Strategies for Dealing with Missing Values

Tsung-Jiun Huang

2025-03-13

目錄


```
# R Interface to Python
library(reticulate)               # Make R and Python interoperable, allowing R to call Python code.
use_python("C:/Users/user/anaconda3/python.exe", required = TRUE)  # Finding Anaconda's Python path
py_config()
```

python: C:/Users/user/anaconda3/python.exe libpython: C:/Users/user/anaconda3/python312.dll pythonhome: C:/Users/user/anaconda3 version: 3.12.7 | packaged by Anaconda, Inc. | (main, Oct 4 2024, 13:17:27) [MSC v.1929 64 bit (AMD64)] Architecture: 64bit numpy: C:/Users/user/anaconda3/Lib/site-packages/numpy numpy_version: 1.26.4

NOTE: Python version was forced by use_python() function

```
library(Hmisc)                    # data analysis and report tools
library(ggplot2)

# read Titanic dataset
df <- read.csv("C:/Users/user/Downloads/Titanic.csv")
latex(describe(df), descript = "descriptive statistics", file = '', caption.placement = 'top')
```

df
12 Variables 891 Observations

PassengerId

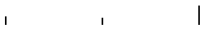
n	missing	distinct	Info	Mean	pMedian	Gmd	.05	.10	.25	.50	.75	.90	.95
891	0	891	1	446	446	297.3	45.5	90.0	223.5	446.0	668.5	802.0	846.5

lowest : 1 2 3 4 5, highest: 887 888 889 890 891

Survived

n	missing	distinct	Info	Sum	Mean
891	0	2	0.71	342	0.3838

Pclass



n	missing	distinct	Info	Mean	pMedian	Gmd
891	0	3	0.81	2.309	2.5	0.8631

Value	1	2	3
Frequency	216	184	491
Proportion	0.242	0.207	0.551

For the frequency table, variable is rounded to the nearest 0

Name

n	missing	distinct
891	0	891

lowest : Abbing, Mr. Anthony | Abbott, Mr. Rossmore Edward | Abbott, Mrs. Stanton (Rosa Hunt) | Abelson, Mr
highest: Youssef, Mr. Gerious | Yrois, Miss. Henriette ("Mrs Harbeck") | Zabour, Miss. Hileni | Zabour, Mis

Sex

n	missing	distinct
891	0	2

Value	female	male
Frequency	314	577
Proportion	0.352	0.648

Age

n	missing	distinct	Info	Mean	pMedian	Gmd	.05	.10	.25	.50	.75	.90	.95
714	177	88	0.999	29.7	29	16.21	4.00	14.00	20.12	28.00	38.00	50.00	56.00

lowest : 0.42 0.67 0.75 0.83 0.92, highest: 70 70.5 71 74 80

SibSp

n	missing	distinct	Info	Mean	pMedian	Gmd
891	0	7	0.669	0.523	0.5	0.823

Value	0	1	2	3	4	5	8
Frequency	608	209	28	16	18	5	7
Proportion	0.682	0.235	0.031	0.018	0.020	0.006	0.008

For the frequency table, variable is rounded to the nearest 0

Parch

n	missing	distinct	Info	Mean	pMedian	Gmd
891	0	7	0.556	0.3816	0	0.6259

Value	0	1	2	3	4	5	6
Frequency	678	118	80	5	4	5	1
Proportion	0.761	0.132	0.090	0.006	0.004	0.006	0.001

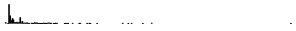
For the frequency table, variable is rounded to the nearest 0

Ticket

n	missing	distinct
891	0	681

lowest : 110152 110413 110465 110564 110813
highest: W./C. 6608 W./C. 6609 W.E.P. 5734 W/C 14208 WE/P 5735

Fare



n	missing	distinct	Info	Mean	pMedian	Gmd	.05	.10	.25
891	0	248	1	32.2	19.6	36.78	7.225	7.550	7.910

.50	.75	.90	.95
14.454	31.000	77.958	112.079

lowest : 0 4.0125 5 6.2375 6.4375 , highest: 227.525 247.521 262.375 263 512.329

Cabin

n	missing	distinct
204	687	147

lowest : A10 A14 A16 A19 A20, highest: F33 F38 F4 G6 T

Embarked

```
   n missing distinct
 889       2        3

Value          C     Q     S
Frequency    168    77   644
Proportion 0.189 0.087 0.724
```

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

```
# !pip -q install fancyimpute
```

安裝相關套件

```
import pandas as pd                    #
import numpy as np                     #
from sklearn.linear_model import LinearRegression    #    (     )
from sklearn.linear_model import LogisticRegression   #    (     )
from sklearn.ensemble import RandomForestClassifier   #
from sklearn.ensemble import RandomForestRegressor   #
from sklearn.model_selection import train_test_split  #
from sklearn.preprocessing import LabelEncoder     #
from sklearn.impute import SimpleImputer         #
from fancyimpute import IterativeImputer         #
from sklearn.metrics import accuracy_score       #

#   Titanic
df = pd.read_csv("C:/Users/user/Downloads/Titanic.csv")
```

```
#
df.isnull().sum()
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

看一下數據資料(看前15行)

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

```
    PassengerId  Survived  Pclass  ...     Fare Cabin  Embarked
0             1         0       3  ...   7.2500   NaN         S
1             2         1       1  ...  71.2833   C85         C
2             3         1       3  ...   7.9250   NaN         S
3             4         1       1  ...  53.1000  C123         S
4             5         0       3  ...   8.0500   NaN         S
5             6         0       3  ...   8.4583   NaN         Q
6             7         0       1  ...  51.8625   E46         S
```

```
7             8         0       3  ...  21.0750   NaN         S
8             9         1       3  ...  11.1333   NaN         S
9            10         1       2  ...  30.0708   NaN         C
10           11         1       3  ...  16.7000    G6         S
11           12         1       1  ...  26.5500  C103         S
12           13         0       3  ...   8.0500   NaN         S
13           14         0       3  ...  31.2750   NaN         S
14           15         0       3  ...   7.8542   NaN         S

[15 rows x 12 columns]
```

先進行Label Encoding

```
labelencoder = LabelEncoder()

#  'Sex'
df['Sex'] = labelencoder.fit_transform(df['Sex'].values)

#  'Embarked'
df['Embarked'] = labelencoder.fit_transform(df['Embarked'].values)
```

確認數據資料(看前15行)

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

```
    PassengerId  Survived  Pclass  ...     Fare  Cabin  Embarked
0             1         0       3  ...   7.2500    NaN         2
1             2         1       1  ...  71.2833    C85         0
2             3         1       3  ...   7.9250    NaN         2
3             4         1       1  ...  53.1000   C123         2
4             5         0       3  ...   8.0500    NaN         2
5             6         0       3  ...   8.4583    NaN         1
6             7         0       1  ...  51.8625    E46         2
7             8         0       3  ...  21.0750    NaN         2
8             9         1       3  ...  11.1333    NaN         2
9            10         1       2  ...  30.0708    NaN         0
10           11         1       3  ...  16.7000     G6         2
11           12         1       1  ...  26.5500   C103         2
12           13         0       3  ...   8.0500    NaN         2
13           14         0       3  ...  31.2750    NaN         2
14           15         0       3  ...   7.8542    NaN         2

[15 rows x 12 columns]
```

```
#     Survived
#           PassengerId    Name Ticket(  ) Cabin(  )
df = df[['Survived', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Pclass', 'Embarked']]
```

```
#
print(df.shape)
```

```
(891, 8)
```

Deletion（削除法）

```
#
df_deleted = df.dropna()

#
print('    dropna sex     891-177=714')
```

```
    dropna sex     891-177=714
```

```
print(df_deleted.shape)
```

```
(714, 8)
```

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

#
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=12)

#
model = LogisticRegression(max_iter=2000)
model.fit(X_train, y_train)
```

```
LogisticRegression(max_iter=2000)
```

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

```
Accuracy (Deletion Method): 0.7972027972027972
```

Mean Imputation（平均値補値）

```
#    "mean"
imputer = SimpleImputer(strategy="mean")
df_imputed = df.copy()

#  'Age'
df_imputed[['Age']] = imputer.fit_transform(df[['Age']])

#  'Embarked'
imputer_mode = SimpleImputer(strategy="mean")
df_imputed[['Embarked']] = imputer_mode.fit_transform(df[['Embarked']])

#
print('              ')
```

```
print(df_imputed.shape)
```

```
(891, 8)
```

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

#
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=12)

#
model.fit(X_train, y_train)
```

```
LogisticRegression(max_iter=2000)
```

```
y_pred2 = model.predict(X_test)
accuracy_imputation = accuracy_score(y_test, y_pred2)
print("Accuracy (Imputation Method):", accuracy_imputation)
```

```
Accuracy (Imputation Method): 0.7877094972067039
```

Regression Imputation（線性迴歸補值）

```
#    'Age'
df_missing = df[df['Age'].isnull()]
df_non_missing = df.dropna(subset=['Age'])

# X, y      Age
X_train_age = df_non_missing[['Sex', 'SibSp', 'Parch', 'Fare', 'Pclass', 'Embarked']]
y_train_age = df_non_missing['Age']

#
regressor = LinearRegression()
regressor.fit(X_train_age, y_train_age)
```

```
LinearRegression()
```

```
#              0.1~90
df_missing.loc[:, 'Age'] = np.clip(np.round(regressor.predict(df_missing[['Sex', 'SibSp', 'Parch', 'Fare

#
df_filled = pd.concat([df_missing, df_non_missing])

#
print('               ')
```

```
print(df_filled.shape)
```

```
(891, 8)
```

```
#
print(df_filled['Age'].head(10))
```

```
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
```

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

#
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=12)

#
model.fit(X_train, y_train)
```

```
LogisticRegression(max_iter=2000)
```

```
y_pred3 = model.predict(X_test)
accuracy_regression = accuracy_score(y_test, y_pred3)
print("Accuracy (Regression Imputation):", accuracy_regression)
```

```
Accuracy (Regression Imputation): 0.7821229050279329
```

MICE（多重插補法）單獨對每個變數建模

```
#   MICE
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)
```

```
(891, 8)
```

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

#
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=12)

#
model.fit(X_train, y_train)
```

```
LogisticRegression(max_iter=2000)
```

```
y_pred4 = model.predict(X_test)
accuracy_MICE = accuracy_score(y_test, y_pred4)
print("Accuracy (MICE Method):", accuracy_MICE)
```

```
Accuracy (MICE Method): 0.7877094972067039
```

MICE（多重插補法）對所有資料變數建模

```
#   MICE
imputer = IterativeImputer()
df_imputed = imputer.fit_transform(df)

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

```
(891, 8)
```

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

#
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=12)

#
model.fit(X_train, y_train)
```

```
LogisticRegression(max_iter=2000)
```

```
y_pred4 = model.predict(X_test)
accuracy_MICE2 = accuracy_score(y_test, y_pred4)
print("Accuracy (MICE2 Method):", accuracy_MICE2)
```

```
Accuracy (MICE2 Method): 0.8156424581005587
```

綜合版本

```
#
df = pd.read_csv("C:/Users/user/Downloads/Titanic.csv")

#   Embarked      2
df = df.dropna(subset=["Embarked"])

#   Age   Name      Title      Title
#       ( ", ")    ([1])     ( ".")
df["Title"] = df["Name"].str.split(", ", expand=True)[1].str.split(".", expand=True)[0]

#    Title   (mean)
age_means = df.groupby("Title")["Age"].mean().to_dict()

#    Age      Title
df["Age"] = df.apply(lambda row: age_means[row["Title"]] if pd.isnull(row["Age"]) else row["Age"], axis=

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

print(df.shape)
```

```
(889, 13)
```

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

#
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=12)

#
model = LogisticRegression(penalty='l1', solver='liblinear')
model.fit(X_train, y_train)
```

```
LogisticRegression(penalty='l1', solver='liblinear')
```

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

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

```
Accuracy (Comprehensive Approach): 0.8202247191011236
```

```
#
df = pd.read_csv("C:/Users/user/Downloads/Titanic.csv")

#   Embarked      2
df = df.dropna(subset=["Embarked"])

#   Age   Name     Title     Title
#       ( ", ")   ([1])    ( ".")
df["Title"] = df["Name"].str.split(", ", expand=True)[1].str.split(".", expand=True)[0]

#    Title   (mean)
age_means = df.groupby("Title")["Age"].mean().to_dict()

#    Age      Title
df["Age"] = df.apply(lambda row: age_means[row["Title"]] if pd.isnull(row["Age"]) else row["Age"], axis=

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

print(df.shape)
```

```
(889, 13)
```

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

#
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=12)

#
model = RandomForestClassifier(n_estimators=1500, random_state=12)
model.fit(X_train, y_train)
```

```
RandomForestClassifier(n_estimators=1500, random_state=12)
```

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

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

```
Accuracy (Comprehensive Approach): 0.848314606741573
```

```
print("Accuracy (Deletion Method):", accuracy_deletion)
```

```
Accuracy (Deletion Method): 0.7972027972027972
```

```
print("Accuracy (Imputation Method):", accuracy_imputation)
```

```
Accuracy (Imputation Method): 0.7877094972067039
```

```
print("Accuracy (Regression Imputation):", accuracy_regression)
```

```
Accuracy (Regression Imputation): 0.7821229050279329
```

```
print("Accuracy (MICE Method):", accuracy_MICE)
```

```
Accuracy (MICE Method): 0.7877094972067039
```

```
print("Accuracy (MICE2 Method):", accuracy_MICE2)
```

```
Accuracy (MICE2 Method): 0.8156424581005587
```

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

```
Accuracy (combined Approach): 0.848314606741573
```