Retail

September 4, 2022

1 Importing dataset and libraries

[1]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[30]:

```
df=pd.read_csv('test_data_hidden.csv')
# df= pd.read_excel('train.xlsx')
df.head()
```

[30]:

```
   Store  DayOfWeek        Date  Sales  Customers  Open  Promo  StateHoliday  \
0      1          5  2015-07-31   5263        555     1      1             0
1      2          5  2015-07-31   6064        625     1      1             0
2      3          5  2015-07-31   8314        821     1      1             0
3      4          5  2015-07-31  13995       1498     1      1             0
4      5          5  2015-07-31   4822        559     1      1             0

   SchoolHoliday
0              1
1              1
2              1
3              1
4              1
```

2 EDA and Visualization

[31]:

```
# shape of dataset
print('Shape of dataset')
print('*'*30)
print(f'rows : {df.shape[0]} ')
print(f'columns : {df.shape[1]}')
```

```
Shape of dataset
******************************
rows : 34565
columns : 9
```

[32]:

```
# info
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 34565 entries, 0 to 34564
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Store          34565 non-null  int64
 1   DayOfWeek      34565 non-null  int64
 2   Date           34565 non-null  object
 3   Sales          34565 non-null  int64
 4   Customers      34565 non-null  int64
 5   Open           34565 non-null  int64
 6   Promo          34565 non-null  int64
 7   StateHoliday   34565 non-null  int64
 8   SchoolHoliday  34565 non-null  int64
dtypes: int64(8), object(1)
memory usage: 2.4+ MB
```

[33]:

```
# nan values in dataset
df.isna().sum()
```

[33]:

```
Store            0
DayOfWeek        0
Date             0
Sales            0
Customers        0
Open             0
Promo            0
StateHoliday     0
SchoolHoliday    0
dtype: int64
```

[34]:

```
# no missing values
```

[35]:

```
#unique values
df['StateHoliday'].unique()
```

[35]:

```
array([0])
```

[36]:

```
sh={'0':0, 'a':1, 'b':2, 'c':3,0:0}
```

[37]:

```
df['StateHoliday']=df['StateHoliday'].map(sh)
```

[38]:

```
df['SchoolHoliday'].unique()
```

[38]:

```
array([1, 0])
```

[39]:

```
# statistical info
df.describe().T
```

[39]:

```
                 count         mean          std  min     25%     50%     75%  \
Store          34565.0   558.000000   321.877302  1.0   279.0   558.0   837.0
DayOfWeek      34565.0     4.000000     1.917688  1.0     2.0     4.0     6.0
Sales          34565.0  6142.705511  3606.356960  0.0  4325.0  6085.0  8063.0
Customers      34565.0   643.827224   435.207851  0.0   445.0   610.0   812.0
Open           34565.0     0.873369     0.332564  0.0     1.0     1.0     1.0
Promo          34565.0     0.419355     0.493461  0.0     0.0     0.0     1.0
StateHoliday   34565.0     0.000000     0.000000  0.0     0.0     0.0     0.0
SchoolHoliday  34565.0     0.369651     0.482717  0.0     0.0     0.0     1.0

                   max
Store           1115.0
DayOfWeek          7.0
Sales          32547.0
Customers       4783.0
Open               1.0
Promo              1.0
StateHoliday       0.0
SchoolHoliday      1.0
```

[40]:

```
df.head()
```

[40]:

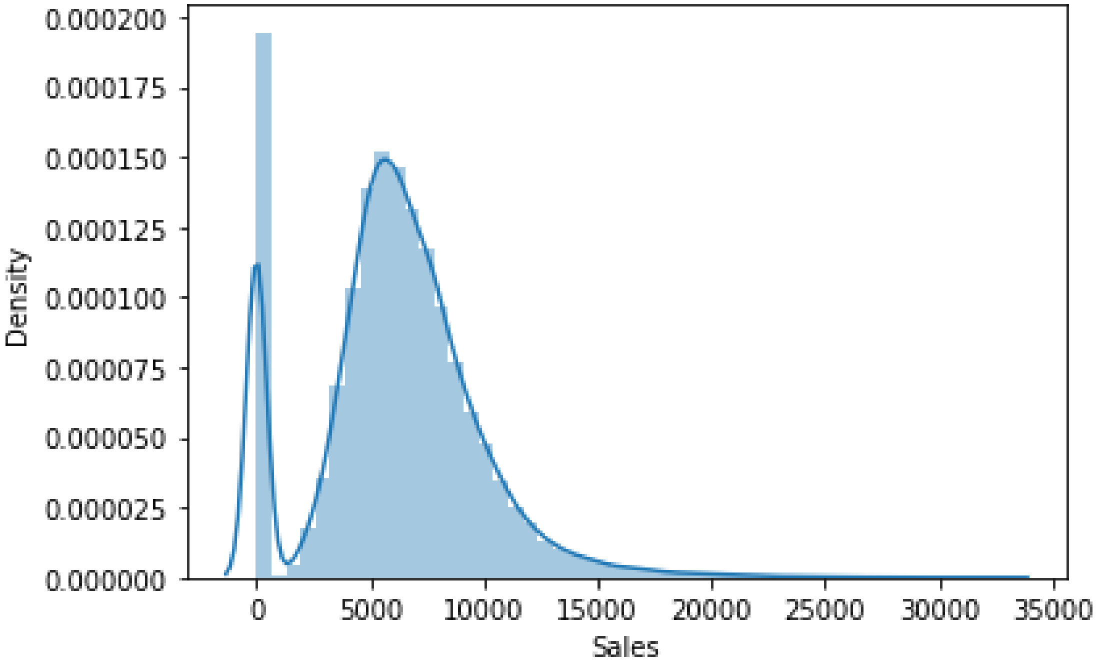
```
   Store  DayOfWeek        Date  Sales  Customers  Open  Promo  StateHoliday  \
0      1          5  2015-07-31   5263        555     1      1             0
1      2          5  2015-07-31   6064        625     1      1             0
2      3          5  2015-07-31   8314        821     1      1             0
3      4          5  2015-07-31  13995       1498     1      1             0
4      5          5  2015-07-31   4822        559     1      1             0

   SchoolHoliday
0              1
1              1
2              1
3              1
4              1
```

[41]:

```
# distribution plot
sns.distplot(df['Sales'])
```

```
/usr/local/lib/python3.9/site-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

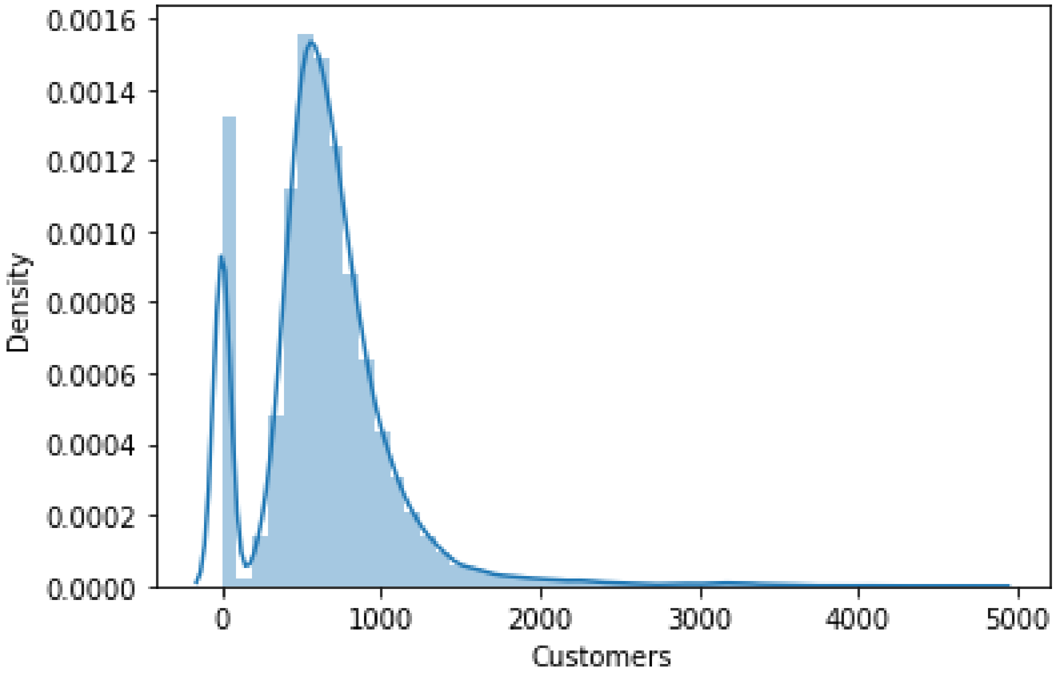
```
[41]: <AxesSubplot:xlabel='Sales', ylabel='Density'>
```



```
[42]: sns.distplot(df['Customers'])
```

```
/usr/local/lib/python3.9/site-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

```
[42]: <AxesSubplot:xlabel='Customers', ylabel='Density'>
```



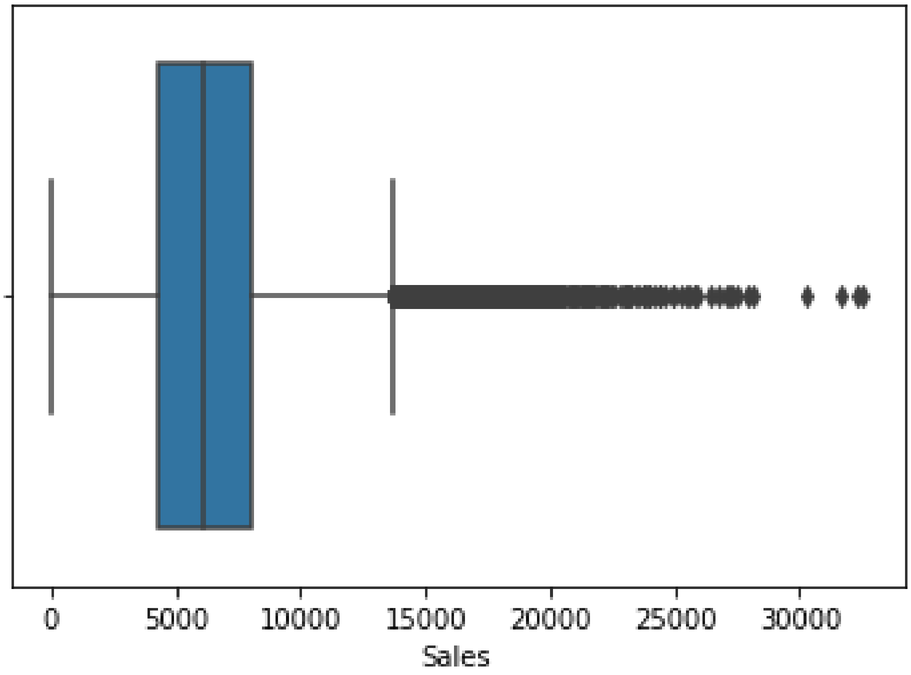
[43]:

```
# boxplots
sns.boxplot(df['Sales'])
```

```
/usr/local/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

[43]:

```
<AxesSubplot:xlabel='Sales'>
```



```
[44]: #feature engineering
      df['day']=pd.to_datetime(df['Date'], format='%Y-%m-%d').dt.day
      df['month']=pd.to_datetime(df['Date'], format='%Y-%m-%d').dt.month
      df['year']=pd.to_datetime(df['Date'], format='%Y-%m-%d').dt.year
```

```
[45]: df.head()
```

```
[45]:    Store  DayOfWeek        Date  Sales  Customers  Open  Promo  StateHoliday  \
      0      1          5  2015-07-31   5263        555     1      1             0
      1      2          5  2015-07-31   6064        625     1      1             0
      2      3          5  2015-07-31   8314        821     1      1             0
      3      4          5  2015-07-31  13995       1498     1      1             0
      4      5          5  2015-07-31   4822        559     1      1             0

         SchoolHoliday  day  month  year
      0              1   31      7  2015
      1              1   31      7  2015
      2              1   31      7  2015
      3              1   31      7  2015
      4              1   31      7  2015
```

```
[46]: df.tail()
```

```
[46]:        Store  DayOfWeek        Date  Sales  Customers  Open  Promo  \
      34560   1111          3  2015-07-01   3701        351     1      1
      34561   1112          3  2015-07-01  10620        716     1      1
      34562   1113          3  2015-07-01   8222        770     1      1
      34563   1114          3  2015-07-01  27071       3788     1      1
      34564   1115          3  2015-07-01   7701        447     1      1

             StateHoliday  SchoolHoliday  day  month  year
      34560             0              1    1      7  2015
      34561             0              1    1      7  2015
      34562             0              0    1      7  2015
      34563             0              0    1      7  2015
      34564             0              0    1      7  2015
```

```
[47]: df.drop('Date', axis=1, inplace=True)
```

```
[48]: df.head()
```

```
[48]:    Store  DayOfWeek  Sales  Customers  Open  Promo  StateHoliday  \
      0      1          5   5263        555     1      1             0
      1      2          5   6064        625     1      1             0
      2      3          5   8314        821     1      1             0
      3      4          5  13995       1498     1      1             0
      4      5          5   4822        559     1      1             0

         SchoolHoliday  day  month  year
      0              1   31      7  2015
      1              1   31      7  2015
      2              1   31      7  2015
      3              1   31      7  2015
      4              1   31      7  2015
```

```
[49]: df['StateHoliday'].unique()
```

```
[49]: array([0])
```

```
[50]: y=df['Sales']
      X=df.drop('Sales', axis=1)
```

3 Model Building

```
[51]: from sklearn.model_selection import cross_val_score, train_test_split
      X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,␣
       ↪random_state=42)
```

```
[52]: # linear regression
      from sklearn.linear_model import LinearRegression
```

```
lr=LinearRegression()
lr.fit(X_train,y_train)
pred_lr=lr.predict(X_test)
score_lr=cross_val_score(lr,X,y,cv=5)
print(score_lr)
```

```
[0.8272264  0.74292577 0.79467188 0.8325215  0.82853658]
```

[53]:

```
score_lr.mean()
```

[53]:

```
0.8051764243690333
```

[54]:

```
from sklearn.metrics import mean_absolute_error, mean_squared_error
mae_lr=mean_absolute_error(y_test,pred_lr)
print(mae_lr)
mse_lr=mean_squared_error(y_test,pred_lr)
print(mae_lr)
```

```
1020.9858020361147
1020.9858020361147
```

[55]:

```
from sklearn.tree import DecisionTreeRegressor
dt=DecisionTreeRegressor()
dt.fit(X_train,y_train)
pred_dt=dt.predict(X_test)
score_dt=cross_val_score(dt,X,y,cv=5)
print(score_dt)
```

```
[0.79298482 0.72923396 0.78229266 0.81046471 0.8135094 ]
```

[56]:

```
score_dt.mean()
```

[56]:

```
0.785697108855009
```

[57]:

```
mae_dt=mean_absolute_error(y_test,pred_dt)
print(mae_dt)
mse_dt=mean_squared_error(y_test,pred_dt)
print(mae_dt)
```

```
1031.1549255026762
1031.1549255026762
```

[71]:

```
from sklearn.ensemble import RandomForestRegressor
rf=RandomForestRegressor()
rf.fit(X_train,y_train)
pred_rf=rf.predict(X_test)
score_rf=cross_val_score(rf,X,y,cv=4)
print(score_rf)
```

```
[0.86766729 0.90337174 0.89618014 0.89128587]
```

[72]:
```
score_rf.mean()
```

[72]:
```
0.8896262596783888
```

[73]:
```
mae_rf=mean_absolute_error(y_test,pred_rf)
print(mae_rf)
mse_rf=mean_squared_error(y_test,pred_rf)
print(mae_rf)
```

```
816.019401128309
816.019401128309
```

[74]:
```
# best model is Random forest regressor with accuracy with greater than 89%
```

[ ]:
```

```