**PRODUCT DEMAND PREDICTION USING MACHINE LEARNING**

STEP01: Loading Dataset and performing various data preprocessing techniques

# importing libraries  
import pandas as pd  
import scipy  
import numpy as np  
from sklearn.preprocessing import MinMaxScaler  
import seaborn as sns  
import matplotlib.pyplot as plt

# Loading Dataset  
df=pd.read\_csv('PoductDemand.csv')

df.info()

<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 150150 entries, 0 to 150149  
Data columns (total 5 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 ID 150150 non-null int64   
 1 Store ID 150150 non-null int64   
 2 Total Price 150149 non-null float64  
 3 Base Price 150150 non-null float64  
 4 Units Sold 150150 non-null int64   
dtypes: float64(2), int64(3)  
memory usage: 5.7 MB

df.shape

(150150, 5)

#STASTICAL ANYALSIS  
df.describe()

ID Store ID Total Price Base Price \  
count 150150.000000 150150.000000 150149.000000 150150.000000   
mean 106271.555504 9199.422511 206.626751 219.425927   
std 61386.037861 615.591445 103.308516 110.961712   
min 1.000000 8023.000000 41.325000 61.275000   
25% 53111.250000 8562.000000 130.387500 133.237500   
50% 106226.500000 9371.000000 198.075000 205.912500   
75% 159452.750000 9731.000000 233.700000 234.412500   
max 212644.000000 9984.000000 562.162500 562.162500   
  
 Units Sold   
count 150150.000000   
mean 51.674206   
std 60.207904   
min 1.000000   
25% 20.000000   
50% 35.000000

75% 62.000000   
max 2876.000000

df.head(5)

ID Store ID Total Price Base Price Units Sold  
0 1 8091 99.0375 111.8625 20  
1 2 8091 99.0375 99.0375 28  
2 3 8091 133.9500 133.9500 19  
3 4 8091 133.9500 133.9500 44  
4 5 8091 141.0750 141.0750 52

df.tail(10)

ID Store ID Total Price Base Price Units Sold  
150140 212633 9984 327.0375 327.0375 15  
150141 212634 9984 163.8750 210.9000 204  
150142 212635 9984 205.9125 205.9125 20  
150143 212636 9984 205.9125 205.9125 12  
150144 212637 9984 239.4000 239.4000 23  
150145 212638 9984 235.8375 235.8375 38  
150146 212639 9984 235.8375 235.8375 30  
150147 212642 9984 357.6750 483.7875 31  
150148 212643 9984 141.7875 191.6625 12  
150149 212644 9984 234.4125 234.4125 15

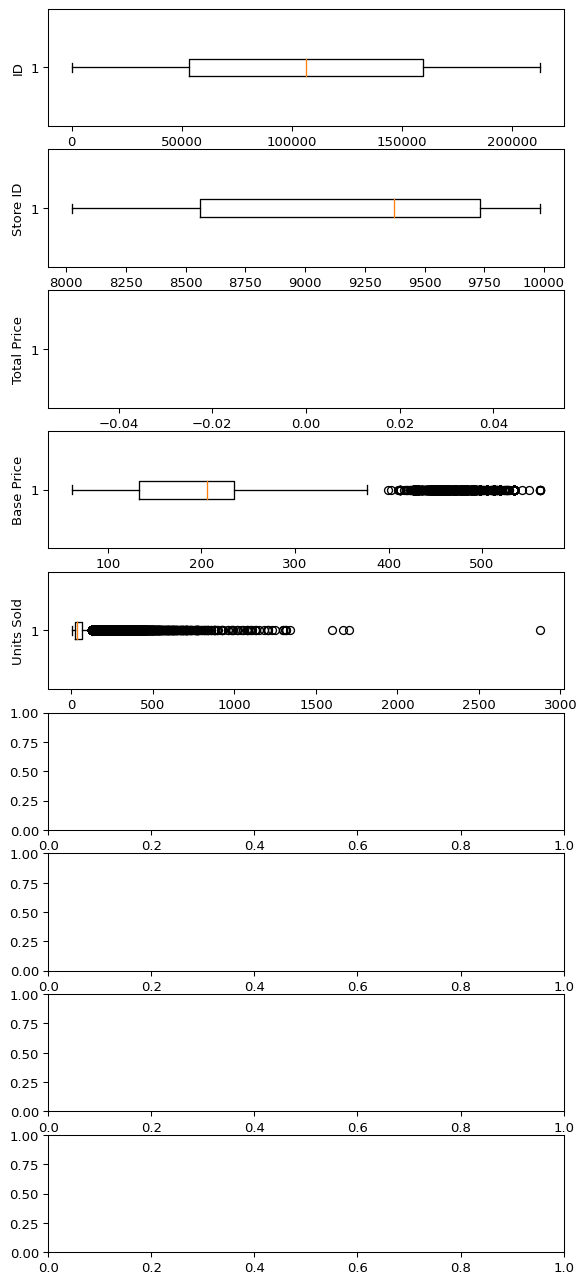
df.isnull().sum

<bound method NDFrame.\_add\_numeric\_operations.<locals>.sum of ID Store ID Total Price Base Price Units Sold  
0 False False False False False  
1 False False False False False  
2 False False False False False  
3 False False False False False  
4 False False False False False  
... ... ... ... ... ...  
150145 False False False False False  
150146 False False False False False  
150147 False False False False False  
150148 False False False False False  
150149 False False False False False  
  
[150150 rows x 5 columns]>

# Checking The Outliers  
fig, axs = plt.subplots(9,1,dpi=95, figsize=(7,17))  
i = 0  
for col in df.columns:  
axs[i].boxplot(df[col], vert=False)  
axs[i].set\_ylabel

(col)i+=1

plt.show()



#drop the outliers  
# Identify the quartiles  
q1, q3 = np.percentile(df['Base Price'], [25, 75])  
# Calculate the interquartile range  
iqr = q3 - q1  
# Calculate the lower and upper bounds  
lower\_bound = q1 - (1.5 \* iqr)  
upper\_bound = q3 + (1.5 \* iqr)  
# Drop the outliers  
clean\_data = df[(df['Base Price'] >= lower\_bound)   
 & (df['Base Price'] <= upper\_bound)]  
  
# Identify the quartiles  
q1, q3 = np.percentile(clean\_data['Total Price'], [25, 75])  
# Calculate the interquartile range  
iqr = q3 - q1  
# Calculate the lower and upper bounds  
lower\_bound = q1 - (1.5 \* iqr)  
upper\_bound = q3 + (1.5 \* iqr)  
# Drop the outliers  
clean\_data = clean\_data[(clean\_data['Total Price'] >= lower\_bound)   
 & (clean\_data['Total Price'] <= upper\_bound)]  
# Identify the quartiles  
q1, q3 = np.percentile(clean\_data['Units Sold'], [25, 75])  
# Calculate the interquartile range  
iqr = q3 - q1  
# Calculate the lower and upper bounds  
lower\_bound = q1 - (1.5 \* iqr)  
upper\_bound = q3 + (1.5 \* iqr)  
# Drop the outliers  
clean\_data = clean\_data[(clean\_data['Units Sold'] >= lower\_bound)   
 & (clean\_data['Units Sold'] <= upper\_bound)]

#CORRELATION  
corr = df.corr()  
   
plt.figure(dpi=130)  
sns.heatmap(df.corr(), annot=True, fmt= '.2f')  
plt.show()



#Separate independent features and Target Variables  
  
# separate array into input and output components  
X = df.drop(columns =['Units Sold'])

# Normalization  
  
# initialising the MinMaxScaler  
scaler = MinMaxScaler(feature\_range=(0, 1))  
   
# learning the statistical parameters for each of the data and transforming  
rescaledX = scaler.fit\_transform(X)  
rescaledX[:5]

array([[0.00000000e+00, 3.46761856e-02, 1.10807114e-01, 1.00995733e-01],  
 [4.70271770e-06, 3.46761856e-02, 1.10807114e-01, 7.53911807e-02],  
 [9.40543540e-06, 3.46761856e-02, 1.77838577e-01, 1.45092461e-01],  
 [1.41081531e-05, 3.46761856e-02, 1.77838577e-01, 1.45092461e-01],  
 [1.88108708e-05, 3.46761856e-02, 1.91518468e-01, 1.59317212e-01]])

#Standardization  
from sklearn.preprocessing import StandardScaler  
   
scaler = StandardScaler().fit(X)  
rescaledX = scaler.transform(X)  
rescaledX[:5]

array([[-1.73119024, -1.80058741, -1.04143988, -0.96937748],  
 [-1.73117395, -1.80058741, -1.04143988, -1.08495828],  
 [-1.73115766, -1.80058741, -0.70349469, -0.77032167],  
 [-1.73114137, -1.80058741, -0.70349469, -0.77032167],  
 [-1.73112507, -1.80058741, -0.63452629, -0.70611012]])