Data Preprocessing Tools[¶](http://localhost:8888/notebooks/ML_data/data_preprocessing_tools.ipynb#Data-Preprocessing-Tools)



**## Importing the libraries**

Importing the libraries[¶](http://localhost:8888/notebooks/ML_data/data_preprocessing_tools.ipynb#Importing-the-libraries)

In [1]:



import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

. . .



xxxxxxxxxx

**## Importing the dataset and splitting it into features and dependent variable**

Importing the dataset and splitting it into features and dependent variable[¶](http://localhost:8888/notebooks/ML_data/data_preprocessing_tools.ipynb#Importing-the-dataset-and-splitting-it-into-features-and-dependent-variable)

In [2]:



dataset = pd.read\_csv('Data.csv')

print(dataset)

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

y = dataset.iloc[:, -1].values

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 [3]:



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]]

. . .

In [4]:



print(y)

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

. . .



**## Taking care of missing data**

Taking care of missing data[¶](http://localhost:8888/notebooks/ML_data/data_preprocessing_tools.ipynb#Taking-care-of-missing-data)

In [5]:



x

from sklearn.impute import SimpleImputer

imputer = SimpleImputer(missing\_values=np.nan, strategy='mean')

# creating the object of the simpleImputer class to replace our missing witht he mean

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

# we use the fit function to fit the imputer on the data set

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

# using the tranform function we are actuall replacing the missing values

. . .

In [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 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]]

. . .



**## Encoding categorical data**

Encoding categorical data[¶](http://localhost:8888/notebooks/ML_data/data_preprocessing_tools.ipynb#Encoding-categorical-data)



**### Encoding the Independent Variable**

Encoding the Independent Variable[¶](http://localhost:8888/notebooks/ML_data/data_preprocessing_tools.ipynb#Encoding-the-Independent-Variable)

In [7]:



x

from sklearn.compose import ColumnTransformer

from sklearn.preprocessing import OneHotEncoder

ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [0])], remainder='passthrough')

# creating the object for ColumnTransformer with onehotencoder for 1st column

X = np.array(ct.fit\_transform(X))

# fitting and tranformer the initial data set and then converting it into a numpy array.

. . .

In [8]:



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]]

. . .



**### Encoding the Dependent Variable**

Encoding the Dependent Variable[¶](http://localhost:8888/notebooks/ML_data/data_preprocessing_tools.ipynb#Encoding-the-Dependent-Variable)

In [9]:



from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

y = le.fit\_transform(y)

. . .

In [10]:



print(y)

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

. . .



**## Splitting the dataset into the Training set and Test set**

Splitting the dataset into the Training set and Test set[¶](http://localhost:8888/notebooks/ML_data/data_preprocessing_tools.ipynb#Splitting-the-dataset-into-the-Training-set-and-Test-set)

In [11]:



xxxxxxxxxx

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)

#generally we prefer 75:25 ration for train set and test set but since the data here is less i used 80:20 ratio

. . .

In [12]:



#np.set\_printoptions(suppress=True)

print(X\_train)

[[0.0 1.0 0.0 40.0 63777.77777777778]

[1.0 0.0 0.0 37.0 67000.0]

[0.0 0.0 1.0 27.0 48000.0]

[0.0 0.0 1.0 38.77777777777778 52000.0]

[1.0 0.0 0.0 48.0 79000.0]

[0.0 0.0 1.0 38.0 61000.0]

[1.0 0.0 0.0 44.0 72000.0]

[1.0 0.0 0.0 35.0 58000.0]]

. . .

In [13]:



xxxxxxxxxx

print(X\_test)

[[0.0 1.0 0.0 30.0 54000.0]

[0.0 1.0 0.0 50.0 83000.0]]

. . .

In [14]:



xxxxxxxxxx

print(y\_train)

[1 1 1 0 1 0 0 1]

. . .

In [15]:



xxxxxxxxxx

print(y\_test)

[0 0]

. . .



**## Feature Scalling**

Feature Scalling[¶](http://localhost:8888/notebooks/ML_data/data_preprocessing_tools.ipynb" \l "Feature-Scalling)

In [16]:



from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X = sc.fit\_transform(X)

. . .

In [17]:



np.set\_printoptions(suppress=True)

print(X)

[[ 1.22474487 -0.65465367 -0.65465367 0.75887436 0.74947325]

[-0.81649658 -0.65465367 1.52752523 -1.71150388 -1.43817841]

[-0.81649658 1.52752523 -0.65465367 -1.27555478 -0.89126549]

[-0.81649658 -0.65465367 1.52752523 -0.11302384 -0.25320042]

[-0.81649658 1.52752523 -0.65465367 0.17760889 0. ]

[ 1.22474487 -0.65465367 -0.65465367 -0.54897294 -0.52665688]

[-0.81649658 -0.65465367 1.52752523 0. -1.0735698 ]

[ 1.22474487 -0.65465367 -0.65465367 1.34013983 1.38753832]

[-0.81649658 1.52752523 -0.65465367 1.63077256 1.75214693]

[ 1.22474487 -0.65465367 -0.65465367 -0.25834021 0.29371249]]

. . .