作業二 資管碩二 呂晟維 r10725012

(助教抱歉，由於我寫完 hw2 才看到您的評語，這次先繳交 py 檔+PDF，下次就改繳交單一份 html)

Q1 Curse of Dimensionality

(a) 試簡述何謂維度的詛咒？試列舉一案例說明。

Ans:

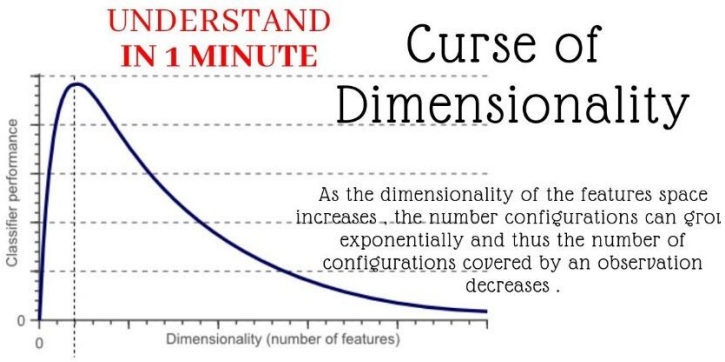
模型的參數數量與模型本身的複雜度與資料集的特徵維度呈正相關，

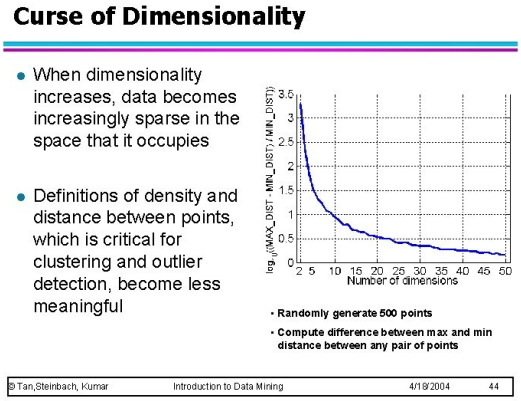
模型的參數數量越多，使模型達成收斂需要的資料筆數也就越多，當樣本數不足時會導致

1. 花很長時間學習但演算法不易收斂
2. 收斂時出現多重解 (multiple solutions) 或過度配適

簡而言之，特徵從零變多模型的預測能力一開始會提升，當過了最適的特徵個數，預測績效立即呈現指數遞減。(如下圖左)

舉例: 植栽的數據集有花齡、品種、施肥、澆水量等特徵；若特徵數量越多，特徵空間中的樣本點就會變得更疏離，對樣本取距離 min 和 max 會顯得沒有差別，降低模型的適配度。(如下圖右)





(b) 避免維度詛咒的方法有哪些？

Ans:

1. 避免使用過多特徵，僅使用最適的特徵個數數量的作為訓練資料。
2. 檢查特徵間是否有共線性關係，若有則整合或剔除具共線性關係的特徵們。

(c) 試找一個開放數據 (e.g. Kaggle 開放數據或第一次作業紅酒數據集)並選一種方法 (e.g. 線性迴歸或決策樹)，用模擬方法 固定樣本數但逐步增加變數個數，試著重新繪製圖 3.12，呈現維度與預測 (或分類)績效間的關係。

Ans: 我使用 hw1 的紅酒數據集，一共有 28 個特徵以及一個標籤欄位。重要變數如(1)(2)之程式輸出，重要程度為 f18 > f2 > f14 > … > f12。

(1) 整體資料先做線性迴歸。

(2) 依 p value 小至大排序選出重要變數。

```python
# (1) 整體資料先做線性迴歸；
# (2) 依 p value 小至大排序選出重要變數；
furnace_model = sm.OLS(furnace_y, furnace_X_const)
furnace_result = furnace_model.fit()
# print(furnace_result.summary())
print('rsquared:', furnace_result.rsquared, '\nrsquared_adj:', furnace_result.rsquared_adj)

furnace_pvalues = furnace_result.pvalues
print(furnace_pvalues.sort_values().head(3)) # p 越小影響力越大
print(furnace_pvalues.sort_values().tail(3)) # p 越大影響力越小

sorted_pvalues = list(furnace_pvalues.sort_values().keys())
print(f"sorted_pvalues: {sorted_pvalues}")
```

Python

```
sorted_pvalues: ['Intercept', 'f18', 'f2', 'f14', 'f15', 'f22', 'f17', 'f25', 'f6', 'f5', 'f8', 'f10', 'f16',
'f26', 'f1', 'f11', 'f13', 'f21', 'f19', 'f7', 'f24', 'f20', 'f3', 'f23', 'f27', 'f0', 'f4', 'f9', 'f12']
```

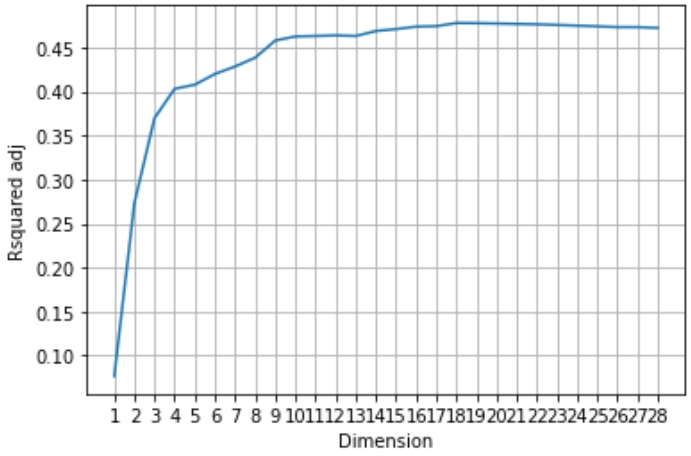
(3) 將重要的變數一個個依序放入迴歸並計算 adjusted R2 作為預測準確度。

變數數量之於準確度的相關性如下圖左，可以看到當變數個數從 1 增加到 4 個時，模型準確度快速增長，18 個變數為峰值。但此資料集在維度增加時，並未展現如下圖右的指數遞減情形，或許可以推論資料集的維度數量適中，未達維度詛咒的標準。

```python
# (3) 將重要的變數一個個依序放入迴歸並計算 adjusted R2 作為預測準確度
# formula string format: 'Label ~ param1 + param2 ...'
numOfTest = len(sorted_pvalues[1:])
rsquared_adj_lst = []

for i in range(1, numOfTest+1):
    formula_str = make_formula(i)
    result = smf.ols(formula=formula_str, data=furnace_data).fit()
    print(f'rsquared_adj with {i} params: {result.rsquared_adj}')
    rsquared_adj_lst.append(result.rsquared_adj)

plt.plot(rsquared_adj_lst)
```

Python

```
rsquared_adj with 1 params: 0.0768328566497618
rsquared_adj with 2 params: 0.2733340635593492
rsquared_adj with 3 params: 0.3698974444769454
rsquared_adj with 4 params: 0.403183398054795
rsquared_adj with 5 params: 0.40785545925955646
```



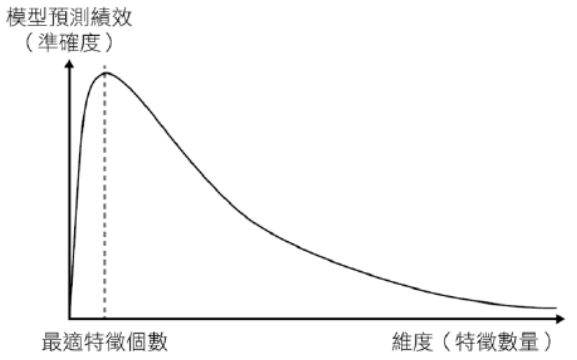


圖 3.12　數據維度與模型的預測的準確度

Q2 Data Quality

(a)試找一個開放數據，會用什麼方法來確認資料品質的好壞？

Ans: 我會由(c)小題的三項指標，來應用於(b)小題的 SOP。

- 數據: https://www.kaggle.com/datasets/segunadedipe/nigerian-car-prices 尼日利亞汽車價格的資料集，共 9 個欄位和 4095 筆資料。

```
# https://www.kaggle.com/datasets/whenamancodes/covid-19-coronavirus-pandemic-dataset
# https://www.kaggle.com/datasets/segunadedipe/nigerian-car-prices
df = pd.read_csv("Nigerian_Car_Prices.csv")
df = df.iloc[:, 1:]
print(df.shape)
df.head()
```

✓ 0.8s　Python

```
(4095, 9)
```

	Make	Year of manufacture	Condition	Mileage	Engine Size	Fuel	Transmission	Price	Build
0	Toyota	2007.0	Nigerian Used	166418.0	2400.0	Petrol	Automatic	3,120,000	NaN
1	Lexus	NaN	NaN	138024.0	NaN	NaN	Automatic	5,834,000	NaN
2	Mercedes-Benz	2008.0	Nigerian Used	376807.0	3000.0	Petrol	Automatic	3,640,000	NaN
3	Lexus	NaN	NaN	213362.0	NaN	NaN	Automatic	3,594,000	NaN
4	Mercedes-Benz	NaN	NaN	106199.0	NaN	NaN	Automatic	8,410,000	NaN

1. 遺漏比例 (Missing value rate)

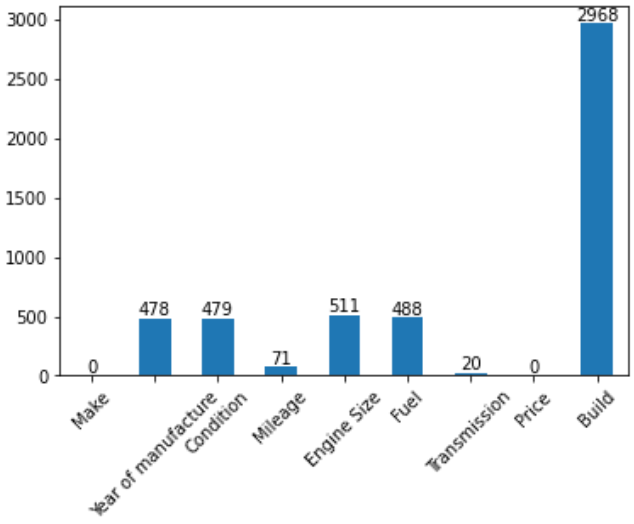
   整體的遺漏比例為 13.61% 是可以接受的。

   而 build 欄位遺漏數為 2968 筆 (比例為 72%)，缺失率非常高，經檢視發現 build 的值皆為 'SUV'車型，空值可能 Sedan, CUV, Truck 等其他車型，故可以以二分法進行填補，不需刪除此欄位。

```
# Missing value rate
# ref https://datatofish.com/check-nan-pandas-dataframe/
numOfNanCell = df.isnull().sum().sum()
numOfTotalCall = df.shape[0]*df.shape[1]
missRate = numOfNanCell / numOfTotalCall
print(f"Missing value rate is {100*missRate:.2f}%. ({numOfNanCell} out of {numOfTotalCall})")
df.isnull().sum().plot(kind='bar')
```

✓ 0.3s　Python　Python　Python

```
Missing value rate is 13.61%. (5015 out of 36855)
```



2. 特徵的獨立性 (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。我們將廠牌當作 Label 的情形下，此數據為 DW 值為 1.7066 具獨立性。

```python
# Independence by durbin_watson
from statsmodels.formula.api import ols
from statsmodels.stats.stattools import durbin_watson
import pandas as pd
import numpy as np

# fit multiple linear regression model
model = ols("Make ~ Q('Year of manufacture') + Condition + Mileage + Q('Engine Size') + \
    Fuel + Transmission + Price + Build", data=df)
res = model.fit()

dw = durbin_watson(res.resid)
print(f"Durbin-Watson: {dw}")
```

✓ 0.1s Python

```
Durbin-Watson: 1.7066677199512363
```

Ps: 進行 OLS 之前需要先填補空值，以及將類別型資料轉為數值資料。

```python
# 進行獨行性測試(OLS建模)之前，需要補空值，把類別轉成數值
def format_float(x: str):
    return x.replace(',','')

df['Price'] = df['Price'].apply(lambda x: format_float(x))

# 數值資料填 mean
for col_name in ['Year of manufacture', 'Mileage', 'Engine Size', 'Price']:
    df[col_name] = pd.to_numeric(df[col_name])
    df[col_name].fillna(value=df[col_name].mean(), inplace=True)
# df.fillna(value=0, inplace=True)

# 類別資料轉成 0,1,2 空值 -1
for col_name in ['Make', 'Condition', 'Fuel', 'Transmission', 'Build']:
    # df[col_name] = df[col_name].astype('category')
    df[col_name] = pd.factorize( df[col_name] )[0]
df.head()
```

✓ 0.1s Python

	Make	Year of manufacture	Condition	Mileage	Engine Size	Fuel	Transmission	Price	Build
0	0	2007.000000	0	166418.0	2400.000000	0	0	3120000	-1
1	1	2007.898535	-1	138024.0	3274.976562	-1	0	5834000	-1
2	2	2008.000000	0	376807.0	3000.000000	0	0	3640000	-1
3	1	2007.898535	-1	213362.0	3274.976562	-1	0	3594000	-1
4	2	2007.898535	-1	106199.0	3274.976562	-1	0	8410000	-1

3. 資訊量/熵 (Entropy)

我們可以對任何類別型的特徵衡量其熵，我們挑 Label (廠牌)作範例，由結果得知有 49 個品牌，Entropy 是 2.367 具有足夠的資訊量。

```python
# Entropy of 'Make', we can see the entropy is high, the infomation in rich enough.
# ref https://stackoverflow.com/questions/15450192/fastest-way-to-compute-entropy-in-python
from math import log, e
def entropy3(labels, base=None):
  vc = pd.Series(labels).value_counts(normalize=True, sort=False)
  print(f'number of brand: {len(vc)}')
  base = e if base is None else base
  return -(vc * np.log(vc)/np.log(base)).sum()

entropy3(df['Make'])
```

✓ 0.8s Python Python Python

```
number of brand: 49

2.367840566848608
```

```python
df['Make'].value_counts(normalize=True, sort=True).head()
```

✓ 0.9s Python

```
Toyota          0.358730
Lexus           0.113309
Mercedes-Benz   0.106471
Honda           0.104518
Ford            0.048107
Name: Make, dtype: float64
```

(b)確認資料品質的標準作業流程。

Ans:

1. 檢查資料對母體的占比性(Proportion)與時效性(Timeliness)。=>離群值偵測
2. 檢查資料的格式與豐富性，如一致性(consistency)、多元性(diversity)。=>正規化

3. 檢查資料的安全性(Confidentiality、Integrity、Availability) =>遺漏值填補
4. 檢查資料的準確及完善，如準確性(accuracy)、數據解釋性(interpretability)。=>建模

(c)試建議三個可能衡量數據品質的量化指標 (i.e. KPIs)。

Ans:

1. 遺漏比例 (Missing value rate): num of missing data / num of total data.
2. 特徵的獨立性 (Independence): Durbin-Watson Test, A value of 2.0 indicates there is no autocorrelation detected in the sample.
3. 資訊量/熵 (Entropy): Shannon Entropy can describe the information richness of data (uncertainty). We can calculate the Entropy of each feature as well as the Entropy of label.

Q3 決策支援

1. 適應性: 隨著時間的推移和模型是否需要重新訓練以維持績效水平，不適應性稱為 drift(飄移)，細部包括 drift detection(偵測), drift understanding(何原因發生飄移), drift adaptation(如何更新模型)。
2. 擴充性: Transfer Learning，將 A 主題上訓練的模型，搬移至 B 主題使用，此舉是基於 A、B 主題的數據在空間分布上有可能有特徵空間的交集。例如將 Tesla 自小客車的自駕行車模型，轉移至大貨車的自動駕駛上。
3. 我的數據資料較為老舊，汽車出廠年份大約為 2000 多年，對於 2022 年的汽車適應性可能有偏移的現象，可再加入新樣本作訓練。若論應用於他國汽車買賣的擴充性，我覺得應該適配，因為汽車的規格在全球化之下有一定的共通規範，對於估價的應用，具有擴充性。

Q4 Missing Value

遺漏值填補的方法包括了統計量填補 、預測式與生成式填補

(a) 試說明這些方法分別適用於什麼樣情形?

- ✓ 統計量填補 (statistics imputation): 以 mean, min, max 或不偏估計量來填補的方法。以 mean 填補可以保持平均數不變，以 min, max 填補可以樂觀或悲觀的態度做解釋。當我們需要解釋資料的樣貌，或是演算法不能接受空值時，此法是最直接的做法。
- ✓ 預測式填補 (predictive imputation): 有 KNNI, MICE 等方式，MICE 會先用 mean 來填補所有的遺漏值，再逐一建立預測模型來填補一個欄位。此方法適用於生成與原始資料分布相似的資料點，引用論文的一句話: If the original variable is skewed, the imputed values will also be skewed. If the original variable is bounded by 0 and 100, the imputed values will also be bounded by 0 and 100.
- ✓ 生成式填補 (generative imputation): 是將「生成對抗網路」(generative adversarial network, GAN) 的模型應用在填補上，以「生成模型」生成填補數據，再以「判別模型」判別生成後的數據為實際還是是被生成的。適用於圖像生成、聲音生成等領域，被大量應用"創造"上，如 NovelAI,繪圖軟體輔助渲染背景等。GAN 只須建單一模型，也改善預測式填補需建多個不同模型的缺點，考量效能時可擇此法。

Ref :

1. https://statisticalhorizons.com/predictive-mean-matching/
2. https://www.ibm.com/docs/zh-tw/spss-statistics/25.0.0?topic=values-missing-value-analysis

(b) 為什麼某特徵存在大量遺漏值不宜直接刪除？

因為這些"有值"的樣本有機會代表了某種 Integrity，很可能是 systematic missing，如某年度以後的機台才能紀錄零件承受的壓力，而大部分的機台都無此紀錄，因此我們需要對這個壓力欄位做額外的填補而非直接刪除來得合常理。

Q5 鋼板判別模型 (30%)

在 UCI Machine Learning Repository 開放數據中包含了一個鋼板缺陷數據 (steel plates faults dataset https://archive.ics.uci.edu/ml/datasets/steel+plates+faults)，一共包含了 1,941 個觀測值，而每個觀測值具有 27 個特徵以及作為目標值的 7 種缺陷。試挑選出凹凸不平(Bumps)以及刮痕(K_Scratch)兩種缺陷進行分析。試著參考網路資源學習並撰寫程式， 使用此數據回答下列問題。

(a) 試將羅吉斯迴歸分析的結果呈現如下表，並試著解釋任一特徵與目標值之間的關係。

Ans:

```
num of bump samples: 402 out of 1941
num of scatch samples: 391 out of 1941
```

我們先實驗二次判別模型，分別建兩個模型，各自判斷凹凸不平和刮痕:

1. Bumps 凹凸不平=>1, 良品=>0

MDS_Assignment2 > Q5_Logit_BumpModel.txt

```
Warning: Maximum number of iterations has been exceeded.
         Current function value: nan
         Iterations: 35
                    MNLogit Regression Results
==============================================================================
Dep. Variable:                  Bumps   No. Observations:                 1941
Model:                        MNLogit   Df Residuals:                     1914
Method:                           MLE   Df Model:                           26
Date:                Sun, 23 Oct 2022   Pseudo R-squ.:                    nan
Time:                        10:28:44   Log-Likelihood:                   nan
converged:                     False   LL-Null:                       -990.11
Covariance Type:           nonrobust   LLR p-value:                      nan
==============================================================================
       Bumps=1      coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          -12.9897        nan        nan        nan        nan        nan
X_Minimum        0.0441      0.050      0.879      0.379     -0.054      0.142
X_Maximum       -0.0436      0.050     -0.870      0.384     -0.142      0.055
Y_Minimum        0.1252      0.023      5.348      0.000      0.079      0.171
Y_Maximum       -0.1252      0.023     -5.348      0.000     -0.171     -0.079
Pixels_Areas    -0.0022      0.002     -1.414      0.157     -0.005      0.001
X_Perimeter     -0.0373      0.011     -3.288      0.001     -0.060     -0.015
Y_Perimeter      0.0970      0.021      4.560      0.000      0.055      0.139
Sum_of_Luminosity  1.582e-05  1.41e-05      1.122      0.262  -1.18e-05  4.34e-05
Minimum_of_Luminosity  0.0172      0.010      1.734      0.083     -0.002      0.037
Maximum_of_Luminosity  0.0315      0.012      2.521      0.012      0.007      0.056
Length_of_Conveyer  0.0013      0.001      1.846      0.065  -7.93e-05      0.003
```

```
    # confusion matrix of c
    # model_bump.pred_table
    pred = np.array(model_b
    table = np.histogram2d(
    table
```

✓ 0.2s

```
array([[1451.,   88.],
       [ 229.,  173.]])
```

TypeOfSteel_A300	-6.0621	nan	nan	nan	nan	nan
TypeOfSteel_A400	-6.9937	nan	nan	nan	nan	nan
Steel_Plate_Thickness	-0.0051	0.002	-3.208	0.001	-0.008	-0.002
Edges_Index	1.1659	0.235	4.960	0.000	0.705	1.627
Empty_Index	-3.2953	3.045	-1.082	0.279	-9.264	2.673
Square_Index	1.1437	0.369	3.097	0.002	0.420	1.868
Outside_X_Index	43.4036	66.951	0.648	0.517	-87.818	174.625
Edges_X_Index	-1.8628	0.744	-2.503	0.012	-3.321	-0.404
Edges_Y_Index	5.1532	1.001	5.147	0.000	3.191	7.116
Outside_Global_Index	0.4553	0.331	1.375	0.169	-0.194	1.104
LogOfAreas	-1.9522	3.785	-0.516	0.606	-9.371	5.467
Log_X_Index	6.2231	4.181	1.488	0.137	-1.971	14.418
Log_Y_Index	4.2860	4.020	1.066	0.286	-3.592	12.164
Orientation_Index	-1.2111	1.149	-1.054	0.292	-3.464	1.042
Luminosity_Index	-6.8484	2.574	-2.661	0.008	-11.894	-1.803
SigmoidOfAreas	-0.3590	0.760	-0.472	0.637	-1.848	1.130

2. Scratch 刮痕=>1，良品=>0

MDS_Assignment2 > Q5_Logit_ScatchModel.txt

```
Optimization terminated successfully.
         Current function value: 0.055606
         Iterations 16
                          MNLogit Regression Results
==============================================================================
Dep. Variable:              K_Scatch   No. Observations:                 1941
Model:                       MNLogit   Df Residuals:                     1914
Method:                         MLE   Df Model:                           26
Date:                Sun, 23 Oct 2022   Pseudo R-squ.:                0.8893
Time:                        10:40:10   Log-Likelihood:                -107.93
converged:                      True   LL-Null:                       -975.15
Covariance Type:            nonrobust   LLR p-value:                    0.000
==============================================================================
```

K_Scatch=1	coef	std err	z	P>|z|	[0.025	0.975]
const	8.0169	nan	nan	nan	nan	nan
X_Minimum	-0.0646	0.061	-1.067	0.286	-0.183	0.054
X_Maximum	0.0632	0.061	1.044	0.296	-0.055	0.182
Y_Minimum	-0.0162	0.013	-1.226	0.220	-0.042	0.010
Y_Maximum	0.0162	0.013	1.226	0.220	-0.010	0.042
Pixels_Areas	0.0017	0.001	2.788	0.005	0.001	0.003
X_Perimeter	0.0300	0.008	3.944	0.000	0.015	0.045
Y_Perimeter	-0.0373	0.017	-2.236	0.025	-0.070	-0.005
Sum_of_Luminosity	-1.682e-05	6.27e-06	-2.681	0.007	-2.91e-05	-4.52e-06
Minimum_of_Luminosity	-0.0248	0.012	-2.040	0.041	-0.049	-0.001
Maximum_of_Luminosity	0.0005	0.019	0.029	0.977	-0.036	0.037
Length_of_Conveyer	-0.0018	0.004	-0.404	0.686	-0.010	0.007

```
# confusion matrix of cla
model_scatch.pred_table()
```

✓ 0.1s

```
array([[1536.,   14.],
       [  25.,  366.]])
```

TypeOfSteel_A300	4.7089	nan	nan	nan	nan	nan
TypeOfSteel_A400	3.3064	nan	nan	nan	nan	nan
Steel_Plate_Thickness	-0.2575	0.097	-2.658	0.008	-0.447	-0.068
Edges_Index	-2.6033	0.964	-2.701	0.007	-4.492	-0.714
Empty_Index	-13.2467	4.768	-2.778	0.005	-22.591	-3.902
Square_Index	2.9832	0.924	3.229	0.001	1.172	4.794
Outside_X_Index	-95.5002	83.009	-1.150	0.250	-258.194	67.194
Edges_X_Index	7.3119	2.017	3.625	0.000	3.358	11.266
Edges_Y_Index	-5.4754	2.564	-2.136	0.033	-10.500	-0.451
Outside_Global_Index	2.8886	0.924	3.126	0.002	1.077	4.700
LogOfAreas	-2.9010	4.454	-0.651	0.515	-11.631	5.829
Log_X_Index	-8.3077	5.612	-1.480	0.139	-19.307	2.692
Log_Y_Index	16.4139	4.719	3.478	0.001	7.164	25.664
Orientation_Index	-10.5780	2.156	-4.906	0.000	-14.804	-6.352
Luminosity_Index	15.1091	3.379	4.472	0.000	8.486	21.732
SigmoidOfAreas	3.5560	1.547	2.299	0.022	0.524	6.588

由上結果可看出Outside_X_Index特徵在Bump模型的coef是43.40，表示強烈的正相關影響力；而在Scatch模型的coef是-95.50，表示強烈的負相關。因此Outside_X_Index特徵可有效鑑別兩種不良品。

(b) 基於上述(a)的結果，將上述特徵以t value進行排序後，哪些特徵的迴歸係數在統計上是顯著的呢(p-value<0.01)？

```
p_values_bump = model_bump.pvalues
p_values_bump = p_values_bump[p_values_bump[0] < 0.01]
p_values_bump.sort_values(by=0)
```

✓ 0.1s

	0
Y_Minimum	8.905653e-08
Y_Maximum	8.905915e-08
Edges_Y_Index	2.647692e-07
Edges_Index	7.053109e-07
Y_Perimeter	5.118999e-06
X_Perimeter	1.010214e-03
Steel_Plate_Thickness	1.335119e-03
Square_Index	1.956418e-03
Luminosity_Index	7.801371e-03

```
p_values_scatch = model_scatch.pvalues
p_values_scatch = p_values_scatch[p_values_scatch[0] < 0.01]
p_values_scatch.sort_values(by=0)
```

✓ 0.2s

	0
Orientation_Index	9.283976e-07
Luminosity_Index	7.765839e-06
X_Perimeter	8.019708e-05
Edges_X_Index	2.894543e-04
Log_Y_Index	5.051630e-04
Square_Index	1.242161e-03
Outside_Global_Index	1.772720e-03
Pixels_Areas	5.305806e-03
Empty_Index	5.463242e-03
Edges_Index	6.909960e-03
Sum_of_Luminosity	7.335425e-03
Steel_Plate_Thickness	7.864281e-03

Scatch模型與各種index較為相關。(右圖)
也可以看到Bump模型對於與Y有關的特徵，和與Edge有關的特徵較相關。(左圖)

(c) 試問配適一個羅吉斯迴歸模型是否合適？試若配適不佳，試說明其可能的原因為何？
若照上題各為了一個不良品特徵建模，則結果合適。
綜合Scatch模型的Pseudo R Square為0.8893，和兩模型的confusion matrix來看，Logistic regression表現不錯。

(d) 試問配適一個線性判別分析模型是否合適？若配適不佳，試說明其可能的原因為何？
因為判別分析是生成模型，建構各個類別的機率分布(如Naïve Bayes)，再形成一個分類器。生成模型不像判別模型會受到二元分類前後順序的相依性影響。在多類別的情況下(這題class = 3)，生成模型較有優勢。

我們先把資料轉成3個類別，有別於Q1~Q3的二元類別。

```
# 生成模型可一次預測多類別，將資料做成多類別的
# 0: 正常無缺陷, 1: Bumps, 2: K_Scatch
K_Scatch_ids = df.index[df['K_Scatch'] == 1].tolist()
Bumps_ids = df.index[df['Bumps'] == 1].tolist()
y_true = [0]*len(df)

for i in Bumps_ids:
    y_true[i] = 1
for i in K_Scatch_ids:
    y_true[i] = 2

from collections import Counter
Counter(y_true)
```

✓ 0.5s Python

```
Counter({0: 1148, 2: 391, 1: 402})
```

建立一個線性判別分析模型，可以看到預測 Normal 和 KScatch 的準確度不錯。

```python
import numpy as np
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
clf = LinearDiscriminantAnalysis()
clf.fit(x, y_true)
```

Python

```python
from sklearn.metrics import classification_report
y_pred = clf.predict(x)
report = classification_report(y_true, y_pred, target_names=['Noraml','Bumps','K_Scatch'])
print(report)
```

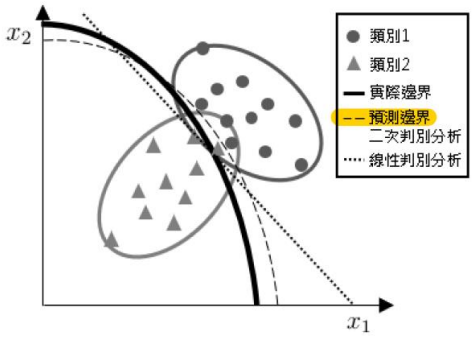
✓ 0.1s Python

```
              precision    recall  f1-score   support

      Noraml       0.80      0.89      0.84      1148
       Bumps       0.61      0.48      0.54       402
    K_Scatch       0.96      0.84      0.90       391

    accuracy                           0.79      1941
   macro avg       0.79      0.74      0.76      1941
weighted avg       0.79      0.79      0.79      1941
```

(e) 試問配適一個二次判別分析模型是否合適？若配適不佳，試說明其可能的原因為何？

依照講義p49頁的敘述，二次判別分析會建立了這個假設，分別估計類別間各自多變量常態分配的共變異矩陣，表現應比線性模型來得準確。(bias變精確，variance變低)

類效果不佳。若使用二次判別分析的結果如圖中虛線曲線，與真實的決策邊界較為相近。



可以看到二次判別分析模型，分辨 Normal 和 KScatch 更優秀(precision 0.92, 0.99)。也注意到sklearn警告我們資料集具有共線性，也許是造成bumps被overfit的原因，其recall為0.94。

```python
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
clf = QuadraticDiscriminantAnalysis()
clf.fit(x, y_true)

from sklearn.metrics import classification_report
y_pred = clf.predict(x)
report = classification_report(y_true, y_pred, target_names=['Noraml','Bumps','K_Scatch'])
print(report)
```

✓ 0.1s Python

```
              precision    recall  f1-score   support

      Noraml       0.92      0.43      0.59      1148
       Bumps       0.36      0.94      0.52       402
    K_Scatch       0.99      0.91      0.95       391

    accuracy                           0.63      1941
   macro avg       0.76      0.76      0.68      1941
weighted avg       0.82      0.63      0.64      1941


c:\Users\luweb\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\discriminant_analysis.py:887:
UserWarning: Variables are collinear
```