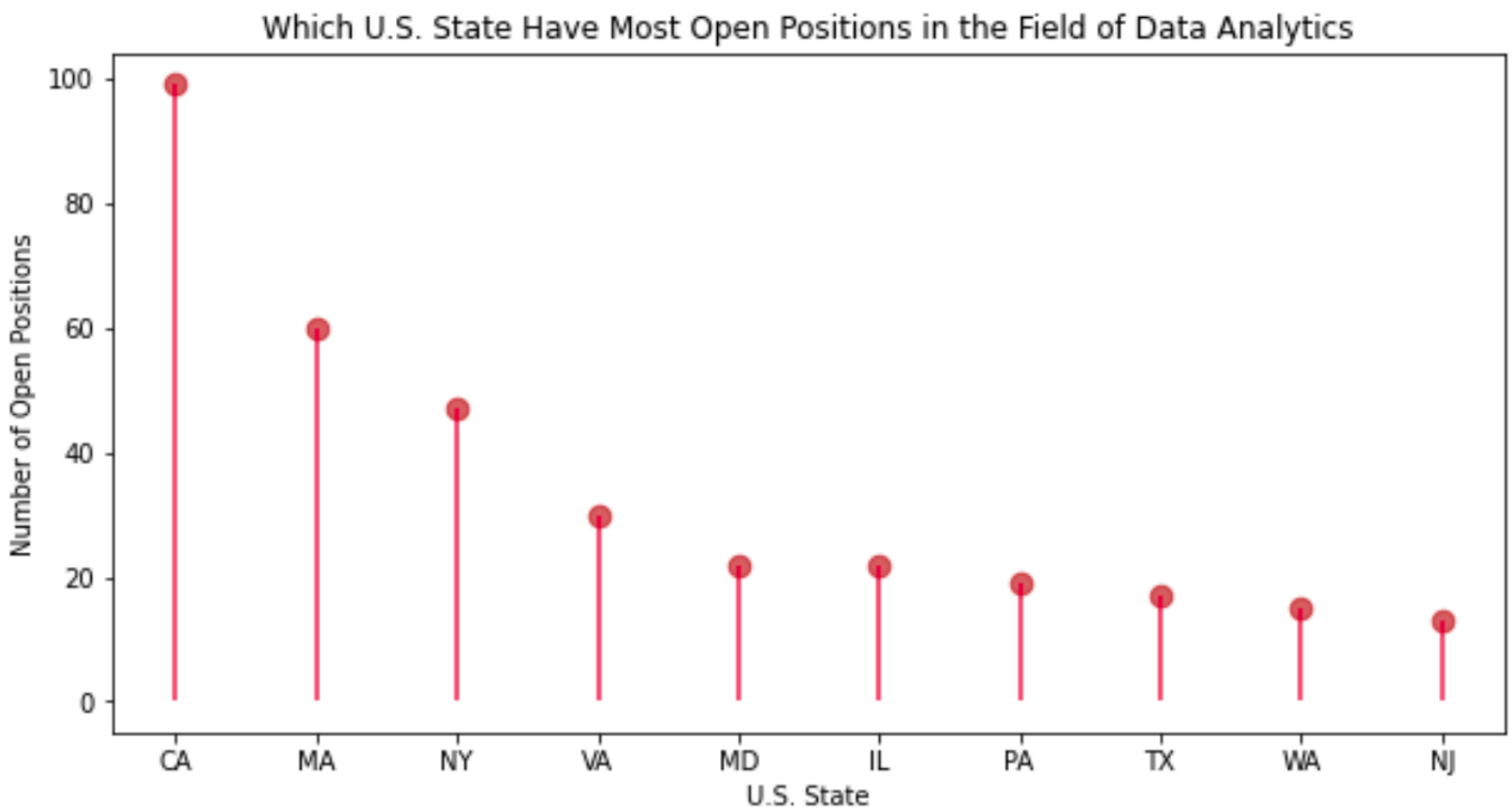
```
state_jobs = ds_data.groupby('Location (State)')['Job Title'].count().sort_values(ascending=False).head(10)
state_jobs = state_jobs.to_frame().reset_index()
```

✓ 0.3s Python

```
plt.figure(figsize=(10,5))
ax = plt.subplot()
ax.vlines(x=state_jobs['Location (State)'], ymin=0, ymax=state_jobs['Job Title'], color='crimson', alpha=0.7, linewidth=2)
ax.scatter(x=state_jobs['Location (State)'], y=state_jobs['Job Title'], s=75, color='firebrick', alpha=0.7)
plt.xlabel('U.S. State')
plt.ylabel('Number of Open Positions')

plt.title('Which U.S. State Have Most Open Positions in the Field of Data Analytics')
plt.show()
```

✓ 0.1s Python

結果呈現，綜合工作地區與薪水關係圖，California State 不僅配給薪水高，提供的工作機會也是相當多，可以用量化的方式看出大城市的經濟規模



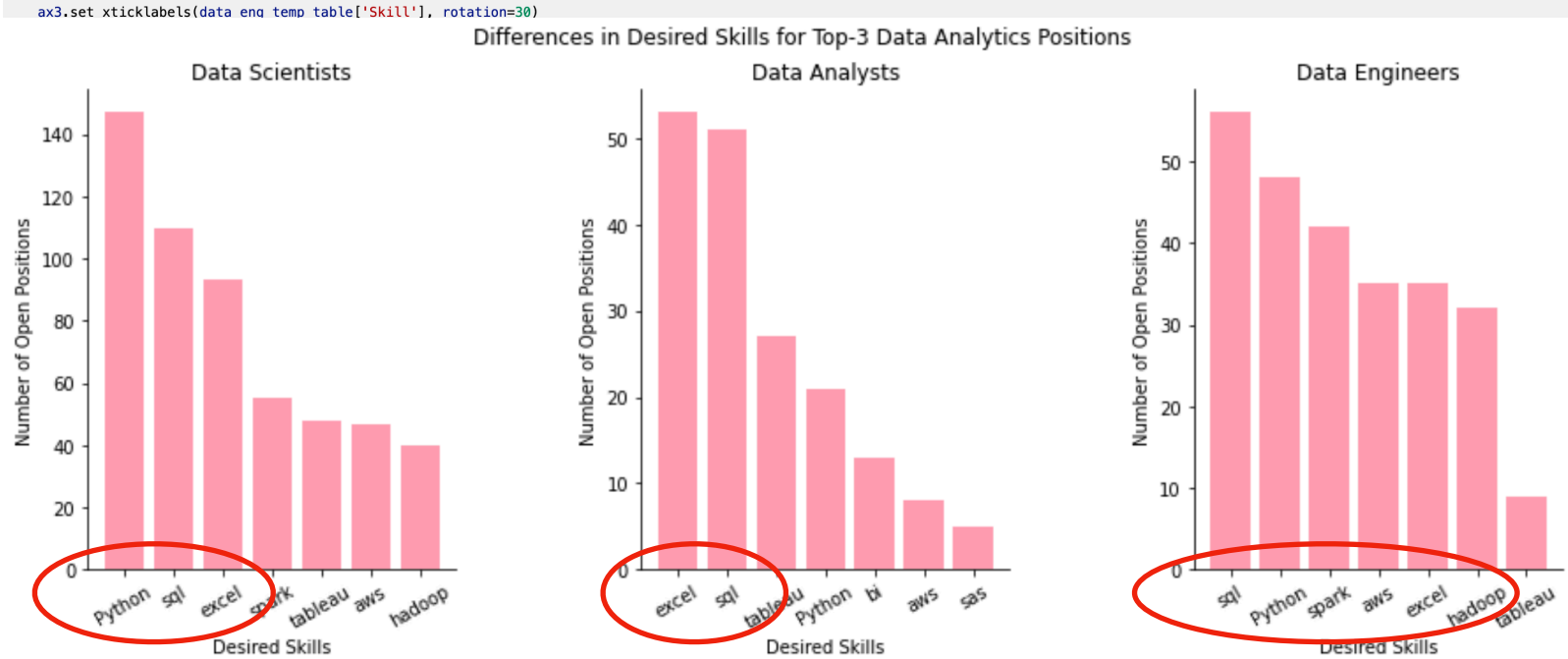
個人取向觀察結果：前幾大需求職業最需要何種技能

結果關係圖可以幫助學生或求職人有一個好的方向充實自己

```
list_of_skills = ['Python', 'spark', 'aws','excel', 'sql', 'sas', 'keras', 'pytorch', 'scikit', 'tensor', 'hadoop','tableau', 'bi', 'flink', 'mongo', 'goog

data_science_temp_table = pd.DataFrame()

for idx, skills in enumerate(list_of_skills):
    value = ds_data[ds_data['Profession'] == 'Data Scientist'][skills].sum()
    data_science_temp_table.loc[idx,'Skill'] = skills
    data_science_temp_table.loc[idx,'Count'] = value

data_science_temp_table = data_science_temp_table.sort_values(by='Count', ascending=False).head(7)

data_analyst_temp_table = pd.DataFrame()

for idx, skills in enumerate(list_of_skills):
    value = ds_data[ds_data['Profession'] == 'Data Analyst'][skills].sum()
    data_analyst_temp_table.loc[idx,'Skill'] = skills
    data_analyst_temp_table.loc[idx,'Count'] = value

data_analyst_temp_table = data_analyst_temp_table.sort_values(by='Count', ascending=False).head(7)

data_eng_temp_table = pd.DataFrame()

for idx, skills in enumerate(list_of_skills):
    value = ds_data[ds_data['Profession'] == 'Data Engineer'][skills].sum()
    data_eng_temp_table.loc[idx,'Skill'] = skills
    data_eng_temp_table.loc[idx,'Count'] = value

data_eng_temp_table = data_eng_temp_table.sort_values(by='Count', ascending=False).head(7)
```

✓ 0.6s Python

```python
plt.figure(figsize=(15,5))

ax1 = plt.subplot(1,3,1)
plt.bar(range(len(data_science_temp_table)),data_science_temp_table['Count'], color='crimson', alpha=0.4)
plt.xlabel('Desired Skills')
plt.ylabel('Number of Open Positions')
ax1.set_xticks(range(len(data_science_temp_table)))
ax1.set_xticklabels(data_science_temp_table['Skill'], rotation=30)
plt.title("Data Scientists")
for s in ['top', 'right']:
    ax1.spines[s].set_visible(False)

ax2 = plt.subplot(1,3,2)
plt.bar(range(len(data_analyst_temp_table)),data_analyst_temp_table['Count'], color='crimson', alpha=0.4)
plt.xlabel('Desired Skills')
plt.ylabel('Number of Open Positions')
ax2.set_xticks(range(len(data_analyst_temp_table)))
ax2.set_xticklabels(data_analyst_temp_table['Skill'], rotation=30)
plt.title("Data Analysts")
for s in ['top', 'right']:
    ax2.spines[s].set_visible(False)

ax3 = plt.subplot(1,3,3)
plt.bar(range(len(data_eng_temp_table)),data_eng_temp_table['Count'], color='crimson', alpha=0.4)
plt.xlabel('Desired Skills')
plt.ylabel('Number of Open Positions')
ax3.set_xticks(range(len(data_eng_temp_table)))
ax3.set_xticklabels(data_eng_temp_table['Skill'], rotation=30)
```



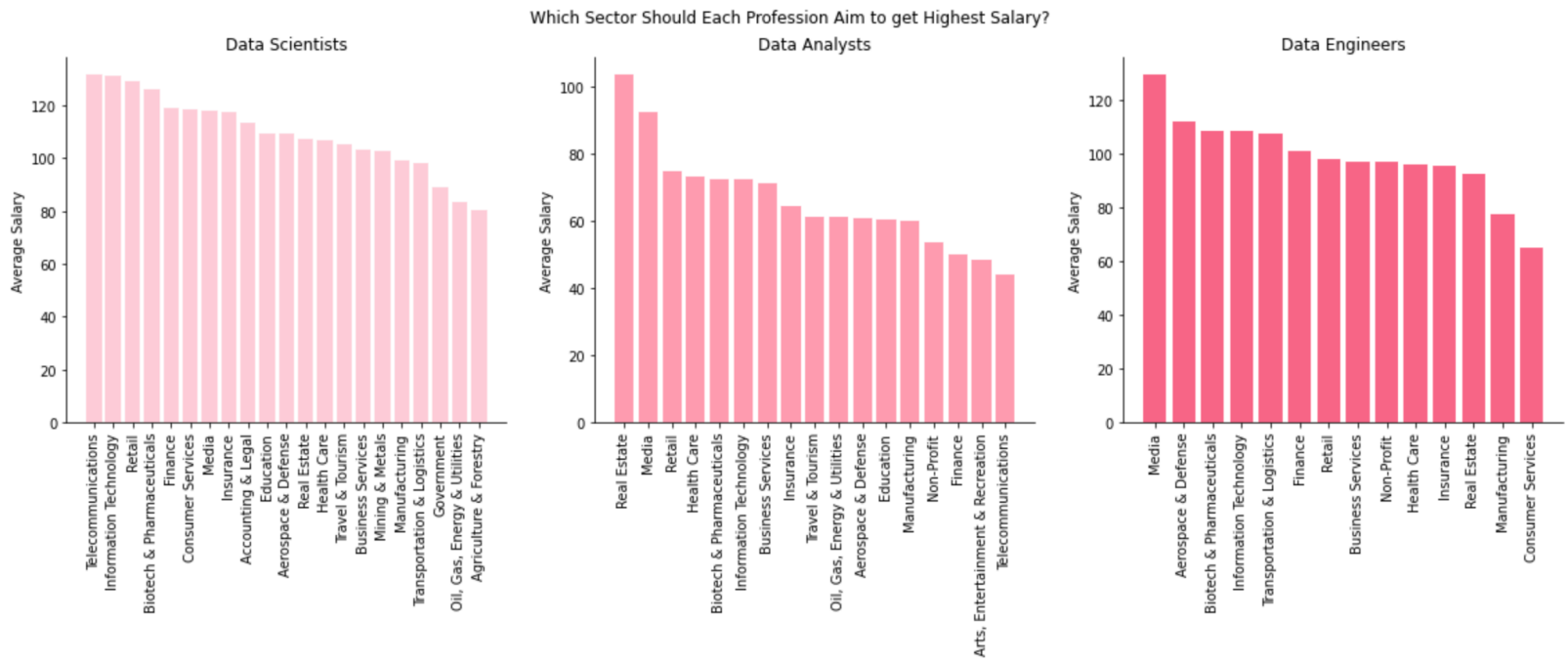
產業與薪水量化分佈圖

幫助人找出自己最想要的高薪產業

```python
sector_salary_ds = ds_data[ds_data['Profession'] == 'Data Scientist'].groupby('Sector')['Avg Salary(K)'].mean().sort_values(ascending=False)

sector_salary_da = ds_data[ds_data['Profession'] == 'Data Analyst'].groupby('Sector')['Avg Salary(K)'].mean().sort_values(ascending=False)

sector_salary_de = ds_data[ds_data['Profession'] == 'Data Engineer'].groupby('Sector')['Avg Salary(K)'].mean().sort_values(ascending=False)
```

✓ 0.2s Python

```python
plt.figure(figsize=(20,5))

ax1 = plt.subplot(1,3,1)
plt.bar(range(len(sector_salary_ds)),sector_salary_ds, color='crimson', alpha=0.2)
ax1.set_xticks(range(len(sector_salary_ds)))
ax1.set_xticklabels(sector_salary_ds.index, rotation=90)
plt.ylabel('Average Salary')
plt.title("Data Scientists")
for s in ['top', 'right']:
    ax1.spines[s].set_visible(False)

ax2 = plt.subplot(1,3,2)
plt.bar(range(len(sector_salary_da)),sector_salary_da, color='crimson', alpha=0.4)
ax2.set_xticks(range(len(sector_salary_da)))
ax2.set_xticklabels(sector_salary_da.index, rotation=90)
plt.ylabel('Average Salary')
plt.title("Data Analysts")
for s in ['top', 'right']:
    ax2.spines[s].set_visible(False)

ax3 = plt.subplot(1,3,3)
plt.bar(range(len(sector_salary_de)),sector_salary_de, color='crimson', alpha=0.6)
ax3.set_xticks(range(len(sector_salary_de)))
ax3.set_xticklabels(sector_salary_de.index, rotation=90)
plt.ylabel('Average Salary')
plt.title("Data Engineers")
for s in ['top', 'right']:
    ax3.spines[s].set_visible(False)

plt.suptitle("Which Sector Should Each Profession Aim to get Highest Salary?")
plt.show()
```



結果探討：

這份code的結論都是以視覺化的圖表、分佈圖為主，是一種把資料價值做提高的做法，人們可以依照這些圖表去做選擇工作、公司則可以作為參考，有機會進行內部改革，由此看出這份Dataset的價值。

Insight:

這兩份 code 分別對這份資料產出不同形式的結果，對於初入資料分析的我來說有很大的幫助，因為為了不同樣的目的，這兩位作者在做資料前處理時選擇不同的處理法，我實在見識甚廣。

第一份code為了能進行好的模型建立，對於資料量有一定的執著，因此他利用統計學，盡可能把遺失值用合理的值補齊，雖然未必貼於現實，但也讓建立的模型更有普遍性，意即不會overfitting，所以模型帶來的結果就會是一個值得參考的薪水數值，讓人們在看這些資料時不會認爲是固定資訊，而是能自行輸入狀況來預測薪水的互動式結果；而第二份code，作者不倚靠Machine learning，反而運用資料分析的一些直覺，為我們示範化簡

大量資料並用圖表顯示資料，而這些圖表往往能看出一些薪水配發端倪、特性，這種特性並不是一開始就能窺探，因此這樣的資料不失為一種價值資訊，畢竟在做資料前處理時，這位作者不進行補值，而是移除過多遺失值欄位，讓資料保有原本的特性。

就結論來說，如果這兩者code的概念能做整合，應該能發展出一套小型的資料分析系統；我認為，一開始要像第二位作者，先移除遺失值過多的欄位、畫成圖表，就能先看出人們想要看到的欄位關係群，再來才是將遺失值數目少的欄位做補值，也就是說學習第一位作者的統計補值技巧；最後則是把此過程做標準化流程，待過去年份更多的資料輸入、做處理後，建出「鑑往知來」的模型，而不是硬把資料「 補值 」形成偏離現實的模型。