

amazon-sales-dataset

June 21, 2024

1 Amazon Sales Data - Analysis

- 1. Load libraries and read the data
 - 1.1. Load libraries
 - 1.2. Read the data
 - 1.3. Missing values
- 2. Feature engineering and selection
 - 2.1. Checking for anomalies
 - 2.2. Creating new variable
 - 2.3. Feature Encoding
 - 2.4. Drop redundant features
 - 2.5. Feature Scaling
- 3. Exploratory Data Analysis (EDA)
 - 3.1. Histograms
 - 3.2. Correlation Heatmap

2 1. Load libraries and read the data

2.1 1.1. Load libraries

```
[1]: import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import zscore
import warnings
warnings.filterwarnings('ignore')
```

2.2 1.2. Read the data

```
[3]: df = pd.read_csv('D:/unified mentor/um inside/Amazon Sales data.csv')
```

```
[4]: #Check for the first five rows of the dataset
df.head()
```

```
[4]:
```

	Region	Country	Item Type	\
0	Australia and Oceania	Tuvalu	Baby Food	
1	Central America and the Caribbean	Grenada	Cereal	
2	Europe	Russia	Office Supplies	
3	Sub-Saharan Africa	Sao Tome and Principe	Fruits	
4	Sub-Saharan Africa	Rwanda	Office Supplies	

	Sales Channel	Order Priority	Order Date	Order ID	Ship Date	Units Sold	\
0	Offline	H	5/28/2010	669165933	6/27/2010	9925	
1	Online	C	8/22/2012	963881480	9/15/2012	2804	
2	Offline	L	5/2/2014	341417157	5/8/2014	1779	
3	Online	C	6/20/2014	514321792	7/5/2014	8102	
4	Offline	L	2/1/2013	115456712	2/6/2013	5062	

	Unit Price	Unit Cost	Total Revenue	Total Cost	Total Profit
0	255.28	159.42	2533654.00	1582243.50	951410.50
1	205.70	117.11	576782.80	328376.44	248406.36
2	651.21	524.96	1158502.59	933903.84	224598.75
3	9.33	6.92	75591.66	56065.84	19525.82
4	651.21	524.96	3296425.02	2657347.52	639077.50

```
[5]: #Rows and columns of the dataset
df.shape
# There are 100 rows and 14 columns
```

```
[5]: (100, 14)
```

```
[ ]: df.describe().T
```

2.3 1.3. Missing values

```
[6]: print(f'The number of null values in the dataset is {df.isnull().sum().sum()}')
```

The number of null values in the dataset is 0

```
[7]: print(f'The number of duplicated rows in the dataset is {df.duplicated().sum()}')
```

The number of duplicated rows in the dataset is 0

3 2. Feature engineering and selection

3.1 2.1. Checking for anomalies

```
[8]: cat_var = ['Region', 'Country', 'Item Type', 'Sales Channel', 'Order Priority']
```

```
[9]: # Lets check for the number of each of the labels in the categorical variables.  
# This is also helpful to check for any redundant or anomaly in the data.
```

```
for index in range(2,5):  
    vc = df[cat_var[index]].value_counts()  
    print(vc, '\n')
```

```
Clothes          13  
Cosmetics        13  
Office Supplies  12  
Fruits           10  
Personal Care    10  
Household        9  
Beverages        8  
Baby Food        7  
Cereal            7  
Vegetables       6  
Snacks            3  
Meat              2  
Name: Item Type, dtype: int64
```

```
Offline         50  
Online          50  
Name: Sales Channel, dtype: int64
```

```
H      30  
L      27  
C      22  
M      21  
Name: Order Priority, dtype: int64
```

3.2 2.2. Creating new variable

```
[10]: #Lets convert the order date and ship date to datetime  
df['Order Date'] = pd.to_datetime(df['Order Date'])  
df['Ship Date'] = pd.to_datetime(df['Ship Date'])
```

```
[11]: # Adding a new feature 'Days Shipping' to the dataset 'df' which calculates the  
# number of days for the order to ship.  
df['Days Shipping'] = df['Ship Date'] - df['Order Date']
```

3.3 2.3. Feature Encoding

```
[12]: df['Sales Channel'] = df['Sales Channel'].str.replace('Offline', '0')
df['Sales Channel'] = df['Sales Channel'].str.replace('Online', '1')
```

```
[13]: df['Order Priority'] = df['Order Priority'].str.replace('C', '0')
df['Order Priority'] = df['Order Priority'].str.replace('L', '1')
df['Order Priority'] = df['Order Priority'].str.replace('M', '2')
df['Order Priority'] = df['Order Priority'].str.replace('H', '3')
```

```
[14]: df[['Sales Channel', 'Order Priority']] = df[['Sales Channel', 'Order Priority']].astype(int)
```

```
[15]: # Encoding a categorical variable using categorical encoder
for feature in cat_var:
    if feature == 'Item Type':
        print('\n')
        print('feature:', feature)
        print(pd.Categorical(df[feature].unique()))
        print(pd.Categorical(df[feature].unique()).codes)
        df[feature] = pd.Categorical(df[feature]).codes
```

```
feature: Item Type
['Baby Food', 'Cereal', 'Office Supplies', 'Fruits', 'Household', ...,
'Clothes', 'Cosmetics', 'Beverages', 'Meat', 'Snacks']
Length: 12
Categories (12, object): ['Baby Food', 'Beverages', 'Cereal', 'Clothes', ...,
'Office Supplies', 'Personal Care', 'Snacks', 'Vegetables']
[ 0  2  8  5  6 11  9  3  4  1  7 10]
```

3.4 2.4. Drop redundant features

```
[16]: # Dropping some redundant features
df.drop(['Country', 'Region', 'Order ID', 'Ship Date', 'Order Date'], axis=1, u
        ↪inplace=True)
```

```
[17]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Item Type        100 non-null    int8   
 1   Sales Channel    100 non-null    int32  
 2   Order Priority   100 non-null    int32  

```

```

3   Units Sold      100 non-null    int64
4   Unit Price     100 non-null    float64
5   Unit Cost      100 non-null    float64
6   Total Revenue   100 non-null    float64
7   Total Cost      100 non-null    float64
8   Total Profit    100 non-null    float64
9   Days Shipping   100 non-null    timedelta64[ns]
dtypes: float64(5), int32(2), int64(1), int8(1), timedelta64[ns](1)
memory usage: 6.5 KB

```

3.5 2.5. Feature Scaling

```
[18]: # Selecting specific numerical type data from the dataset for scaling.
num_dtypes = df.select_dtypes(['int8', 'int32', 'int64', 'float64'])
```

```
[19]: # Scaling the data using z-score
df[num_dtypes.columns] = num_dtypes.apply(zscore)
```

```
[20]: df.head()
# Data is scaled. The value range lies in the range -1 to +1.
```

```

[20]:   Item Type  Sales Channel  Order Priority  Units Sold  Unit Price \
0   -1.617958           -1.0      1.245352    1.724988  -0.091639
1   -0.987175           1.0      -1.404333   -0.836083  -0.303148
2    0.905174           -1.0      -0.521105   -1.204725   1.597400
3   -0.041001           1.0      -1.404333    1.069345  -1.140863
4    0.905174           -1.0      -0.521105   -0.023992   1.597400

      Unit Cost  Total Revenue  Total Cost  Total Profit Days Shipping
0   -0.168895     0.798622    0.603092    1.168192    30 days
1   -0.394831     -0.548427   -0.559505   -0.442948    24 days
2    1.783101     -0.147989    0.001945   -0.497510     6 days
3   -0.983250     -0.893431   -0.811994   -0.967494    15 days
4    1.783101     1.323690    1.599939    0.452390     5 days

```

4 3. Exploratory Data Analysis (EDA)

```
[22]: df.describe().T

# This description of the data is the proof that after normalization the data
# revolves
# around mean of value zero and standard deviation of value of 1.
```

```

[22]:          count          mean            std \
Item Type    100.0         0.0        1.005038
Sales Channel 100.0         0.0        1.005038

```

Order Priority	100.0	-0.0	1.005038
Units Sold	100.0	-0.0	1.005038
Unit Price	100.0	-0.0	1.005038
Unit Cost	100.0	0.0	1.005038
Total Revenue	100.0	-0.0	1.005038
Total Cost	100.0	-0.0	1.005038
Total Profit	100.0	0.0	1.005038
Days Shipping	100	23 days 08:38:24	14 days 17:49:19.419899149

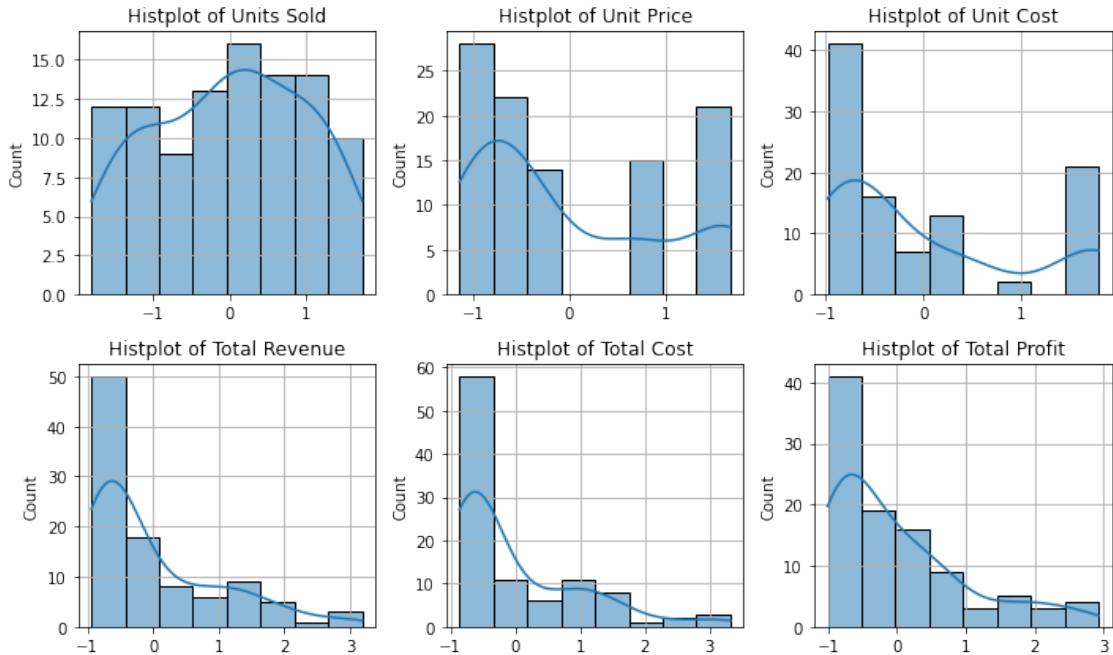
	min	25%	50%	\
Item Type	-1.617958	-0.671784	-0.041001	
Sales Channel	-1.0	-1.0	0.0	
Order Priority	-1.404333	-0.521105	0.362124	
Units Sold	-1.799947	-0.824484	0.091276	
Unit Price	-1.140863	-0.832005	-0.413296	
Unit Cost	-0.98325	-0.828816	-0.447351	
Total Revenue	-0.942113	-0.760486	-0.427596	
Total Cost	-0.86063	-0.707403	-0.526877	
Total Profit	-1.00936	-0.73392	-0.345864	
Days Shipping	0 days 00:00:00	9 days 18:00:00	23 days 12:00:00	

	75%	max
Item Type	0.905174	1.851348
Sales Channel	1.0	1.0
Order Priority	1.245352	1.245352
Units Sold	0.805721	1.724988
Unit Price	0.684432	1.670178
Unit Cost	0.385988	1.783101
Total Revenue	0.577236	3.182718
Total Cost	0.632416	3.317545
Total Profit	0.444944	2.929461
Days Shipping	36 days 06:00:00	50 days 00:00:00

4.1 3.1. Histograms

```
[23]: num_var = ['Units Sold', 'Unit Price', 'Unit Cost', 'Total Revenue', 'Total Cost', 'Total Profit']
```

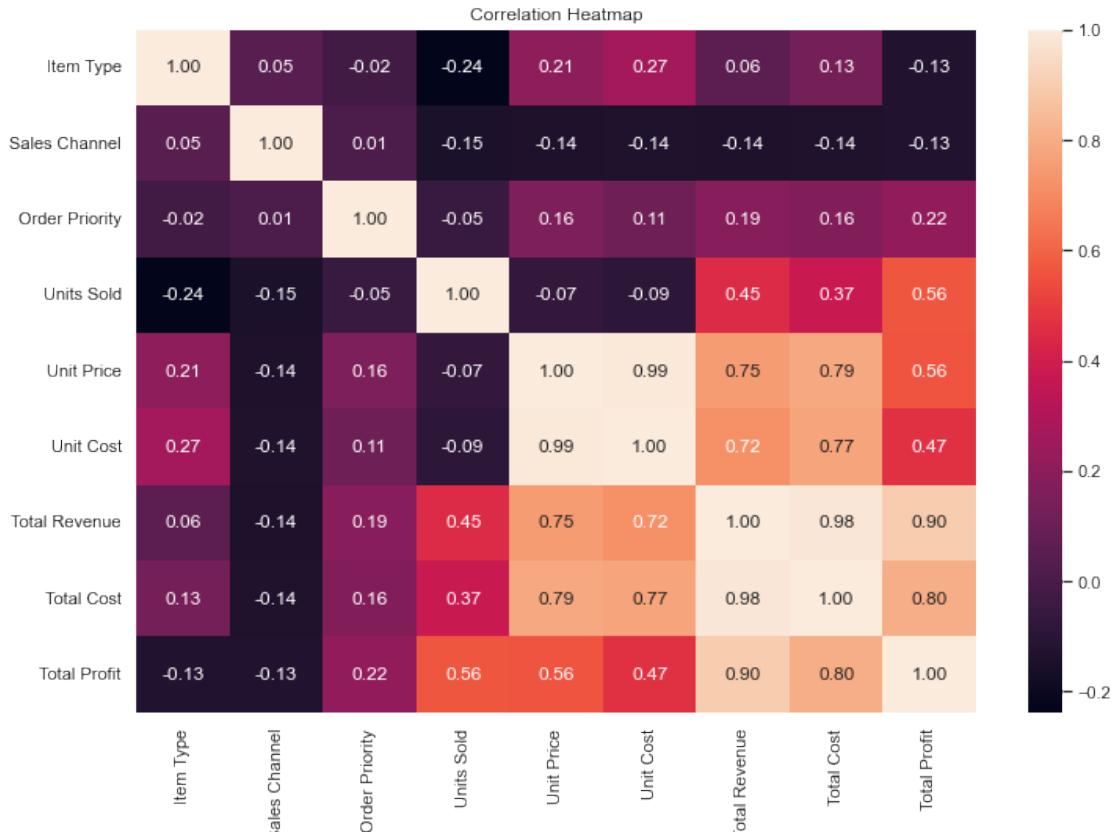
```
[25]: plt.figure(figsize=(10,6))
for i, column in enumerate(num_var):
    plt.subplot(2, 3, i+1)
    sns.histplot(x=df[column], kde=True)
    plt.title(f'Histplot of {column}')
    plt.grid(True)
    plt.xlabel('') # Remove the x axis label
    plt.tight_layout()
```



4.2 3.2. Correlation Heatmap

```
[26]: plt.figure(figsize=(12,8))
sns.set(font_scale=1)
sns.heatmap(df.corr(), fmt='.2f', annot=True)
plt.title('Correlation Heatmap')
plt.show()

# The variables like
# (1) 'Unit Cost', 'Unit Price' (correlation=0.99)
# (2) 'Total Cost', 'Total Revenue' (correlation=0.98)
# (3) 'Total Cost', 'Total Profit' (correlation=0.80)
# (4) 'Total Profit', 'Total Revenue' (correlation=0.90)
# (5) 'Total Cost', 'Unit Price' (correlation=0.79) have very high correlation
# with each other.
# These variables should be removed from the dataset by analysing the Variation
# Inflation Factor
# of the variables or else they might affect the accuracy and prediction of the
# ML model.
```



```
[27]: var_pplot = ['Units Sold','Unit Price','Unit Cost','Total Revenue','Total Cost','Total Profit','Days Shipping']
sns.pairplot(df[var_pplot], diag_kind='kde')

# The kde plot for Unit Price and Unit Cost have bimodal curve which means they do not have normal distributions rather
# they have two modes within the same dataset.
```

[27]: <seaborn.axisgrid.PairGrid at 0x117619b1ca0>

