HIDDEN Traps: Handling Missing Data in Regression and linear model.

RE6144027 黃琮竣

3.14.2025

Outline

- Regression and Data Issues
- Missing Data
- Handle Missing Data
- Case Study
- Conclusion

INTRODUCTION TO REGRESSION & DATA ISSUES

What does regression do?

預測結果、分析、解釋變數間關係的統計方法，
探討自變數**x**與應變數**y**之間的線性關係。

Why do missing data occur?

1. 數據收集或儲存過程中發生錯誤，遺漏部分資料。
2. 收集資料者不願透露資訊或未給予合適回答選項。
3. 手動輸入錯誤，使資料有所缺失或不合理值出現。

INTRODUCTION TO REGRESSION & DATA ISSUES

Why Missing Data Matters?

Missing data 將對分析估計造成影響，包括 **Bias(** 預測平均誤差，影響模型精確度**)**、**Variance(**誤差差異程度、預測變化大**)** 等。

此將進一步影響到**R-squared**、**MSE**、**MAE**等模型評估方法，導致模型**Overfitting**、**Underfitting**，進而得出錯誤分析結論。

當資料不完美時，迴歸模型是否仍然可靠？

線性迴歸模型仍然可以使用，但其**可靠性**取決於**資料問題的嚴重程度**以及**是否妥善處理缺失值與異常值！**

MISSING DATA

What is missing data?

缺失值(**Missing Value**)指的是在蒐集數據的過程中發生人為或機器上的疏失，導致資料缺失的情況。

	A	B	C	D	E	F	G	H	I	J	K	L
1	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
2	1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171	7.25		S
3	2	1	1	Cumings, Mrs. John Bradl	female	38	1	0	PC 17599	71.2833	C85	C
4	3	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2. 3101282	7.925		S
5	4	1	1	Futrelle, Mrs. Jacques Hea	female	35	1	0	113803	53.1	C123	S
6	5	0	3	Allen, Mr. William Henry	male	35	0	0	373450	8.05		S
7	6	0	3	Moran, Mr. James	male	NA	0	0	330877	8.4583		Q
8	7	0	1	McCarthy, Mr. Timothy J	male	54	0	0	17463	51.8625	E46	S
9	8	0	3	Palsson, Master. Gosta Le	male	2	3	1	349909	21.075		S
10	9	1	3	Johnson, Mrs. Oscar W (E	female	27	0	2	347742	11.1333		S
11	10	1	2	Nasser, Mrs. Nicholas (Ac	female	14	1	0	237736	30.0708		C
12	11	1	3	Sandstrom, Miss. Marguer	female	4	1	1	PP 9549	16.7	G6	S
13	12	1	1	Bonnell, Miss. Elizabeth	female	58	0	0	113783	26.55	C103	S
14	13	0	3	Saundercock, Mr. William	male	20	0	0	A/5. 2151	8.05		S
15	14	0	3	Andersson, Mr. Anders Jo	male	39	1	5	347082	31.275		S
16	15	0	3	Vestrom, Miss. Hulda Am	female	14	0	0	350406	7.8542		S
17	16	1	2	Hewlett, Mrs. (Mary D Ki	female	55	0	0	248706	16		S
18	17	0	3	Rice, Master. Eugene	male	2	4	1	382652	29.125		Q
19	18	1	2	Williams, Mr. Charles Eug	male	NA	0	0	244373	13		S
20	19	0	3	Vander Planke, Mrs. Juliu	female	31	1	0	345763	18		S
21	20	1	3	Masselmani, Mrs. Fatima	female	NA	0	0	2649	7.225		C
22	21	0	2	Fynney, Mr. Joseph J	male	35	0	0	239865	26		S
23	22	1	2	Beesley, Mr. Lawrence	male	34	0	0	248698	13	D56	S
24	23	1	3	McGowan, Miss. Anna "A	female	15	0	0	330923	8.0292		Q

MISSING DATA - 缺失值類型

MAR

Missing at Random

該缺失值出現**跟缺失欄位本身無關**，但**與其他欄位有相關**，缺失值發生是條件隨機的，也因此若有足夠的其他變數來解釋缺失，數據仍可以被建模並填補。

如：學生資料中「學生數學成績」與「出席率」，發現經常缺席的學生較少提交數學成績，他們可能錯過考試。

MCAR

Missing Completely at Random

資料缺失是**完全隨機**的，與**自己或其他欄位皆無關**。也因為缺失數據沒有系統性偏差，刪除並不會影響其他的數據。

如：問卷資料部分遺失，但遺失原因可能是被損毀或弄丟了，並非跟問卷內容有關聯，因為缺失是隨機的狀況。

MNAR

Missing Not at Random

數據缺失**與該變數自身的值有關**，即缺失的發生不是隨機的，需要**額外數據來源或統計方法補償**此缺失。

如：做問卷調查時，薪水是較為敏感的資訊，較高者可能因為隱私等考量而選擇不填寫，這種情況就並非是隨機的。

MISSING DATA

缺失值導致問題

1. 使線性模型有錯誤的解釋跟決策，導致結果扭曲。
2. 影響變數間相關性，使得錯誤分析或共線性問題。
3. 讓線性模型的係數變得不穩定、統計顯著性降低。

缺失值處理目的

將數據資料中的缺失值，根據相應的處理策略來填入適當的值，以便後續進行有效的資料分析和預測。

而不同的缺失值處理策略，也會對迴歸分析結果造成不一樣的影響。

MISSING DATA - 缺失值處理方法

1

Deletion(刪除)

直接將資料刪除。MCAR

1. 缺失欄位刪除(column deletion)
2. 該筆資料刪除 (row deletion)

缺點：容易丟失大量數據。

DataFrame.dropna()

2

imputaion(補值)

通過**對缺失原因的了解，對其進行合理的補值**。

1. 統計數值補值(Median, Mode, Mean)
2. 近似資料補值 (KNN)
3. 向前向後補值 （資料前後有相依性）

缺點：會影響到數據方差。

SimpleImputer(missing_values=np.nan, strategy='mean')

3

線性迴歸填補

如果缺失值只與少數變數相關，可以使用 **線性迴歸** 來填補。 MAR

計算快速，適用於簡單具線性關係的數據。

當一個變數 X 有缺失值時，我們可以**使用其他變數來建立一個迴歸模型**，然後用此模型來預測缺失值。

4

MICE(多重差補)

如果多個變數同時有缺失，且變數之間的關聯較複雜，可以使用 **MICE進行**多重插補。

不僅使用單一迴歸模型來填補缺失值，而是**進行多次隨機插補**，逐步選擇有缺失值的變數來進行預測、差補，再繼續換下個變數繼續補值，考慮插補的**不確定性**，直到最終收斂、穩定。

缺失值處理方法 - MICE

MICE

Multiple Imputation by Chained Equations

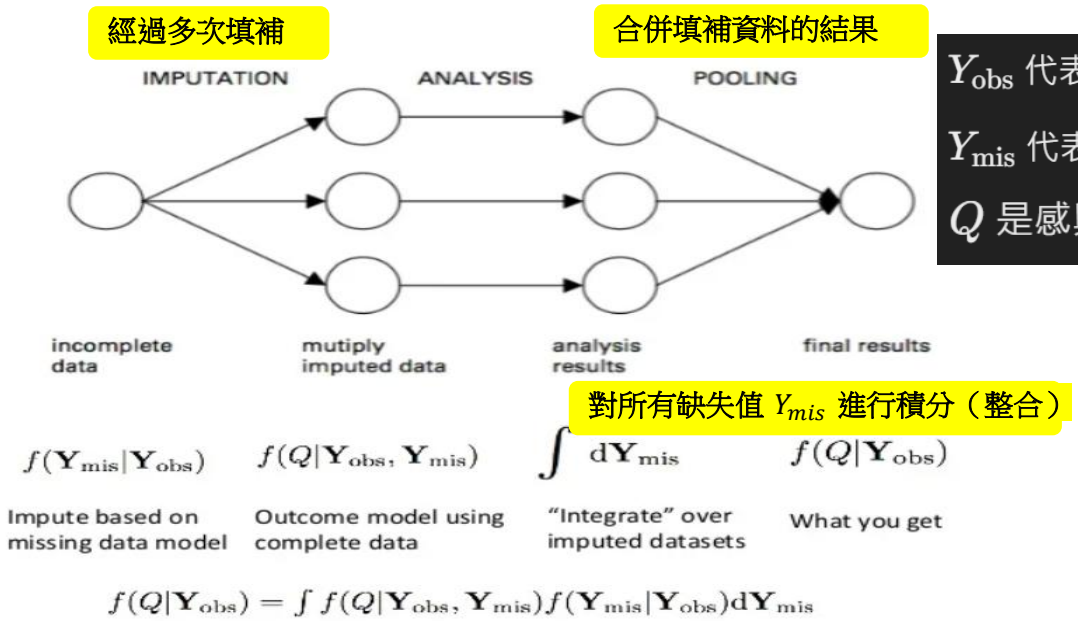
一種**多重補值**方法，主要用於處理數據中的缺失值。

對每個有缺失值的欄位，使用其餘變數來訓練迴歸模型，並**逐步預測及填補**這些缺失值。

通過迴圈來重複此步驟多次，直到收斂為止。

最後生成**多個填補數據集**，進行分析並合併結果。

MICE示意圖



Y_{obs} 代表已觀測的數據

Y_{mis} 代表缺失的數據

Q 是感興趣的統計量

$$f(Q|\mathbf{Y}_{\text{obs}}) = \int f(Q|\mathbf{Y}_{\text{obs}}, \mathbf{Y}_{\text{mis}}) f(\mathbf{Y}_{\text{mis}}|\mathbf{Y}_{\text{obs}}) \mathrm{d}\mathbf{Y}_{\text{mis}}$$

實例操作

讓我們進行深入實作

安裝相關套件

該工具各種矩陣運算，缺失值填補方法的套件，如：KNN、IterativeImputer等。

```
[1]  pip -q install fancyimpute
```

安裝相關套件

```
[2]  import pandas as pd                                          # 數據分析套件
     import numpy as np                                           # 矩陣數學運算套件
     from sklearn.linear_model import LinearRegression           # 線性迴歸套件(擅長處理數值型資料)
     from sklearn.linear_model import LogisticRegression         # 邏輯迴歸套件(擅長處理變數型資料)
     from sklearn.ensemble import RandomForestClassifier         # 隨機森林分類器套件
     from sklearn.ensemble import RandomForestRegressor          # 隨機森林迴歸分析套件
     from sklearn.model_selection import train_test_split        # 分割資料集的套件
     from sklearn.preprocessing import LabelEncoder              # 類別編碼套件
     from sklearn.impute import SimpleImputer                    # 使用簡單策略完成缺失值的套件
     from fancyimpute import IterativeImputer                    # 以循環方式，將具有缺失值的每個特徵建模，為其他特徵的函數來估算缺失值的策略。
     from sklearn.metrics import accuracy_score                  # 可以衡量分類模型整體性能的套件

     # 讀取 Titanic 數據集
     df = pd.read_csv("titanic.csv")
```

觀察數據、發現有類別型資料

看一下數據資料(看前15行)

```
[4] print(df.head(15))
```

```
    PassengerId  Survived  Pclass  \
0             1         0       3
1             2         1       1
2             3         1       3
3             4         1       1
4             5         0       3
5             6         0       3
6             7         0       1
7             8         0       3
8             9         1       3
9            10         1       2
10           11         1       3
11           12         1       1
12           13         0       3
13           14         0       3
14           15         0       3

                                                Name     Sex   Age  SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
2                             Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                           Allen, Mr. William Henry    male  35.0      0
5                                   Moran, Mr. James    male   NaN      0
6                            McCarthy, Mr. Timothy J    male  54.0      0
7                     Palsson, Master. Gosta Leonard    male   2.0      3
8  Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)  female  27.0      0
9                Nasser, Mrs. Nicholas (Adele Achem)  female  14.0      1
10               Sandstrom, Miss. Marguerite Rut  female   4.0      1
11                      Bonnell, Miss. Elizabeth  female  58.0      0
12                Saundercock, Mr. William Henry    male  20.0      0
13                   Andersson, Mr. Anders Johan    male  39.0      1
14          Vestrom, Miss. Hulda Amanda Adolfina  female  14.0      0

    Parch            Ticket     Fare Cabin Embarked
0       0         A/5 21171   7.2500   NaN        S
1       0          PC 17599  71.2833   C85        C
2       0  STON/02. 3101282   7.9250   NaN        S
3       0            113803  53.1000  C123        S
4       0            373450   8.0500   NaN        S
5       0            330877   8.4583   NaN        Q
6       0             17463  51.8625   E46        S
7       1            349909  21.0750   NaN        S
8       2            347742  11.1333   NaN        S
9       0            237736  30.0708   NaN        C
10      1        PP 9549     16.7000    G6        S
```

Label encoding

確認數據資料(看前15行)

```
print(df.head(15))
```

```
    PassengerId  Survived  Pclass  \
0             1         0       3
1             2         1       1
2             3         1       3
3             4         1       1
4             5         0       3
5             6         0       3
6             7         0       1
7             8         0       3
8             9         1       3
9            10         1       2
10           11         1       3
11           12         1       1
12           13         0       3
13           14         0       3
14           15         0       3

                                                Name  Sex   Age  SibSp  \
0                            Braund, Mr. Owen Harris    1  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th...    0  38.0      1
2                             Heikkinen, Miss. Laina    0  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)    0  35.0      1
4                           Allen, Mr. William Henry    1  35.0      0
5                                   Moran, Mr. James    1   NaN      0
6                            McCarthy, Mr. Timothy J    1  54.0      0
7                     Palsson, Master. Gosta Leonard    1   2.0      3
8  Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)    0  27.0      0
9                Nasser, Mrs. Nicholas (Adele Achem)    0  14.0      1
10               Sandstrom, Miss. Marguerite Rut    0   4.0      1
11                      Bonnell, Miss. Elizabeth    0  58.0      0
12                Saundercock, Mr. William Henry    1  20.0      0
13                   Andersson, Mr. Anders Johan    1  39.0      1
14          Vestrom, Miss. Hulda Amanda Adolfina    0  14.0      0

    Parch            Ticket     Fare Cabin Embarked
0       0         A/5 21171   7.2500   NaN        2
1       0          PC 17599  71.2833   C85        0
2       0  STON/02. 3101282   7.9250   NaN        2
3       0            113803  53.1000  C123        2
4       0            373450   8.0500   NaN        2
5       0            330877   8.4583   NaN        1
6       0             17463  51.8625   E46        2
7       1            349909  21.0750   NaN        2
8       2            347742  11.1333   NaN        2
9       0            237736  30.0708   NaN        0
10      1        PP 9549     16.7000    G6        2
```

數據集前處理&確認預處理後資料大小

```
[7]  #  選擇與生存率（Survived）相關的數值型變數
     #  由於主要是去分析跟存活相關的機率，而PassengerId不具特殊意義，Name、Ticket(船票號碼)、Cabin(船艙號碼)相對跟存活關聯性較少，因此此處先捨去這四個特徵。
     df  =  df[['Survived',  'Sex',  'Age',  'SibSp',  'Parch',  'Fare',  'Pclass',  'Embarked']]

[8]  #  了解現在的處理後的檔案大小
     print(df.shape)

     (891,  8)
```

實作四種方式並測試準確度 - DELETION（刪除法）

Deletion（刪除法）

[9]
```
# 刪除缺失值
df_deleted = df.dropna()

# 了解現在的處理後的檔案大小
print('由此可以看出dropna將sex的缺失值皆刪掉了，891-177=714')
print(df_deleted.shape)

# X, y特徵選擇
X = df_deleted[['Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Pclass', 'Embarked']]
y = df_deleted['Survived']

# 分割訓練與測試集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=12)

# 訓練邏輯回歸模型
model = LogisticRegression(max_iter=2000)
model.fit(X_train, y_train)

# 預測與評估
y_pred1 = model.predict(X_test)
accuracy_deletion = accuracy_score(y_test, y_pred1)
print("Accuracy (Deletion Method):", accuracy_deletion)
```

```
由此可以看出dropna將sex的缺失值皆刪掉了，891-177=714
(714, 8)
Accuracy (Deletion Method): 0.7972027972027972
```

實作四種方式並測試準確度 - MEAN IMPUTATION（平均值補值）

Mean Imputation（平均值補值）

```
[10] #  使用平均值"mean"策略補值
     imputer  =  SimpleImputer(strategy="mean")
     df_imputed  =  df.copy()

     #  對'Age'欄位進行補值
     df_imputed['Age']]  =  imputer.fit_transform(df[['Age']])

     #  對'Embarked'欄位進行補值
     imputer_mode  =  SimpleImputer(strategy="mean")
     df_imputed[['Embarked']]  =  imputer_mode.fit_transform(df[['Embarked']])

     #  了解現在的處理後的檔案大小
     print('由此可以看出通過平均值補值的方式，資料數量都沒變。')
     print(df_imputed.shape)

     #  X,  y特徵選擇
     X  =  df_imputed[['Sex',  'Age',  'SibSp',  'Parch',  'Fare',  'Pclass',  'Embarked']]
     y  =  df_imputed['Survived']

     #  分割訓練與測試集
     X_train,  X_test,  y_train,  y_test  =  train_test_split(X,  y,  test_size=0.2,  random_state=12)

     #  預測與評估
     model.fit(X_train,  y_train)
     y_pred2  =  model.predict(X_test)
     accuracy_imputation  =  accuracy_score(y_test,  y_pred2)
     print("Accuracy  (Imputation  Method):",  accuracy_imputation)
```

```
由此可以看出通過平均值補值的方式，資料數量都沒變。
(891, 8)
Accuracy (Imputation Method): 0.7877094972067039
```

實作四種方式並測試準確度 - REGRESSION IMPUTATION（線性迴歸補值）

Regression Imputation（線性迴歸補值）

```
[11] #  找出並分類'Age'缺失的數據
     df_missing  =  df[df['Age'].isnull()]
     df_non_missing  =  df.dropna(subset=['Age'])

     #  X,  y特徵選擇來進一步預測  Age
     X_train_age  =  df_non_missing[['Sex',  'SibSp',  'Parch',  'Fare',  'Pclass',  'Embarked']]
     y_train_age  =  df_non_missing['Age']

     #  定義要進行補值的線性迴歸模型
     regressor  =  LinearRegression()
     regressor.fit(X_train_age,  y_train_age)

     #  預測缺失值、並且讓預測結果維持是小數第一位且範圍在0.1~90歲的資料型態
     df_missing.loc[:,  'Age']  =  np.clip(np.round(regressor.predict(df_missing[['Sex',  'SibSp',  'Parch',  'Fare',  'Pclass',  'Embarked']]),  1),  0.1,  90)

     #  將填補的預測值合併拼接回原數據
     df_filled  =  pd.concat([df_missing,  df_non_missing])

     #  了解現在的處理後的檔案大小
     print('由此可以看出通過線性迴歸補值的方式，資料數量也都沒變。')
     print(df_filled.shape)

     #  了解現在的補植後的資料長怎樣
     print(df_filled['Age'].head(10))

     #  X,  y特徵選擇
     X  =  df_filled[['Sex',  'Age',  'SibSp',  'Parch',  'Fare',  'Pclass',  'Embarked']]
     y  =  df_filled['Survived']

     #  分割訓練與測試集
     X_train,  X_test,  y_train,  y_test  =  train_test_split(X,  y,  test_size=0.2,  random_state=12)

     #  預測與評估
     model.fit(X_train,  y_train)
     y_pred3  =  model.predict(X_test)
     accuracy_regression  =  accuracy_score(y_test,  y_pred3)
     print("Accuracy  (Regression  Imputation):",  accuracy_regression)
```

```
由此可以看出通過線性迴歸補值的方式，資料數量也都沒變。
(891, 8)
5     27.5
17    35.6
19    23.4
26    26.6
28    24.4
29    28.5
31    31.8
32    24.4
36    26.6
42    26.6
Name: Age, dtype: float64
Accuracy (Regression Imputation): 0.7821229050279329
```

實作四種方式並測試準確度 - MICE（多重插補法）

MICE（多重插補法）單獨對每個變數建模

```
# 使用 MICE 進行補值
imputer = IterativeImputer()
df_mice = df.copy()
df_mice['Age'] = imputer.fit_transform(df[['Age']])
df_mice['Embarked'] = imputer.fit_transform(df[['Embarked']])

print(df_mice.shape)

# X, y特徵選擇
X = df_mice['Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Pclass', 'Embarked']]
y = df_mice['Survived']

# 分割訓練與測試集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=12)

# 預測與評估
model.fit(X_train, y_train)
y_pred4 = model.predict(X_test)
accuracy_MICE = accuracy_score(y_test, y_pred4)
print("Accuracy (MICE Method):", accuracy_MICE)
```

```
(891, 8)
Accuracy (MICE Method): 0.7877094972067039
```

MICE（多重插補法）對所有資料變數建模

```
# 使用 MICE 進行多重插補
imputer = IterativeImputer()
df_imputed = imputer.fit_transform(df)

df = pd.DataFrame(df_imputed, columns=['Survived', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Pclass', 'Embarked'])
print(df.shape)

# X, y特徵選擇
X = df['Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Pclass', 'Embarked']]
y = df['Survived']

# 分割訓練與測試集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=12)

# 預測與評估
model.fit(X_train, y_train)
y_pred4 = model.predict(X_test)
accuracy_MICE2 = accuracy_score(y_test, y_pred4)
print("Accuracy (MICE2 Method):", accuracy_MICE2)
```

```
(891, 8)
Accuracy (MICE2 Method): 0.8156424581005587
```

綜合版

綜合版本

```python
# 讀取數據集
df = pd.read_csv("titanic.csv")

# 刪除 Embarked 缺失值（因為只有 2 筆，相對影響不大。）
df = df.dropna(subset=["Embarked"])

# 因為 Age 和 Name 的相關性建立 Title 特徵，並根據 Title 來進行後續年齡補值
# 先將姓和名分開成兩部分(根據", ")並拿第二部分([1])，然後進一步分割(根據".")得到，並取第一部分得到職稱。
df["Title"] = df["Name"].str.split(", ", expand=True)[1].str.split(".", expand=True)[0]

# 計算每個 Title 的平均(mean)年齡並將結果轉換成字典
age_means = df.groupby("Title")["Age"].mean().to_dict()

# 將缺失的 Age 補上字典中對應 Title 的平均年齡
df["Age"] = df.apply(lambda row: age_means[row["Title"]] if pd.isnull(row["Age"]) else row["Age"], axis=1)

# 轉換 Sex 和 Embarked 為數值類別
df["Sex"] = df["Sex"].map({"male": 0, "female": 1})
df["Embarked"] = df["Embarked"].map({"C": 0, "Q": 1, "S": 2})

print(df.shape)

# 特徵選擇
X = df[['Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Pclass', 'Embarked']]
y = df['Survived']

# 分割訓練與測試集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=12)

# 訓練邏輯回歸模型
model = LogisticRegression(penalty='l1', solver='liblinear')
model.fit(X_train, y_train)

# 預測
y_pred = model.predict(X_test)
accuracy_combined1 = accuracy_score(y_test, y_pred)

print("Accuracy (Comprehensive Approach):", accuracy_combined1)
```

```
(889, 13)
Accuracy (Comprehensive Approach): 0.8202247191011236
```

```python
[18] # 讀取數據集
df = pd.read_csv("titanic.csv")

# 刪除 Embarked 缺失值（因為只有 2 筆，相對影響不大。）
df = df.dropna(subset=["Embarked"])

# 因為 Age 和 Name 的相關性建立 Title 特徵，並根據 Title 來進行後續年齡補值
# 先將姓和名分開成兩部分(根據", ")並拿第二部分([1])，然後進一步分割(根據".")得到，並取第一部分得到職稱。
df["Title"] = df["Name"].str.split(", ", expand=True)[1].str.split(".", expand=True)[0]

# 計算每個 Title 的平均(mean)年齡並將結果轉換成字典
age_means = df.groupby("Title")["Age"].mean().to_dict()

# 將缺失的 Age 補上字典中對應 Title 的平均年齡
df["Age"] = df.apply(lambda row: age_means[row["Title"]] if pd.isnull(row["Age"]) else row["Age"], axis=1)

# 轉換 Sex 和 Embarked 為數值類別
df["Sex"] = df["Sex"].map({"male": 0, "female": 1})
df["Embarked"] = df["Embarked"].map({"C": 0, "Q": 1, "S": 2})

print(df.shape)

# 特徵選擇
X = df[['Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Pclass', 'Embarked']]
y = df['Survived']

# 分割訓練與測試集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=12)

# 訓練模型
model = RandomForestClassifier(n_estimators=1500, random_state=12)
model.fit(X_train, y_train)

# 預測
y_pred = model.predict(X_test)
accuracy_combined = accuracy_score(y_test, y_pred)

print("Accuracy (Comprehensive Approach):", accuracy_combined)
```

```
(889, 13)
Accuracy (Comprehensive Approach): 0.848314606741573
```

綜合版

綜合版本

```
# 讀取數據集
df = pd.read_csv("titanic.csv")

# 刪除 Embarked 缺失值（因為只有 2 筆，相對影響不大。）
df = df.dropna(subset=["Embarked"])

# 因為 Age 和 Name 的相關性建立 Title 特徵，並根據 Title 來進行後續年齡補值
# 先將姓和名分開成兩部分(根據", ")並拿第二部分([1])，然後進一步分割(根據".")得到，並取第一部分得到職稱。
df["Title"] = df["Name"].str.split(", ", expand=True)[1].str.split(".", expand=True)[0]

# 計算每個 Title 的平均(mean)年齡並將結果轉換成字典
age_means = df.groupby("Title")["Age"].mean().to_dict()

# 將缺失的 Age 補上字典中對應 Title 的平均年齡
df["Age"] = df.apply(lambda row: age_means[row["Title"]] if pd.isnull(row["Age"]) else row["Age"], axis=1)

# 轉換 Sex 和 Embarked 為數值類別
df["Sex"] = df["Sex"].map({"male": 0, "female": 1})
df["Embarked"] = df["Embarked"].map({"C": 0, "Q": 1, "S": 2})

print(df.shape)

# 特徵選擇
X = df[['Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Pclass', 'Embarked']]
y = df['Survived']

# 分割訓練與測試集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=12)

# 訓練邏輯回歸模型
model = LogisticRegression(penalty='l1', solver='liblinear')
model.fit(X_train, y_train)

# 預測
y_pred = model.predict(X_test)
accuracy_combined1 = accuracy_score(y_test, y_pred)

print("Accuracy (Comprehensive Approach):", accuracy_combined1)
```

```
(889, 13)
Accuracy (Comprehensive Approach): 0.8202247191011236
```

刪除 Embarked 欄位有缺失值的資料
（因為只有 2 筆，相對影響不大。）

	Column Name	Missing Count	Percentage(%)
0	Cabin	687	77.10%
1	Age	177	19.87%
2	Embarked	2	0.22%
3	Fare	0	0.00%
4	Ticket	0	0.00%
5	Parch	0	0.00%
6	SibSp	0	0.00%
7	Sex	0	0.00%
8	Name	0	0.00%
9	Pclass	0	0.00%
10	Survived	0	0.00%
11	PassengerId	0	0.00%

綜合版

綜合版本

```
# 讀取數據集
df = pd.read_csv("titanic.csv")

# 刪除 Embarked 缺失值（因為只有 2 筆，相對影響不大。）
df = df.dropna(subset=["Embarked"])

# 因為 Age 和 Name 的相關性建立 Title 特徵，並根據 Title 來進行後續年齡補值
# 先將姓和名分開成兩部分(根據", ")並拿第二部分([1])，然後進一步分割(根據".")得到，並取第一部分得到職稱。
df["Title"] = df["Name"].str.split(", ", expand=True)[1].str.split(".", expand=True)[0]

# 計算每個 Title 的平均(mean)年齡並將結果轉換成字典
age_means = df.groupby("Title")["Age"].mean().to_dict()

# 將缺失的 Age 補上字典中對應 Title 的平均年齡
df["Age"] = df.apply(lambda row: age_means[row["Title"]] if pd.isnull(row["Age"]) else row["Age"], axis=1)

# 轉換 Sex 和 Embarked 為數值類別
df["Sex"] = df["Sex"].map({"male": 0, "female": 1})
df["Embarked"] = df["Embarked"].map({"C": 0, "Q": 1, "S": 2})

print(df.shape)

# 特徵選擇
X = df[['Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Pclass', 'Embarked']]
y = df['Survived']

# 分割訓練與測試集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=12)

# 訓練邏輯回歸模型
model = LogisticRegression(penalty='l1', solver='liblinear')
model.fit(X_train, y_train)

# 預測
y_pred = model.predict(X_test)
accuracy_combined1 = accuracy_score(y_test, y_pred)

print("Accuracy (Comprehensive Approach):", accuracy_combined1)
```

```
(889, 13)
Accuracy (Comprehensive Approach): 0.8202247191011236
```

外國人的稱謂和職稱、年紀會有些關係，因此我們處理“ Name ”欄位，將Name做處理後取出，命名為新特徵“Title”。

Name
Braun , Mr. Owen Harris
Cuming , Mrs. ohn Bradley (Florence Briggs Thayer)
Heikkin n, Miss. Laina
Futrell , Mrs. acques Heath (Lily May Peel)
Alle , Mr. William Henry
Mora n, Mr. ames
McCarth y, Mr. Timothy J
Palsso , Master. Gosta Leonard
Johns n, Mrs. Oscar W (Elisabeth Vilhelmina Berg)
Nasse , Mrs. Nicholas (Adele Achem)

綜合版

綜合版本

```python
# 讀取數據集
df = pd.read_csv("titanic.csv")

# 刪除 Embarked 缺失值（因為只有 2 筆，相對影響不大。）
df = df.dropna(subset=["Embarked"])

# 因為 Age 和 Name 的相關性建立 Title 特徵，並根據 Title 來進行後續年齡補值
# 先將姓和名分開成兩部分(根據", ")並拿第二部分([1])，然後進一步分割(根據".")得到，並取第一部分得到職稱。
df["Title"] = df["Name"].str.split(", ", expand=True)[1].str.split(".", expand=True)[0]

# 計算每個 Title 的平均(mean)年齡並將結果轉換成字典
age_means = df.groupby("Title")["Age"].mean().to_dict()

# 將缺失的 Age 補上字典中對應 Title 的平均年齡
df["Age"] = df.apply(lambda row: age_means[row["Title"]] if pd.isnull(row["Age"]) else row["Age"], axis=1)

# 轉換 Sex 和 Embarked 為數值類別
df["Sex"] = df["Sex"].map({"male": 0, "female": 1})
df["Embarked"] = df["Embarked"].map({"C": 0, "Q": 1, "S": 2})

print(df.shape)

# 特徵選擇
X = df[['Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Pclass', 'Embarked']]
y = df['Survived']

# 分割訓練與測試集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=12)

# 訓練邏輯回歸模型
model = LogisticRegression(penalty='l1', solver='liblinear')
model.fit(X_train, y_train)

# 預測
y_pred = model.predict(X_test)
accuracy_combined1 = accuracy_score(y_test, y_pred)

print("Accuracy (Comprehensive Approach):", accuracy_combined1)
```

```
(889, 13)
Accuracy (Comprehensive Approach): 0.8202247191011236
```

1. 定義一個名為 age_means 的字典。
2. 使用groupby的方式，按照“Title”的欄位做為標籤做分組。
3. 對 “Age” 計算各組的平均值 [.mean()]。
4. 再將其轉換為字典格式 [.to_dict()]，用來填補"Age"缺失值。

	Title	Age_Mean
0	Capt	70.000000
1	Col	58.000000
2	Don	40.000000
3	Dr	42.000000
4	Jonkheer	38.000000
5	Lady	48.000000
6	Major	48.500000
7	Master	4.574167
8	Miss	21.662069
9	Mlle	24.000000
10	Mme	24.000000
11	Mr	32.368090
12	Mrs	35.654206
13	Ms	28.000000
14	Rev	43.166667
15	Sir	49.000000
16	the Countess	33.000000

綜合版

綜合版本

```
# 讀取數據集
df = pd.read_csv("titanic.csv")

# 刪除 Embarked 缺失值（因為只有 2 筆，相對影響不大。）
df = df.dropna(subset=["Embarked"])

# 因為 Age 和 Name 的相關性建立 Title 特徵，並根據 Title 來進行後續年齡補值
# 先將姓和名分開成兩部分(根據", ")並拿第二部分([1])，然後進一步分割(根據".")得到，並取第一部分得到職稱。
df["Title"] = df["Name"].str.split(", ", expand=True)[1].str.split(".", expand=True)[0]

# 計算每個 Title 的平均(mean)年齡並將結果轉換成字典
age_means = df.groupby("Title")["Age"].mean().to_dict()

# 將缺失的 Age 補上字典中對應 Title 的平均年齡
df["Age"] = df.apply(lambda row: age_means[row["Title"]] if pd.isnull(row["Age"]) else row["Age"], axis=1)

# 轉換 Sex 和 Embarked 為數值類別
df["Sex"] = df["Sex"].map({"male": 0, "female": 1})
df["Embarked"] = df["Embarked"].map({"C": 0, "Q": 1, "S": 2})

print(df.shape)

# 特徵選擇
X = df[['Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Pclass', 'Embarked']]
y = df['Survived']

# 分割訓練與測試集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=12)

# 訓練邏輯回歸模型
model = LogisticRegression(penalty='l1', solver='liblinear')
model.fit(X_train, y_train)

# 預測
y_pred = model.predict(X_test)
accuracy_combined1 = accuracy_score(y_test, y_pred)

print("Accuracy (Comprehensive Approach):", accuracy_combined1)
```

```
(889, 13)
Accuracy (Comprehensive Approach): 0.8202247191011236
```

1. 使用apply方法，對 DataFrame 的每一列（row）應用函數進行處理。
2. axis=1，表示按列 (row) 操作。
3. 檢查 row[“Title”]欄位，並根據 age_means 字典中對應 Title 的平均年齡來替換 Age 欄位的缺失值。
4. 檢查Age 欄位是否為缺失值。若是缺失值的話，則從age_means 字典中，根據 row["Title"] 取得該類別對應的平均年齡作為填充值。
5. 如果 row["Age"] 不是 NaN，則保持原值不變。

將”Sex”和”Embarked”轉為數值類別。

準確率比較

```
[16] print("Accuracy (Deletion Method):", accuracy_deletion)
     print("Accuracy (Imputation Method):", accuracy_imputation)
     print("Accuracy (Regression Imputation):", accuracy_regression)
     print("Accuracy (MICE Method):", accuracy_MICE)
     print("Accuracy (MICE2 Method):", accuracy_MICE2)
     print("Accuracy (combined Approach):", accuracy_combined)
```

```
Accuracy (Deletion Method): 0.7972027972027972
Accuracy (Imputation Method): 0.7877094972067039
Accuracy (Regression Imputation): 0.7821229050279329
Accuracy (MICE Method): 0.7877094972067039
Accuracy (MICE2 Method): 0.8156424581005587
Accuracy (combined Approach): 0.848314606741573
```

結論

選擇合適方法

根據資料特性和分析目標，選擇最適合的缺失值處理方法，如Listwise deletion、 imputation、MICE 等。

謹慎評估

資料的前處理對於後續線性模型的分析預測有一定差異性。評估每種方法的優缺點，並考慮其對模型結果的影響，將能有效幫助數據分析的工作。

參考資料

1. 【簡單線性迴歸分析(Simple regression analysis)-統計說明與SPSS操作】
   https://www.yongxi-stat.com/simple-regression-analysis/
2. 迴歸分析與缺失值處理: 策略與技巧
   https://blog.csdn.net/universsky2015/article/details/137312662
3. Regression回歸問題的評估指標
   https://bc165870081.medium.com/regression回歸問題的評估指標-4fb9b7c9a993
4. 資料科學常見觀念：缺失值處理
   https://medium.com/@AppliedDataScienceWeeklyNew/資料科學常見觀念-缺失值處理-8d62547d0c13
5. DAY08 資料前處理-缺失值處理方法
   https://ithelp.ithome.com.tw/m/articles/10260675
6. pandas.DataFrame.dropna
   https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.dropna.html
7. SimpleImputer
   https://scikit-learn.org/stable/modules/generated/sklearn.impute.SimpleImputer.html
8. **Day 6. 使用[R]進行遺失值處理**
   https://ithelp.ithome.com.tw/m/articles/10291434

感謝聆聽