Data Collection and Preprocessing Phase

Date	04 June 2024
Team ID	SWTID1720096620
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	`data.shape` → `(10999, 12)`
	`data.ndim` → `2`
	`data.info()`
	`<class 'pandas.core.frame.DataFrame'>`
	`RangeIndex: 10999 entries, 0 to 10998`
	`Data columns (total 12 columns):`
	`#  Column  Non-Null Count  Dtype`
	`---  ------  --------------  -----`
	`0  ID  10999 non-null  int64`
	`1  Warehouse_block  10999 non-null  object`
	`2  Mode_of_Shipment  10999 non-null  object`
	`3  Customer_care_calls  10999 non-null  int64`
	`4  Customer_rating  10999 non-null  int64`
	`5  Cost_of_the_Product  10999 non-null  int64`
	`6  Prior_purchases  10999 non-null  int64`
	`7  Product_importance  10999 non-null  object`
	`8  Gender  10999 non-null  object`
	`9  Discount_offered  10999 non-null  int64`
	`10  Weight_in_gms  10999 non-null  int64`
	`11  Reached.on.Time_Y.N  10999 non-null  int64`
	`dtypes: int64(8), object(4)`
	`memory usage: 1.0+ MB`

```
unique_values = df.nunique()
print(unique_values)
```

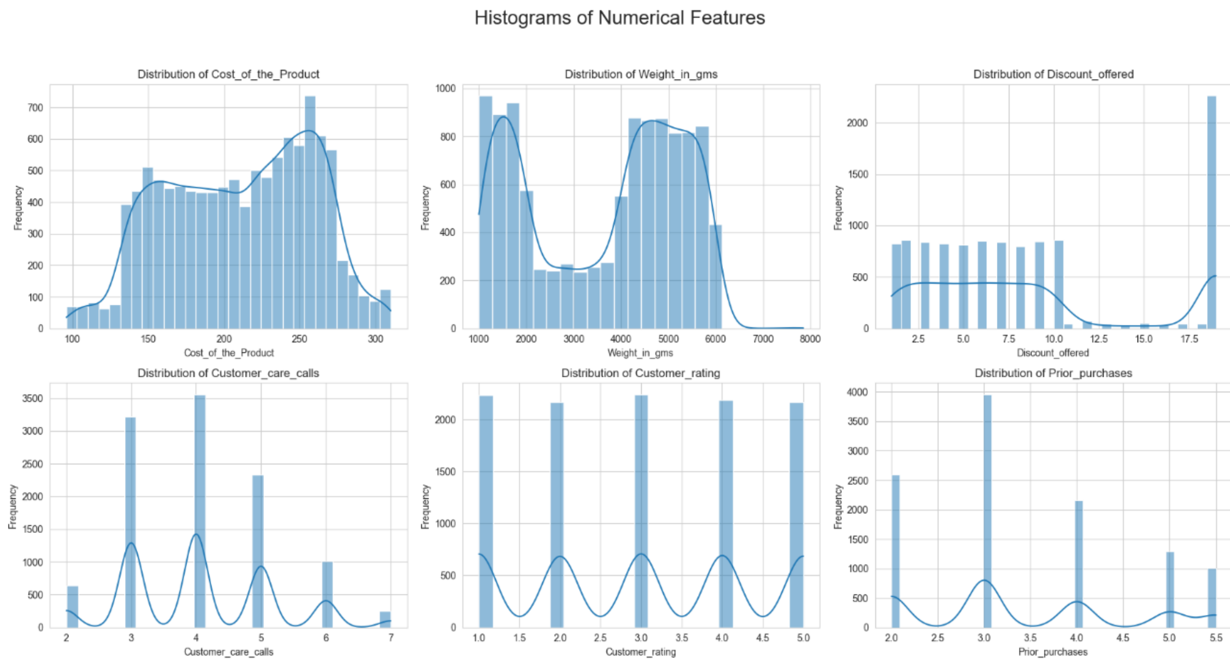
```
Warehouse_block          5
Mode_of_Shipment         3
Customer_care_calls      6
Customer_rating          5
Cost_of_the_Product    215
Prior_purchases          5
Product_importance       3
Gender                   2
Discount_offered        19
Weight_in_gms         4034
Reached.on.Time_Y.N      2
dtype: int64
```

Univariate Analysis

```
data.describe()
```

	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

```
numerical_features = ['Cost_of_the_Product', 'Weight_in_gms', 'Discount_offered', 'Customer_care_calls',
                      'Customer_rating', 'Prior_purchases']

fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(18, 10))
fig.suptitle('Histograms of Numerical Features', fontsize=20)

for i, feature in enumerate(numerical_features):
    row = i // 3
    col = i % 3
    sns.histplot(df[feature], kde=True, ax=axes[row, col])
    axes[row, col].set_title(f'Distribution of {feature}')
    axes[row, col].set_xlabel(feature)
    axes[row, col].set_ylabel('Frequency')

plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()
```

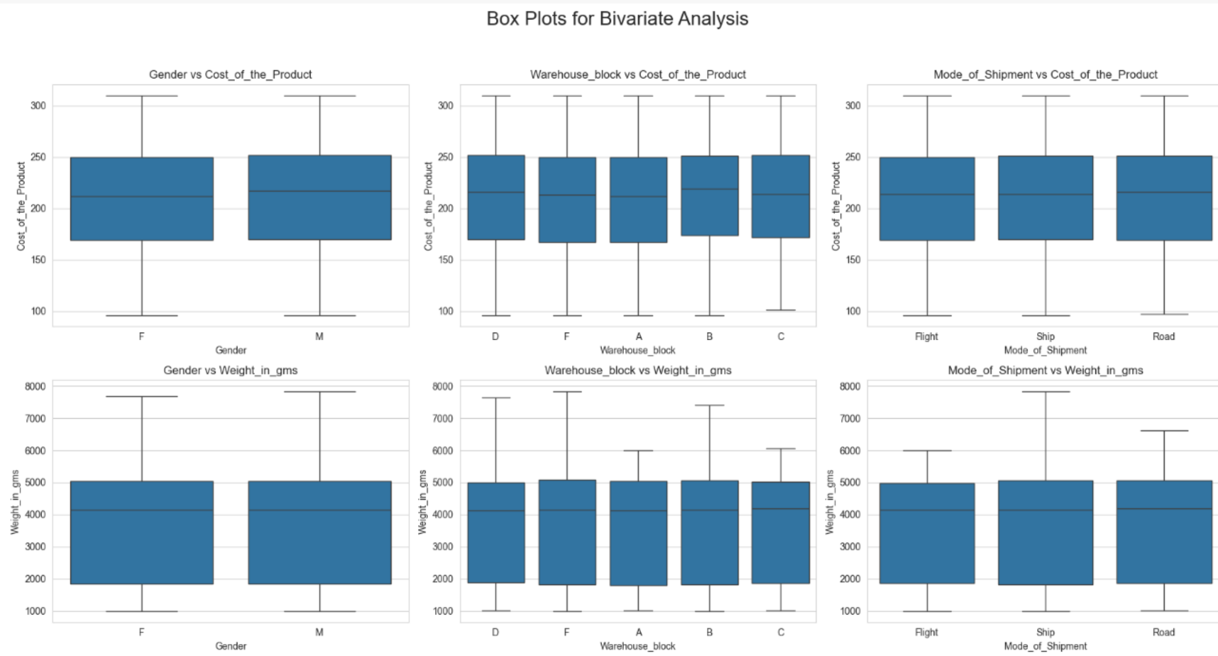


Bivariate Analysis

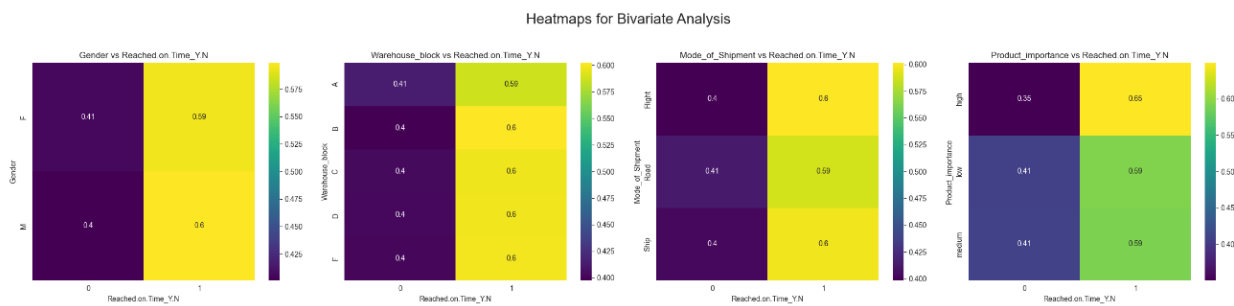
```
df.corr()
```

	Warehouse_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	Cost_of_the_Product	Prior_purchases	Product_importance	Gender	Discount_offered	Weight_in_gms	Reached.on.Time_Y.N
Warehouse_block	1.000000	0.000617	0.014496	0.010169	-0.006679	-0.006632	0.004260	-0.003700	0.007794	0.004086	0.005214
Mode_of_Shipment	0.000617	1.000000	-0.020164	0.001679	0.006681	-0.006336	0.004911	-0.011288	0.001722	-0.000797	-0.000535
Customer_care_calls	0.014496	-0.020164	1.000000	0.012209	0.323182	0.264801	0.006273	0.002545	-0.133149	-0.276615	-0.067126
Customer_rating	0.010169	0.001679	0.012209	1.000000	0.009270	0.008450	0.003157	0.002775	-0.001346	-0.001897	0.013119
Cost_of_the_Product	-0.006679	0.006681	0.323182	0.009270	1.000000	0.180123	0.006366	0.019759	-0.143876	-0.132604	-0.073587
Prior_purchases	-0.006632	-0.006336	0.264801	0.008450	0.180123	1.000000	0.013841	-0.008808	-0.119570	-0.253856	-0.074934
Product_importance	0.004260	0.004911	0.006273	0.003157	0.006366	0.013841	1.000000	-0.009865	-0.007683	0.001652	-0.023483
Gender	-0.003700	-0.011288	0.002545	0.002775	0.019759	-0.008808	-0.009865	1.000000	-0.012533	0.003573	0.004689
Discount_offered	0.007794	0.001722	-0.133149	-0.001346	-0.143876	-0.119570	-0.007683	-0.012533	1.000000	-0.389933	0.410716
Weight_in_gms	0.004086	-0.000797	-0.276615	-0.001897	-0.132604	-0.253856	0.001652	0.003573	-0.389933	1.000000	-0.268793
Reached.on.Time_Y.N	0.005214	-0.000535	-0.067126	0.013119	-0.073587	-0.074934	-0.023483	0.004689	0.410716	-0.268793	1.000000

```python
categorical_features = ['Gender', 'Warehouse_block', 'Mode_of_Shipment', 'Product_importance']
numerical_features = ['Cost_of_the_Product', 'Weight_in_gms', 'Discount_offered']

fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(18, 10))
fig.suptitle('Box Plots for Bivariate Analysis', fontsize=20)

for i, cat_feature in enumerate(categorical_features[:3]):
    sns.boxplot(x=cat_feature, y=numerical_features[0], data=df, ax=axes[0, i])
    axes[0, i].set_title(f'{cat_feature} vs {numerical_features[0]}')
    axes[0, i].set_xlabel(cat_feature)
    axes[0, i].set_ylabel(numerical_features[0])

for i, cat_feature in enumerate(categorical_features[:3]):
    sns.boxplot(x=cat_feature, y=numerical_features[1], data=df, ax=axes[1, i])
    axes[1, i].set_title(f'{cat_feature} vs {numerical_features[1]}')
    axes[1, i].set_xlabel(cat_feature)
    axes[1, i].set_ylabel(numerical_features[1])

plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()
```



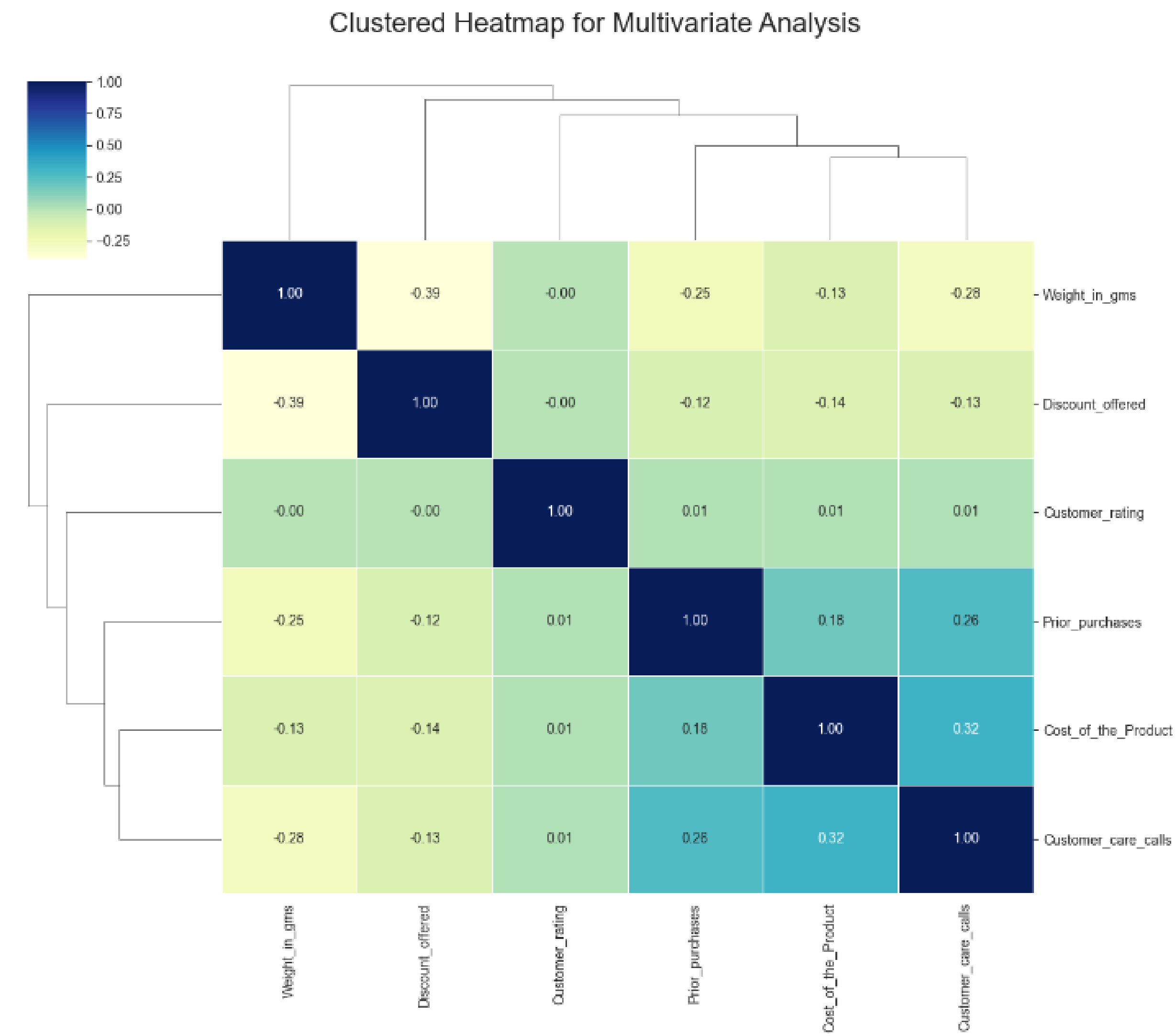
```python
categorical_features = ['Gender', 'Warehouse_block', 'Mode_of_Shipment', 'Product_importance']
target_feature = 'Reached.on.Time_Y.N'
heatmaps_data = [
    pd.crosstab(df[cat_feature], df[target_feature], normalize='index')
    for cat_feature in categorical_features
]
fig, axes = plt.subplots(nrows=1, ncols=4, figsize=(24, 6))
fig.suptitle('Heatmaps for Bivariate Analysis', fontsize=20)

for i, cat_feature in enumerate(categorical_features):
    sns.heatmap(heatmaps_data[i], annot=True, cmap='viridis', ax=axes[i])
    axes[i].set_title(f'{cat_feature} vs {target_feature}')
    axes[i].set_xlabel(target_feature)
    axes[i].set_ylabel(cat_feature)

plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()
```



Multivariate Analysis

```python
numerical_features = ['Cost_of_the_Product', 'Weight_in_gms', 'Discount_offered', 'Customer_care_calls',
                      'Customer_rating', 'Prior_purchases']

plt.figure(figsize=(16, 12))
sns.clustermap(df[numerical_features].corr(), annot=True, linewidths=0.5, fmt=".2f", cmap="YlGnBu", figsize=(12, 10))
plt.suptitle('Clustered Heatmap for Multivariate Analysis', fontsize=20, y=1.05)
plt.show()
```

```
<Figure size 1600x1200 with 0 Axes>
```



```
correlation_matrix = df.corr()
plt.figure(figsize=(14, 10))
sns.set_style('whitegrid')
heatmap = sns.heatmap(
    correlation_matrix,
    annot=True,
    linewidths=0.5,
    fmt=".2f",
    cmap="YlGnBu",
    cbar_kws={'shrink': 0.8, 'label': 'Correlation Coefficient'},
    square=True,
    annot_kws={'size': 10, 'weight': 'bold'}
)
plt.title('Correlation Matrix Heatmap', fontsize=18, weight='bold')
plt.xticks(rotation=45, ha='right', fontsize=12)
plt.yticks(rotation=0, fontsize=12)
plt.xlabel('Features', fontsize=14, weight='bold')
plt.ylabel('Features', fontsize=14, weight='bold')
plt.tight_layout()
plt.show()
```

<table>
<tr><td>Outliers and Anomalies</td><td>

```
numeric_cols = data.select_dtypes(include=['float64', 'int64']).columns

# Function to cap outliers using IQR method
def cap_outliers(series):
    Q1 = series.quantile(0.25)
    Q3 = series.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return series.apply(lambda x: lower_bound if x < lower_bound else (upper_bound if x > upper_bound else x))

for col in numeric_cols:
    if col != 'ID':
        data[col] = cap_outliers(data[col])
data.shape
```

```
(10999, 12)
```

</td></tr>
<tr><td colspan="2">Data Preprocessing Code Screenshots</td></tr>
<tr><td>Loading Data</td><td>

```
data=pd.read_csv('Train.csv')
```

</td></tr>
<tr><td>Handling Missing Data</td><td>

```
missing_data_summary = data.isnull().sum()
print(missing_data_summary)
```

```
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_dropped_rows = data.dropna()
```

</td></tr>
<tr><td>Data Transformation</td><td>

```
from sklearn.preprocessing import LabelEncoder
```

```
le=LabelEncoder()
columns=['Warehouse_block','Mode_of_Shipment','Product_importance','Gender']
for column in columns:
    df[column] = le.fit_transform(df[column])
df.head()
df
```

</td></tr>
</table>

	Warehouse_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	Cost_of_the_Product	Prior_purchases	Product_importance	Gender
0	3	0	4	2	177	3.0	1	0
1	4	0	4	5	216	2.0	1	1
2	0	0	2	2	183	4.0	1	1
3	1	0	3	3	176	4.0	2	1
4	2	0	2	2	184	3.0	2	0
...	...	...	...	...	...	...	...	...
10994	0	2	4	1	252	5.0	2	0
10995	1	2	4	1	232	5.0	2	0
10996	2	2	5	4	242	5.0	1	0
10997	4	2	5	2	223	5.5	2	1
10998	3	2	2	5	155	5.0	1	0

Discount_offered	Weight_in_gms	Reached.on.Time_Y.N
19.0	1233	1
19.0	3088	1
19.0	3374	1
10.0	1177	1
19.0	2484	1
...	...	...
1.0	1538	1
6.0	1247	0
4.0	1155	0
2.0	1210	0
6.0	1639	0

```
from sklearn.preprocessing import StandardScaler
```

```
sc=StandardScaler()
x=sc.fit_transform(x)
x
```

```
array([[ 0.4471892 , -2.00415767, -0.04771132, ..., -0.99176046,
         1.70774793, -1.46823975],
       [ 1.11803399, -2.00415767, -0.04771132, ...,  1.00830799,
         1.70774793, -0.33389333],
       [-1.56534517, -2.00415767, -1.79988745, ...,  1.00830799,
         1.70774793, -0.15900218],
       ...,
       [-0.22365559,  0.63834175,  0.82837675, ..., -0.99176046,
        -0.75321157, -1.51593733],
       [ 1.11803399,  0.63834175,  0.82837675, ...,  1.00830799,
        -1.08133951, -1.48230442],
       [ 0.4471892 ,  0.63834175, -1.79988745, ..., -0.99176046,
        -0.42508364, -1.2199677 ]])
```

Feature Engineering	`df=data.drop(['ID'], axis=1 )` `df.head()`
Save Processed Data	`data.to_csv('Train_cleaned.csv', index=False)`

Output of `df.head()`:

	Warehouse_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	Cost_of_the_Product	Prior_purchases	Product_importance	Gender	Discount_offered	Weight_in_gms	Reached.on.Time_Y.N
0	D	Flight	4	2	177	3.0	low	F	19.0	1233	1
1	F	Flight	4	5	216	2.0	low	M	19.0	3088	1
2	A	Flight	2	2	183	4.0	low	M	19.0	3374	1
3	B	Flight	3	3	176	4.0	medium	M	10.0	1177	1
4	C	Flight	2	2	184	3.0	medium	F	19.0	2484	1