Data Preprocessing Tools

Importing the libraries

In [0]:

```
import numpy as np
import matplotlib.pyplot  as plt
import pandas as pd
```

Importing the dataset

In [0]:

```
dataset = pd.read_csv('Data.csv')
# factors or independent variables are th params on what we are gonna predicted the depen
dent variable
#usually the last column is the dependent variable

x = dataset.iloc[:, :-1].values # :-1 means from zero index to last( index of last one is
-1)
# iloc stands for locate indexes
y= dataset.iloc[:, -1].values   # -1 is index of last column
```

In [0]:

```
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 [0]:

```
print(y)
```

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

Taking care of missing data

In [0]:

```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values=np.nan,strategy='mean')
#options are mean, median ,most_frequent,constant
imputer.fit(x[:,1:3]) #fit imputer on X
#select the columns that have numeric value

x[:, 1:3]=imputer.transform(x[:, 1:3])  # fit the imputer on the data
```

In [0]:

```
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:

one cloumn with categories , it will be difficult for machine larning model to produce out come for the In dependent variable hence we encode the string to numbers eg: France 0 , germany 1 spain 2

But there is not relational order bwteen the countries n we want to prevent it hence we dont do 0 ,1 ,2

Hence we OneHot Encoding

OneHotEncoding transforms the Country column into 3 columns, similarly 5 countries then 5 columsn

it creates binary vectors for each country

france 100 spain 001 germany 010

hence no simple 1 2 3

IT IS DONE FOR processing data sets containing categorical variables

ALso purchased value is Yes and No , so we have to replace it with 0 and 1

In [0]:

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder

ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(),[0])], remainder='passt
hrough')
#encoder means we want to do encoding , OneHotEncoder is the name of the class that will
proceed to do encoding ,0 is the country column
#passthrough we want to keep the column that wont be applied transformation i.e Age and S
alary

x = np.array(ct.fit_transform(x))   # we need the output of the transforamtion as a numpy
array for the future transformation
```

In [0]:

```
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

In [0]:

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y=le.fit_transform(y)
```

In [0]:

```
print(y)
```

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

Splitting the dataset into the Training set and Test set

The data set is divided into two data sets , training and testing data set

Training is used to train the model

Testing is used to test the model and evaluating the performance . the test set is not worked during the training

we create a pair of matrix xtest , xtrain , ytest , ytrain

In [0]:

```
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=1)
#test size 20% and
#andom_state=1 is Controls the shuffling applied to the data before applying the split.
```

In [0]:

```
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]]
```

In [0]:

```
print(x_test)
```

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

In [0]:

```
print(y_train)
```

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

In [0]:

```
print(y_test)
```

```
[0 1]
```

Feature Scaling

Scaling all the variables , so that all considered at the same scale

To prevent dominance of one variable over the other and prevent preference in machine learning model

It uses Mean and Standard Deviation

Hence to not inlcude the test data set value in the scaling to calcuate Mean and SD , feature scaling is not done before Splitting the data into test and train

2 types

Standardisation & Normalisation

Here we will use Standardisation

In [0]:

```
from sklearn.preprocessing import StandardScaler
sc= StandardScaler()

x_train[:,3:] = sc.fit_transform(x_train[:,3:])

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

In [0]:

```
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]]
```

In [0]:

```
print(x_test)
```

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