Data Preprocessing in Python

October 10, 2021

1 Importing the libraries

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

2 Importing the dataset

[2]:
```
data_set = pd.read_csv("Data.csv")
```

[3]:
```
x = data_set.iloc[:,:-1].values
```

[4]:
```
y = data_set.iloc[:, -1].values
```

[5]:
```
print(data_set)
```

```
   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
```

[6]:
```
print(x)
```

```
[['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]]
```

[7]:

```
print(y)
```

```
['No' 'Yes' 'No' 'No' 'Yes' 'Yes' 'No' 'Yes' 'No' 'Yes']
```

3 Taking care of missing data

[8]:

```
imputer = SimpleImputer(missing_values = np.nan, strategy="mean")
```

[9]:

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

[10]:

```
print(x)
```

```
[['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]]
```

3.1 Encoding categorical data

3.1.1 Encoding the independent Variable

[11]:

```
ct = ColumnTransformer(transformers=[('encoder',OneHotEncoder(),[0])],␣
 ↪remainder ='passthrough')
x = ct.fit_transform(x)
```

[12]:

```
print(x)
```

```
[[1.0 0.0 0.0 44.0 72000.0]
 [0.0 0.0 1.0 27.0 48000.0]
 [0.0 1.0 0.0 30.0 54000.0]
 [0.0 0.0 1.0 38.0 61000.0]
 [0.0 1.0 0.0 40.0 63777.77777777778]
 [1.0 0.0 0.0 35.0 58000.0]
 [0.0 0.0 1.0 38.77777777777778 52000.0]
 [1.0 0.0 0.0 48.0 79000.0]
 [0.0 1.0 0.0 50.0 83000.0]
 [1.0 0.0 0.0 37.0 67000.0]]
```

3.2 Encoding the Dependent Variable

[13]:

```
le = LabelEncoder()
y = le.fit_transform(y)
```

[14]:

```
print(y)
```

```
[0 1 0 0 1 1 0 1 0 1]
```

3.3 Splitting the dataset into the Training set and Test set

[15]:

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.
 ↪2,random_state = 1)
```

[16]:

```
print(x_train)
```

```
[[0.0 0.0 1.0 38.77777777777778 52000.0]
 [0.0 1.0 0.0 40.0 63777.77777777778]
 [1.0 0.0 0.0 44.0 72000.0]
 [0.0 0.0 1.0 38.0 61000.0]
 [0.0 0.0 1.0 27.0 48000.0]
 [1.0 0.0 0.0 48.0 79000.0]
 [0.0 1.0 0.0 50.0 83000.0]
 [1.0 0.0 0.0 35.0 58000.0]]
```

[17]:

```
print(x_test)
```

```
[[0.0 1.0 0.0 30.0 54000.0]
 [1.0 0.0 0.0 37.0 67000.0]]
```

[18]:

```
print(y_train)
```

```
[0 1 0 0 1 1 0 1]
```

[19]:

```
print(y_test)
```

```
[0 1]
```

3.4 Feature Scaling

[20]:

```
sc = StandardScaler()
x_train[:, 3:] = sc.fit_transform(x_train[:,3:])
x_test[:, 3:] = sc.transform(x_test[:,3:])
```

[21]:

```
print(x_train)
```

```
[[0.0 0.0 1.0 -0.19159184384578545 -1.0781259408412425]
 [0.0 1.0 0.0 -0.014117293757057777 -0.07013167641635372]
 [1.0 0.0 0.0 0.566708506533324 0.633562432710455]
 [0.0 0.0 1.0 -0.30453019390224867 -0.30786617274297867]
 [0.0 0.0 1.0 -1.9018011447007988 -1.420463615551582]
 [1.0 0.0 0.0 1.1475343068237058 1.232653363453549]
 [0.0 1.0 0.0 1.4379472069688968 1.5749910381638885]
 [1.0 0.0 0.0 -0.7401495441200351 -0.5646194287757332]]
```

```
[22]: print(x_test)
```

```
[[0.0 1.0 0.0 -1.4661817944830124 -0.9069571034860727]
 [1.0 0.0 0.0 -0.44973664397484414 0.2056403393225306]]
```