Logistic regression (with three categories)

1. Import and visualize data set

In [1]:
```
import pandas as pd
```

In [2]:
```
df = pd.read_excel('wine_3.xlsx')
```

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

Out[3]:

	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	Type1	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	Type1	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	Type1	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	Type1	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	Type1	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 [4]:
```
x=df.drop('Class',axis=1)
```

In [5]:
```
y=df['Class']
```

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

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

Out[7]:
```
array(['Tpye3', 'Type1', 'Type2'], dtype=object)
```

In [8]:
```
df.describe()
```

Out[8]:

	Alcohol	Malic acid	Ash	Alcalinity of ash	Magnesium	Total phenols	Flavanoids	Nonflavanoid phenols	Proanthocyanins	Color intensity	Hue	OD280/OD315 of diluted wines	Proline
count	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000
mean	13.000618	2.336348	2.366517	19.494944	99.741573	2.295112	2.029270	0.361854	1.590899	5.058090	0.957449	2.611685	746.893258
std	0.811827	1.117146	0.274344	3.339564	14.282484	0.625851	0.998859	0.124453	0.572359	2.318286	0.228572	0.709990	314.907474
min	11.030000	0.740000	1.360000	10.600000	70.000000	0.980000	0.340000	0.130000	0.410000	1.280000	0.480000	1.270000	278.000000
25%	12.362500	1.602500	2.210000	17.200000	88.000000	1.742500	1.205000	0.270000	1.250000	3.220000	0.782500	1.937500	500.500000
50%	13.050000	1.865000	2.360000	19.500000	98.000000	2.355000	2.135000	0.340000	1.555000	4.690000	0.965000	2.780000	673.500000
75%	13.677500	3.082500	2.557500	21.500000	107.000000	2.800000	2.875000	0.437500	1.950000	6.200000	1.120000	3.170000	985.000000
max	14.830000	5.800000	3.230000	30.000000	162.000000	3.880000	5.080000	0.660000	3.580000	13.000000	1.710000	4.000000	1680.000000

2. Adjust data scaling

In [9]:
```
from sklearn.preprocessing import StandardScaler
```

In [10]:
```
scaler = StandardScaler()
```

In [11]:
```
x = scaler.fit_transform(x)
```

In [12]:
```
pd.DataFrame(x).describe()
```

Out[12]:

	0	1	2	3	4	5	6	7	8	9	10	
count	1.780000e+02	1.780000e+02	1.780000e+02	1.780000e+02	1.780000e+02	1.780000e+02	1.780000e+02	1.780000e+02	1.780000e+02	1.780000e+02	1.780000e+02	1.780000e+
mean	-8.619821e-16	-8.357859e-17	-8.657245e-16	-1.160121e-16	-1.995907e-17	-2.972030e-16	-4.016762e-16	4.079134e-16	-1.699639e-16	-1.122697e-17	3.717376e-16	2.919013e
std	1.002821e+00	1.002821e+00	1.002821e+00	1.002821e+00	1.002821e+00	1.002821e+00	1.002821e+00	1.002821e+00	1.002821e+00	1.002821e+00	1.002821e+00	1.002821e+
min	-2.434235e+00	-1.432983e+00	-3.679162e+00	-2.671018e+00	-2.088255e+00	-2.107246e+00	-1.695971e+00	-1.868234e+00	-2.069034e+00	-1.634288e+00	-2.094732e+00	-1.895054e+
25%	-7.882448e-01	-6.587486e-01	-5.721225e-01	-6.891372e-01	-8.244151e-01	-8.854682e-01	-8.275393e-01	-7.401412e-01	-5.972835e-01	-7.951025e-01	-7.675624e-01	-9.522483e
50%	6.099988e-02	-4.231120e-01	-2.382132e-02	1.518295e-03	-1.222817e-01	9.595986e-02	1.061497e-01	-1.760948e-01	-6.289785e-02	-1.592246e-01	3.312687e-02	2.377348e
75%	8.361286e-01	6.697929e-01	6.981085e-01	6.020883e-01	5.096384e-01	8.089974e-01	8.490851e-01	6.095413e-01	6.291754e-01	4.939560e-01	7.131644e-01	7.885875e
max	2.259772e+00	3.109192e+00	3.156325e+00	3.154511e+00	4.371372e+00	2.539515e+00	3.062832e+00	2.402403e+00	3.485073e+00	3.435432e+00	3.301694e+00	1.960915e+

3. Split data set to training data and testing data

In [13]:
```
from sklearn.model_selection import train_test_split
```

In [14]:
```
xtrain, xtest, ytrain, ytest = train_test_split(x,y, test_size=0.3)
```

4. Build the linear regressing model

In [15]:
```
from sklearn.linear_model import LogisticRegression
```

In [16]:
```
logmodel = LogisticRegression()
```

In [17]:
```
logmodel.fit(xtrain, ytrain)
```

Out[17]:
```
LogisticRegression()
```

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

In [18]:
```
ypred = logmodel.predict(xtest)
```

6. Evaluate the model with confusion metrix

In [19]:
```
from sklearn.metrics import classification_report, confusion_matrix
```

In [20]:
```
print(confusion_matrix(ytest, ypred))
```

```
[[13  0  0]
 [ 0 20  0]
 [ 0  0 21]]
```

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

```
              precision    recall  f1-score   support

       Tpye3       1.00      1.00      1.00        13
       Type1       1.00      1.00      1.00        20
       Type2       1.00      1.00      1.00        21

    accuracy                           1.00        54
   macro avg       1.00      1.00      1.00        54
weighted avg       1.00      1.00      1.00        54
```

In [ ]:
```

```