一、 資料預處理

1. 匯入資料

1.1 匯入資料

In [225]:
```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder

df = pd.read_csv('student-mat.csv',sep = ';')
df
```

executed in 49ms, finished 22:03:48 2020-12-19

Out[225]:

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famrel	freetime	goout	Dalc	Walc	health	absences	G1	G2	G3
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	4	3	4	1	1	3	6	5	6	6
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	5	3	3	1	1	3	4	5	5	6
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	4	3	2	2	3	3	10	7	8	10
3	GP	F	15	U	GT3	T	4	2	health	services	...	3	2	2	1	1	5	2	15	14	15
4	GP	F	16	U	GT3	T	3	3	other	other	...	4	3	2	1	2	5	4	6	10	10
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
390	MS	M	20	U	LE3	A	2	2	services	services	...	5	5	4	4	5	4	11	9	9	9
391	MS	M	17	U	LE3	T	3	1	services	services	...	2	4	5	3	4	2	3	14	16	16
392	MS	M	21	R	GT3	T	1	1	other	other	...	5	5	3	3	3	3	3	10	8	7
393	MS	M	18	R	LE3	T	3	2	services	other	...	4	4	1	3	4	5	0	11	12	10
394	MS	M	19	U	LE3	T	1	1	other	at_home	...	3	2	3	3	3	5	5	8	9	9

395 rows × 33 columns

2. 需要 One-hot encoding 的類別行欄位整理

1.2 需要one-hot encoding的類別型欄位

In [226]:
```
print(set(df['school']))
print(set(df['sex']))
print(set(df['address']))
print(set(df['famsize']))
print(set(df['Pstatus']))
print(set(df['Mjob'])) ## nominal
print(set(df['Fjob'])) ## nominal
print(set(df['reason'])) ## nominal
print(set(df['guardian'])) ## nominal
print(set(df['schoolsup']))
print(set(df['famsup']))
print(set(df['paid']))
print(set(df['activities']))
print(set(df['nursery']))
print(set(df['higher']))
print(set(df['internet']))
print(set(df['romantic']))
```

executed in 18ms, finished 22:03:48 2020-12-19

```
{'GP', 'MS'}
{'M', 'F'}
{'R', 'U'}
{'GT3', 'LE3'}
{'T', 'A'}
{'health', 'other', 'at_home', 'services', 'teacher'}
{'health', 'other', 'at_home', 'services', 'teacher'}
{'other', 'home', 'course', 'reputation'}
{'other', 'father', 'mother'}
{'yes', 'no'}
{'yes', 'no'}
{'yes', 'no'}
{'yes', 'no'}
{'yes', 'no'}
{'yes', 'no'}
{'yes', 'no'}
{'yes', 'no'}
```

1.3 將需要做onehot encoding的欄位做成一表格

In [227]:
```
nominal_df = df[['school','sex','address','famsize','Pstatus','schoolsup','famsup','paid','activities','nursery',
                 'higher','internet','romantic','Mjob','Fjob','reason','guardian']]
nominal_df
```

executed in 37ms, finished 22:03:48 2020-12-19

Out[227]:

	school	sex	address	famsize	Pstatus	schoolsup	famsup	paid	activities	nursery	higher	internet	romantic	Mjob	Fjob	reason	guardian
0	GP	F	U	GT3	A	yes	no	no	no	yes	yes	no	no	at_home	teacher	course	mother
1	GP	F	U	GT3	T	no	yes	no	no	no	yes	yes	no	at_home	other	course	father
2	GP	F	U	LE3	T	yes	no	yes	no	yes	yes	yes	no	at_home	other	other	mother
3	GP	F	U	GT3	T	no	yes	yes	yes	yes	yes	yes	yes	health	services	home	mother
4	GP	F	U	GT3	T	no	yes	yes	no	yes	yes	no	no	other	other	home	father
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
390	MS	M	U	LE3	A	no	yes	yes	no	yes	yes	no	no	services	services	course	other
391	MS	M	U	LE3	T	no	no	no	no	no	yes	yes	no	services	services	course	mother
392	MS	M	R	GT3	T	no	no	no	no	no	yes	no	no	other	other	course	other
393	MS	M	R	LE3	T	no	no	no	no	no	yes	yes	no	services	other	course	mother
394	MS	M	U	LE3	T	no	no	no	no	yes	yes	yes	no	other	at_home	course	father

395 rows × 17 columns

1.4 執行one hot encoding

In [228]:
```
data_dummies = pd.get_dummies(nominal_df)#, drop_first = True)
data_dummies
```

executed in 76ms, finished 22:03:49 2020-12-19

Out[228]:

	school_GP	school_MS	sex_F	sex_M	address_R	address_U	famsize_GT3	famsize_LE3	Pstatus_A	Pstatus_T	...	Fjob_other	Fjob_services	Fjob_teach
0	1	0	1	0	0	1	1	0	1	0	...	0	0	
1	1	0	1	0	0	1	1	0	0	1	...	1	0	
2	1	0	1	0	0	1	0	1	0	1	...	1	0	
3	1	0	1	0	0	1	1	0	0	1	...	0	1	
4	1	0	1	0	0	1	1	0	0	1	...	1	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
390	0	1	0	1	0	1	0	1	1	0	...	0	1	
391	0	1	0	1	0	1	0	1	0	1	...	0	1	
392	0	1	0	1	1	0	1	0	0	1	...	1	0	
393	0	1	0	1	1	0	0	1	0	1	...	1	0	
394	0	1	0	1	0	1	0	1	0	1	...	0	0	

395 rows × 43 columns

3. 合併所有資料

1.5 將處理後的資料與原資料concat起來

In [229]:
```
df_new = pd.concat([df,data_dummies],axis=1)
df_new
```

executed in 57ms, finished 22:03:49 2020-12-19

Out[229]:

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	Fjob_other	Fjob_services	Fjob_teacher	reason_course	reason_home
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	0	0	1	1	0
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	1	0	0	1	0
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	1	0	0	0	0
3	GP	F	15	U	GT3	T	4	2	health	services	...	0	1	0	0	1
4	GP	F	16	U	GT3	T	3	3	other	other	...	1	0	0	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
390	MS	M	20	U	LE3	A	2	2	services	services	...	0	1	0	1	0
391	MS	M	17	U	LE3	T	3	1	services	services	...	0	1	0	1	0
392	MS	M	21	R	GT3	T	1	1	other	other	...	1	0	0	1	0
393	MS	M	18	R	LE3	T	3	2	services	other	...	1	0	0	1	0
394	MS	M	19	U	LE3	T	1	1	other	at_home	...	0	0	0	1	0

395 rows × 76 columns

4. 篩選要丟入模型的 X 與 y 欄位

1.6 整理成丟入模型需使用的X與y

In [230]:
```
df_new2 = df_new.drop(['school','sex','address','famsize','Pstatus','schoolsup','famsup','paid','activities','nur
                       'higher','internet','romantic','Mjob','Fjob','reason','guardian'], axis = 1)
df_new2
```
executed in 52ms, finished 22:03:50 2020-12-19

Out[230]:

	age	Medu	Fedu	traveltime	studytime	failures	famrel	freetime	goout	Dalc	...	Fjob_other	Fjob_services	Fjob_teacher	reason_course	reason_hom
0	18	4	4	2	2	0	4	3	4	1	...	0	0	1	1	
1	17	1	1	1	2	0	5	3	3	1	...	1	0	0	1	
2	15	1	1	1	2	3	4	3	2	2	...	1	0	0	0	
3	15	4	2	1	3	0	3	2	2	1	...	0	1	0	0	
4	16	3	3	1	2	0	4	3	2	1	...	1	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
390	20	2	2	1	2	2	5	5	4	4	...	0	1	0	1	
391	17	3	1	2	1	0	2	4	5	3	...	0	1	0	1	
392	21	1	1	1	1	3	5	5	3	3	...	1	0	0	1	
393	18	3	2	3	1	0	4	4	1	3	...	1	0	0	1	
394	19	1	1	1	1	0	3	2	3	3	...	0	0	0	1	

395 rows × 59 columns

In [231]:
```
df_x = df_new2.drop(["G3"], axis = 1)
df_x
```
executed in 32ms, finished 22:03:51 2020-12-19

Out[231]:

	age	Medu	Fedu	traveltime	studytime	failures	famrel	freetime	goout	Dalc	...	Fjob_other	Fjob_services	Fjob_teacher	reason_course	reason_hom
0	18	4	4	2	2	0	4	3	4	1	...	0	0	1	1	
1	17	1	1	1	2	0	5	3	3	1	...	1	0	0	1	
2	15	1	1	1	2	3	4	3	2	2	...	1	0	0	0	
3	15	4	2	1	3	0	3	2	2	1	...	0	1	0	0	
4	16	3	3	1	2	0	4	3	2	1	...	1	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
390	20	2	2	1	2	2	5	5	4	4	...	0	1	0	1	
391	17	3	1	2	1	0	2	4	5	3	...	0	1	0	1	
392	21	1	1	1	1	3	5	5	3	3	...	1	0	0	1	
393	18	3	2	3	1	0	4	4	1	3	...	1	0	0	1	
394	19	1	1	1	1	0	3	2	3	3	...	0	0	0	1	

395 rows × 58 columns

In [232]:
```
df_y = df_new2['G3']
df_y = pd.DataFrame(df_y)
df_y
```
executed in 19ms, finished 22:03:51 2020-12-19

Out[232]:

	G3
0	6
1	6
2	10

1.7 轉換丟入模型的值之型態

In [233]:
```
X = df_x.iloc[:,:].values
y = df_y.iloc[:,-1].values
```
executed in 6ms, finished 22:03:52 2020-12-19

二、 切割訓練&測試資料集，並將資料及標準化

→ train_test_split(X,y,test_size = 0.3, random_state= 3)

1.8 train test 切割

In [280]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,random_state=3)

sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)
```
executed in 28ms, finished 22:13:08 2020-12-19

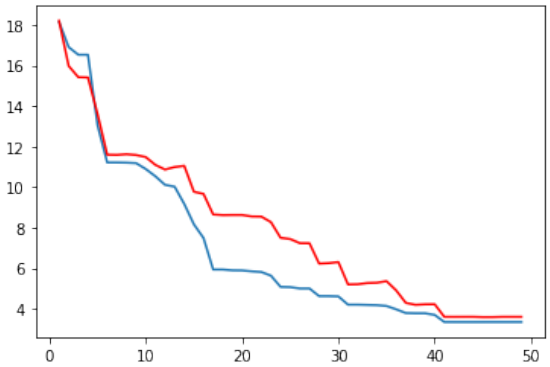
三、 PCA 降維 → 以 linear regression 為標準選取最佳維度

In [270]:

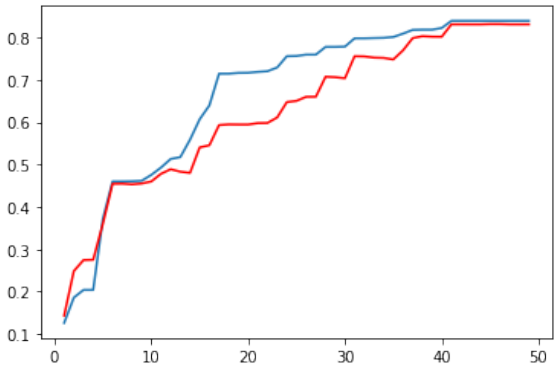
```
## PCA降維

from sklearn.decomposition import PCA

from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score

mse_scores_train = []
mse_scores_test = []
r2_scores_train = []
r2_scores_test = []
coef = []
times = [x for x in range(1,50)]

for i in times:

    pca = PCA(n_components = i)
    X_train_pca = pca.fit_transform(X_train_std)
    X_test_pca = pca.transform(X_test_std)

    slr = LinearRegression()

    slr.fit(X_train_pca,y_train)
    coef.append(slr.coef_)
    y_train_pred = slr.predict(X_train_pca)
    y_test_pred = slr.predict(X_test_pca)

    mse_scores_train.append(mean_squared_error(y_train,y_train_pred))
    mse_scores_test.append(mean_squared_error(y_test,y_test_pred))
    r2_scores_train.append(r2_score(y_train,y_train_pred))
    r2_scores_test.append(r2_score(y_test,y_test_pred))
```

executed in 239ms, finished 22:09:16 2020-12-19

→ MSE (train 為藍色，test 為紅色):



→ R2 (train 為藍色，test 為紅色):



→ 選取最佳 n_components = 43

四、 n_components=43 下 模型最佳結果

1. linear regression：

✓ **模型與參數設定：**

In [287]:
```
1 slr
```
executed in 15ms, finished 01:08:11 2020-12-20

Out[287]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

✓ **特徵 coef 與誤差指標：**

In [286]:
```
1  slr = LinearRegression()
2
3  slr.fit(X_train_pca,y_train)
4  print(slr.coef_)
5
6  y_train_pred = slr.predict(X_train_pca)
7  y_test_pred = slr.predict(X_test_pca)
8
9  print('MSE train: %.3f, test: %.3f'%(
10             mean_squared_error(y_train,y_train_pred),
11             mean_squared_error(y_test,y_test_pred)))
12 print('R^2 train: %.3f, test: %.3f'%(
13             r2_score(y_train,y_train_pred),
14             r2_score(y_test,y_test_pred)))
```
executed in 21ms, finished 01:08:06 2020-12-20

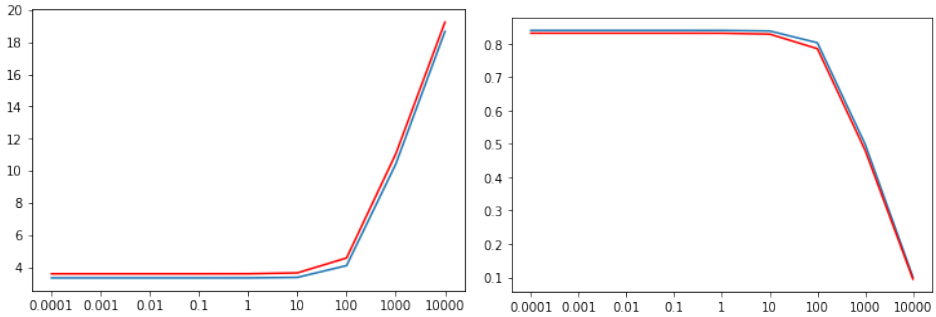
```
[-7.23303109e-01  5.92078494e-01  3.53591634e-01 -1.03462784e-02
 -1.11829403e+00 -8.52503155e-01 -1.73132565e-02 -5.76414966e-02
  1.08645955e-01  3.79415104e-01  4.41537498e-01  4.84176606e-01
  2.33082463e-01 -7.20205994e-01  8.09238286e-01  6.76578777e-01
  1.06157272e+00  3.32525535e-02  1.73931996e-01 -6.82317313e-02
  1.99488047e-01 -1.70579178e-01  4.10256092e-01 -7.58427263e-01
 -1.12198390e-01 -2.66497191e-01  6.44171963e-03 -6.90650968e-01
  3.26794071e-02  1.30411678e-01  7.49890900e-01  1.62749527e-02
  1.43526986e-01 -1.47805551e-01  2.81554760e-01 -5.97588964e-01
 -6.72892768e-01 -1.79450293e-01 -4.24730511e-02  6.29746176e-01
 -1.78454985e+00  1.35114495e+13  1.33687847e+13]
MSE train: 3.339, test: 3.595
R^2 train: 0.839, test: 0.831
```

2. Ridge regression

✓ **調參情況（gamma）：**

→ alpha = [0.0001,0.001,0.01,0.1,1,10,100,1000,10000]

→ **MSE（左圖）與 R2（右圖）折線圖 （train 為藍色，test 為紅色）：**



✓ **模型與參數設定：(alpha = 1)**

```
In [293]:  1  ridge
executed in 12ms, finished 01:10:58 2020-12-20

Out[293]: Ridge(alpha=1, copy_X=True, fit_intercept=True, max_iter=None, normalize=False,
      random_state=None, solver='auto', tol=0.001)
```

✓ **特徵 coef 與誤差指標：**

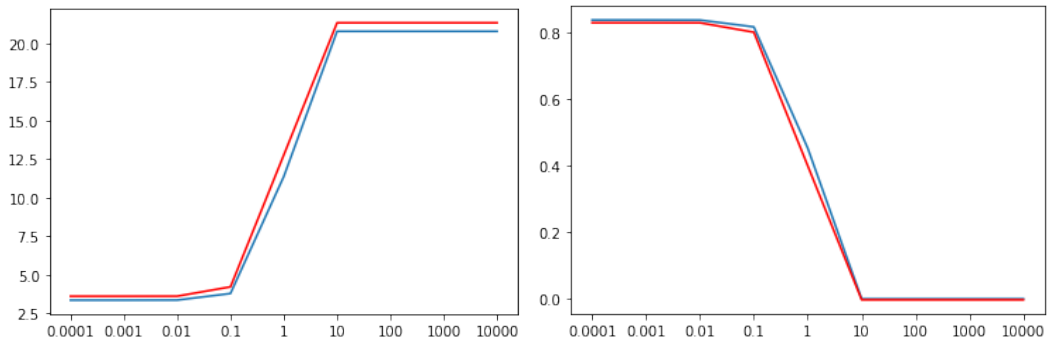
```
In [292]:
 1  ridge = Ridge(alpha=1).fit(X_train,y_train) ## alpha = 1
 2
 3  ridge.fit(X_train_pca,y_train)
 4  print(ridge.coef_)
 5
 6  y_train_pred = ridge.predict(X_train_pca)
 7  y_test_pred = ridge.predict(X_test_pca)
 8
 9  print('MSE train: %.3f, test: %.3f'%(
10              mean_squared_error(y_train,y_train_pred),
11              mean_squared_error(y_test,y_test_pred)))
12  print('R^2 train: %.3f, test: %.3f'%(
13              r2_score(y_train,y_train_pred),
14              r2_score(y_test,y_test_pred)))
executed in 22ms, finished 01:10:53 2020-12-20

[-7.22781136e-01  5.91473184e-01  3.53166326e-01 -1.03334937e-02
 -1.11684144e+00 -8.51280623e-01 -1.72872367e-02 -5.75525348e-02
  1.08452569e-01  3.78733825e-01  4.40670068e-01  4.83203209e-01
  2.32580106e-01 -7.18623851e-01  8.07332285e-01  6.74925417e-01
  1.05879147e+00  3.31619037e-02  1.73441653e-01 -6.80247543e-02
  1.98837902e-01 -1.70022373e-01  4.08886983e-01 -7.55589514e-01
 -1.11763969e-01 -2.65468089e-01  6.80536961e-03 -6.87494871e-01
  3.25672463e-02  1.29848702e-01  7.46162131e-01  1.61863353e-02
  1.42696781e-01 -1.46829081e-01  2.79547989e-01 -5.92946338e-01
 -6.66915192e-01 -1.77620369e-01 -4.19379512e-02  6.18730891e-01
 -1.72770876e+00  1.69372204e-14 -1.72682162e-15]
MSE train: 3.340, test: 3.597
R^2 train: 0.839, test: 0.831
```

3. Lasso Regression

✓ **調參情況（alpha）：**

→ alpha = [0.0001,0.001,0.01,0.1,1,10,100,1000,10000]

→ **MSE(左圖) 與 R2（右圖）折線圖 （train 為藍色，test 為紅色）：**



✓ **模型與參數設定：(alpha = 0.01)**

```
In [299]:  1  lasso
```
executed in 11ms, finished 01:15:50 2020-12-20

```
Out[299]: Lasso(alpha=0.01, copy_X=True, fit_intercept=True, max_iter=1000,
      normalize=False, positive=False, precompute=False, random_state=None,
      selection='cyclic', tol=0.0001, warm_start=False)
```

✓ **特徵 coef 與誤差指標：**

```
In [298]:  1  lasso = Lasso(alpha=0.01).fit(X_train,y_train)
           2
           3  lasso.fit(X_train_pca,y_train)
           4  print(lasso.coef_)
           5
           6  y_train_pred = lasso.predict(X_train_pca)
           7  y_test_pred = lasso.predict(X_test_pca)
           8
           9  print('MSE train: %.3f, test: %.3f'%(
          10              mean_squared_error(y_train,y_train_pred),
          11              mean_squared_error(y_test,y_test_pred)))
          12  print('R^2 train: %.3f, test: %.3f'%(
          13              r2_score(y_train,y_train_pred),
          14              r2_score(y_test,y_test_pred)))
```
executed in 33ms, finished 01:15:44 2020-12-20

```
[-7.21309912e-01  5.89253927e-01  3.50267845e-01 -6.93157345e-03
 -1.11470431e+00 -8.48539493e-01 -1.31590497e-02 -5.33752272e-02
  1.03724468e-01  3.74450320e-01  4.36104617e-01  4.78616673e-01
  2.27121060e-01 -7.14129496e-01  8.02722289e-01  6.69817666e-01
  1.05432275e+00  2.56635695e-02  1.66127807e-01 -5.98310350e-02
  1.90502048e-01 -1.61457513e-01  4.01001946e-01 -7.48052913e-01
 -1.01677224e-01 -2.55692203e-01  0.00000000e+00 -6.78025182e-01
  1.98378398e-02  1.17386668e-01  7.36090269e-01  1.23703443e-03
  1.27474333e-01 -1.29452291e-01  2.61741993e-01 -5.75978823e-01
 -6.48154825e-01 -1.51015548e-01 -7.25732365e-03  5.80609813e-01
 -1.69374669e+00  0.00000000e+00 -0.00000000e+00]
MSE train: 3.345, test: 3.598
R^2 train: 0.839, test: 0.831
```

我的分析與比較

模型	參數設定	MSE	R2	各特徵相關性（參數）
linear	預設	Train: 3.339 Test: 3.595	Train: 0.839 Test: 0.831	[-7.23303109e-01 5.92078494e-01 3.53591634e-01 -1.03462784e-02 -1.11829403e+00 -8.52503155e-01 -1.73132565e-02 -5.76414966e-02 1.08645955e-01 3.79415104e-01 4.41537498e-01 4.84176606e-01 2.33082463e-01 -7.20205994e-01 8.09238286e-01 6.76578777e-01 1.06157272e+00 3.32525535e-02 1.73931996e-01 -6.82317313e-02 1.99488047e-01 -1.70579178e-01 4.10256092e-01 -7.58427263e-01 -1.12198390e-01 -2.66497191e-01 6.44171963e-03 -6.90650968e-01 3.26794071e-02 1.30411678e-01 7.49890900e-01 1.62749527e-02 1.43526986e-01 -1.47805551e-01 2.81554760e-01 -5.97588964e-01 -6.72892768e-01 -1.79450293e-01 -4.24730511e-02 6.29746176e-01 -1.78454985e+00 1.35114495e+13 1.33687847e+13]
Ridge	alpha = 1	Train: 3.340 Test: 3.597	Train: 0. 839 Test: 0. 831	[-7.22781136e-01 5.91473184e-01 3.53166326e-01 -1.03334937e-02 -1.11684144e+00 -8.51280623e-01 -1.72872367e-02 -5.75525348e-02 1.08452569e-01 3.78733825e-01 4.40670068e-01 4.83203209e-01 2.32580106e-01 -7.18623851e-01 8.07332285e-01 6.74925417e-01 1.05879147e+00 3.31619037e-02 1.73441653e-01 -6.80247543e-02 1.98837902e-01 -1.70022373e-01 4.08886983e-01 -7.55589514e-01 -1.11763969e-01 -2.65468089e-01 6.80536961e-03 -6.87494871e-01 3.25672463e-02 1.29848702e-01 7.46162131e-01 1.61863353e-02 1.42696781e-01 -1.46829081e-01 2.79547989e-01 -5.92946338e-01 -6.66915192e-01 -1.77620369e-01 -4.19379512e-02 6.18730891e-01 -1.72770876e+00 1.69372204e-14 -1.72682162e-15]
Lasso	alpha = 0.01	Train: 3.345 Test: 3.598	Train: 0.839 Test: 0.831	[-7.21309912e-01 5.89253927e-01 3.50267845e-01 -6.93157345e-03 -1.11470431e+00 -8.48539493e-01 -1.31590497e-02 -5.33752272e-02 1.03724468e-01 3.74450320e-01 4.36104617e-01 4.78616673e-01 2.27121060e-01 -7.14129496e-01 8.02722289e-01 6.69817666e-01 1.05432275e+00 2.56635695e-02 1.66127807e-01 -5.98310350e-02 1.90502048e-01 -1.61457513e-01 4.01001946e-01 -7.48052913e-01 -1.01677224e-01 -2.55692203e-01 0.00000000e+00 -6.78025182e-01 1.98378398e-02 1.17386668e-01 7.36090269e-01 1.23703443e-03 1.27474333e-01 -1.29452291e-01 2.61741993e-01 -5.75978823e-01 -6.48154825e-01 -1.51015548e-01 -7.25732365e-03 5.80609813e-01 -1.69374669e+00 0.00000000e+00 -0.00000000e+00]

- **（加分題）有關於共線性的問題？**

→分析：

整理結果後發現，基本上三者回歸模型之評估指標表現都相差不大，但模型卻是各有不同的特點：

1. linear regression：這就是我們一般聽到的線性迴歸模型，它考慮了所有的特徵與預測結果的相關性，因此模型也很容易受到離群值的影響而有誤差。另外，多個特徵丟入模型也可能容易造成 overfitting 或是模型效能不佳的結果，就是我們一般所說的「共線性」的問題。
2. Ridge regression：這是為了解決共線性所延伸出來的一種方法，它使用 L2 正規化，避免模型複雜度太高以及過度配適的問題，適合較低維稠密的特

徵。在選取特徵時，會將影響預測成果相關性越大的特徵之相關係數調整愈大，反之，則愈小，因此，通常在做模型時會首選 Ridge 而非 Lasso。

3. Lasso regression：這是為了解決共線性所延伸出來的一種方法，它使用 L1 正規化，避免模型過度配適的問題，適合較高維稀疏的特徵，例如特徵很多時卻只有一小部分是真正重要的，則選擇此方法建模。在選取特徵時，會使某些係數變為 0，因此也可能影響模型預測正確性。但 Lasso 也有較好地理解性，因為使用較少特徵，也較好解釋模型。

→ 結論：

從這次的建模成果可以看到，三者的預測準確程度都差不多，且從相關係數也可以觀察到特徵之間存在著共線性的關係。例如：Lasso 將一些較不重要的特徵之係數轉為 0，卻還是對預測結果未有什麼影響。但值得注意的是，我們觀察了只會將不重要特徵之權重調低的 Ridge 方法，它採取的特徵雖較 Lasso 來得多與雜，卻有比 Lasso 來得稍微好一點點的表現，此點也印證了上面我們對模型方法所做的分析。