機器學習
資料預處理

授課老師：林彥廷

資料預處理

Get the dataset
Importing the Libraries
Importing the Dataset
Missing Data
Categorical Data
Splitting the Dataset into the Training set and Test set
Feature Scaling
Data Preprocessing Template

Get the dataset

- 本課程請上數位學習平台下載
- 有興趣的可上Kaggle平台

Get the dataset

	A	B	C	D
1	Country	Age	Salary	Purchased
2	France	44	72000	No
3	Spain	27	48000	Yes
4	Germany	30	54000	No
5	Spain	38	61000	No
6	Germany	40		Yes
7	France	35	58000	Yes
8	Spain		52000	No
9	France	48	79000	Yes
10	Germany	50	83000	No
11	France	37	67000	Yes

自變量與應變量

$f(x) = y$

- 自變量：x
- 應變量：y

資料預處理

Get the dataset
Importing the Libraries
Importing the Dataset
Missing Data
Categorical Data
Splitting the Dataset into the Training set and Test set
Feature Scaling
Data Preprocessing Template

Importing the libraries

- import numpy as np
  - 提供矩陣數學運算方法
- import matplotlib.pyplot as plt
  - 提供畫圖方法
- import pandas as pd
  - 提供讀取資料集的方法

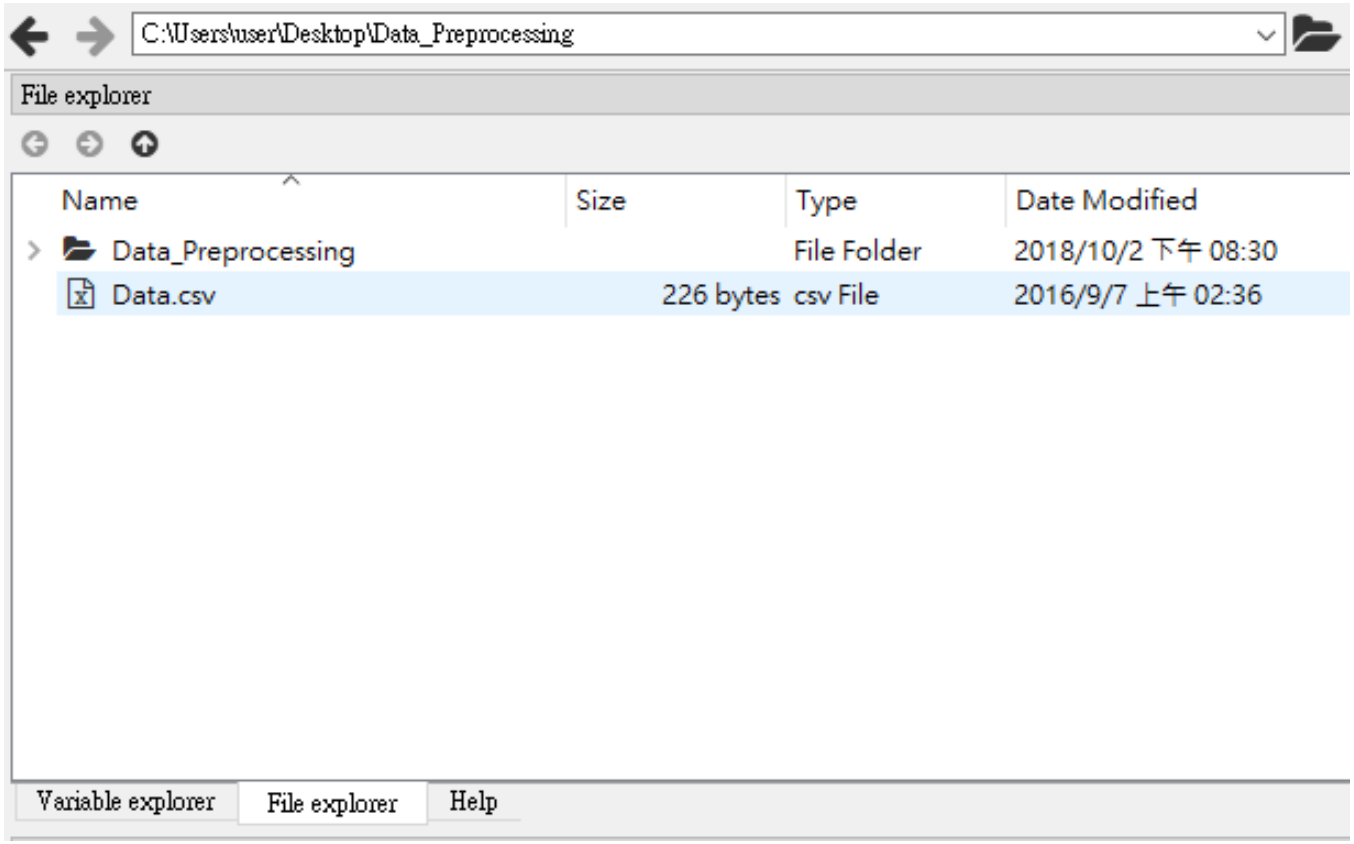
```
1 # Data Preprocessing
2
3 # Importing the libraries
4 import numpy as np
5 import matplotlib.pyplot as plt
6 import pandas as pd
7
```

資料預處理

Get the dataset
Importing the Libraries
Importing the Dataset
Missing Data
Categorical Data
Splitting the Dataset into the Training set and Test set
Feature Scaling
Data Preprocessing Template

Importing the dataset

- 選取資料集檔案路徑



Importing the dataset

- 透過pandas libarary中的read_csv方法來讀取
- dataset = pd.read_csv('Data.csv')

```
1 # Data Preprocessing
2
3 # Importing the libraries
4 import numpy as np
5 import matplotlib.pyplot as plt
6 import pandas as pd
7
8 # Importing the dataset
9 dataset = pd.read_csv('Data.csv')
```

Name	Type	Size	Value
dataset	DataFrame	(10, 4)	Column names: Country, Age, Salary, Purchased

Variable explorer | File explorer | Help

處理自變量與應變量

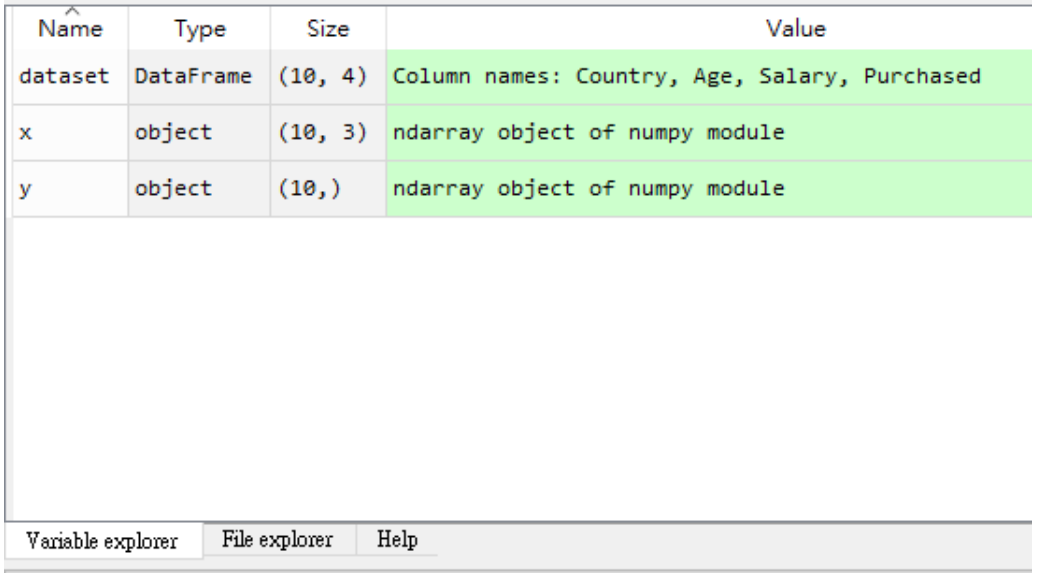
- 使用pandas中的iloc方法取出自變量與應變量所需要的values
- x = dataset.iloc[:, :-1].values
- y = dataset.iloc[:, 3].values

	A	B	C	D
1	Country	Age	Salary	Purchased
2	France	44	72000	No
3	Spain	27	48000	Yes
4	Germany	30	54000	No
5	Spain	38	61000	No
6	Germany	40		Yes
7	France	35	58000	Yes
8	Spain		52000	No
9	France	48	79000	Yes
10	Germany	50	83000	No
11	France	37	67000	Yes

- iloc[row, column]
- :表示所有rows or columns
- :-1表示所有rows or columns但不包含最後一個row or column

處理自變量與應變量

```
1  # Data Preprocessing
2
3  # Importing the libraries
4  import numpy as np
5  import matplotlib.pyplot as plt
6  import pandas as pd
7
8  # Importing the dataset
9  dataset = pd.read_csv('Data.csv')
10 x = dataset.iloc[:, :-1].values
11 y = dataset.iloc[:, 3].values
```



Name	Type	Size	Value
dataset	DataFrame	(10, 4)	Column names: Country, Age, Salary, Purchased
x	object	(10, 3)	ndarray object of numpy module
y	object	(10,)	ndarray object of numpy module

Variable explorer | File explorer | Help

資料預處理

Get the dataset
Importing the Libraries
Importing the Dataset
Missing Data
Categorical Data
Splitting the Dataset into the Training set and Test set
Feature Scaling
Data Preprocessing Template

Missing data

	A	B	C	D
1	Country	Age	Salary	Purchased
2	France	44	72000	No
3	Spain	27	48000	Yes
4	Germany	30	54000	No
5	Spain	38	61000	No
6	Germany	40		Yes
7	France	35	58000	Yes
8	Spain		52000	No
9	France	48	79000	Yes
10	Germany	50	83000	No
11	France	37	67000	Yes

Country	Age	Salary	Purchased
France	44	72000	No
Spain	27	48000	Yes
Germany	30	54000	No
Spain	38	61000	No
Germany	40	nan	Yes
France	35	58000	Yes
Spain	nan	52000	No
France	48	79000	Yes
Germany	50	83000	No
France	37	67000	Yes

Missing data

- 使用scikit-learn (sklearn)
  - 提供資料分析與資料處理的方法
- 使用sklearn中的preprocessing
  - 提供預處理的方法
- 使用sklearn中的preprocessing的SimpleImputer類別
  - 提供missing data處理的方法

```
from sklearn.impute import SimpleImputer
```

Missing data

```
imputer = SimpleImputer(missing_values=np.nan, strategy="mean", fill_value=None)
```

missing_values : number, string, np.nan (default) or None

The placeholder for the missing values. All occurrences of *missing_values* will be imputed. For pandas' dataframes with nullable integer dtypes with missing values, *missing_values* should be set to *np.nan*, since *pd.NA* will be converted to *np.nan*.

strategy : string, default='mean'

The imputation strategy.

- If "mean", then replace missing values using the mean along each column. Can only be used with numeric data.
- If "median", then replace missing values using the median along each column. Can only be used with numeric data.
- If "most_frequent", then replace missing using the most frequent value along each column. Can be used with strings or numeric data.
- If "constant", then replace missing values with fill_value. Can be used with strings or numeric data.

New in version 0.20: strategy="constant" for fixed value imputation.

fill_value : string or numerical value, default=None

When strategy == "constant", fill_value is used to replace all occurrences of missing_values. If left to the default, fill_value will be 0 when imputing numerical data and "missing_value" for strings or object data types.

Missing data

```
21 imputer = imputer.fit(x[:, 1:3])
```

- 透過imputer.fit()進行資料擬合
- 設定要進行缺失資料fit的範圍
- x[:, 1:3] ➔ x[all rows, 1 & 2 columns]
- 1:3表示第2及第3個columns

Missing data

```
23 x[:, 1:3] = imputer.transform(X[:, 1:3])
```

- 透過imputer.transform()進行缺失資料處理
- 設定要進行缺失資料轉換的範圍
- x[:, 1:3] ➔ x[all rows, 1 & 2 columns]
- 1:3表示第2及第3個columns

- 將轉換的結果設定回原本的資料集合中

資料預處理

Get the dataset
Importing the Libraries
Importing the Dataset
Missing Data
Categorical Data
Splitting the Dataset into the Training set and Test set
Feature Scaling
Data Preprocessing Template

Categorical data

	A	B	C	D
1	Country	Age	Salary	Purchased
2	France	44	72000	No
3	Spain	27	48000	Yes
4	Germany	30	54000	No
5	Spain	38	61000	No
6	Germany	40		Yes
7	France	35	58000	Yes
8	Spain		52000	No
9	France	48	79000	Yes
10	Germany	50	83000	No
11	France	37	67000	Yes

Categorical data

- 使用scikit-learn (sklearn)
  - 提供資料分析與資料處理的方法
- 使用sklearn中的preprocessing
  - 提供預處理的方法
- 使用sklearn中的preprocessing的LabelEncoder類別
  - 提供標籤編碼處理的方法

```
26 from sklearn.preprocessing import LabelEncoder
```

Categorical data

```
28 labelencoder_x = LabelEncoder()
```

- 宣告LabelEncoder物件

```
30 labelencoder_x.fit_transform(x[:, 0])
```

```
In [21]: labelencoder_x = LabelEncoder()
    ...: labelencoder_x.fit_transform(x[:, 0])
Out[21]: array([0, 2, 1, 2, 1, 0, 2, 0, 1, 0], dtype=int64)

In [22]: x
Out[22]:
array([['France', 44.0, 72000.0],
       ['Spain', 27.0, 48000.0],
       ['Germany', 30.0, 54000.0],
       ['Spain', 38.0, 61000.0],
       ['Germany', 40.0, 63777.77777777778],
       ['France', 35.0, 58000.0],
```

Categorical data

```
30 x[:, 0] = labelencoder_x.fit_transform(x[:, 0])
```

- 將轉換的結果設定回原本的資料集合中

```
In [24]: x
Out[24]:
array([[0, 44.0, 72000.0],
       [2, 27.0, 48000.0],
       [1, 30.0, 54000.0],
       [2, 38.0, 61000.0],
       [1, 40.0, 63777.77777777778],
       [0, 35.0, 58000.0],
```

Categorical data

- Dummy Encoding 虛擬編碼

Country
France
Spain
Germany
Spain
Germany
France
Spain
France
Germany
France

France	Spain	Germany
1	0	0
0	1	0
0	0	1
0	1	0
0	0	1
1	0	0
0	1	0
1	0	0
0	0	1
1	0	0

Categorical data

- 使用scikit-learn (sklearn)
  - 提供資料分析與資料處理的方法
- 使用sklearn中的preprocessing
  - 提供預處理的方法
- 使用sklearn中的preprocessing的OneHotEncoder類別
  - 提供虛擬編碼處理的方法

```
26 from sklearn.preprocessing import LabelEncoder, OneHotEncoder
```

Categorical data

- 使用scikit-learn (sklearn)
  - 提供資料分析與資料處理的方法
- 使用sklearn中的preprocessing
  - 提供預處理的方法
- 使用sklearn中的preprocessing的OneHotEncoder類別
  - 提供虛擬編碼處理的方法

```
from sklearn.compose import ColumnTransformer
```

Categorical data

- 使用ColumnTransformer方法來進行虛擬編碼

```
ct = ColumnTransformer([("Country", OneHotEncoder(), [0])]  ,  remainder='passthrough')
```

Definition : `ColumnTransformer(*, self, transformers, remainder='drop', sparse_threshold=0.3, n_jobs=None, transformer_weights=None, verbose=False)`

Applies transformers to columns of an array or pandas DataFrame.

This estimator allows different columns or column subsets of the input to be transformed separately and the features generated by each transformer will be concatenated to form a single feature space. This is useful for heterogeneous or columnar data, to combine several feature extraction mechanisms or transformations into a single transformer.

transformers : list of tuples

List of (name, transformer, columns) tuples specifying the transformer objects to be applied to subsets of the data.

name : str

Like in Pipeline and FeatureUnion, this allows the transformer and its parameters to be set using `set_params` and searched in grid search.

transformer : {'drop', 'passthrough'} or estimator

Estimator must support fit and transform. Special-cased strings 'drop' and 'passthrough' are accepted as well, to indicate to drop the columns or to pass them through untransformed, respectively.

columns : str, array-like of str, int, array-like of int, array-like of bool, slice or callable

Indexes the data on its second axis. Integers are interpreted as positional columns, while strings can reference DataFrame columns by name. A scalar string or int should be used where `transformer` expects X to be a 1d array-like (vector), otherwise a 2d array will be passed to the transformer. A callable is passed the input data `X` and can return any of the above. To select multiple columns by name or dtype, you can use `make_column_selector`.

Categorical data

- 使用ColumnTransformer方法來進行虛擬編碼

```
ct = ColumnTransformer([("Country", OneHotEncoder(), [0])]  ,  remainder='passthrough')
```

remainder : {'drop', 'passthrough'} or estimator, default='drop'

By default, only the specified columns in *transformers* are transformed and combined in the output, and the non-specified columns are dropped. (default of `'drop'`). By specifying `remainder='passthrough'`, all remaining columns that were not specified in *transformers* will be automatically passed through. This subset of columns is concatenated with the output of the transformers. By setting `remainder` to be an estimator, the remaining non-specified columns will use the `remainder` estimator. The estimator must support fit and transform. Note that using this feature requires that the DataFrame columns input at fit and transform have identical order.

Categorical data

```
X = ct.fit_transform(x)
```

- 使用ColumnTransformer中的fit_transform()方法進行轉換，並且轉換為array()

```
In [28]: x
Out[28]:
array([[1.00000000e+00, 0.00000000e+00, 0.00000000e+00,
4.40000000e+01,
        7.20000000e+04],
       [0.00000000e+00, 0.00000000e+00, 1.00000000e+00,
2.70000000e+01,
        4.80000000e+04],
       [0.00000000e+00, 1.00000000e+00, 0.00000000e+00,
```

	0	1	2	3	4
0	1	0	0	44	72000
1	0	0	1	27	48000
2	0	1	0	30	54000
3	0	0	1	38	61000
4	0	1	0	40	63777.8
5	1	0	0	35	58000
6	0	0	1	38.7778	52000
7	1	0	0	48	79000
8	0	1	0	50	83000
9	1	0	0	37	67000

Categorical data

```
37 labelencoder_y = LabelEncoder()
38 y = labelencoder_y.fit_transform(y)
```

- 應變量能夠自動被識別為類別，所以不需要做虛擬編碼，直接使用LabelEncoder即可
- 使用LabelEncoder中的fit_transform()方法進行轉換

Categorical data

```
37 labelencoder_y = LabelEncoder()
38 y = labelencoder_y.fit_transform(y)
```

	0
0	0
1	1
2	0
3	0
4	1
5	1
6	0
7	1
8	0
9	1

Country	Age	Salary	Purchased
France	44	72000	No
Spain	27	48000	Yes
Germany	30	54000	No
Spain	38	61000	No
Germany	40	nan	Yes
France	35	58000	Yes
Spain	nan	52000	No
France	48	79000	Yes
Germany	50	83000	No
France	37	67000	Yes

資料預處理

Get the dataset
Importing the Libraries
Importing the Dataset
Missing Data
Categorical Data
Splitting the Dataset into the Training set and Test set
Feature Scaling
Data Preprocessing Template

分割訓練集合與測試集合

- 使用sklearn中的model_selection的train_test_split類別
  - 提供分割訓練集合與測試集合的方法

```
40 from sklearn.model_selection import train_test_split
```

分割訓練集合與測試集合

```
42 x_train, x_test, y_train, y_test = train_test_split(
43         x, y, test_size = 0.2, random_state = 0)
```

***arrays :** sequence of indexables with same length / shape[0]
Allowed inputs are lists, numpy arrays, scipy-sparse matrices or pandas dataframes.

test_size : float, int, None, optional
If float, should be between 0.0 and 1.0 and represent the proportion of the dataset to include in the test split. If int, represents the absolute number of test samples. If None, the value is set to the complement of the train size. By default, the value is set to 0.25. The default will change in version 0.21. It will remain 0.25 only if `train_size` is unspecified, otherwise it will complement the specified `train_size`.

分割訓練集合與測試集合

```
42 x_train, x_test, y_train, y_test = train_test_split(
43         x, y, test_size = 0.2, random_state = 0)
```

random_state : int, RandomState instance or None, optional (default=None)

If int, random_state is the seed used by the random number generator; If RandomState instance, random_state is the random number generator; If None, the random number generator is the RandomState instance used by *np.random*.

分割訓練集合與測試集合

```
42 x_train, x_test, y_train, y_test = train_test_split(
43         x, y, test_size = 0.2, random_state = 0)
```

x_train - NumPy array

	0	1	2	3	4
0	0	1	0	40	63777.8
1	1	0	0	37	67000
2	0	0	1	27	48000
3	0	0	1	38.7778	52000
4	1	0	0	48	79000
5	0	0	1	38	61000
6	1	0	0	44	72000
7	1	0	0	35	58000

Format Resize Background color

y_train - NumPy array

	0
0	1
1	1
2	1
3	0
4	1
5	0
6	0

Format Resize Background color OK Cancel

x_test - NumPy array

	0	1	2	3	4
0	0	1	0	30	54000
1	0	1	0	50	83000

y_test - NumPy array

	0
0	0
1	0

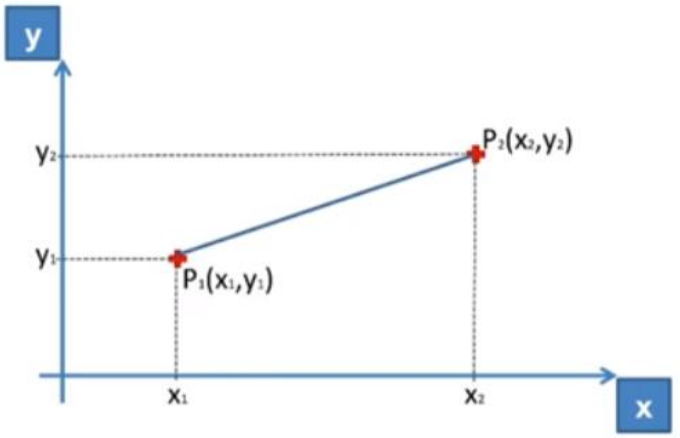
資料預處理

Get the dataset
Importing the Libraries
Importing the Dataset
Missing Data
Categorical Data
Splitting the Dataset into the Training set and Test set
Feature Scaling
Data Preprocessing Template

Feature scaling

- 特徵縮放

Country	Age	Salary	Purchased
France	44	72000	No
Spain	27	48000	Yes
Germany	30	54000	No
Spain	38	61000	No
Germany	40	63777.78	Yes
France	35	58000	Yes
Spain	38.78	52000	No
France	48	79000	Yes
Germany	50	83000	No
France	37	67000	Yes



Euclidean Distance between P_1 and $P_2 = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

Feature scaling

- 特徵縮放
  - 標準化與正規化

Standardisation	Normalisation
$x_{stand} = \frac{x - mean(x)}{StandardDeviation(x)}$	$x_{norm} = \frac{x - min(x)}{max(x) - min(x)}$

Feature scaling

- 使用sklearn中的preprocessing的StandardScaler類別
  - 提供標準化的方法

```
45 from sklearn.preprocessing import StandardScaler
```

Feature scaling

```
47 sc_x = StandardScaler()
48 x_train = sc_x.fit_transform(x_train)
49 x_test = sc_x.transform(x_test)
```

- 建立StandardScaler()物件
- 透過StandardScaler()中的fit_transform()方法對x_train進行擬合並轉換
- 因為sc_x已經被擬合過，故49行的sc_x不需要再用fit_transform()方法，可直接使用transform()方法

Feature scaling

```
47 sc_x = StandardScaler()
48 x_train = sc_x.fit_transform(x_train)
49 x_test = sc_x.transform(x_test)
```

x_train - NumPy array

	0	1	2	3	4
0	-1	2.64575	-0.774597	0.263068	0.123815
1	1	-0.377964	-0.774597	-0.253501	0.461756
2	-1	-0.377964	1.29099	-1.9754	-1.53093
3	-1	-0.377964	1.29099	0.0526135	-1.11142
4	1	-0.377964	-0.774597	1.64059	1.7203
5	-1	-0.377964	1.29099	-0.0813118	-0.167514
6	1	-0.377964	-0.774597	0.951826	0.986148
7	1	-0.377964	-0.774597	-0.597881	-0.482149

x_test - NumPy array

	0	1	2	3	4
0	-1	2.64575	-0.774597	-1.45883	-0.901663
1	-1	2.64575	-0.774597	1.98496	2.13981

資料預處理

Get the dataset
Importing the Libraries
Importing the Dataset
Missing Data
Categorical Data
Splitting the Dataset into the Training set and Test set
Feature Scaling
Data Preprocessing Template

THE END

ytlin@mail.nptu.edu.tw