day1数据预处理

笔记本:	100days		
创建时间:	2019/1/10 16:03	**更新时间:**	2019/1/10 16:06
URL:	http://localhost:8888/notebooks/Day1/demo.ipynb		

Day 1: Data Prepocessing(数据预处理)¶

Step 1: Importing the libraries(导入库)¶

In [48]:

2

```
import numpy as np
```

```
import pandas as pd
```

. . .

Step 2: Importing dataset(导入数据)¶

In [15]:

2

```
dataset = pd.read_csv('Data.csv')
```

```
dataset
```

Out[15]:

	COUNTRY	AGE	SALARY	PURCHASED
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	NaN	Yes
5	France	35.0	58000.0	Yes
6	Spain	NaN	52000.0	No
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No
9	France	37.0	67000.0	Yes

. . .

In [24]:

2

```
X = dataset.iloc[ : , :-1].values
```

```
X
```

Out[24]:

```
array([['France', 44.0, 72000.0],
['Spain', 27.0, 48000.0],
['Germany', 30.0, 54000.0],
['Spain', 38.0, 61000.0],
['Germany', 40.0, nan],
['France', 35.0, 58000.0],
['Spain', nan, 52000.0],
['France', 48.0, 79000.0],
['Germany', 50.0, 83000.0],
['France', 37.0, 67000.0]], dtype=object)
```

. . .

In [17]:

2

```
Y= dataset.iloc[:,3].values
```

```
Y
```

Out[17]:

```
array(['No', 'Yes', 'No', 'No', 'Yes', 'Yes', 'No', 'Yes', 'No', 'Yes'],
dtype=object)
```

. . .

Step 3: Handling the missing data(处理缺失数据)¶

In [34]:

4

```
from sklearn.preprocessing import Imputer
```

```
imputer = Imputer(missing_values="NaN",strategy="mean",axis=0)
```

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

```
X
```

Out[34]:

```
array([['France', 44.0, 72000.0],
['Spain', 27.0, 48000.0],
['Germany', 30.0, 54000.0],
['Spain', 38.0, 61000.0],
['Germany', 40.0, nan],
['France', 35.0, 58000.0],
['Spain', nan, 52000.0],
['France', 48.0, 79000.0],
['Germany', 50.0, 83000.0],
['France', 37.0, 67000.0]], dtype=object)
```

. . .

In [36]:

2

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

```
X
```

Out[36]:

```
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],
['Spain', 38.77777777777778, 52000.0],
['France', 48.0, 79000.0],
['Germany', 50.0, 83000.0],
['France', 37.0, 67000.0]], dtype=object)
```

. . .

Step 4: Encoding categorical data(解析分类数据)¶

In [41]:

4

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

```
labelencoder_X=LabelEncoder()
```

```
X[:,0]=labelencoder_X.fit_transform(X[:,0])
```

```
X
```

Out[41]:

```
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],
[2, 38.77777777777778, 52000.0],
[0, 48.0, 79000.0],
[1, 50.0, 83000.0],
[0, 37.0, 67000.0]], dtype=object)
```

. . .

Creating a dummy variable(创建虚拟变量)¶

In [43]:

3

```
onehotencoder = OneHotEncoder(categorical_features=[0])
```

```
X=onehotencoder.fit_transform(X).toarray()
```

```
X
```

Out[43]:

```
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, 3.00000000e+01,
5.40000000e+04],
[0.00000000e+00, 0.00000000e+00, 1.00000000e+00, 3.80000000e+01,
6.10000000e+04],
[0.00000000e+00, 1.00000000e+00, 0.00000000e+00, 4.00000000e+01,
6.37777778e+04],
[1.00000000e+00, 0.00000000e+00, 0.00000000e+00, 3.50000000e+01,
5.80000000e+04],
[0.00000000e+00, 0.00000000e+00, 1.00000000e+00, 3.87777778e+01,
5.20000000e+04],
[1.00000000e+00, 0.00000000e+00, 0.00000000e+00, 4.80000000e+01,
7.90000000e+04],
[0.00000000e+00, 1.00000000e+00, 0.00000000e+00, 5.00000000e+01,
8.30000000e+04],
[1.00000000e+00, 0.00000000e+00, 0.00000000e+00, 3.70000000e+01,
6.70000000e+04]])
```

. . .

In [44]:

```
xxxxxxxxxx
```

```
labelencoder_Y=LabelEncoder()
```

```
Y=labelencoder_Y.fit_transform(Y)
```

```
Y
```

Out[44]:

```
array([0, 1, 0, 0, 1, 1, 0, 1, 0, 1], dtype=int64)
```

. . .

Step 5: Splitting the datasets into training sets and Test sets(拆分数据集为训练数据和测试数据)¶

In [45]:

```
x
```

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

```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0.2,random_state=0)
```

```
X_train, X_test, Y_train, Y_test
```

Out[45]:

```
(array([[0.00000000e+00, 1.00000000e+00, 0.00000000e+00, 4.00000000e+01,
6.37777778e+04],
[1.00000000e+00, 0.00000000e+00, 0.00000000e+00, 3.70000000e+01,
6.70000000e+04],
[0.00000000e+00, 0.00000000e+00, 1.00000000e+00, 2.70000000e+01,
4.80000000e+04],
[0.00000000e+00, 0.00000000e+00, 1.00000000e+00, 3.87777778e+01,
5.20000000e+04],
[1.00000000e+00, 0.00000000e+00, 0.00000000e+00, 4.80000000e+01,
7.90000000e+04],
[0.00000000e+00, 0.00000000e+00, 1.00000000e+00, 3.80000000e+01,
6.10000000e+04],
[1.00000000e+00, 0.00000000e+00, 0.00000000e+00, 4.40000000e+01,
7.20000000e+04],
[1.00000000e+00, 0.00000000e+00, 0.00000000e+00, 3.50000000e+01,
5.80000000e+04]]),
array([[0.0e+00, 1.0e+00, 0.0e+00, 3.0e+01, 5.4e+04],
[0.0e+00, 1.0e+00, 0.0e+00, 5.0e+01, 8.3e+04]]),
array([1, 1, 1, 0, 1, 0, 0, 1], dtype=int64),
array([0, 0], dtype=int64))
```

. . .

Step 6: Feature Scaling(特征量化)¶

In [46]:

5

```
from sklearn.preprocessing import StandardScaler

sc_X = StandardScaler()

X_train = sc_X.fit_transform(X_train)

X_test=sc_X.transform(X_test)

X_train, X_test
```

Out[46]:

```
(array([[-1. , 2.64575131, -0.77459667, 0.26306757, 0.12381479],
[ 1. , -0.37796447, -0.77459667, -0.25350148, 0.46175632],
[-1. , -0.37796447, 1.29099445, -1.97539832, -1.53093341],
[-1. , -0.37796447, 1.29099445, 0.05261351, -1.11141978],
[ 1. , -0.37796447, -0.77459667, 1.64058505, 1.7202972 ],
[-1. , -0.37796447, 1.29099445, -0.0813118 , -0.16751412],
[ 1. , -0.37796447, -0.77459667, 0.95182631, 0.98614835],
[ 1. , -0.37796447, -0.77459667, -0.59788085, -0.48214934]]),
array([[-1. , 2.64575131, -0.77459667, -1.45882927, -0.90166297],
[-1. , 2.64575131, -0.77459667, 1.98496442, 2.13981082]]))
```

. . .