LAB 2 : Data Preprocessing Tools

Mohammed Meraj Mohammed Ashfaque

T2 32

TY Computer

Dataset Used : concert.csv

Import Liabraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Import Dataset

In [28]:
```
dataset = pd.read_csv("concert.csv")
```

EDA Steps

In [29]:
```
dataset.head()
```

Out[29]:

	Concert Name	Start Time	Tickets Sold	Ticket Price	Sold Out
0	Rock Revolution	6	52000	1250	Yes
1	Jazz Nights	8	18500	1000	No
2	Classical Waves	3	9800	1100	No
3	Metal Thunder	10	15000	1300	Yes
4	Pop Sensation	5	77000	950	No

In [30]:
```
dataset.columns
```

Out[30]:
```
Index(['Concert Name', 'Start Time', 'Tickets Sold', 'Ticket Price',
       'Sold Out'],
      dtype='object')
```

In [31]:
```
dataset.shape
```

Out[31]:
```
(8, 5)
```

In [32]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8 entries, 0 to 7
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   Concert Name  8 non-null      object
 1   Start Time    8 non-null      int64 
 2   Tickets Sold  8 non-null      int64 
 3   Ticket Price  8 non-null      int64 
 4   Sold Out      8 non-null      object
dtypes: int64(3), object(2)
memory usage: 452.0+ bytes
```

In [33]:
```
# for numerical feaatures
dataset.describe()
```

Out[33]:

	Start Time	Tickets Sold	Ticket Price
count	8.00000	8.000000	8.000000
mean	6.50000	27137.500000	1056.250000
std	2.44949	24579.720881	174.104853
min	3.00000	8600.000000	800.000000
25%	4.75000	10850.000000	937.500000
50%	6.50000	16750.000000	1050.000000
75%	8.25000	31750.000000	1175.000000
max	10.00000	77000.000000	1300.000000

In [35]:
```
# categorical features
dataset.describe(include = object)
```

Out[35]:

	Concert Name	Sold Out
count	8	8
unique	8	2
top	Rock Revolution	No
freq	1	5

Preprocessing Steps

Step 1 : Divide dataframe into input and output Featutres

In [42]:
```
# dataset.iloc [row_index,column_index]
# dataset.iloc[: , :-1]
X = dataset.iloc[:,:-1]
Y = dataset.iloc[:,-1]
```

In [43]:
```
print(X)
```

```
      Concert Name  Start Time  Tickets Sold  Ticket Price
0   Rock Revolution           6         52000          1250
1       Jazz Nights           8         18500          1000
2   Classical Waves           3          9800          1100
3     Metal Thunder          10         15000          1300
4     Pop Sensation           5         77000           950
5       Indie Vibes           4         11200          1150
6     Blues Evening           7         25000           800
7  Electronic Beats           9          8600           900
```

In [44]:
```
print(Y)
```

```
0    Yes
1     No
2     No
3    Yes
4     No
5    Yes
6     No
7     No
Name: Sold Out, dtype: object
```

Step 2: Handle the missing values in Dataset

In [45]:
```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values=np.nan,
                        strategy='mean')
imputer.fit(X.iloc[:,1:3])
X.iloc[:,1:3] = imputer.transform(X.iloc[:,1:3])
```

In [46]:
```
print(X)
```

```
       Concert Name  Start Time  Tickets Sold  Ticket Price
0   Rock Revolution           6         52000          1250
1       Jazz Nights           8         18500          1000
2   Classical Waves           3          9800          1100
3     Metal Thunder          10         15000          1300
4     Pop Sensation           5         77000           950
5       Indie Vibes           4         11200          1150
6     Blues Evening           7         25000           800
7  Electronic Beats           9          8600           900
```

Step 3 : Encoding categorical data

A. Encoding the Independent Variable (i/p feature/X)

In X we have Concert Name as categorical feature

It has 8 categories

Hence used One hot encoder

In [47]:
```
X['Concert Name'].value_counts()
```

Out[47]:
```
Concert Name
Rock Revolution     1
Jazz Nights         1
Classical Waves     1
Metal Thunder       1
Pop Sensation       1
Indie Vibes         1
Blues Evening       1
Electronic Beats    1
Name: count, dtype: int64
```

In [48]:
```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder

ct = ColumnTransformer(transformers=[('encoder',
                                      OneHotEncoder(),
                                      [0])],
                       remainder='passthrough')
X = np.array(ct.fit_transform(X))
```

In [49]:
```
print(X)
```

```
[[0.00e+00 0.00e+00 0.00e+00 0.00e+00 0.00e+00 0.00e+00 0.00e+00 1.00e+00
  6.00e+00 5.20e+04 1.25e+03]
 [0.00e+00 0.00e+00 0.00e+00 0.00e+00 1.00e+00 0.00e+00 0.00e+00 0.00e+00
  8.00e+00 1.85e+04 1.00e+03]
 [0.00e+00 1.00e+00 0.00e+00 0.00e+00 0.00e+00 0.00e+00 0.00e+00 0.00e+00
  3.00e+00 9.80e+03 1.10e+03]
 [0.00e+00 0.00e+00 0.00e+00 0.00e+00 0.00e+00 1.00e+00 0.00e+00 0.00e+00
  1.00e+01 1.50e+04 1.30e+03]
 [0.00e+00 0.00e+00 0.00e+00 0.00e+00 0.00e+00 0.00e+00 1.00e+00 0.00e+00
  5.00e+00 7.70e+04 9.50e+02]
 [0.00e+00 0.00e+00 0.00e+00 1.00e+00 0.00e+00 0.00e+00 0.00e+00 0.00e+00
  4.00e+00 1.12e+04 1.15e+03]
 [1.00e+00 0.00e+00 0.00e+00 0.00e+00 0.00e+00 0.00e+00 0.00e+00 0.00e+00
  7.00e+00 2.50e+04 8.00e+02]
 [0.00e+00 0.00e+00 1.00e+00 0.00e+00 0.00e+00 0.00e+00 0.00e+00 0.00e+00
  9.00e+00 8.60e+03 9.00e+02]]
```

Encoding the Dependent Variable (o/p feature/ target / Y)

In [50]:
```
Y.value_counts()
```

Out[50]:
```
Sold Out
No     5
Yes    3
Name: count, dtype: int64
```

As it has only two categoris

Hence used Label Encoder

In [51]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
Y = le.fit_transform(Y)
```

In [52]:
```
print(Y)
```

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

Step 4 : Splitting Data into Training and Testing

In [55]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test = train_test_split(X,
                                                 Y,
                                          test_size =0.3,
                                          random_state = 1)
```

In [56]:
```
print(X_train)
```

```
[[1.00e+00 0.00e+00 0.00e+00 0.00e+00 0.00e+00 0.00e+00 0.00e+00 0.00e+00
  7.00e+00 2.50e+04 8.00e+02]
 [0.00e+00 0.00e+00 0.00e+00 0.00e+00 0.00e+00 0.00e+00 0.00e+00 1.00e+00
  6.00e+00 5.20e+04 1.25e+03]
 [0.00e+00 0.00e+00 0.00e+00 0.00e+00 0.00e+00 0.00e+00 1.00e+00 0.00e+00
  5.00e+00 7.70e+04 9.50e+02]
 [0.00e+00 0.00e+00 0.00e+00 0.00e+00 0.00e+00 1.00e+00 0.00e+00 0.00e+00
  1.00e+01 1.50e+04 1.30e+03]
 [0.00e+00 0.00e+00 0.00e+00 1.00e+00 0.00e+00 0.00e+00 0.00e+00 0.00e+00
  4.00e+00 1.12e+04 1.15e+03]]
```

Step 5 : Feature Scaling

In [57]:
```
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 [58]:
```
print(X_train)
```

```
[[ 1.          0.          0.         -0.5         0.         -0.5
  -0.5        -0.5         0.29138576 -0.442377   -1.54133261]
 [ 0.          0.          0.         -0.5         0.         -0.5
  -0.5         2.         -0.19425717  0.63952327  0.85039041]
 [ 0.          0.          0.         -0.5         0.         -0.5
   2.         -0.5        -0.6799001   1.64128279 -0.7440916 ]
 [ 0.          0.          0.         -0.5         0.          2.
  -0.5        -0.5         1.74831455 -0.84308081  1.11613741]
 [ 0.          0.          0.          2.          0.         -0.5
  -0.5        -0.5        -1.16554303 -0.99534825  0.3188964 ]]
```

In [ ]:
```

```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js