1. (35%) Linear Regression Analysis for Wine Quality

使用 OLS 進行迴歸分析

```
df_furnace = pd.read_csv('/content/MDS_Assignment1_furnace.csv')
Y = df_furnace['grade']
X = df_furnace.iloc[:,:-1]
X = sm.add_constant(X)
results = sm.OLS(Y, X).fit()
results.summary()
```

(a) (10%) Show the results of regression analysis as follows.

	estimate	std. error	t-value	p-value
f0	-5.546435e-03	2.481004e-02	-0.223556	8.231798e-01
f1	4.397579e-02	2.799403e-02	1.570899	1.167404e-01
f2	3.139592e-01	3.477425e-02	9.028495	2.430368e-18
f3	1.858547e-02	4.153747e-02	0.447439	6.547220e-01
f4	-3.476121e-03	3.819912e-02	-0.091000	9.275234e-01
f5	-7.399292e-02	2.980243e-02	-2.482781	1.331207e-02
f6	-7.104964e-02	2.591382e-02	-2.741766	6.295708e-03
f7	2.351933e-02	2.756285e-02	0.853298	3.938391e-01
f8	4.101353e-02	1.889593e-02	2.170495	3.036614e-02
f9	2.674570e-16	4.052498e-17	6.599806	9.154186e-11
f10	-4.459473e-02	2.104490e-02	-2.119028	3.450386e-02
f11	-2.915346e-02	2.027138e-02	-1.438159	1.509174e-01
f12	-5.926334e-04	2.200147e-02	-0.026936	9.785198e-01
f13	3.363141e-02	2.382459e-02	1.411626	1.585856e-01
f14	-1.832175e-01	2.059106e-02	-8.897917	6.896905e-18
f15	-1.060875e-01	1.906256e-02	-5.565229	3.971326e-08
f16	-3.577142e-02	2.037004e-02	-1.756080	7.959189e-02
f17	6.331038e-02	1.857249e-02	3.408826	6.967770e-04
f18	-1.903848e-01	2.070661e-02	-9.194397	6.355895e-19
f19	2.775466e-02	2.641510e-02	1.050712	2.938197e-01
f20	1.256291e-02	1.951939e-02	0.643612	5.200766e-01
f21	-3.567369e-02	2.824686e-02	-1.262926	2.071131e-01
f22	7.468009e-02	2.113791e-02	3.532993	4.429125e-04
f23	-8.768870e-03	1.983889e-02	-0.442004	6.586476e-01
f24	1.930274e-02	2.411629e-02	0.800403	4.237986e-01
f25	-6.792171e-02	1.994242e-02	-3.405891	7.041639e-04
f26	-3.600558e-02	2.215419e-02	-1.625227	1.046465e-01
f27	-6.201440e-03	1.916890e-02	-0.323516	7.464190e-01

R-squared: 0.4953759883189155
Adjusted R-squared: 0.47236104184021743

(b) (5%) The fitting of the linear regression is a good idea? If yes, why? If no, why? What's the possible reason of poor fitting?

在回歸分析中，R-squared 越接近 1 代表模型表現越好。在此例題中，R-squared < 0.5，模型表現並不佳。而 Adjusted R-squared < 0.5，代表我們所採用和未採用的自變量存在著問題。

(c) (5%) Based on the results, rank the independent variables by p-values and which one are statistically significant variables with p-values<0.01?

p-value 可以視為出錯的機率，在 p-value < 0.01 的篩選條件下，得到 9 個變數滿足該條件，p-value 有小到大依序是：f18 < f2 < f14 < f9 < f15 < f22 < f17 < f25 < f6

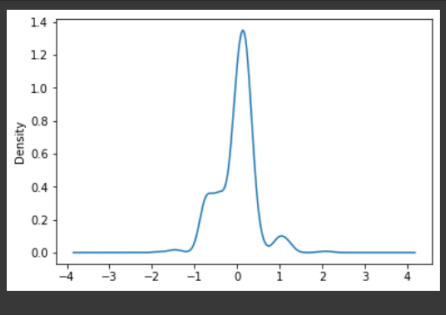
```
print('significant variables with p-value < 0.01: \n'
      ,result_df[result_df['p-value']<0.01].sort_values(by=['p-value']).index)
```

```
significant variables with p-value < 0.01:
 Index(['f18', 'f2', 'f14', 'f9', 'f15', 'f22', 'f17', 'f25', 'f6'], dtype='object')
```

(d) (15%) Testify the underlying assumptions of regression (1) Normality, (2) Independence, and (3) Homogeneity of Variance with respect to residual.

(1) Normality

```
# 繪製殘差密度來檢查正態性
import matplotlib.pyplot as plt
plt.figure()
results.resid.plot.density()
plt.show()
```



使用 Shapiro-Wilk 常態性檢定，H0：常態 ；H1：非常態。 若 p-value < 0.05，我們就可以拒絕虛無假設(H0)，並宣稱這個檢定在統計上是顯著的。此題顯示 p-value <0.05，拒絕 H0，表示殘差分佈為非常態，迴歸模型的常態性不通過。

```
from scipy import stats
shapiro_test = stats.shapiro(results.resid)
shapiro_test.pvalue
```

```
7.536043131996446e-16
```

(2) Independence

採用最常使用的 Durbin-Watson 之 D 檢定法，當 DW 值接近 2 左右，通常沒有違反獨立性的假設

```
from statsmodels.stats.stattools import durbin_watson
print('DW: ',durbin_watson(results.resid))
```

```
DW:  2.004053480472211
```

(3) Homogeneity of Variance with respect to residual

使用 BP 檢定進行驗證，此檢定假設 H0：同質性；H1：異質性

顯示 P-value < 0.05，拒絕 H0，代表殘差不具有同質性，因此線性模型可能有問題。

```
from statsmodels.stats import diagnostic as dia
het = dia.het_breuschpagan(results.resid,X)
print('p-value: ',het[-1])
```

```
p-value:  5.311815675352488e-20
```

2. (35%) Association Rule- Market Basket Analysis

(1) (10%) How to handle the raw dataset via data preprocessing?

Step1. 讀取資料

```
groceries_df = pd.read_csv('/content/MDS_Assignment1_groceries.csv','\r',header=None)
groceries_df
```

```
/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:3326: FutureWarning: In a
  exec(code_obj, self.user_global_ns, self.user_ns)
```

	0
0	citrus fruit,semi-finished bread,margarine,rea...
1	tropical fruit,yogurt,coffee
2	whole milk
3	pip fruit,yogurt,cream cheese ,meat spreads
4	other vegetables,whole milk,condensed milk,lon...
...	...
9830	sausage,chicken,beef,hamburger meat,citrus fru...
9831	cooking chocolate
9832	chicken,citrus fruit,other vegetables,butter,y...
9833	semi-finished bread,bottled water,soda,bottled...
9834	chicken,tropical fruit,other vegetables,vinega...

9835 rows × 1 columns

Step2. 透過 transaction encoder 將每個產品設一個 column，若該筆交易有購買該產品則顯示 True，否則顯示 False。最後將整理好的資料存入 groceries_df2 (dataframe)中。

```
# 分割 data 中的商品
products = [data[0].split(',') for data in groceries_df.values]
# 使用mlxtend作 feature encoding
te = TransactionEncoder()
te_ary = te.fit(products).transform(products)
# 建立整理好的dataframe
groceries_df2 = pd.DataFrame(te_ary, columns=te.columns_, dtype='bool')
groceries_df2
```

	Instant food products	UHT-milk	abrasive cleaner	artif. sweetener	baby cosmetics	baby food	bags	baking powder	bathroom cleaner	beef	...	turkey	vir
0	False	False	False	False	False	False	False	False	False	False	...	False	
1	False	False	False	False	False	False	False	False	False	False	...	False	
2	False	False	False	False	False	False	False	False	False	False	...	False	
3	False	False	False	False	False	False	False	False	False	False	...	False	
4	False	False	False	False	False	False	False	False	False	False	...	False	
...	...	...	...	...	...	...	...	...	...	...	...	...	
9830	False	False	False	False	False	False	False	False	False	True	...	False	
9831	False	False	False	False	False	False	False	False	False	False	...	False	
9832	False	False	False	False	False	False	False	False	False	False	...	False	
9833	False	False	False	False	False	False	False	False	False	False	...	False	

(2) (10%) What's the top 5 association rules? Show the support, confidence, and lift to each specific rule, respectively?

Step1. 使用 apriori 演算法，設置 minimum support = 0.001 計算出 frequent itemsets。

Step2. 使用 association_rules( )建立關聯規則，可以得知每個關聯規則的 support, confidence 和 lift。

Step3. 根據題意，篩選出 confidence > 0.15 的規則，再用 lift 值去排序，得到 top5 rules。

```
# generate frequent item sets that have a support of at least 0.1%
frequent_itemsets = apriori(groceries_df2, min_support=0.001, use_colnames=True)
frequent_itemsets
```

	support	itemsets
0	0.008033	(Instant food products)
1	0.033452	(UHT-milk)
2	0.003559	(abrasive cleaner)
3	0.003254	(artif. sweetener)
4	0.017692	(baking powder)
...	...	...
13487	0.001017	(other vegetables, citrus fruit, root vegetabl...
13488	0.001017	(other vegetables, root vegetables, yogurt, wh...
13489	0.001322	(other vegetables, root vegetables, yogurt, wh...
13490	0.001322	(other vegetables, root vegetables, yogurt, wh...
13491	0.001118	(other vegetables, root vegetables, yogurt, wh...

```
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)
rules
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift
0	(Instant food products)	(bottled water)	0.008033	0.110524	0.001017	0.126582	1.145296
1	(bottled water)	(Instant food products)	0.110524	0.008033	0.001017	0.009200	1.145296
2	(Instant food products)	(butter)	0.008033	0.055414	0.001220	0.151899	2.741145
3	(butter)	(Instant food products)	0.055414	0.008033	0.001220	0.022018	2.741145
4	(Instant food products)	(citrus fruit)	0.008033	0.082766	0.001118	0.139241	1.682347
...	...	...	...	...	...	...	...
101425	(root vegetables)	(other vegetables, yogurt, whole milk, whipped...	0.108998	0.002440	0.001118	0.010261	4.204952
101426	(yogurt)	(other vegetables, root vegetables, whole milk...	0.139502	0.001932	0.001118	0.008017	4.150107
101427	(whole milk)	(other vegetables, root vegetables, yogurt, wh...	0.255516	0.001729	0.001118	0.004377	2.532361
101428	(whipped/sour cream)	(other vegetables, root vegetables, yogurt, wh...	0.071683	0.003559	0.001118	0.015603	4.384397
101429	(tropical fruit)	(other vegetables, root vegetables, yogurt, wh...	0.104931	0.002339	0.001118	0.010659	4.557845

```
top5 = rules[rules['confidence']>=0.15].sort_values(by=['lift'],ascending=False)[:5]
top5
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift
9816	(liquor)	(bottled beer, red/blush wine)	0.011083	0.004881	0.001932	0.174312	35.715787
9813	(bottled beer, red/blush wine)	(liquor)	0.004881	0.011083	0.001932	0.395833	35.715787
101222	(whole milk, oil, tropical fruit)	(other vegetables, yogurt, root vegetables)	0.002542	0.012913	0.001017	0.400000	30.976378
101215	(oil, yogurt, root vegetables)	(whole milk, other vegetables, tropical fruit)	0.001932	0.017082	0.001017	0.526316	30.811404
100940	(whole milk, domestic eggs, other vegetables, ...	(butter, tropical fruit)	0.003355	0.009964	0.001017	0.303030	30.411255

(3) (5%) Please provide/guess the "story" to interpret one of top-5 rules you are interested in.

(oil, yogurt, root vegetables) ⇨ (whole milk, other vegetables, tropical fruit)

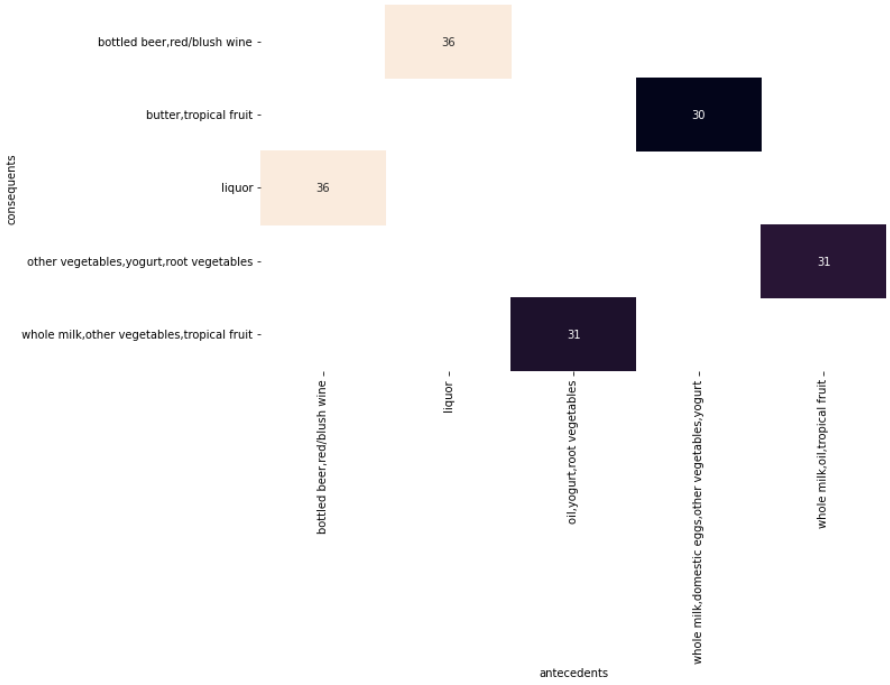
一位正在瘦身減脂的女性，早餐吃低熱量的優格，午晚餐則選擇自己煮根莖類食物當作主食以補充優良的澱粉，為了增加纖維攝取量和營養，又另外添購了蔬菜和油。水果和牛奶則是可以自製優酪乳的食材，增加多樣性。

(4) (10%) Give a visualization graph of your association rules.

主要透過兩種不同的方式針對 top5 rules 進行資料的視覺化。

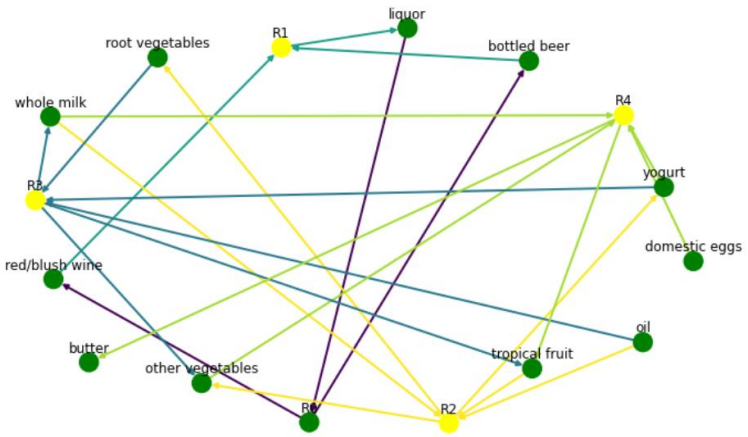
1. Heatmap：依據 lift

```
import matplotlib.pyplot as plt
import seaborn as sns
# Convert antecedents and consequents into strings
top5['antecedents'] = top5['antecedents'].apply(lambda a: ','.join(list(a)))
top5['consequents'] = top5['consequents'].apply(lambda a: ','.join(list(a)))

# Transform antecedent, consequent, and support columns into matrix
lift_table = top5.pivot(index='consequents', columns='antecedents', values='lift')

plt.figure(figsize=(10,6))
sns.heatmap(lift_table, annot=True, cbar=False)

plt.show()
```



2. Graph (套件: networkx)

一個顏色的線代表一條關聯規則，圖中有五個規則節點作為 antecedents 和 consequents 的橋樑，依序 top5 將規則節點取名為(R1, R2, R3, R4, R5)。多個 antecedents 會先共同指向規則節點，再從規則節點指向 consequents。



```python
import networkx as nx
import numpy as np
def draw_graph(rules, rules_to_show):

    G1 = nx.DiGraph()
    color_map=[]
    N = 5
    colors = np.random.rand(N)
    strs=['R1', 'R2', 'R3', 'R4', 'R5']
    plt.figure(figsize=(10,6))


    for i in range (rules_to_show):
        G1.add_nodes_from(['R'+str(i)])

        for a in top5.iloc[i]['antecedents']:

            G1.add_nodes_from([a])

            G1.add_edge(a,'R'+str(i),color=colors[i],weight = 2)

        for c in top5.iloc[i]['consequents']:

            G1.add_nodes_from([c])

            G1.add_edge('R'+str(i),c,color=colors[i],weight=2)
```

```python
for node in G1:
    found_a_string = False
    for item in strs:
        if node==item:
            found_a_string = True
    if found_a_string:
        color_map.append('yellow')
    else:
        color_map.append('green')


edges = G1.edges()
print(edges)
colors = [G1[u][v]['color'] for u,v in edges]
weights = [G1[u][v]['weight'] for u,v in edges]

pos = nx.spring_layout(G1,k=10)
nx.draw(G1, pos,node_color=color_map, edge_color=colors,
        width=weights, font_size=16, with_labels=False)

for p in pos:  # raise text positions
    pos[p][1] += 0.07
nx.draw_networkx_labels(G1, pos)
plt.show()
```

3. (30%) Manufacturing System Analysis

(a) (10%)根據 Little’ s Law，試計算各工作站的產出率 TH 於下表；試問瓶頸站的產出率 rb、最小生產週期時間(總加工時間，$T0$)、關鍵在製品水準($W0$)各為多少？

```python
df = pd.DataFrame({'工作站編號':[1,2,3,4,5],'機台數':[3,3,6,2,5],'加工時間(小時)':[5,8,12,4,12]})
df['TH'] = df['機台數']/ df['加工時間(小時)']
df
```

	工作站編號	機台數	加工時間(小時)	TH
0	1	3	5	0.600000
1	2	3	8	0.375000
2	3	6	12	0.500000
3	4	2	4	0.500000
4	5	5	12	0.416667

```python
bottle_neck_index = min(range(len(df['TH'])), key=df['TH'].__getitem__)
W_0 = df['機台數'][bottle_neck_index]
r_b = df['TH'][bottle_neck_index]
T_0 = df['加工時間(小時)'][bottle_neck_index]
print('瓶頸站的產出率rb = {}、最小生產週期時間(總加工時間，T0) = {}、關鍵在製品水準(W0) = {}'.format(r_b,T_0,W_0))
```

```
瓶頸站的產出率rb = 0.375、最小生產週期時間(總加工時間，T0) = 8、關鍵在製品水準(W0) = 3
```

(b) (10%)試給出最佳績效(best case)下，最大的產出率(THbest)與最小生產週期時間(CTbest)的計算公式？

在 best case 下的 THBest 和 CTBest 即瓶頸站的最佳 TH 和最佳 CT，故此產線

的 best case 使用的參數是瓶頸站的W_0、r_b和T_0。

$$\mathbf{CT(w)} = \begin{cases} \boldsymbol{T_0} & , \boldsymbol{w \le W_0} \\ \dfrac{\boldsymbol{w}}{\boldsymbol{r_b}} & , \boldsymbol{w > W_0} \end{cases}$$

$$\mathbf{TH(w)} = \begin{cases} \dfrac{\boldsymbol{w}}{\boldsymbol{T_0}} & , \boldsymbol{w \le W_0} \\ \boldsymbol{r_b} & , \boldsymbol{w > W_0} \end{cases}$$

```
CT = []; TH=[]; WIP=[]
for w in range(1,11):
        WIP.append(w)
        if w <= W_0:
                CT.append(T_0)
                TH.append(w/T_0)
        else:
                CT.append(w/r_b)
                TH.append(r_b)

CT_best = min(CT)
TH_best = max(TH)
print('CT_best: ',CT_best)
print('TH_best: ',TH_best)
```

```
CT_best:  8
TH_best:  0.375
```

(c) (10%)根據該問題的產線，試程式撰寫建立一模擬模型(或用套裝軟體、數值分析)來 驗證，當在製品 WIP 數量超過工廠產能時，其生產週期將嚴重惡化。也就是當產線的投 料速度(投產量)大於產線的產出率，此時生產系統將處於非穩態的狀態(non-steady state)。 試用圖表呈現 WIP、CT 與 TH 之間惡化的關係。(提示：講義 22-29 頁)

1. 建立一個 function，其 input 值為計算好 TH 的 dataframe 以及 w 值的模擬上限。
2. 透過公式，計算出 WIP 從 1 到 w 的 CT 值和 TH 值，並存入陣列中。
3. 透過折線圖繪出 CT、TH 和 WIP 之間的關係。
4. 可以看出當 WIP 的數量大於瓶頸站的W_0後，TH 持平、CT 值上升，呈現惡化的現象。

```python
def little_Law(df, W):
    bottle_neck_index = min(range(len(df['TH'])), key=df['TH'].__getitem__)
    W_0 = df['機台數'][bottle_neck_index]
    r_b = df['TH'][bottle_neck_index]
    T_0 = df['加工時間(小時)'][bottle_neck_index]
    CT = []; TH=[]; WIP=[]
    for w in range(1,W+1):
        WIP.append(w)
        if w <= W_0:
            CT.append(T_0)
            TH.append(w/T_0)
        else:
            CT.append(w/r_b)
            TH.append(r_b)

    plt.figure(figsize=(8, 4))

    # 設置x軸刻度間隔為1
    x_major_locator = MultipleLocator(1)
    ax = plt.gca()
    ax.xaxis.set_major_locator(x_major_locator)
    plt.plot(WIP, CT)
    plt.xlabel('WIP', fontsize="14")
    plt.ylabel('CT', fontsize="14")
    plt.title('Best Case: CT-WIP', fontsize="18")
    plt.show()

    plt.figure(figsize=(8, 4))
    plt.plot(WIP, TH)
    plt.xlabel('WIP', fontsize="14")
    plt.ylabel('TH', fontsize="14")
    plt.title('Best Case: TH-WIP', fontsize="18")
    plt.show()
```