

Importing the libraries/packages

```
In [1]: # Libraries
import numpy as np # used for handling numbers
import pandas as pd # used for handling the dataset
```

```
In [2]: from sklearn.impute import SimpleImputer # used for handling missing data
```

To import the dataset into a variable

```
In [3]: dataset = pd.read_csv('Datapreprocessing.csv')
```

```
In [5]: dataset
```

```
Out[5]:
```

	Region	Age	Income	Online Shopper
0	India	49.0	86400.0	No
1	Brazil	32.0	57600.0	Yes
2	USA	35.0	64800.0	No
3	Brazil	43.0	73200.0	No
4	USA	45.0	NaN	Yes
5	India	40.0	69600.0	Yes
6	Brazil	NaN	62400.0	No
7	India	53.0	94800.0	Yes
8	USA	55.0	99600.0	No
9	India	42.0	80400.0	Yes

```
In [6]: dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 4 columns):
#   Column          Non-Null Count  Dtype
---  -
0   Region          10 non-null    object
1   Age             9 non-null     float64
2   Income          9 non-null     float64
3   Online Shopper  10 non-null    object
dtypes: float64(2), object(2)
memory usage: 448.0+ bytes
```

```
In [7]: dataset.describe()
```

```
Out[7]:
```

	Age	Income
count	9.000000	9.000000
mean	43.777778	76533.333333
std	7.693793	14718.695594
min	32.000000	57600.000000
25%	40.000000	64800.000000
50%	43.000000	73200.000000
75%	49.000000	86400.000000
max	55.000000	99600.000000

```
In [8]: dataset.head()
```

```
Out[8]:
```

	Region	Age	Income	Online Shopper
0	India	49.0	86400.0	No
1	Brazil	32.0	57600.0	Yes
2	USA	35.0	64800.0	No
3	Brazil	43.0	73200.0	No
4	USA	45.0	NaN	Yes

```
In [9]: dataset.tail()
```

```
Out[9]:
```

	Region	Age	Income	Online Shopper
5	India	40.0	69600.0	Yes
6	Brazil	NaN	62400.0	No
7	India	53.0	94800.0	Yes
8	USA	55.0	99600.0	No
9	India	42.0	80400.0	Yes

How To check null Value

`#isnull()` : This will return boolean value for every column in the data frame, i.e. if the value is null it returns True, and False values are other than null.

```
In [10]: dataset.isnull()
```

```
Out[10]:
```

	Region	Age	Income	Online Shopper
0	False	False	False	False
1	False	False	False	False
2	False	False	False	False
3	False	False	False	False
4	False	False	True	False
5	False	False	False	False
6	False	True	False	False
7	False	False	False	False
8	False	False	False	False
9	False	False	False	False

isnull(). sum() : This code will give you total number of null values in each features in the data frame.

```
In [11]: dataset.isnull().sum()
```

```
Out[11]: Region      0
Age              1
Income           1
Online Shopper   0
dtype: int64
```

```
In [27]: #dataset.fillna(0) #You can fill the missing values with the code df.fillna(0)
```

```
In [12]: dataset
```

```
Out[12]:
```

	Region	Age	Income	Online Shopper
0	India	49.0	86400.0	No
1	Brazil	32.0	57600.0	Yes
2	USA	35.0	64800.0	No
3	Brazil	43.0	73200.0	No
4	USA	45.0	NaN	Yes
5	India	40.0	69600.0	Yes
6	Brazil	NaN	62400.0	No
7	India	53.0	94800.0	Yes
8	USA	55.0	99600.0	No
9	India	42.0	80400.0	Yes

Splitting the attributes into independent and dependent attributes

```
In [13]: from sklearn.model_selection import train_test_split # used for splitting train
```

```
In [14]: X = dataset.iloc[:, :-1].values # attributes to determine Independent variable  
Y = dataset.iloc[:, -1].values # dependent variable / Class
```

```
In [15]: X
```

```
Out[15]: array([[ 'India', 49.0, 86400.0],  
                [ 'Brazil', 32.0, 57600.0],  
                [ 'USA', 35.0, 64800.0],  
                [ 'Brazil', 43.0, 73200.0],  
                [ 'USA', 45.0, nan],  
                [ 'India', 40.0, 69600.0],  
                [ 'Brazil', nan, 62400.0],  
                [ 'India', 53.0, 94800.0],  
                [ 'USA', 55.0, 99600.0],  
                [ 'India', 42.0, 80400.0]], dtype=object)
```

```
In [16]: X=dataset[['Region', 'Age', 'Income']] # attributes to determine Independent vari
```

```
In [17]: X # Display Independent variable / Class
```

```
Out[17]:
```

	Region	Age	Income
0	India	49.0	86400.0
1	Brazil	32.0	57600.0
2	USA	35.0	64800.0
3	Brazil	43.0	73200.0
4	USA	45.0	NaN
5	India	40.0	69600.0
6	Brazil	NaN	62400.0
7	India	53.0	94800.0
8	USA	55.0	99600.0
9	India	42.0	80400.0

```
In [18]: Y
```

```
Out[18]: array(['No', 'Yes', 'No', 'No', 'Yes', 'Yes', 'No', 'Yes', 'No', 'Yes'],  
              dtype=object)
```

```
In [19]: Y=dataset[['Online Shopper']] # attributes to determine Independent variable /
```

```
In [20]: Y
```

```
Out[20]:
```

	Online Shopper
0	No
1	Yes
2	No
3	No
4	Yes
5	Yes
6	No
7	Yes
8	No
9	Yes

Handling of Missing Data

#Well the first idea is to remove the lines in the observations where there is some missing data. But that can be quite dangerous because imagine this data set contains crucial information. It would be quite dangerous to remove an observation. So we need to figure out a better idea to

handle this problem. And another idea that's actually the most common idea to handle missing

```
#In dataset, we have two values missing, one for age column in 7th data row  
and for Income column in 5th data row. Missing values should be handled  
during the data analysis
```

```
In [53]: from sklearn.impute import SimpleImputer # used for handling missing data
```

Handling the missing data and replace missing values with nan from numpy and replace with mean of all the other values

```
In [56]: #imputer = SimpleImputer(missing_values=np.nan, strategy='mean')  
#imputer = imputer.fit(X[:, 1:])  
#X[:, 1:] = imputer.transform(X[:, 1:])
```

Filling missing values with 0 dataset.fillna(0)

```
In [21]: dataset
```

```
Out[21]:
```

	Region	Age	Income	Online Shopper
0	India	49.0	86400.0	No
1	Brazil	32.0	57600.0	Yes
2	USA	35.0	64800.0	No
3	Brazil	43.0	73200.0	No
4	USA	45.0	NaN	Yes
5	India	40.0	69600.0	Yes
6	Brazil	NaN	62400.0	No
7	India	53.0	94800.0	Yes
8	USA	55.0	99600.0	No
9	India	42.0	80400.0	Yes

```
In [24]: new_df = dataset.fillna(0) #Filling missing values with 0
```

```
In [25]: new_df
```

```
Out[25]:
```

	Region	Age	Income	Online Shopper
0	India	49.0	86400.0	No
1	Brazil	32.0	57600.0	Yes
2	USA	35.0	64800.0	No
3	Brazil	43.0	73200.0	No
4	USA	45.0	0.0	Yes
5	India	40.0	69600.0	Yes
6	Brazil	0.0	62400.0	No
7	India	53.0	94800.0	Yes
8	USA	55.0	99600.0	No
9	India	42.0	80400.0	Yes

Filling NaN values with forward fill value

```
In [ ]: #used method df.fillna(method="ffill") If we use forward fill that simply means
```

```
In [26]: new_df1 = dataset.fillna(method="ffill")  
new_df1
```

```
Out[26]:
```

	Region	Age	Income	Online Shopper
0	India	49.0	86400.0	No
1	Brazil	32.0	57600.0	Yes
2	USA	35.0	64800.0	No
3	Brazil	43.0	73200.0	No
4	USA	45.0	73200.0	Yes
5	India	40.0	69600.0	Yes
6	Brazil	40.0	62400.0	No
7	India	53.0	94800.0	Yes
8	USA	55.0	99600.0	No
9	India	42.0	80400.0	Yes

**#Filling NaN values in Backward Direction using
method df.fillna(method="bfill")new_df**

```
In [28]: new_df2 = dataset.fillna(method="bfill")
new_df2
```

```
Out[28]:
```

	Region	Age	Income	Online Shopper
0	India	49.0	86400.0	No
1	Brazil	32.0	57600.0	Yes
2	USA	35.0	64800.0	No
3	Brazil	43.0	73200.0	No
4	USA	45.0	69600.0	Yes
5	India	40.0	69600.0	Yes
6	Brazil	53.0	62400.0	No
7	India	53.0	94800.0	Yes
8	USA	55.0	99600.0	No
9	India	42.0	80400.0	Yes

```
In [29]: dataset
```

```
Out[29]:
```

	Region	Age	Income	Online Shopper
0	India	49.0	86400.0	No
1	Brazil	32.0	57600.0	Yes
2	USA	35.0	64800.0	No
3	Brazil	43.0	73200.0	No
4	USA	45.0	NaN	Yes
5	India	40.0	69600.0	Yes
6	Brazil	NaN	62400.0	No
7	India	53.0	94800.0	Yes
8	USA	55.0	99600.0	No
9	India	42.0	80400.0	Yes

Filling missing values Mean between value was a NaN value(Mean of Previous NAN and after NAN

```
In [30]: new_df3 = dataset.interpolate()
```


In [31]: new_df3

Out[31]:

	Region	Age	Income	Online Shopper
0	India	49.0	86400.0	No
1	Brazil	32.0	57600.0	Yes
2	USA	35.0	64800.0	No
3	Brazil	43.0	73200.0	No
4	USA	45.0	71400.0	Yes
5	India	40.0	69600.0	Yes
6	Brazil	46.5	62400.0	No
7	India	53.0	94800.0	Yes
8	USA	55.0	99600.0	No
9	India	42.0	80400.0	Yes

In [34]: dataset

Out[34]:

	Region	Age	Income	Online Shopper
0	India	49.0	86400.0	No
1	Brazil	32.0	57600.0	Yes
2	USA	35.0	64800.0	No
3	Brazil	43.0	73200.0	No
4	USA	45.0	NaN	Yes
5	India	40.0	69600.0	Yes
6	Brazil	NaN	62400.0	No
7	India	53.0	94800.0	Yes
8	USA	55.0	99600.0	No
9	India	42.0	80400.0	Yes

Dropna() used to delete those rows were continuing NaN values were dropped.

```
In [32]: new_df4 = dataset.dropna()  
new_df4
```

```
Out[32]:
```

	Region	Age	Income	Online Shopper
0	India	49.0	86400.0	No
1	Brazil	32.0	57600.0	Yes
2	USA	35.0	64800.0	No
3	Brazil	43.0	73200.0	No
5	India	40.0	69600.0	Yes
7	India	53.0	94800.0	Yes
8	USA	55.0	99600.0	No
9	India	42.0	80400.0	Yes

```
In [33]: dataset
```

```
Out[33]:
```

	Region	Age	Income	Online Shopper
0	India	49.0	86400.0	No
1	Brazil	32.0	57600.0	Yes
2	USA	35.0	64800.0	No
3	Brazil	43.0	73200.0	No
4	USA	45.0	NaN	Yes
5	India	40.0	69600.0	Yes
6	Brazil	NaN	62400.0	No
7	India	53.0	94800.0	Yes
8	USA	55.0	99600.0	No
9	India	42.0	80400.0	Yes

Deleting the rows having all NaN values

`new_df = df.dropna(how='all')` Those rows in which all the values are NaN values will be deleted.
If the row even has one value even then it will not be dropped.

```
In [35]: new_df5 = dataset.dropna(how='all')
new_df5
```

```
Out[35]:
```

	Region	Age	Income	Online Shopper
0	India	49.0	86400.0	No
1	Brazil	32.0	57600.0	Yes
2	USA	35.0	64800.0	No
3	Brazil	43.0	73200.0	No
4	USA	45.0	NaN	Yes
5	India	40.0	69600.0	Yes
6	Brazil	NaN	62400.0	No
7	India	53.0	94800.0	Yes
8	USA	55.0	99600.0	No
9	India	42.0	80400.0	Yes

encode categorical data

```
In [51]: from sklearn.compose import ColumnTransformer # encode categorical data to disc
from sklearn.preprocessing import LabelEncoder, OneHotEncoder # used for encoding
```

```
In [ ]:
```

splitting the dataset into training set and test set

```
In [37]: from sklearn.model_selection import train_test_split # used for splitting train
```

```
In [38]: X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random
```

```
In [39]: X_train
```

```
Out[39]:
```

	Region	Age	Income
9	India	42.0	80400.0
1	Brazil	32.0	57600.0
6	Brazil	NaN	62400.0
7	India	53.0	94800.0
3	Brazil	43.0	73200.0
0	India	49.0	86400.0
5	India	40.0	69600.0

In [41]: Y_train

Out[41]:

	Online Shopper
9	Yes
1	Yes
6	No
7	Yes
3	No
0	No
5	Yes

In [42]: X_test

Out[42]:

	Region	Age	Income
2	USA	35.0	64800.0
8	USA	55.0	99600.0
4	USA	45.0	NaN

In [43]: Y_test

Out[43]:

	Online Shopper
2	No
8	No
4	Yes

In []: