CBE109040 蔣幸容

資料集合： insurance.csv

資料特徵欄位說明：

自變量

age：被保險人年紀，連續型欄位

sex：被保險人性別，分類型欄位

bmi：被保險人身體質量指數，連續型欄位

children：被保險人子女人數，連續型欄位

smoker：被保險人是否抽菸，分類型欄位

region：被保險人所在地區，分類型欄位

---

應變量

charges：被保險人保險費用，連續型欄位

若資料儲存格為空格，表示該資料缺失

分析需求說明

1. 請採用多元線性回歸進行分析，以被保險人的年紀、性別、身體質量指數、子女人數、是否抽菸等特徵預測被保險人的保險費用。並採用反向淘汰方法挑選出適合於本次需求分析的多元線性回歸模型的被保險人特徵。

回答區

1. 請將反向淘汰過程中每一次的 summary 表格呈現於此，並說明反向淘汰過程，如何判斷哪些被保險人特徵被淘汰。

比較 P-value，若 P-value > 0.05，則刪除

先刪除最大的 P-value(為逐步淘汰步驟)

```
x_train = np.append(arr = np.ones((1070,1)).astype(int), values = x_train, axis = 1)
x_opt = x_train[:, [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]]
x_opt = np.array(x_opt, dtype=float)
regressor_OLS = sm.OLS(endog = y_train, exog = x_opt).fit()
regressor_OLS.summary()
```

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| const | -262.7259 | 631.239 | -0.416 | 0.677 | -1501.344 | 975.892 |
| x1 | -24.9517 | 379.157 | -0.066 | 0.948 | -768.934 | 719.031 |
| x2 | -1.193e+04 | 377.713 | -31.576 | 0.000 | -1.27e+04 | -1.12e+04 |
| x3 | 1.166e+04 | 409.727 | 28.467 | 0.000 | 1.09e+04 | 1.25e+04 |
| x4 | 410.1389 | 360.497 | 1.138 | 0.256 | -297.230 | 1117.508 |
| x5 | 193.0088 | 353.720 | 0.546 | 0.585 | -501.061 | 887.079 |
| x6 | -510.0725 | 399.811 | -1.276 | 0.202 | -1294.582 | 274.437 |
| x7 | -355.8011 | 361.938 | -0.983 | 0.326 | -1065.996 | 354.394 |
| x8 | 251.7084 | 13.568 | 18.552 | 0.000 | 225.085 | 278.331 |
| x9 | 337.3665 | 32.326 | 10.436 | 0.000 | 273.936 | 400.797 |
| x10 | 434.1064 | 157.033 | 2.764 | 0.006 | 125.977 | 742.236 |

I. 删除 x1(x_train[:, [1]])

```
x_opt = x_train[:, [0, 2, 3, 4, 5, 6, 7, 8, 9, 10]]
x_opt = np.array(x_opt, dtype=float)
regressor_OLS = sm.OLS(endog = y_train, exog = x_opt).fit()
regressor_OLS.summary()
```

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| const | -269.1933 | 623.249 | -0.432 | 0.666 | -1492.133 | 953.747 |
| x1 | -1.193e+04 | 375.635 | -31.757 | 0.000 | -1.27e+04 | -1.12e+04 |
| x2 | 1.166e+04 | 405.058 | 28.786 | 0.000 | 1.09e+04 | 1.25e+04 |
| x3 | 408.4649 | 359.430 | 1.136 | 0.256 | -296.809 | 1113.739 |
| x4 | 191.7487 | 353.036 | 0.543 | 0.587 | -500.978 | 884.475 |
| x5 | -512.2530 | 398.248 | -1.286 | 0.199 | -1293.696 | 269.190 |
| x6 | -357.1538 | 361.184 | -0.989 | 0.323 | -1065.869 | 351.561 |
| x7 | 251.7289 | 13.558 | 18.567 | 0.000 | 225.125 | 278.332 |
| x8 | 337.3051 | 32.297 | 10.444 | 0.000 | 273.931 | 400.679 |
| x9 | 433.8203 | 156.899 | 2.765 | 0.006 | 125.954 | 741.687 |

II. 删除 const(x_train[:, [0]])

```
x_opt = x_train[:, [2, 3, 4, 5, 6, 7, 8, 9, 10]]
x_opt = np.array(x_opt, dtype=float)
regressor_OLS = sm.OLS(endog = y_train, exog = x_opt).fit()
regressor_OLS.summary()
```

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| x1 | -1.211e+04 | 746.584 | -16.219 | 0.000 | -1.36e+04 | -1.06e+04 |
| x2 | 1.148e+04 | 781.656 | 14.687 | 0.000 | 9946.664 | 1.3e+04 |
| x3 | 318.7338 | 473.482 | 0.673 | 0.501 | -610.332 | 1247.799 |
| x4 | 102.0176 | 463.662 | 0.220 | 0.826 | -807.780 | 1011.815 |
| x5 | -601.9841 | 545.607 | -1.103 | 0.270 | -1672.575 | 468.607 |
| x6 | -446.8849 | 487.468 | -0.917 | 0.359 | -1403.394 | 509.624 |
| x7 | 251.7289 | 13.558 | 18.567 | 0.000 | 225.125 | 278.332 |
| x8 | 337.3051 | 32.297 | 10.444 | 0.000 | 273.931 | 400.679 |
| x9 | 433.8203 | 156.899 | 2.765 | 0.006 | 125.954 | 741.687 |

III. 删除 x4(x_train[:, [5]])

```python
x_opt = x_train[:, [2, 3, 4, 6, 7, 8, 9, 10]]
x_opt = np.array(x_opt, dtype=float)
regressor_OLS = sm.OLS(endog = y_train, exog = x_opt).fit()
regressor_OLS.summary()
```

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| x1 | -1.201e+04 | 1105.513 | -10.861 | 0.000 | -1.42e+04 | -9837.291 |
| x2 | 1.158e+04 | 1144.873 | 10.117 | 0.000 | 9335.976 | 1.38e+04 |
| x3 | 216.7162 | 551.436 | 0.393 | 0.694 | -865.312 | 1298.745 |
| x4 | -704.0017 | 550.177 | -1.280 | 0.201 | -1783.559 | 375.556 |
| x5 | -548.9025 | 541.364 | -1.014 | 0.311 | -1611.167 | 513.362 |
| x6 | 251.7289 | 13.558 | 18.567 | 0.000 | 225.125 | 278.332 |
| x7 | 337.3051 | 32.297 | 10.444 | 0.000 | 273.931 | 400.679 |
| x8 | 433.8203 | 156.899 | 2.765 | 0.006 | 125.954 | 741.687 |

IV. 删除 x3(x_train[:, [4]])

```python
x_opt = x_train[:, [2, 3, 6, 7, 8, 9, 10]]
x_opt = np.array(x_opt, dtype=float)
regressor_OLS = sm.OLS(endog = y_train, exog = x_opt).fit()
regressor_OLS.summary()
```

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| x1 | -1.191e+04 | 1077.051 | -11.057 | 0.000 | -1.4e+04 | -9795.921 |
| x2 | 1.168e+04 | 1115.009 | 10.479 | 0.000 | 9495.923 | 1.39e+04 |
| x3 | -812.0227 | 476.409 | -1.704 | 0.089 | -1746.832 | 122.787 |
| x4 | -656.0713 | 467.483 | -1.403 | 0.161 | -1573.365 | 261.223 |
| x5 | 251.8756 | 13.547 | 18.592 | 0.000 | 225.293 | 278.458 |
| x6 | 337.4341 | 32.283 | 10.452 | 0.000 | 274.089 | 400.780 |
| x7 | 433.5323 | 156.835 | 2.764 | 0.006 | 125.792 | 741.273 |

V. 删除 x4(x_train[:, [7]])

```python
x_opt = x_train[:, [2, 3, 6, 8, 9, 10]]
x_opt = np.array(x_opt, dtype=float)
regressor_OLS = sm.OLS(endog = y_train, exog = x_opt).fit()
regressor_OLS.summary()
```

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| x1 | -1.203e+04 | 1073.926 | -11.205 | 0.000 | -1.41e+04 | -9925.765 |
| x2 | 1.159e+04 | 1113.346 | 10.407 | 0.000 | 9401.605 | 1.38e+04 |
| x3 | -568.0117 | 443.750 | -1.280 | 0.201 | -1438.737 | 302.714 |
| x4 | 252.4123 | 13.548 | 18.631 | 0.000 | 225.828 | 278.997 |
| x5 | 333.4455 | 32.172 | 10.364 | 0.000 | 270.317 | 396.574 |
| x6 | 420.1346 | 156.615 | 2.683 | 0.007 | 112.825 | 727.444 |

VI. 删除 x3(x_train[:, [6]])

```python
x_opt = x_train[:, [2, 3, 8, 9, 10]]
x_opt = np.array(x_opt, dtype=float)
regressor_OLS = sm.OLS(endog = y_train, exog = x_opt).fit()
regressor_OLS.summary()
```

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| x1 | -1.184e+04 | 1063.627 | -11.132 | 0.000 | -1.39e+04 | -9753.151 |
| x2 | 1.173e+04 | 1107.945 | 10.588 | 0.000 | 9556.639 | 1.39e+04 |
| x3 | 252.9818 | 13.545 | 18.677 | 0.000 | 226.404 | 279.560 |
| x4 | 321.4421 | 30.784 | 10.442 | 0.000 | 261.037 | 381.847 |
| x5 | 427.2585 | 156.563 | 2.729 | $\boxed{0.006}$ | 120.051 | 734.466 |

VII. 刪除 x5(x_train[:, [10]])

```
x_opt = x_train[:, [2, 3, 8, 9]]
x_opt = np.array(x_opt, dtype=float)
regressor_OLS = sm.OLS(endog = y_train, exog = x_opt).fit()
regressor_OLS.summary()
```

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| x1 | -1.15e+04 | 1059.668 | -10.856 | 0.000 | -1.36e+04 | -9424.945 |
| x2 | 1.21e+04 | 1102.765 | 10.976 | 0.000 | 9940.609 | 1.43e+04 |
| x3 | 255.0359 | 13.565 | 18.801 | 0.000 | 228.419 | 281.653 |
| x4 | 322.6407 | 30.874 | 10.450 | 0.000 | 262.059 | 383.222 |

P-value 皆小於 0.05，反向淘汰完畢

2. 請將完成的程式碼全選複製貼上於此，並於程式碼中加入適當註解。

```
# Importing the Libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the Dataset
dataset = pd.read_csv("insurance.csv")
x = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 6].values

dataset.info() #檢查哪裡有缺失資料

# Missing Data
from sklearn.impute import SimpleImputer

#age 和 bmi 的缺失資料以平均值填入
imputer = SimpleImputer(missing_values=np.nan, strategy="mean", fill_value=None)
imputer = imputer.fit(x[:, [0, 2]])
x[:, [0, 2]] = imputer.transform(x[:, [0, 2]])

#sex、children 和 smoker 的缺失資料以最常出現的值填入
imputer = SimpleImputer(missing_values=np.nan, strategy="most_frequent", fill_value=None)
imputer = imputer.fit(x[:, [1, 3, 4]])
x[:, [1, 3, 4]] = imputer.transform(x[:, [1, 3, 4]])
```

```python

#charges 的缺失資料以平均值填入
y = np.reshape(y, (-1, 1))
imputer = SimpleImputer(missing_values=np.nan, strategy="mean", fill_value=None)
imputer = imputer.fit(y[:, :])
y[:, :] = imputer.transform(y[:, :])

# Categorical Data
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.compose import ColumnTransformer

#sex、smoker 和 region 為分類型欄位，需做標籤編碼與虛擬變量
labelencorder_x = LabelEncoder()
x[:, 1] = labelencorder_x.fit_transform(x[:, 1])
x[:, 4] = labelencorder_x.fit_transform(x[:, 4])
x[:, 5] = labelencorder_x.fit_transform(x[:, 5])

ct = ColumnTransformer([("sex", OneHotEncoder(), [1]), ("smoker", OneHotEncoder(), [4]), ("region", OneHotEncoder(), [5])] , remainder="passthrough")
X = ct.fit_transform(x)

#將虛擬變量的其中一行刪除，因為會違反無多重共線性
X = X[:,1:]

# Splitting the Dataset into the Training set and Test set
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# Feature Scaling
# 因為 LinearRegression 方法自帶特徵縮放的功能，故不用做
# 資料預處理完成

# Multiple Linear Regression
from sklearn.linear_model import LinearRegression

regressor = LinearRegression()
regressor.fit(x_train, y_train)
y_pred = regressor.predict(x_test)

# Building the optimal model using Backward Elimination
import statsmodels.api as sm

x_train = np.append(arr = np.ones((1070,1)).astype(int), values = x_train, axis = 1)
```

```
x_opt = x_train[:, [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]]
x_opt = np.array(x_opt, dtype=float)
regressor_OLS = sm.OLS(endog = y_train, exog = x_opt).fit()
regressor_OLS.summary()

x_opt = x_train[:, [0, 2, 3, 4, 5, 6, 7, 8, 9, 10]]
x_opt = np.array(x_opt, dtype=float)
regressor_OLS = sm.OLS(endog = y_train, exog = x_opt).fit()
regressor_OLS.summary()

x_opt = x_train[:, [2, 3, 4, 5, 6, 7, 8, 9, 10]]
x_opt = np.array(x_opt, dtype=float)
regressor_OLS = sm.OLS(endog = y_train, exog = x_opt).fit()
regressor_OLS.summary()

x_opt = x_train[:, [2, 3, 4, 6, 7, 8, 9, 10]]
x_opt = np.array(x_opt, dtype=float)
regressor_OLS = sm.OLS(endog = y_train, exog = x_opt).fit()
regressor_OLS.summary()

x_opt = x_train[:, [2, 3, 6, 7, 8, 9, 10]]
x_opt = np.array(x_opt, dtype=float)
regressor_OLS = sm.OLS(endog = y_train, exog = x_opt).fit()
regressor_OLS.summary()

x_opt = x_train[:, [2, 3, 6, 8, 9, 10]]
x_opt = np.array(x_opt, dtype=float)
regressor_OLS = sm.OLS(endog = y_train, exog = x_opt).fit()
regressor_OLS.summary()

x_opt = x_train[:, [2, 3, 8, 9, 10]]
x_opt = np.array(x_opt, dtype=float)
regressor_OLS = sm.OLS(endog = y_train, exog = x_opt).fit()
regressor_OLS.summary()

x_opt = x_train[:, [2, 3, 8, 9]]
x_opt = np.array(x_opt, dtype=float)
regressor_OLS = sm.OLS(endog = y_train, exog = x_opt).fit()
regressor_OLS.summary()
```