Data Collection and Preprocessing Phase

Date	11 July 2024
Team ID	SWTID1720151584
Project Title	E-Commerce Shipping Prediction Using Machine Learning
Maximum Marks	6 Marks

Data Exploration and Preprocessing Template

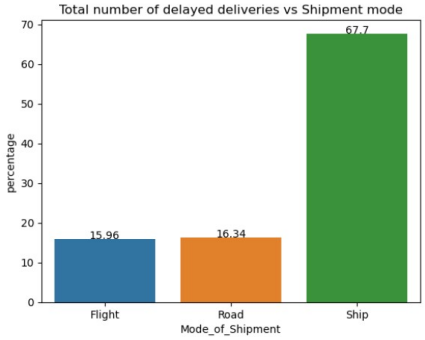
Identifies data sources, assesses quality issues like missing values and duplicates, and implements resolution plans to ensure accurate and reliable analysis.

Section	Description
Data Overview	• Internal: • Historical order data (order ID, product details, customer information, shipping method, delivery time) • Product catalog data (product weight, dimensions) • External (potential): • Real-time carrier data (shipping rates, transit times) • Weather data (location-based, impacting delivery times) • Holiday calendars (potential delay
Univariate Analysis	[16]: `#Basic summary statistics` `dataset.describe()` [16]:

	ID	Customer_care_calls	Customer_rating	Cost_of_the_Product	Prior_purchases	Discount_offered	Weight_in_gms	Reached.on.Time_Y.N
count	10999.00000	10999.000000	10999.000000	10999.000000	10999.000000	10999.000000	10999.000000	10999.000000
mean	5500.00000	4.054459	2.990545	210.196836	3.567597	13.373216	3634.016729	0.596691
std	3175.28214	1.141490	1.413603	48.063272	1.522860	16.205527	1635.377251	0.490584
min	1.00000	2.000000	1.000000	96.000000	2.000000	1.000000	1001.000000	0.000000
25%	2750.50000	3.000000	2.000000	169.000000	3.000000	4.000000	1839.500000	0.000000
50%	5500.00000	4.000000	3.000000	214.000000	3.000000	7.000000	4149.000000	1.000000
75%	8249.50000	5.000000	4.000000	251.000000	4.000000	10.000000	5050.000000	1.000000
max	10999.00000	7.000000	5.000000	310.000000	10.000000	65.000000	7846.000000	1.000000

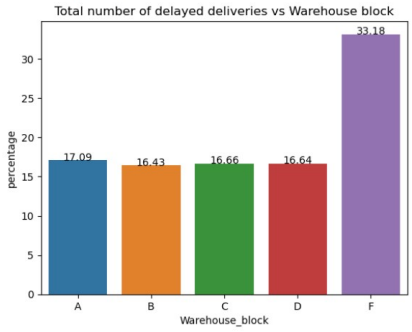
[24]:
```
data_v2=pd.DataFrame((data_v1.groupby(['Mode_of_Shipment'])['ID'].count())/len(data_v1)*100)
data_v2=data_v2.reset_index()
visual=sns.barplot(x="Mode_of_Shipment", y="ID", data=data_v2 )
for index, row in data_v2.iterrows():
    visual.text(row.name,row.ID, round(row.ID,2), color='black', ha="center")
plt.title('Total number of delayed deliveries vs Shipment mode')
plt.ylabel('percentage')
```

[24]: Text(0, 0.5, 'percentage')



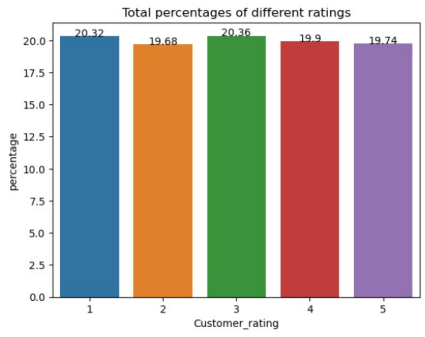
[30]:
```
data_v3=pd.DataFrame((data_v1.groupby(['Warehouse_block'])['ID'].count())/len(data_v1)*100)
data_v3=data_v3.reset_index()
visual=sns.barplot(x="Warehouse_block", y="ID", data=data_v3 )
for index, row in data_v3.iterrows():
    visual.text(row.name,row.ID, round(row.ID,2), color='black', ha="center")
plt.title('Total number of delayed deliveries vs Warehouse block')
plt.ylabel('percentage')
```

[30]: Text(0, 0.5, 'percentage')

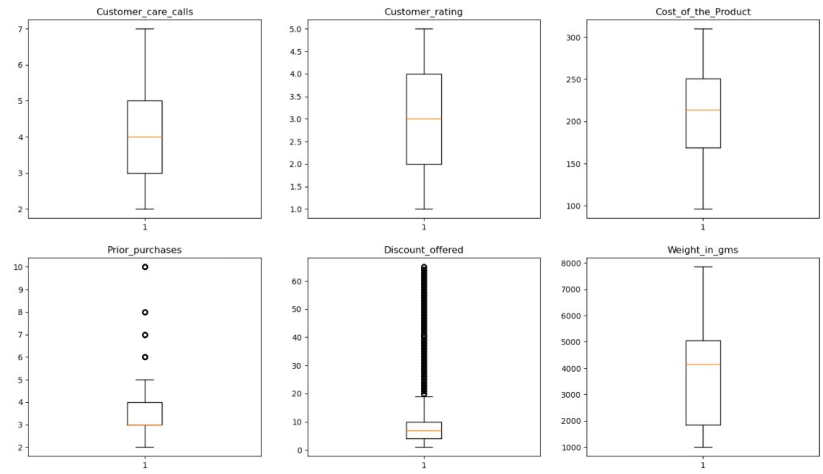


[32]:
```
data_v4=pd.DataFrame((dataset.groupby(['Customer_rating'])['ID'].count())/len(dataset)*100)
data_v4=data_v4.reset_index()
visual=sns.barplot(x="Customer_rating", y="ID", data=data_v4 )
for index, row in data_v4.iterrows():
    visual.text(row.name,row.ID, round(row.ID,2), color='black', ha="center")
plt.title('Total percentages of different ratings')
plt.ylabel('percentage')
```

[32]: Text(0, 0.5, 'percentage')

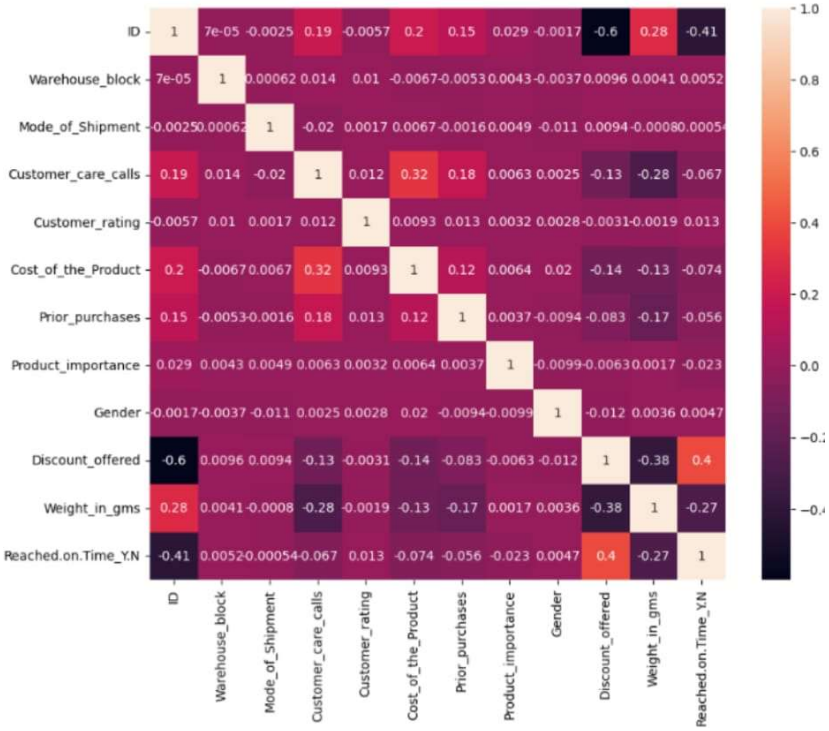


Bivariate Analysis	We expect a positive correlation, meaning locations further away (higher distance) will tend to have longer delivery times. This helps identify factors influencing delivery times.



Multivariate Analysis

Heavy items shipped far via the standard method might be slower



Outliers and Anomalies

```
[50]: def check_outliers(arr):
          Q1 = np.percentile(arr, 25, interpolation = 'midpoint')
          Q3 = np.percentile(arr, 75, interpolation = 'midpoint')
          IQR = Q3 - Q1

      #Above Upper bound
          upper=Q3+1.5*IQR
          upper_array=np.array(arr>=upper)
          print(' '*3,len(upper_array[upper_array == True]), 'are over the upper bound:', upper)

          #Below Lower bound
          lower=Q1-1.5*IQR
          lower_array=np.array(arr<=lower)
          print(' '*3, len(lower_array[lower_array == True]), 'are less than the lower bound:', lower, '\n')

      for i in dataset.drop(columns=[ 'Warehouse_block', 'Mode_of_Shipment', 'Gender', 'Reached.on.Time_Y.N', 'ID' ]).columns:
          if str(dataset[i].dtype)=='object':
              continue
          print(i)
          check_outliers(dataset[i])

Customer_care_calls
    0 are over the upper bound: 8.0
    0 are less than the lower bound: 0.0

Customer_rating
    0 are over the upper bound: 7.0
    0 are less than the lower bound: -1.0

Cost_of_the_Product
    0 are over the upper bound: 374.0
    0 are less than the lower bound: 46.0

Prior_purchases
    1003 are over the upper bound: 5.5
    0 are less than the lower bound: 1.5

Discount_offered
    2262 are over the upper bound: 19.0
    0 are less than the lower bound: -5.0

Weight_in_gms
    0 are over the upper bound: 9865.75
    0 are less than the lower bound: -2976.25

C:\Users\harsha\AppData\Local\Temp\ipykernel_18380\2026037688.py:20: DeprecationWarning: the `interpolation=` argument to percentile was renamed to `me
thod=`, which has additional options.
Users of the modes 'nearest', 'lower', 'higher', or 'midpoint' are encouraged to review the method they used. (Deprecated NumPy 1.22)
  check_outliers(dataset[i])
```

Data Preprocessing Code Screenshots

Loading Data

```
[12]: dataset = pd.read_csv(r"C:\Users\harsha\OneDrive\Desktop\Train.csv")
      dataset.head()
```

[12]:

	ID	Warehouse_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	Cost_of_the_Product	Prior_purchases	Product_importance	Gender	Discount_offered
0	1	D	Flight	4	2	177	3	low	F	44
1	2	F	Flight	4	5	216	2	low	M	59
2	3	A	Flight	2	2	183	4	low	M	48
3	4	B	Flight	3	3	176	4	medium	M	10
4	5	C	Flight	2	2	184	3	medium	F	46

Handling Missing Data

```
[22]: #Checking if there is any null values in the dataset
      dataset.isnull().sum()

[22]: ID                   0
      Warehouse_block      0
      Mode_of_Shipment     0
      Customer_care_calls  0
      Customer_rating      0
      Cost_of_the_Product  0
      Prior_purchases      0
      Product_importance   0
      Gender               0
      Discount_offered     0
      Weight_in_gms        0
      Reached.on.Time_Y.N  0
      dtype: int64
```

Data Transformation

```
[34]: dataset['Reached.on.Time_Y.N'].replace({1 : "Yes", 0: "No"}, inplace = True)

[36]: dataset.head()
```

[36]:

	ID	Warehouse_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	Cost_of_the_Product	Prior_purchases	Product_importance	Gender	Discount_offered
0	1	D	Flight	4	2	177	3	low	F	44
1	2	F	Flight	4	5	216	2	low	M	59
2	3	A	Flight	2	2	183	4	low	M	48
3	4	B	Flight	3	3	176	4	medium	M	10
4	5	C	Flight	2	2	184	3	medium	F	46

Feature Engineering

```
X=data.drop(['Reached on Time','Warehouse_block','Mode_of_Shipment','Gender'],axis=1)
y=data['Reached on Time']
X
```

Save Processed Data

```
[98]: X
```

	Customer_care_calls	Customer_rating	Cost_of_the_Product	Prior_purchases	Product_importance	Discount_offered	Weight_in_gms
0	4	2	177	3	low	44	1233
1	4	5	216	2	low	59	3088
2	2	2	183	4	low	48	3374
3	3	3	176	4	medium	10	1177
4	2	2	184	3	medium	46	2484
...	...	...	...	...	...	...	...
10994	4	1	252	5	medium	1	1538
10995	4	1	232	5	medium	6	1247
10996	5	4	242	5	low	4	1155
10997	5	2	223	6	medium	2	1210
10998	2	5	155	5	low	6	1639

10999 rows × 7 columns