CP-2
PRESENTATION

Consignment Price Prediction

PROBLEM STATEMENT

- Create a predictive model to forecast consignment pricing.
- Enable logistics stakeholders to optimize supply chain processes.
- Enhance operational efficiency through informed decision-making.

ABOUT DATA

- *
- THE DATASET CONTAINS 34 PREDICTOR VARIABLES AND 1 PREDICTED VARIABLE
- THE DATASET HAS 10324 ROWS AND 33 COLUMNS OF DATA
- THERE ARE FEW COLUMNS WITH NULL VALUES IN THE DATASET.
- IT IS OBSERVED THAT THERE ARE QUITE A FEW OUTLIERS IN THE DATA.
- HANDLING OUTLIERS USING “LOG TRANSFORMATION”
- WE CAN CONVERT THE OBJECT TYPE COLUMNS TO CATEGORIES (DONE TO SAVE SPACE)

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10324 entries, 0 to 10323
Data columns (total 33 columns):
 #   Column                         Non-Null Count  Dtype
---  ------                         --------------  -----
 0   ID                             10324 non-null  int64
 1   Project Code                   10324 non-null  object
 2   PQ #                           10324 non-null  object
 3   PO / SO #                      10324 non-null  object
 4   ASN/DN #                       10324 non-null  object
 5   Country                        10324 non-null  object
 6   Managed By                     10324 non-null  object
 7   Fulfill Via                    10324 non-null  object
 8   Vendor INCO Term               10324 non-null  object
 9   Shipment Mode                  9964 non-null   object
 10  PQ First Sent to Client Date   10324 non-null  object
 11  PO Sent to Vendor Date         10324 non-null  object
 12  Scheduled Delivery Date        10324 non-null  object
 13  Delivered to Client Date       10324 non-null  object
 14  Delivery Recorded Date         10324 non-null  object
 15  Product Group                  10324 non-null  object
 16  Sub Classification             10324 non-null  object
 17  Vendor                         10324 non-null  object
 18  Item Description               10324 non-null  object
 19  Molecule/Test Type             10324 non-null  object
 20  Brand                          10324 non-null  object
 21  Dosage                         8588 non-null   object
 22  Dosage Form                    10324 non-null  object
 23  Unit of Measure (Per Pack)     10324 non-null  int64
 24  Line Item Quantity             10324 non-null  int64
 25  Line Item Value                10324 non-null  float64
 26  Pack Price                     10324 non-null  float64
 27  Unit Price                     10324 non-null  float64
 28  Manufacturing Site             10324 non-null  object
 29  First Line Designation         10324 non-null  object
 30  Weight (Kilograms)             10324 non-null  object
 31  Freight Cost (USD)             10324 non-null  object
 32  Line Item Insurance (USD)      10037 non-null  float64
dtypes: float64(4), int64(3), object(26)
memory usage: 2.6+ MB
```

Dataset information

- we have 1324 rows and 33 coulmns
- There are few columns with null values
- memory usage: 2.6+ MB
- data type present in dataset
  *float64(4),
  *int64(3),
  *object(26)

statistical summary

```
# Generating descriptive statistics of the DataFrame
# df.describe() method computes summary statistics of numerical columns in the DataFrame
# It includes count, mean, standard deviation, minimum, 25th percentile, median (50th percentile), 75th percentile, and maximum values
df.describe()
```

	ID	Unit of Measure (Per Pack)	Line Item Quantity	Line Item Value	Pack Price	Unit Price	Line Item Insurance (USD)
count	10324.000000	10324.000000	10324.000000	1.032400e+04	10324.000000	10324.000000	10037.000000
mean	51098.968229	77.990895	18332.534870	1.576506e+05	21.910241	0.611701	240.117626
std	31944.332496	76.579764	40035.302961	3.452921e+05	45.609223	3.275808	500.190568
min	1.000000	1.000000	1.000000	0.000000e+00	0.000000	0.000000	0.000000
25%	12795.750000	30.000000	408.000000	4.314593e+03	4.120000	0.080000	6.510000
50%	57540.500000	60.000000	3000.000000	3.047147e+04	9.300000	0.160000	47.040000

Methodology

Data clening:-

- dropping insignificant columns
- Checking for misssing values and duplicates

EDA:-

- perform EDA to understand the distribution,patterns

Data Preprocessing:-

- treating outliers
- filling null values

Model Building:-

- Develope machine learning model using various algorithms

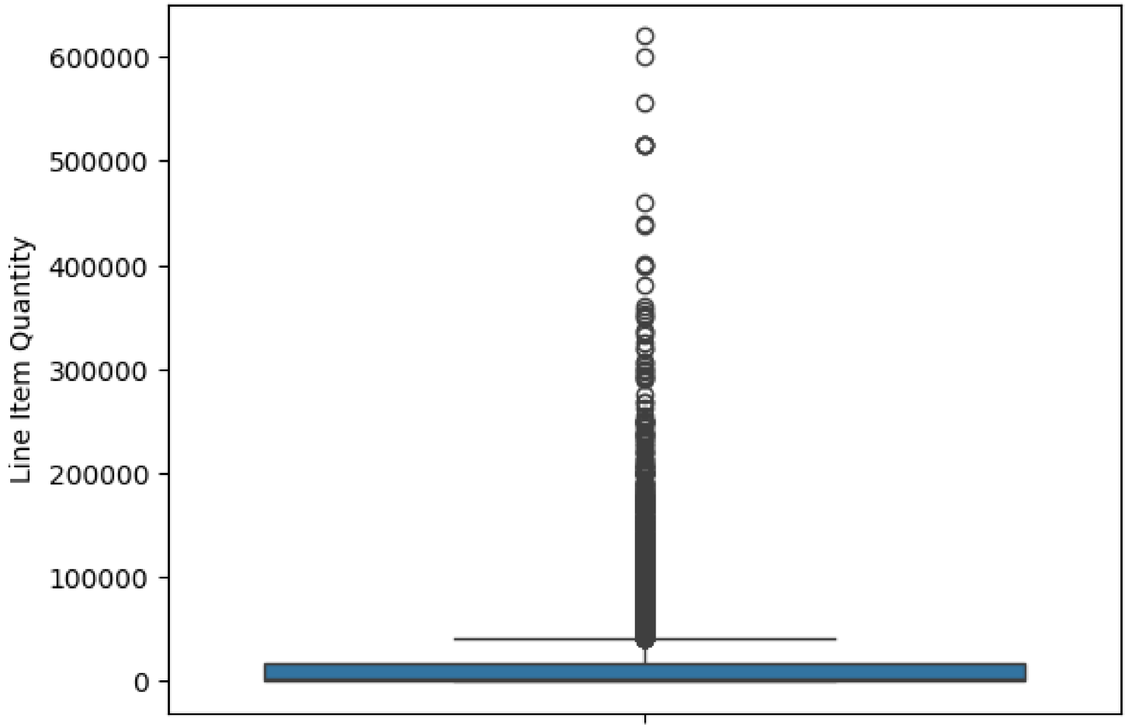
Conclusion:-

- Evaluate model performance and interpret results

Treating outliers using Log transformation

```
# Checking outliers using boxplot
sns.boxplot(df["Line Item Quantity"])
```

```
<Axes: ylabel='Line Item Quantity'>
```

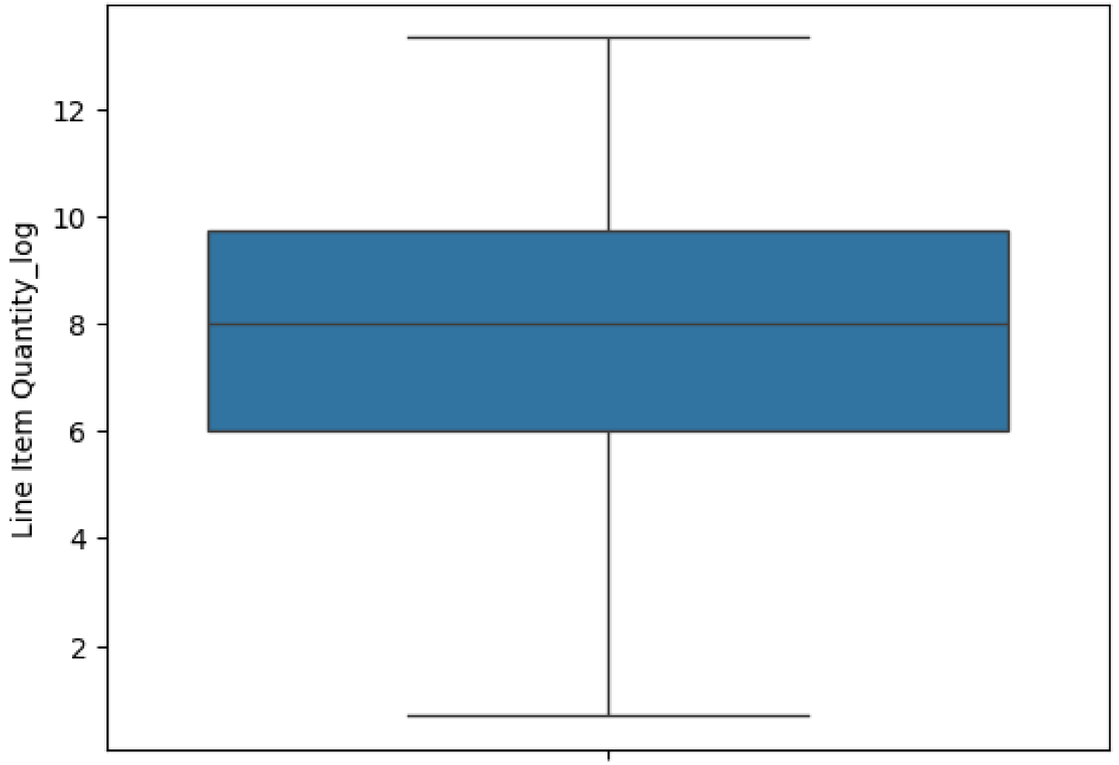


using box-plot to showcase the outliers

After log Transformation

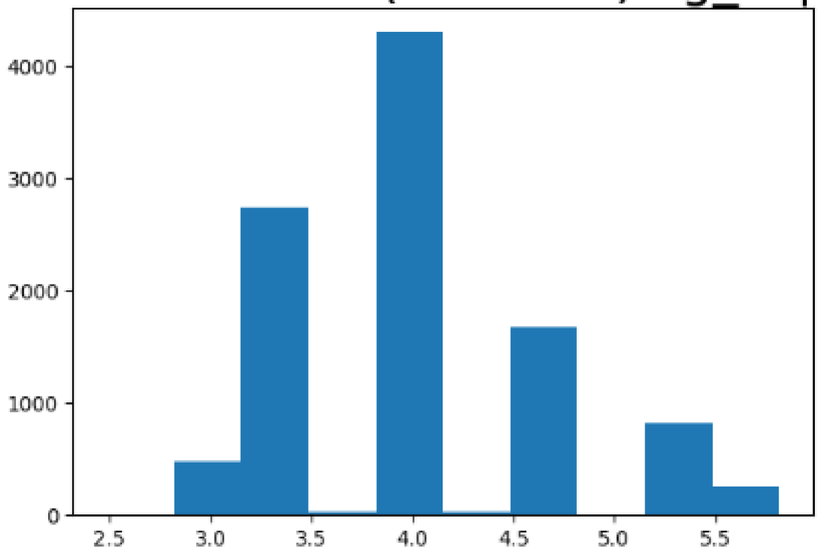
```
# Checking outliers using boxplot
sns.boxplot(df["Line Item Quantity_log"])
```

```
<Axes: ylabel='Line Item Quantity_log'>
```

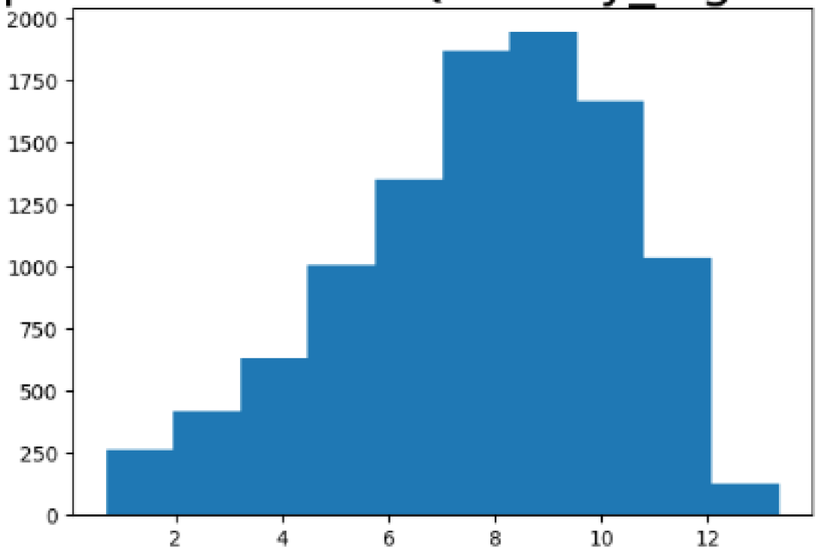


Histograms

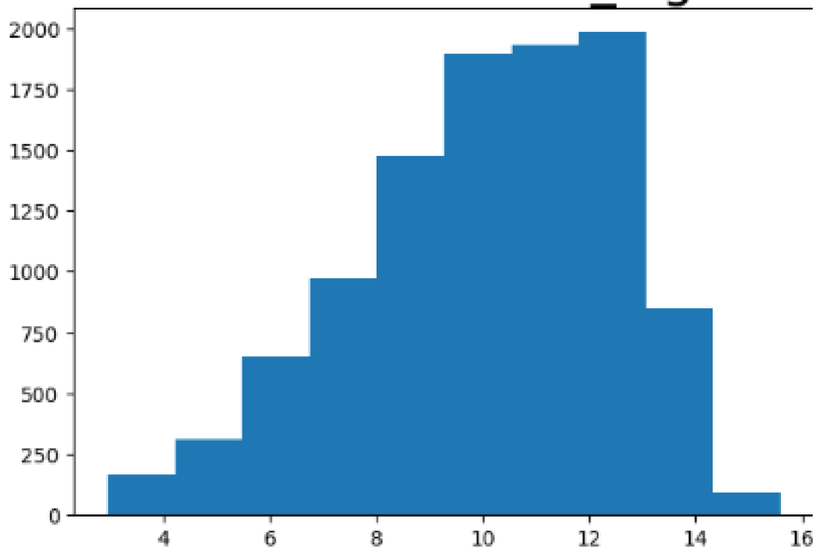
nit of Measure (Per Pack)log_capped



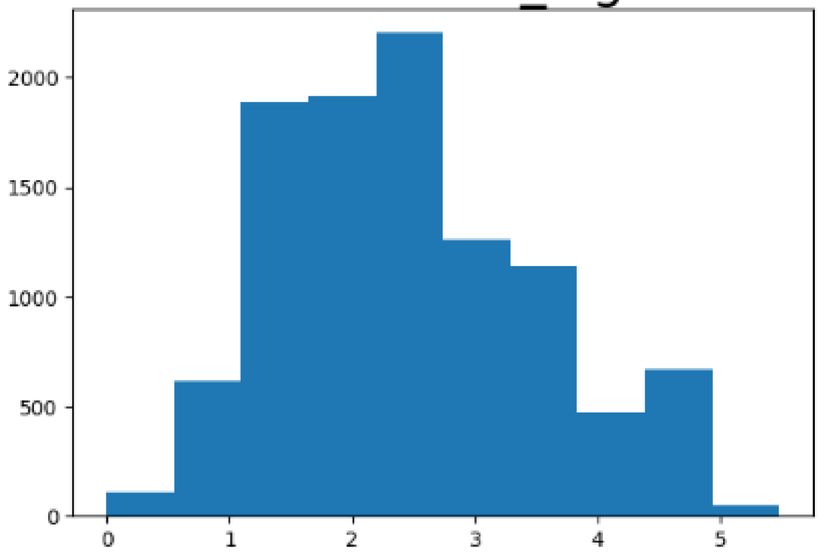
Line Item Quantity_log



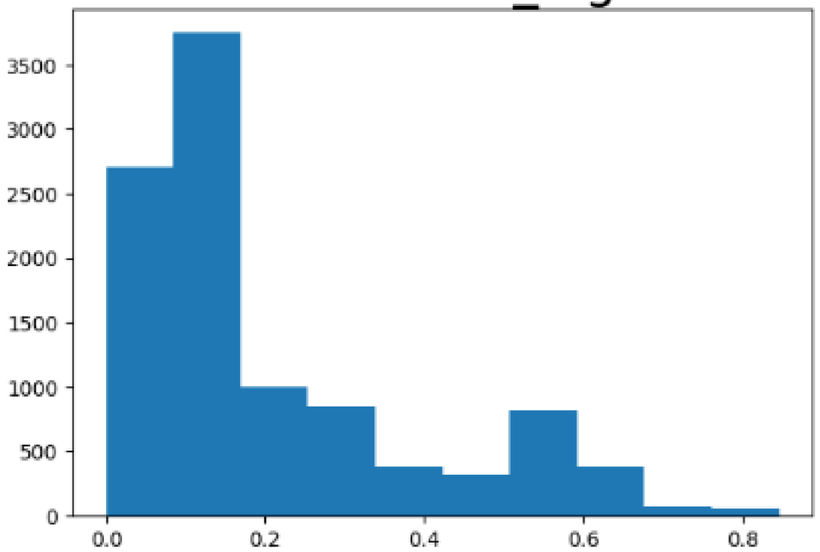
Line Item Value_log



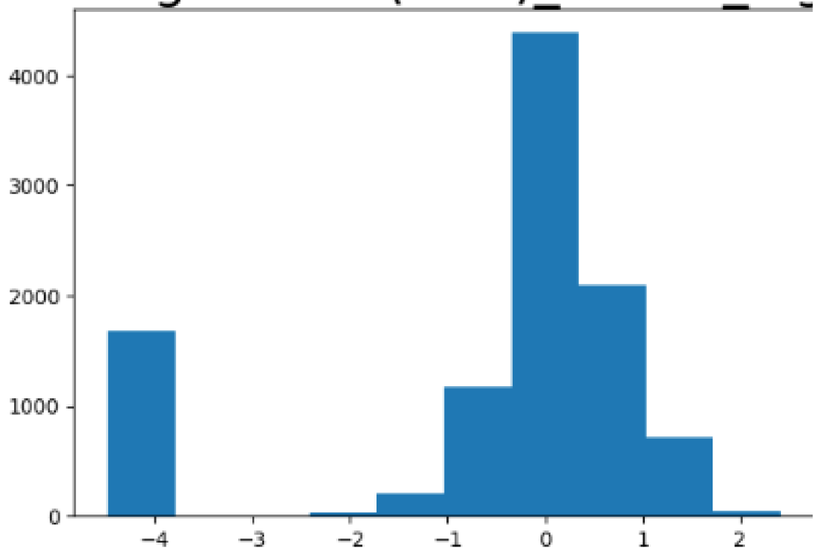
Pack Price_log



Unit Price_log



Freight Cost (USD)_robust_log



Creating dummies

```
[138] # Use pd.get_dummies() to perform one-hot encoding on object columns
      df = pd.get_dummies(df, drop_first=True)
      # This will handle both identifying object columns and performing one-hot encoding in a single step
      df.head()
```

PO / SO _SCMS	PO / SO #_SO	...	Manufacturing Site_Cipla, Goa, India	Manufacturing Site_Hetero Unit III Hyderabad IN	Manufacturing Site_Inverness Japan	Manufacturing Site_MSD, Haarlem, NL	Manufacturing Site_Mylan (formerly Matrix) Nashik	Manufacturing Site_Other	Manufacturing Site_Standard Diagnostics, Korea	Manufacturing Site_Strides, Bangalore, India.	Man Si Bi
1	0	...	0	0	0	0	0	1	0	0	
1	0	...	0	0	0	0	0	0	0	0	
1	0	...	0	0	0	0	0	1	0	0	
1	0	...	0	0	0	0	0	1	0	0	
1	0	...	0	0	0	0	0	0	0	0	

Corelation

```python
# Calculate the correlation between "Line Item Value_log" and c
correlation = df.corrwith(df["Line Item Value_log"])

# Sort the correlation values in descending order
correlation_sorted = correlation.sort_values(ascending=False)

# Display the top 3 factors from a business perspective
top_factors = correlation_sorted

# Print the top 3 factors
print("Top 3 factors from a business perspective:")
```

```
Top 3 factors from a business perspective:
Line Item Value_log                                  1.0000
Line Item Insurance (USD)_log                        0.9567
Line Item Quantity_log                               0.8739
Freight Cost (USD)_robust_log                        0.4602
First Line Designation_Yes                           0.2544
Sub Classification_Adult                             0.2393
Country_Zambia                                       0.1861
Manufacturing Site_Mylan (formerly Matrix) Nashik    0.1725
Fulfill Via_From RDC                                 0.1631
Vendor INCO Term_N/A - From RDC                      0.1631
PO / SO #_SO                                         0.1631
ASN/DN #_DN                                          0.1631
Unit Price_log                                       0.1517
Shipment Mode_Ocean                                  0.1441
Shipment Mode_Air Charter                            0.1342
Country_Nigeria                                      0.1306
Country_Tanzania                                     0.1264
Manufacturing Site_Hetero Unit III Hyderabad IN      0.1031
Pack Price_log                                       0.0965
Country_Zimbabwe                                     0.0875
Country_Mozambique                                   0.0874
Manufacturing Site_Alere Medical Co., Ltd.           0.0794
Manufacturing Site_Aurobindo Unit VII, IN            0.0791
Sub Classification_HIV test                          0.0677
```

Finding top 3 factors which affect the business

Splliting the data in 70% training and 30% testing

```
from sklearn.model_selection import train_test_split
# Define features (X) and target variable (y)
X = df.drop("Line Item Value_log", axis=1)  # Features
y = df[["Line Item Value_log"]]  # Target variable

# Split the data into 70% training and 30% testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42

# Display the shapes of the resulting sets
print("Training set shapes: X_train =", X_train.shape, "y_train =", y_train.shape)
print("Testing set shapes: X_test =", X_test.shape, "y_test =", y_test.shape)
```

```
Training set shapes: X_train = (7226, 64) y_train = (7226, 1)
Testing set shapes: X_test = (3098, 64) y_test = (3098, 1)
```

Scaling the data using MinMaxScaler

```
from sklearn.preprocessing import MinMaxScaler

# Initialize the MinMaxScaler
scaler = MinMaxScaler()

# Fit and transform the training features
X_train = scaler.fit_transform(X_train)

# Transform the test features using the same scaler
X_test= scaler.transform(X_test)
```

Model Building

Using multipal Machine learning algorithm models

- Logistic Regression
- Decision Tree
- Pruned Decision Tree
- Random Forest
- Ada-Boost
- Gradient Boost
- XG Boost

Model Building

Model	MSE_train	RMSE_train	MAE_train	R2_train	Adj_R2_train	MSE_test	RMSE_test	MAE_test	R2_test	Adj_R2_test
Linear Regression	0.335649	0.579352	0.254133	0.943150	0.943016	0.460628	0.678696	0.278130	0.922474	0.922047
Decision Tree	0.000000	0.000000	2.745519	1.000000	1.000000	0.180657	0.425037	2.754550	0.969595	0.969427
Pruned Decision Tree	0.470726	0.686095	0.463714	0.920272	0.920084	0.504683	0.710411	0.476038	0.915060	0.914591
Random Forest	0.152959	0.391100	0.259376	0.974093	0.974032	0.201434	0.448814	0.272230	0.966098	0.965911
Adaboost	0.347483	0.589477	0.423692	0.941146	0.941007	0.399263	0.631873	0.441816	0.932802	0.932432
Gradient Boost	0.387153	0.622216	0.493903	0.934427	0.934272	0.450019	0.670834	0.507253	0.924260	0.923842
XGBoost	0.546936	0.739551	0.539088	0.907364	0.907146	0.583770	0.764048	0.550054	0.901749	0.901207

Conclusion

In conclusion, after evaluating various regression models including Linear Regression, Decision Tree, Pruned Decision Tree, Random Forest, Adaboost, Gradient Boost, and XGBoost, the Random Forest model emerges as the most suitable choice for this particular dataset.

The Random Forest model demonstrates robust performance on both the training and testing sets, exhibiting relatively low errors (MSE, RMSE, MAE) and high R-squared values. It strikes a balance between model complexity and generalization ability, making it less prone to overfitting compared to some other models like Decision Tree and Pruned Decision Tree.

Overall, the Random Forest model provides a reliable and consistent predictive performance, making it the recommended choice for this regression task.

Top 3 factors from a business perspective

Line Item Quantity_log: This variable has a strong positive correlation (0.873926) with the target variable, indicating that the quantity of the line item has a significant impact on its value.

Line Item Insurance (USD)_log: This variable also shows a strong positive correlation (0.956732) with the target variable, suggesting that the insurance cost associated with the line item influences its overall value.

Freight Cost (USD)_robust_log: This variable demonstrates a moderate positive correlation (0.460239) with the target variable, indicating that the freight cost in US dollars plays a notable role in determining the value of the line item.