Logistic regression (with two categories)

1. Import and visualize data set

In [1]:
```
import pandas as pd
df = pd.read_excel('wine_2.xlsx')
```

In [2]:
```
df.head()
```

Out[2]:

	Class	Alcohol	Malic acid	Ash	Alcalinity of ash	Magnesium	Total phenols	Flavanoids	Nonflavanoid phenols	Proanthocyanins	Color intensity	Hue	OD280/OD315 of diluted wines	Proline
0	WineA	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29	5.64	1.04	3.92	1065
1	WineA	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050
2	WineA	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185
3	WineA	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480
4	WineA	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 131 entries, 0 to 130
Data columns (total 14 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   Class                         131 non-null    object 
 1   Alcohol                       131 non-null    float64
 2   Malic acid                    131 non-null    float64
 3   Ash                           131 non-null    float64
 4   Alcalinity of ash             131 non-null    float64
 5   Magnesium                     131 non-null    int64  
 6   Total phenols                 131 non-null    float64
 7   Flavanoids                    131 non-null    float64
 8   Nonflavanoid phenols          131 non-null    float64
 9   Proanthocyanins               131 non-null    float64
 10  Color intensity               131 non-null    float64
 11  Hue                           131 non-null    float64
 12  OD280/OD315 of diluted wines  131 non-null    float64
 13  Proline                       131 non-null    int64  
dtypes: float64(11), int64(2), object(1)
memory usage: 14.5+ KB
```

In [4]:
```
x = df.drop('Class',axis=1)
y = df['Class']
```

In [5]:
```
y.describe()
```

Out[5]:
```
count       131
unique        2
top       WineB
freq         71
Name: Class, dtype: object
```

In [6]:
```
import numpy as np
```

In [7]:
```
np.unique(y)
```

Out[7]:
```
array(['WineA', 'WineB'], dtype=object)
```

2. Adjust data scaling

In [8]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
x = scaler.fit_transform(x)
```

In [9]:
```
pd.DataFrame(x).describe()
```

Out[9]:

	0	1	2	3	4	5	6	7	8	9	10	11	
count	1.310000e+02	1.310000e+02	1.310000e+02	1.310000e+02	1.310000e+02	1.310000e+02	1.310000e+02	1.310000e+02	1.310000e+02	1.310000e+02	1.310000e+02	1.310000e+02	1.31000
mean	3.651024e-15	-3.661193e-16	-1.066789e-15	3.542544e-16	3.305244e-16	-4.406992e-16	4.110368e-16	3.898493e-17	-4.830741e-17	2.949295e-16	-1.664487e-15	-5.678240e-17	-1.27124
std	1.003839e+00	1.003839e+00	1.003839e+00	1.003839e+00	1.003839e+00	1.003839e+00	1.003839e+00	1.003839e+00	1.003839e+00	1.003839e+00	1.003839e+00	1.003839e+00	1.00383
min	-2.169454e+00	-1.398315e+00	-3.324307e+00	-2.423456e+00	-1.954514e+00	-2.580426e+00	-2.581719e+00	-1.832669e+00	-2.474022e+00	-1.806214e+00	-2.164089e+00	-3.348390e+00	-1.45925
25%	-8.032242e-01	-5.239100e-01	-5.775213e-01	-6.605415e-01	-7.844851e-01	-7.202807e-01	-6.898889e-01	-6.373281e-01	-6.039474e-01	-8.175845e-01	-7.178632e-01	-4.392664e-01	-8.93687
50%	5.279530e-02	-2.667321e-01	-6.885731e-02	-8.277973e-02	-1.344690e-01	8.213483e-02	9.386945e-02	-2.695308e-01	-8.551092e-02	-1.822585e-01	-3.902263e-02	3.714072e-02	-2.13860
75%	8.861388e-01	1.047471e-01	6.263168e-01	6.283117e-01	5.155471e-01	7.477750e-01	6.952013e-01	6.039877e-01	4.699567e-01	7.164954e-01	6.398180e-01	7.264105e-01	7.81600
max	2.138989e+00	4.385331e+00	3.017037e+00	3.324534e+00	4.025634e+00	2.489381e+00	3.512677e+00	3.040645e+00	3.395420e+00	2.916893e+00	3.856932e+00	2.145495e+00	2.54543

3. Split data set to training data and testing data

In [10]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3)
```

4. Build the linear regressing model

In [11]:
```
from sklearn.linear_model import LogisticRegression
logmodel = LogisticRegression()
```

In [12]:
```
logmodel.fit(X_train,y_train)
```

Out[12]:
```
LogisticRegression()
```

5. Compare between original data and the result of prediction from the model

```
In [13]: predictions = logmodel.predict(X_test)
         predictions
         len(predictions)

Out[13]: 40
```

```
In [14]: predictions

Out[14]: array(['WineB', 'WineA', 'WineA', 'WineB', 'WineA', 'WineB', 'WineB',
                'WineA', 'WineA', 'WineA', 'WineA', 'WineB', 'WineB', 'WineB',
                'WineA', 'WineB', 'WineB', 'WineB', 'WineA', 'WineB', 'WineA',
                'WineB', 'WineA', 'WineA', 'WineA', 'WineA', 'WineA', 'WineA',
                'WineB', 'WineB', 'WineB', 'WineA', 'WineB', 'WineA', 'WineB',
                'WineA', 'WineA', 'WineB', 'WineB', 'WineB'], dtype=object)
```

6. Evaluate the model with confusion metrix

```
In [22]: from sklearn.metrics import classification_report, confusion_matrix
```

```
In [23]: import pytest
```

```
In [24]: print(confusion_matrix(ytest, ypred))

         ---------------------------------------------------------------------------
         NameError                                 Traceback (most recent call last)
         Input In [24], in <cell line: 1>()
         ----> 1 print(confusion_matrix(ytest, ypred))

         NameError: name 'ytest' is not defined
```

```
In [21]: print(classification_report(ytest,ypred))

         ---------------------------------------------------------------------------
         NameError                                 Traceback (most recent call last)
         Input In [21], in <cell line: 1>()
         ----> 1 print(classification_report(ytest,ypred))

         NameError: name 'ytest' is not defined
```

```
In [ ]:
```