LAB12

August 25, 2019

[1]:

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import LabelEncoder,OneHotEncoder
```

```
/home/rahul/anaconda3/lib/python3.7/importlib/_bootstrap.py:219: RuntimeWarning:
numpy.ufunc size changed, may indicate binary incompatibility. Expected 216, got
192
  return f(*args, **kwds)
/home/rahul/anaconda3/lib/python3.7/importlib/_bootstrap.py:219: RuntimeWarning:
numpy.ufunc size changed, may indicate binary incompatibility. Expected 216, got
192
  return f(*args, **kwds)
```

[3]:

```
datasets = pd.read_csv('Data_for_Categorical_Values.csv')
print("\n\nData :\n",datasets)
print("\n\nData Statistics:\n",datasets.describe())
```

```


Data :
     Country  Age  Salary Purchased
0    France   44   72000        No
1     Spain   27   48000       Yes
2   Germany   30   54000        No
3     Spain   38   61000        No
4   Germany   40   68000       Yes
5    France   35   58000       Yes
6     Spain   39   52000        No
7    France   48   79000       Yes
8   Germany   50   83000        No
9    France   37   67000       Yes
10    Spain   45   55000        No


Data Statistics:
              Age        Salary
```

```
count  11.000000     11.000000
mean   39.363636  63363.636364
std     7.131237  11386.594989
min    27.000000  48000.000000
25%    36.000000  54500.000000
50%    39.000000  61000.000000
75%    44.500000  70000.000000
max    50.000000  83000.000000
```

[4]:
```
X = datasets.iloc[:, :-1].values
Y = datasets.iloc[:, -1].values
print("\n\nInput : \n", X)
print("\n\nOutput: \n", Y)
```

```


Input :
 [['France' 44 72000]
 ['Spain' 27 48000]
 ['Germany' 30 54000]
 ['Spain' 38 61000]
 ['Germany' 40 68000]
 ['France' 35 58000]
 ['Spain' 39 52000]
 ['France' 48 79000]
 ['Germany' 50 83000]
 ['France' 37 67000]
 ['Spain' 45 55000]]


Output:
 ['No' 'Yes' 'No' 'No' 'Yes' 'Yes' 'No' 'Yes' 'No' 'Yes' 'No']
```

[5]:
```
le = LabelEncoder()
X[:,0] = le.fit_transform(X[:,0])
print("\n\nInput :\n",X)
```

```


Input :
 [[0 44 72000]
 [2 27 48000]
 [1 30 54000]
 [2 38 61000]
 [1 40 68000]
 [0 35 58000]
 [2 39 52000]
 [0 48 79000]
 [1 50 83000]
 [0 37 67000]
 [2 45 55000]]
```

[6]:

```python
ohe = OneHotEncoder(categorical_features=[0])
X = ohe.fit_transform(X)
print("\n\nInput : \n", X.toarray())
```

```


Input :
 [[1.0e+00 0.0e+00 0.0e+00 4.4e+01 7.2e+04]
 [0.0e+00 0.0e+00 1.0e+00 2.7e+01 4.8e+04]
 [0.0e+00 1.0e+00 0.0e+00 3.0e+01 5.4e+04]
 [0.0e+00 0.0e+00 1.0e+00 3.8e+01 6.1e+04]
 [0.0e+00 1.0e+00 0.0e+00 4.0e+01 6.8e+04]
 [1.0e+00 0.0e+00 0.0e+00 3.5e+01 5.8e+04]
 [0.0e+00 0.0e+00 1.0e+00 3.9e+01 5.2e+04]
 [1.0e+00 0.0e+00 0.0e+00 4.8e+01 7.9e+04]
 [0.0e+00 1.0e+00 0.0e+00 5.0e+01 8.3e+04]
 [1.0e+00 0.0e+00 0.0e+00 3.7e+01 6.7e+04]
 [0.0e+00 0.0e+00 1.0e+00 4.5e+01 5.5e+04]]

/home/rahul/.local/lib/python3.7/site-
packages/sklearn/preprocessing/_encoders.py:415: FutureWarning: The handling of
integer data will change in version 0.22. Currently, the categories are
determined based on the range [0, max(values)], while in the future they will be
determined based on the unique values.
If you want the future behaviour and silence this warning, you can specify
"categories='auto'".
In case you used a LabelEncoder before this OneHotEncoder to convert the
categories to integers, then you can now use the OneHotEncoder directly.
  warnings.warn(msg, FutureWarning)
```

[7]:

```python
dummy = pd.get_dummies(datasets['Country'])
print("\n\nDummy :\n",dummy)
datasets = datasets.drop(['Country','Purchased'],axis=1)
datasets = pd.concat([dummy,datasets],axis=1)
print("\n\nFinal Data :\n",datasets)
```

```


Dummy :
      France  Germany  Spain
0         1        0      0
1         0        0      1
```

```
2        0        1      0
3        0        0      1
4        0        1      0
5        1        0      0
6        0        0      1
7        1        0      0
8        0        1      0
9        1        0      0
10       0        0      1


Final Data :
     France  Germany  Spain  Age  Salary
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   68000
5        1        0      0   35   58000
6        0        0      1   39   52000
7        1        0      0   48   79000
8        0        1      0   50   83000
9        1        0      0   37   67000
10       0        0      1   45   55000
```

[ ]:

```
```