Ecommerce Shipping Prediction Using Machine Learning

Team ID : SWTID1720082658
Team Members : Sneha Kavitake, Vani Jain, Dheeraj Kosuri, Omkar Prasad

1. INTRODUCTION

1.1 Project overviews :

Ecommerce shipping prediction involves estimating whether a product will be delivered on time. This process is based on analysing various factors such as the package's origin and destination, the shipping method chosen by the customer, the carrier responsible for shipping, and any potential delays or issues that might occur during transit. These predictions are crucial for ecommerce businesses as they impact customer satisfaction and trust.

Key Factors in Ecommerce Shipping Prediction:

1. **Origin and Destination**:
   - The distance between the origin and destination affects the delivery time. Local deliveries typically take less time compared to international shipments, which might face customs and other regulatory delays.
2. **Shipping Method**:
   - Different shipping methods (standard, express, overnight) have varying delivery timelines. Customers' choice of shipping method significantly influences the predicted delivery date.
3. **Carrier Performance**:
   - The performance and reliability of the carrier used for shipping play a crucial role. Some carriers may have better track records for on-time deliveries than others.
4. **Potential Delays**:
   - Various factors can cause delays, such as:
     - **Weather Conditions**: Severe weather can disrupt transportation networks.
     - **Traffic Conditions**: Congestion can slow down delivery vehicles.
     - **Holidays and Weekends**: Non-working days can extend delivery times.
     - **Regulatory Issues**: International shipments might face customs delays.

1.2 Objectives :

1. Enhance Customer Satisfaction:
    - Provide accurate delivery estimates to customers to manage their expectations and improve overall satisfaction with the service.

2. Improve Operational Efficiency:
    - Optimize logistics and inventory management by predicting delivery times accurately, thus reducing costs and improving efficiency.

3. Increase Customer Trust and Loyalty:
    - Build trust and loyalty by consistently delivering products within the estimated time frame, resulting in a positive customer experience.

4. Minimize Delivery Delays:
    - Identify and mitigate potential delays by analysing various factors that impact shipping times, ensuring timely deliveries.

5. Proactive Customer Communication:
    - Enable proactive communication with customers regarding any potential delays, providing timely updates to enhance transparency and trust.

6. Data-Driven Decision Making:
    - Utilize historical and real-time data to make informed decisions regarding shipping methods, carrier selection, and route optimization.

2. Project Initialization and Planning Phase

2.1. Define Problem Statement :

The e-commerce platform is struggling with accurately predicting shipping delivery times, which affects customer satisfaction and trust. Despite having data on shipping origins, destinations, and methods, the current system fails to incorporate real-time factors like traffic, weather, and carrier delays. This results in unreliable delivery estimates, leading to customer frustration and potential loss of business. There is a need for a more advanced machine learning model that can analyse historical and real-time data to provide precise shipping predictions. Implementing such a model would improve delivery accuracy, enhance customer experience, and strengthen the platform's reputation.

2.2. Project Proposal (Proposed Solution) :

The project report outlines a solution to address the challenge of inaccurate shipping delivery predictions faced by e-commerce platforms. Key features includes accurate delivery predictions, real-time updates, seamless integration with e-commerce platforms, scalability, and continuous optimization of machine learning models

Project Overview	
Objective	Develop a machine learning model to accurately predict shipping delivery times, enhancing delivery accuracy, customer satisfaction, and platform reputation.
Scope	Collect historical and real-time shipping data, develop and train models, integrate with e-commerce platforms, provide real-time delivery updates, and ensure scalability for high order volumes.
Problem Statement	
Description	E-commerce platforms struggle with accurately predicting shipping delivery times, failing to account for real-time factors like traffic, weather, and carrier delays, leading to unreliable delivery estimates and customer frustration.
Impact	Solving this problem will enhance delivery accuracy, improve customer satisfaction, boost trust and loyalty, and strengthen the platform's reputation, ultimately driving higher sales and reducing customer churn.

Proposed Solution	
Approach	We will develop machine learning models trained on historical and real-time data to predict shipping delivery times accurately. These models will be integrated with e-commerce platforms to provide seamless and up-to-date delivery information.
Key Features	• Accurate Predictions: Models that factor in distance, traffic, weather, and other variables. • Real-Time Updates: Immediate notifications on delivery status and delays. • Seamless Integration: Easy connection with e-commerce platforms. • Scalability: Efficient handling of large order volumes.

Resource Requirements :

Resource Type	**Description**	**Specification/Allocation**
Hardware		
Computing Resources	CPU/GPU specifications, number of cores	Intel Core i5 10^{th} Gen
Memory	RAM specifications	8 GB
Storage	Disk space for data, models, and logs	1 TB SSD
Software		
Frameworks	Python frameworks	Flask
Libraries	Additional libraries	scikit-learn, pandas, numpy, sklearn, matplotlib
Development Environment	IDE, version control	Jupyter Notebook, Git
Data		
Data	Source, size, format	Kaggle dataset

2.3. Initial Project Planning :

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members	Sprint Start Date	Sprint End Date (Planned)
Sprint-1	Define Problem / Problem Understanding	ECSP-5	Specify The Business Problem	2	Low	Vani Jain	5/7/2024	6/7/2024
Sprint-1	Data Collection & Preparation	ECSP -7	Collect The Dataset	5	Low	Sneha Kavitake	5/7/2024	6/7/2024
Sprint-1	Data Collection & Preparation	ECSP -8	Data Preparation	5	Medium	Sneha Kavitake	6/7/2024	7/7/2024
Sprint-2	Exploratory Data Analysis	ECSP -10	Descriptive Statistics	2	Low	Omkar Prasad	7/7/2024	8/7/2024
Sprint-2	Exploratory Data Analysis	ECSP -11	Visual Analysis	3	Medium	Dheeraj Kosuri	8/7/2024	9/7/2024
Sprint-3	Model Building	ECSP -11	Training The Model In Multiple Algorithms	7	High	Omkar Prasad	8/7/2024	11/7/2024
Sprint-3	Model Building	ECSP -11	Testing The Model	10	High	Sneha Kavitake	9/7/2024	11/7/2024
Sprint-4	Performance Testing & Hyperparameter Tuning	ECSP -11	Testing Model With Multiple Evaluation Metrics	10	High	Vani Jain	9/7/1014	11/7/2024
Sprint-5	Model Deployment	ECSP -11	Integrate With Web Framework	8	High	Dheeraj Kosuri	10/7/2024	11/7/2024

3. Data Collection and Preprocessing Phase

3.1. Data Collection Plan and Raw Data Sources Identified :

We've designed this template to help our team gather the essential data for predicting shipping times in e-commerce using machine learning. It provides a clear roadmap for identifying and organizing raw data sources, ensuring we capture key details like order info, shipping durations, customer data, and logistics. The focus is on collecting high-quality, relevant, and consistent data for an accurate prediction model. Key sections guide us through sourcing, collecting, validating data, and addressing any integration challenges.

Section	Description
Project Overview	The E-Commerce Shipping Prediction project aims to forecast shipping times using machine learning. Key steps include collecting and preprocessing data on orders, customers, and shipping logistics, performing exploratory data analysis, selecting and training regression or classification models, and deploying the model for real-time or batch predictions. The project seeks to provide accurate delivery estimates, optimize shipping operations, and enhance customer satisfaction while addressing challenges like data quality and model scalability.
Data Collection Plan	The dataset was obtained from the E-Commerce Shipping Data from Kaggle.
Raw Data Sources Identified	• ID: ID Number of Customers. • Warehouse block: The Company have big Warehouse which is divided in to block such as A,B,C,D,E. • Mode of shipment:The Company Ships the products in multiple way such as Ship, Flight and Road. • Customer care calls: The number of calls made from enquiry for enquiry of the shipment. • Customer rating: The company has rated from every customer. 1 is the lowest (Worst), 5 is the highest (Best). • Cost of the product: Cost of the Product in US Dollars. • Prior purchases: The Number of Prior Purchase. • Product importance: The company has categorized the product in the various parameter such as low, medium, high. • Gender: Male and Female. • Discount offered: Discount offered on that specific product. • Weight in gms: It is the weight in grams. • Reached on time: It is the target variable, where 1 Indicates that the product has NOT reached on time and 0 indicates it has reached on time.

Source Name	Description	Location/URL	Format	Size	Access Permissions
E-Commerce Shipping Data - Kaggle	The dataset used for model building contained 10999 observations of 12 variables.	https://www.kaggle.com/datasets/prachi13/customer-analytics?select=Train.csv	CSV	430 KB	Public

3.2. Data Quality Report :

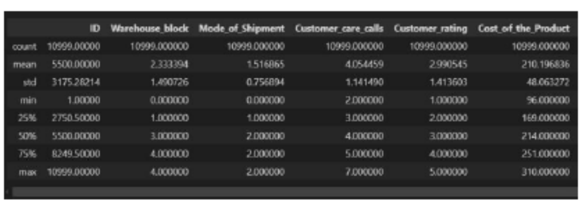
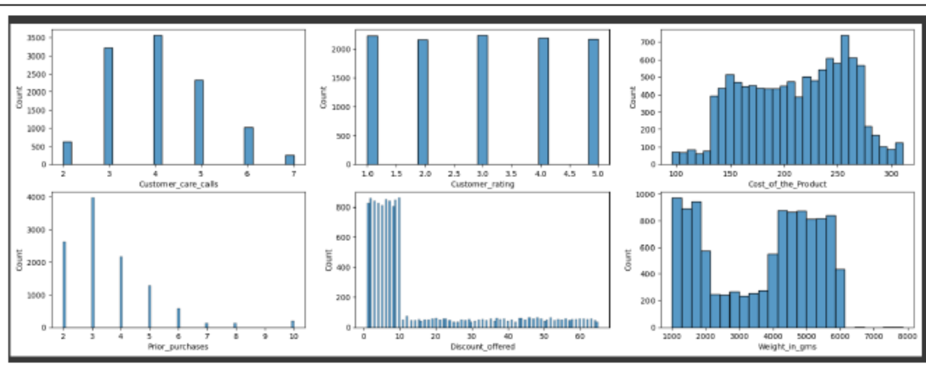
The Data Quality Report will summarise data quality issues from the selected source, including severity levels and resolution plans. It will aid in systematically identifying and rectifying data discrepancies.

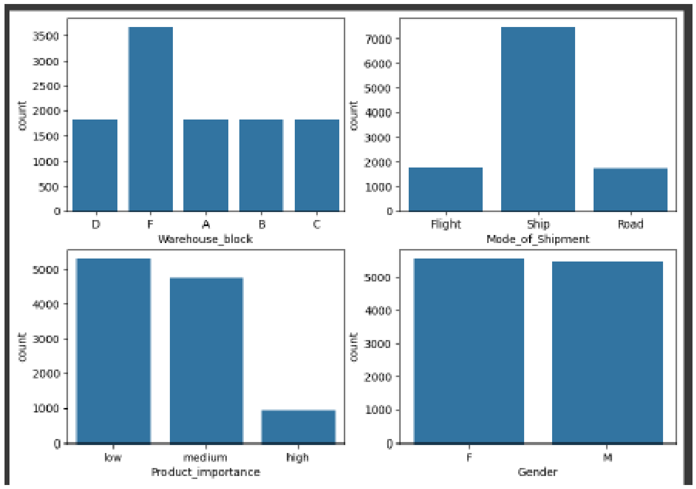
- **Data Integrity**: Maintaining the accuracy and consistency of data across all systems is critical. Regardless of the dataset—whether it pertains to orders, shipping statuses, or warehouse operations—ensuring that the data accurately reflects real-world events is essential for building a reliable predictive model.
- **Data Completeness**: All datasets must be complete with minimal missing values to ensure that the machine learning model has the full context required for accurate predictions. Completeness is a common requirement across all data sources to avoid gaps in the prediction process.
- **Consistency Across Systems**: Data consistency is a fundamental requirement across all datasets. Whether dealing with dates, product types, or shipping methods, having uniform data formats and values is crucial. Inconsistent data can lead to errors in the model and reduce the reliability of predictions.
- **Timeliness**: The timeliness of data is universally important across all datasets. For instance, timely updates on shipping statuses and order processing times are necessary to make real-time predictions. Delays in data updates can lead to outdated predictions and poor customer experiences.
- **Data Validation**: Ensuring the validity of the data across all datasets is essential. This involves making sure that data values fall within acceptable ranges and adhere to expected formats, such as valid postal codes or realistic delivery times.

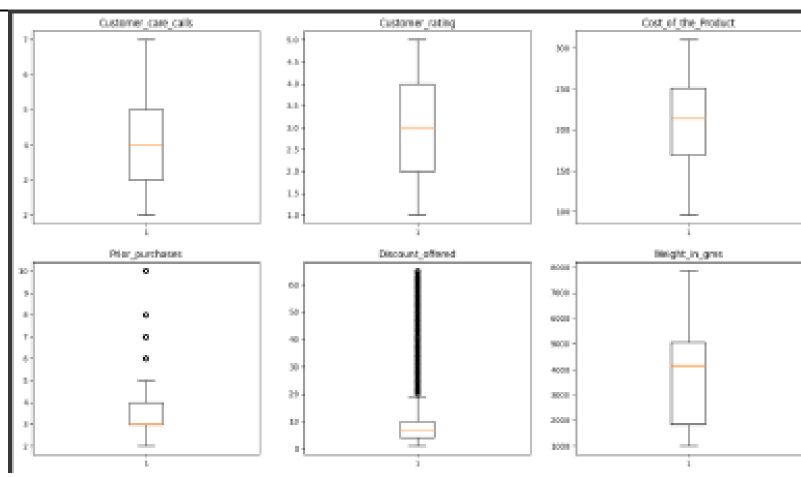
Data Source	Data Quality Issue	Severity	Resolution Plan
https://www.kaggle.com/datasets/prachi13/customer-analytics?select=Train.csv	Imbalanced target variable	Moderate	Handle the class imbalance using SMOTE,SMOTE-TOMEK,ADASYN, or SMOTE-ENN techniques.

3.3. Data Exploration and Preprocessing :

Dataset variables will be statistically analysed to identify patterns and outliers, with Python employed for preprocessing tasks like normalisation and feature engineering. Data cleaning will address missing values and outliers, ensuring quality for subsequent analysis and modelling, and forming a strong foundation for insights and predictions.

<table>
<tr><th>Section</th><th>Description</th></tr>
<tr><td>Data Overview</td><td>Dimensions : 10999 rows x 12 columns

<table>
<tr><th></th><th>ID</th><th>Warehouse_block</th><th>Mode_of_Shipment</th><th>Customer_care_calls</th><th>Customer_rating</th><th>Cost_of_the_Product</th></tr>
<tr><td>count</td><td>10999.00000</td><td>10999.000000</td><td>10999.000000</td><td>10999.000000</td><td>10999.000000</td><td>10999.000000</td></tr>
<tr><td>mean</td><td>5500.00000</td><td>2.333394</td><td>1.516865</td><td>4.054459</td><td>2.990545</td><td>210.196836</td></tr>
<tr><td>std</td><td>3175.28214</td><td>1.490726</td><td>0.756894</td><td>1.141490</td><td>1.413603</td><td>48.063272</td></tr>
<tr><td>min</td><td>1.00000</td><td>0.000000</td><td>0.000000</td><td>2.000000</td><td>1.000000</td><td>96.000000</td></tr>
<tr><td>25%</td><td>2750.50000</td><td>1.000000</td><td>1.000000</td><td>3.000000</td><td>2.000000</td><td>169.000000</td></tr>
<tr><td>50%</td><td>5500.00000</td><td>3.000000</td><td>2.000000</td><td>4.000000</td><td>3.000000</td><td>214.000000</td></tr>
<tr><td>75%</td><td>8249.50000</td><td>4.000000</td><td>2.000000</td><td>5.000000</td><td>4.000000</td><td>251.000000</td></tr>
<tr><td>max</td><td>10999.00000</td><td>4.000000</td><td>2.000000</td><td>7.000000</td><td>5.000000</td><td>310.000000</td></tr>
</table>
</td></tr>
<tr><td>Univariate Analysis</td><td>
</td></tr>
</table>

Bivariate Analysis	
Multivariate Analysis	

Outliers and Anomalies	

Data Preprocessing Code Screenshots

Loading Data

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

	ID	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	1	D	Flight	4	2	177	3	low	F	44	1233	1
1	2	F	Flight	4	5	216	2	low	M	59	3088	1
2	3	A	Flight	2	2	183	4	low	M	48	3374	1
3	4	B	Flight	3	3	176	4	medium	M	10	1177	1
4	5	C	Flight	2	2	184	3	medium	F	46	2484	1

Handling Missing Data

```
[7] data.isnull().sum()

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 Encoding

```
[10] from sklearn.preprocessing import LabelEncoder
     le = LabelEncoder()
     data.Warehouse_block = le.fit_transform(data.Warehouse_block)
     data.Mode_of_Shipment = le.fit_transform(data.Mode_of_Shipment)
     data.Product_importance = le.fit_transform(data.Product_importance)
     data.Gender = le.fit_transform(data.Gender)
     data.head()
```

	ID	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	1	3	0	4	2	177	3	1	0	44	1233	1
1	2	4	0	4	5	216	2	1	1	59	3088	1
2	3	0	0	2	2	183	4	1	1	48	3374	1
3	4	1	0	3	3	176	4	2	1	10	1177	1
4	5	2	0	2	2	184	3	2	0	46	2484	1

Data Transformation

```
[18] from sklearn.preprocessing import StandardScaler
     scale=StandardScaler()
     xnorm_train = scale.fit_transform(x_train)
     xnorm_test = scale.fit_transform(x_test)

[19] from sklearn.preprocessing import MinMaxScaler
     norm=MinMaxScaler()
     x=norm.fit_transform(x)
     x
```

4. Model Development Phase

4.1. Feature Selection Report :

In the forthcoming update, each feature will be accompanied by a brief description. Users will indicate whether it's selected or not, providing reasoning for their decision. This process will streamline decision-making and enhance transparency in feature selection.

Feature	Description	Selected (Yes/No)	Reasoning
ID	Customer IDs.	No	It doesn't help predict delivery time and could make the model less accurate by adding unnecessary information.
Warehouse_block	The Company's Warehouse is segmented into blocks labeled A, B, C, D, and E.	Yes	The warehouse location can affect processing speed and dispatching, which in turn impacts delivery times.
Mode_of_Shipment	The Company uses different transportation methods for shipping products (ship, flight, etc).	Yes	Different transit durations and dependability of shipping methods (air, sea, land) make them essential for forecasting timely deliveries.
Customer_care_calls	The number of calls made to the service center.	Yes	The likelihood of on-time delivery can be influenced by the number of calls, which may reveal potential issues or delays.
Customer_rating	The customers rate their experiences.	Yes	Customer feedback can indirectly influence the effectiveness of handling and shipping operations by indicating previous experiences and levels of satisfaction.
Cost_of_the_Product	The cost of the product.	Yes	Expensive items might get faster processing and shipping, which can impact how quickly they are delivered.
Prior_purchases	The number of prior purchases.	Yes	A customer's buying record can affect the dependability and speed of shipping, since regular customers might be given preferential treatment.

Product_importance	The products are categorised into 3 parameters that are low, medium, and high.	Yes	The priority level of a product will tell us how quickly it needs to be shipped and the method used, affecting how fast it arrives.
Gender	Male or female.	Yes	A customer's gender could be linked to particular delivery preferences, which might affect delivery times.
Discount_offered	Discount offered on a product.	Yes	Products with larger discounts might be sent through slower shipping methods to reduce expenses, thus impacting delivery times.
Weight_in_gms	Weight of the product in grams.	Yes	This can influence the choice of shipping methods and transit times, which can affect delivery accuracy.
Reached on Time Y.N	It is the target variable (1 - product has not reached on time, 0 – product has reached on time.	Yes	This can affect the selection of shipping methods and transit times, thereby impacting delivery accuracy..

4.2. Model Selection Report :

The model selection process involved training various machine learning models, including Random Forest, Decision Tree, Logistic Regression, Logistic RegressionCV, XGBoost (XGB), K-Nearest Neighbors (KNN), and Ridge Classifier. Historical shipping data was used to train these models, and their performance was evaluated through 5-fold cross-validation to ensure robustness. Hyperparameter tuning was conducted using grid search and random search techniques to optimize each model's performance. The models were then compared based on accuracy, precision, recall, F1 score, ROC-AUC, training time, and inference time to determine the best fit for the project.

Random Forest models are particularly valued for their ability to handle complex datasets and provide robust predictions. They combine multiple decision trees to improve predictive accuracy and reduce the risk of over fitting. This ensemble approach makes Random Forest models highly effective in capturing the intricate patterns within shipping data, leading to more reliable delivery time predictions. Additionally, the model's inherent feature importance ranking aids in identifying the most influential factors in predicting delivery times, providing valuable insights for stakeholders.

XGBoost is known for its exceptional performance and scalability in managing large, complex datasets. Using gradient boosting techniques, XGBoost iteratively enhances model accuracy, ensuring reliable predictions even in dynamic e-commerce environments. KNN, with its straightforward approach to pattern recognition, predicts delivery times by comparing new data points with existing observations, making it adaptable to varying shipping conditions.

The report underscores the importance of selecting models that align with the project's objectives of accurately predicting delivery times and optimising logistics operations in e-commerce. Each model's strengths in managing complex shipping dynamics, scalability, interpretability, and predictive accuracy were meticulously evaluated to ensure they effectively contribute to enhancing operational efficiency and customer satisfaction.

4.3. Initial Model Training Code, Model Validation and Evaluation Report :

The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include classification reports, accuracy, and confusion matrices for multiple models, presented through respective screenshots.

Initial Model Training Code:

```
[22] from sklearn import svm
     from sklearn.linear_model import LogisticRegression, LogisticRegressionCV, RidgeClassifier
     from sklearn.neighbors import KNeighborsClassifier
     from sklearn.ensemble import RandomForestClassifier
     from sklearn.model_selection import GridSearchCV
     from xgboost import XGBClassifier
     from sklearn.preprocessing import Normalizer
     from sklearn.metrics import accuracy_score, f1_score, recall_score, precision_score, confusion_matrix

     def model_evaluation(x_train,y_train,x_test,y_test):
         lr=LogisticRegression(random_state=1234)
         lr.fit(x_train,y_train)
         print('LOGISTIC REGRESSION')
         print('Train Score:',lr.score(x_train,y_train))
         print('Test Score:',lr.score(x_test,y_test))
         print()

         lcv=LogisticRegressionCV(random_state=1234)
         lcv.fit(x_train,y_train)
         print('LOGISTIC REGRESSION CV')
         print('Train Score:',lcv.score(x_train,y_train))
         print('Test Score:',lcv.score(x_test,y_test))
         print()
```

```python
[22] from sklearn import svm
     from sklearn.linear_model import LogisticRegression, LogisticRegressionCV, RidgeClassifier
     from sklearn.neighbors import KNeighborsClassifier
     from sklearn.ensemble import RandomForestClassifier
     from sklearn.model_selection import GridSearchCV
     from xgboost import XGBClassifier
     from sklearn.preprocessing import Normalizer
     from sklearn.metrics import accuracy_score, f1_score, recall_score, precision_score, confusion_matrix

     def model_evaluation(x_train,y_train,x_test,y_test):
         lr=LogisticRegression(random_state=1234)
         lr.fit(x_train,y_train)
         print('LOGISTIC REGRESSION')
         print('Train Score:',lr.score(x_train,y_train))
         print('Test Score:',lr.score(x_test,y_test))
         print()

         lcv=LogisticRegressionCV(random_state=1234)
         lcv.fit(x_train,y_train)
         print('LOGISTIC REGRESSION CV')
         print('Train Score:',lcv.score(x_train,y_train))
         print('Test Score:',lcv.score(x_test,y_test))
         print()
```


```python
[22] from sklearn import svm
     from sklearn.linear_model import LogisticRegression, LogisticRegressionCV, RidgeClassifier
     from sklearn.neighbors import KNeighborsClassifier
     from sklearn.ensemble import RandomForestClassifier
     from sklearn.model_selection import GridSearchCV
     from xgboost import XGBClassifier
     from sklearn.preprocessing import Normalizer
     from sklearn.metrics import accuracy_score, f1_score, recall_score, precision_score, confusion_matrix

     def model_evaluation(x_train,y_train,x_test,y_test):
         lr=LogisticRegression(random_state=1234)
         lr.fit(x_train,y_train)
         print('LOGISTIC REGRESSION')
         print('Train Score:',lr.score(x_train,y_train))
         print('Test Score:',lr.score(x_test,y_test))
         print()

         lcv=LogisticRegressionCV(random_state=1234)
         lcv.fit(x_train,y_train)
         print('LOGISTIC REGRESSION CV')
         print('Train Score:',lcv.score(x_train,y_train))
         print('Test Score:',lcv.score(x_test,y_test))
         print()
```


Model Validation and Evaluation Report:

Model	**Classification Report**	**Accuracy**	**Confusion Matrix**

logistic regression	`print(classification_report(y_test,y_pred))`
	<pre> precision recall f1-score support
	
	           0 0.56 0.56 0.56 896
	           1 0.70 0.69 0.70 1304
	
	    accuracy 0.64 2200
	   macro avg 0.63 0.63 0.63 2200
	weighted avg 0.64 0.64 0.64 2200</pre>
		64%	[[503 393]
			 [398 906]]
			`print(confusion_matrix(y_test,y_pred))`

logistic regression CV	`print(classification_report(y_test,y_pred))`
	
	`              precision    recall  f1-score   support`
	`           0       0.56      0.52      0.54       896`
	`           1       0.69      0.72      0.70      1304`
	`    accuracy                           0.64      2200`
	`   macro avg       0.62      0.62      0.62      2200`
	`weighted avg       0.63      0.64      0.64      2200`
		64%	`[[463 433]`
			` [362 942]]`
			`print(confusion_matrix(y_test,y_pred))`

XGBoost	`print(classification_report(y_test,y_pred))`
	
	`              precision    recall  f1-score   support`
	`           0       0.57      0.64      0.60       896`
	`           1       0.73      0.67      0.70      1304`
	`    accuracy                           0.66      2200`
	`   macro avg       0.65      0.65      0.65      2200`
	`weighted avg       0.66      0.66      0.66      2200`
		66%	`[[573 323]`
			` [436 868]]`
			`print(confusion_matrix(y_test,y_pred))`

ridge classifier	`print(classification_report(y_test,y_pred))`
	
	`              precision    recall  f1-score   support`
	`           0       0.56      0.66      0.61       896`
	`           1       0.74      0.65      0.69      1304`
	`    accuracy                           0.65      2200`
	`   macro avg       0.65      0.65      0.65      2200`
	`weighted avg       0.66      0.65      0.66      2200`
		65%	`[[593 303]`
			` [462 842]]`
			`print(confusion_matrix(y_test,y_pred))`

K nearest neighbors	`print(classification_report(y_test,y_pred))`
	
	`              precision    recall  f1-score   support`
	`           0       0.55      0.57      0.56       896`
	`           1       0.70      0.68      0.69      1304`
	`    accuracy                           0.63      2200`
	`   macro avg       0.62      0.62      0.62      2200`
	`weighted avg       0.64      0.63      0.64      2200`
		63%	`[[511 385]`
			` [420 884]]`
			`print(confusion_matrix(y_test,y_pred))`

random forest	`print(classification_report(y_test,y_pred))`
	
	`              precision    recall  f1-score   support`
	`           0       0.56      0.66      0.61       896`
	`           1       0.74      0.65      0.69      1304`
	`    accuracy                           0.65      2200`
	`   macro avg       0.65      0.65      0.65      2200`
	`weighted avg       0.66      0.65      0.66      2200`
		66%	`[[593 303]`
			` [462 842]]`
			`print(confusion_matrix(y_test,y_pred))`

support vector classifier	`print(classification_report(y_test,y_pred))`
	
	`              precision    recall  f1-score   support`
	`           0       0.56      0.82      0.66       896`
	`           1       0.82      0.56      0.66      1304`
	`    accuracy                           0.66      2200`
	`   macro avg       0.69      0.69      0.66      2200`
	`weighted avg       0.71      0.66      0.66      2200`
		66%	`[[734 162]`
			` [578 726]]`
			`print(confusion_matrix(y_test,y_pred))`

5. Model Optimization and Tuning Phase

5.1. Hyper-parameter Tuning Documentation :

Model	Tuned Hyperparameters	Optimal Values
LOGISTIC REGRESSION	Solver,penalty,C,max_iter	Best Cross-Validation Score: 0.6511227264283181 Accuracy with Hyperparameter Tuning and SMOTE: 0.64363636
RANDOM FOREST CLASSIFIER	n_estimators, max_depth, min_samples_split, min_samples_leaf	Best Cross-Validation Score: 0.6839047936339164 Accuracy with Hyperparameter Tuning and SMOTE: 0.6893939393939394
KNN	n_neighbours,weights,metric,p	Best Cross-Validation Score: 0.6530739869775356 Accuracy with Hyperparameter Tuning and SMOTE: 0.6633333333333333
XG BOOST	n_estimators,max_depth,learning_rate,subsample	Best Cross-Validation Score: 0.6822955536990625 Accuracy with Hyperparameter Tuning and SMOTE: 0.690606
SVC	Kernel,C,gamma	Best Cross-Validation Score: 0.6680090799389043 Accuracy with Hyperparameter Tuning and SMOTE: 0.670303

5.2. Performance Metrics Comparison Report :

Model	Baseline Metric	Optimized Metric
LOGISTIC REGRESSION	Accuracy without Hyperparameter Tuning and SMOTE: 0.6284848484848485 Classification Report without Hyperparameter Tuning and SMOTE: precision recall f1-score support 0 0.56 0.53 0.54 1379 1 0.67 0.70 0.69 1921 accuracy 0.63 3300 macro avg 0.62 0.61 0.61 3300 weighted avg 0.63 0.63 0.63 3300 Confusion Matrix without Hyperparameter Tuning and SMOTE: [[725 654] [572 1349]]	Accuracy with Hyperparameter Tuning and SMOTE: 0.64363636 Classification Report with Hyperparameter Tuning and SMOTE: precision recall f1-score support 0 0.55 0.76 0.64 1379 1 0.77 0.56 0.65 1921 accuracy 0.64 3300 macro avg 0.66 0.66 0.64 3300 weighted avg 0.68 0.64 0.64 3300 Confusion Matrix with Hyperparameter Tuning and SMOTE: [[1053 326] [850 1071]]
RANDOM FOREST CLASSIFIER	Accuracy without Hyperparameter Tuning and SMOTE: 0.690000 Classification Report without Hyperparameter Tuning and SMOTE: precision recall f1-score support 0 0.58 0.91 0.71 1379 1 0.89 0.53 0.67 1921 accuracy 0.69 3300 macro avg 0.74 0.72 0.69 3300 weighted avg 0.76 0.69 0.69 3300 Confusion Matrix without Hyperparameter Tuning and SMOTE: [[1250 129] [894 1027]]	Accuracy with Hyperparameter Tuning and SMOTE: 0.6893939393939394 Classification Report with Hyperparameter Tuning and SMOTE: precision recall f1-score support 0 0.58 0.95 0.72 1379 1 0.93 0.50 0.65 1921 accuracy 0.69 3300 macro avg 0.75 0.73 0.69 3300 weighted avg 0.78 0.69 0.68 3300 Confusion Matrix with Hyperparameter Tuning and SMOTE: [[1306 73] [952 969]]
KNN	Accuracy without Hyperparameter Tuning and SMOTE: 0.6669696969696697 Classification Report without Hyperparameter Tuning and SMOTE: precision recall f1-score support 0 0.58 0.75 0.65 1379 1 0.77 0.61 0.68 1921 accuracy 0.67 3300 macro avg 0.68 0.68 0.67 3300 weighted avg 0.69 0.67 0.67 3300 Confusion Matrix without Hyperparameter Tuning and SMOTE: [[1035 344] [755 1166]]	Accuracy with Hyperparameter Tuning and SMOTE: 0.6633333333333333 Classification Report with Hyperparameter Tuning and SMOTE: precision recall f1-score support 0 0.57 0.81 0.67 1379 1 0.80 0.56 0.66 1921 accuracy 0.66 3300 macro avg 0.68 0.68 0.66 3300 weighted avg 0.70 0.66 0.66 3300 Confusion Matrix with Hyperparameter Tuning and SMOTE: [[1111 268] [843 1078]]
XG BOOST	Accuracy without Hyperparameter Tuning and SMOTE: 0.644545 Classification Report without Hyperparameter Tuning and SMOTE: precision recall f1-score support 0 0.57 0.62 0.59 1379 1 0.71 0.66 0.68 1921 accuracy 0.64 3300 macro avg 0.64 0.64 0.64 3300 weighted avg 0.65 0.64 0.65 3300 Confusion Matrix without Hyperparameter Tuning and SMOTE: [[853 526] [647 1274]]	Accuracy with Hyperparameter Tuning and SMOTE: 0.690606 Classification Report with Hyperparameter Tuning and SMOTE: precision recall f1-score support 0 0.58 0.96 0.72 1379 1 0.94 0.50 0.65 1921 accuracy 0.69 3300 macro avg 0.76 0.73 0.69 3300 weighted avg 0.79 0.69 0.68 3300 Confusion Matrix with Hyperparameter Tuning and SMOTE: [[1320 59] [962 959]]
SVC	Accuracy without Hyperparameter Tuning and SMOTE: 0.668788 Classification Report without Hyperparameter Tuning and SMOTE: precision recall f1-score support 0 0.57 0.82 0.67 1379 1 0.81 0.56 0.66 1921 accuracy 0.67 3300 macro avg 0.69 0.69 0.67 3300 weighted avg 0.71 0.67 0.67 3300 Confusion Matrix without Hyperparameter Tuning and SMOTE: [[1129 250] [843 1078]]	Accuracy with Hyperparameter Tuning and SMOTE: 0.670303 Classification Report with Hyperparameter Tuning and SMOTE: precision recall f1-score support 0 0.56 0.92 0.70 1379 1 0.90 0.49 0.63 1921 accuracy 0.67 3300 macro avg 0.73 0.71 0.67 3300 weighted avg 0.76 0.67 0.66 3300 Confusion Matrix with Hyperparameter Tuning and SMOTE: [[1274 105] [983 938]]

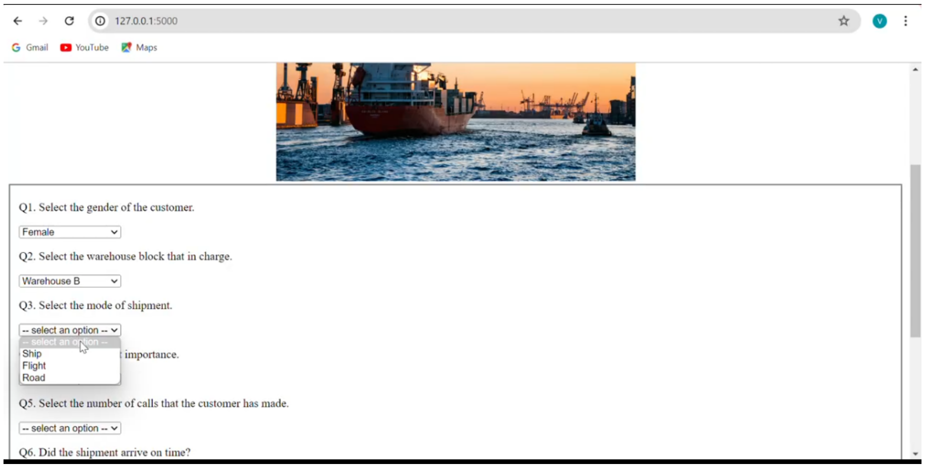
5.3. Final Model Selection Justification :

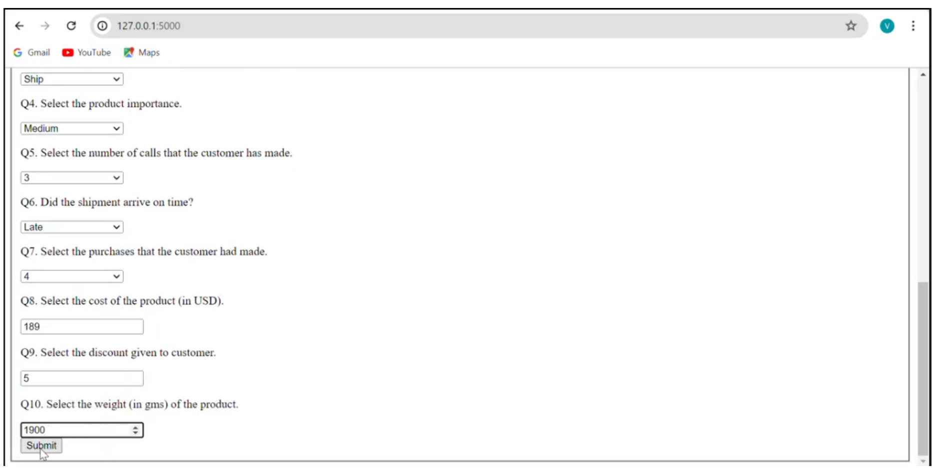
The Random Forest Classifier was chosen as the final model due to its superior performance across multiple evaluation metrics. During hyper parameter tuning, it consistently demonstrated high accuracy, outperforming other models in handling the complexity of the shipping data. The Random Forest Classifier's ensemble approach, which combines the predictions of multiple decision trees, not only enhances predictive accuracy but also reduces the likelihood of over fitting. This capability to capture intricate patterns and relationships within the data ensures more reliable and robust delivery time predictions. Additionally, its inherent feature importance analysis provided valuable insights into the key factors influencing delivery times, making it a powerful tool for both predictive accuracy and interpretability. By aligning with the project's goals of optimising logistics operations and enhancing customer satisfaction, the Random Forest Classifier proved to be the most suitable model, ensuring that the final solution is both effective and scalable in a dynamic e-commerce environment.

Final Model	Reasoning
RANDOM FOREST CLASSIFIER	The RANDOM FOREST CLASSIFIER model was selected for its superior performance, exhibiting high accuracy during hyperparameter tuning. Its ability to handle complex relationships, minimize overfitting, and optimize predictive accuracy aligns with project objectives, justifying its selection as the final model

6. Results

6.1. Output Screenshots :





7. Advantages & Disadvantages

Advantages :

- **Enhanced Customer Experience:** Providing accurate delivery time estimates can boost customer satisfaction by setting reliable expectations for when products will arrive.
- **Informed Decision-Making:** Utilizing data from multiple sources enables more informed decisions in logistics management, leading to improved strategic planning.
- **Operational Optimization:** Using predictive models to optimize logistics can streamline operations, leading to cost savings by minimising delays and improving resource allocation.
- **Resource Optimization:** Accurately predicting delivery times allows for more efficient management of inventory and staffing, reducing waste and maximising resource use.
- **Market Differentiation:** Offering precise delivery time estimates can set the e-commerce business apart from competitors, helping to attract and retain a larger customer base.

Disadvantages :

- **Managing Customer Expectations:** While accurate predictions can improve satisfaction, any discrepancies between expected and actual delivery times might lead to customer dissatisfaction
- **Model Accuracy Limitations:** Predictive models may sometimes fall short in forecasting delivery times due to unexpected events or inaccuracies in the data.
- **Reliance on External Variables:** Factors outside of the business's control, like weather and traffic conditions, can greatly affect delivery times, adding a layer of uncertainty to predictions.
- **Implementation Costs:** Developing and maintaining predictive models and integrating real-time data sources can require significant initial and ongoing investments.
- **Complex Data Integration:** Managing and integrating diverse data sources, such as weather, traffic, and carrier information, can be challenging and require advanced data processing capabilities.

8. Conclusion

Implementing an e-commerce shipping prediction system is a strategic move that can greatly enhance both customer satisfaction and operational efficiency. By utilising advanced machine learning models and integrating real-time data, businesses can offer precise delivery estimates, reducing uncertainty and fostering customer trust. Although challenges exist, such as reliance on data quality and the complexity of integrating various data sources, the advantages—like improved logistics management, cost efficiency, and informed decision-making—far outweigh these hurdles. A well-designed shipping prediction system not only boosts operational efficiency but also ensures more reliable e-commerce operations. This reliability leads to a superior customer experience, with consistent and timely deliveries that build loyalty and strengthen brand perception. Additionally, predictive analytics allow businesses to continuously optimize their operations, quickly adapting to market changes and customer demands. Ultimately, this project on e-commerce shipping estimation will benefit customers and business owners by providing accurate shipping predictions, enabling proactive decision-making and enhancing customer loyalty to e-commerce platforms.

9. Future Scope

Future advancements in e-commerce shipping prediction are set to transform logistics management even further. Machine learning techniques will continue to improve, leading to more accurate delivery time estimates. The integration of real-time data from IoT devices, environmental sensors, and advanced traffic systems will provide deeper insights into shipping dynamics, enabling real-time route optimization and resource allocation. Predictive and prescriptive analytics will not only forecast delivery times but also suggest optimal actions to boost efficiency and cut costs. Blockchain technology offers potential for enhancing supply chain transparency and security, enabling seamless tracking of shipments.

Personalized shipping predictions based on customer segmentation and behaviour analysis will become more common, improving customer satisfaction and retention. As e-commerce expands globally, continuous research in shipping prediction will be essential to meet evolving consumer expectations and market demands. Embracing these technological advancements will allow e-commerce businesses to maintain a competitive edge, delivering high levels of service and operational excellence. This personalized approach, tailored to customer preferences, will optimize resource allocation and inventory management, driving growth and customer loyalty in the dynamic world of e-commerce logistics.

10. Appendix

10.1. Source Code :

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb

import warnings
warnings.filterwarnings('ignore')
```

In [4]:
```
data=pd.read_csv('Train.csv')
data.head()
```

Out[4]:

	ID	Warehouse_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	Cost_of_the_Product	Prior_purchases	Product_imp
0	1	D	Flight	4	2	177	3	
1	2	F	Flight	4	5	216	2	
2	3	A	Flight	2	2	183	4	
3	4	B	Flight	3	3	176	4	
4	5	C	Flight	2	2	184	3	

In [5]:
```
data.shape
```

Out[5]: (10999, 12)

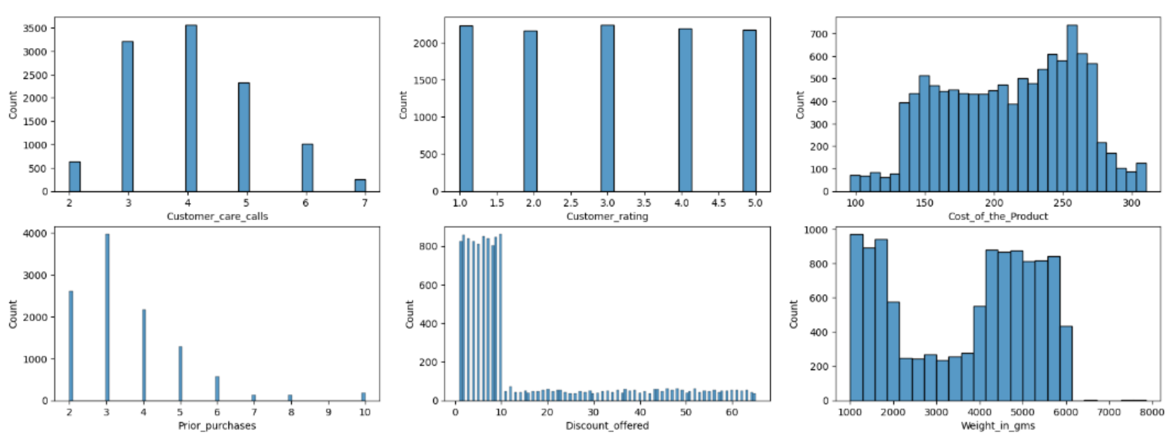
In [6]:
```
data.info()
```

```
<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
```

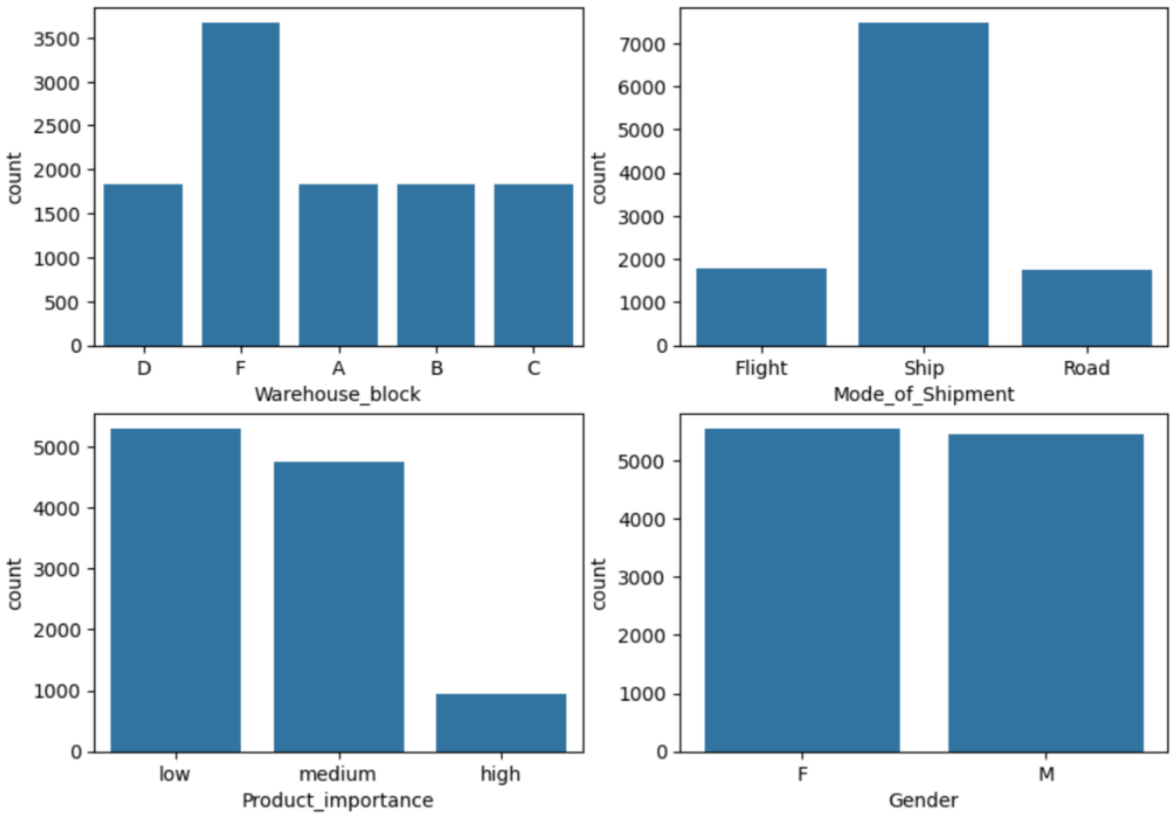
In [7]:
```
data.isnull().sum()
```

Out[7]:
```
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
```

In [8]:
```
plt.figure(figsize=(20,7))
plt.subplot(2,3,1)
sb.histplot(data['Customer_care_calls'])
plt.subplot(2,3,2)
sb.histplot(data['Customer_rating'])
plt.subplot(2,3,3)
sb.histplot(data['Cost_of_the_Product'])
plt.subplot(2,3,4)
sb.histplot(data['Prior_purchases'])
plt.subplot(2,3,5)
sb.histplot(data['Discount_offered'])
plt.subplot(2,3,6)
sb.histplot(data['Weight_in_gms'])
plt.show()
```



In [9]:
```
plt.figure(figsize=(10,7))
plt.subplot(2,2,1)
sb.countplot(data=data,x=data['Warehouse_block'])
plt.subplot(2,2,2)
sb.countplot(data=data,x=data['Mode_of_Shipment'])
plt.subplot(2,2,3)
sb.countplot(data=data,x=data['Product_importance'])
plt.subplot(2,2,4)
sb.countplot(data=data,x=data['Gender'])
plt.show()
```



In [10]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
data.Warehouse_block = le.fit_transform(data.Warehouse_block)
data.Mode_of_Shipment = le.fit_transform(data.Mode_of_Shipment)
data.Product_importance = le.fit_transform(data.Product_importance)
data.Gender = le.fit_transform(data.Gender)
data.head()
```

Out[10]:

	ID	Warehouse_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	Cost_of_the_Product	Prior_purchases	Product_imp
0	1	3	0	4	2	177	3	
1	2	4	0	4	5	216	2	
2	3	0	0	2	2	183	4	
3	4	1	0	3	3	176	4	
4	5	2	0	2	2	184	3	

In [11]:
```
plt.figure(figsize = (18, 7))
sb.heatmap(data.corr(), annot = True, fmt = '0.2f', annot_kws = {'size' : 15}, linewidth = 5, linecolor = 'black')
plt.show()
```

In [12]:

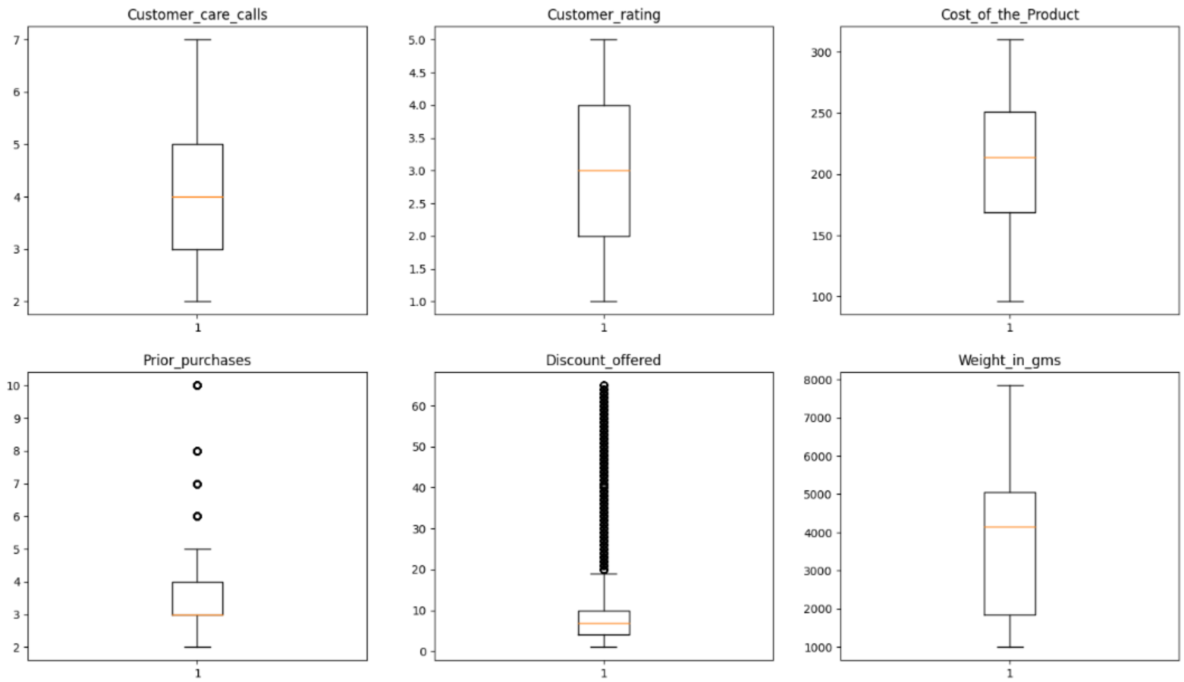
```
data.describe(include='all')
```

Out[12]:

	ID	Warehouse_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	Cost_of_the_Product	Prior_purchases
count	10999.00000	10999.000000	10999.000000	10999.000000	10999.000000	10999.000000	10999.000000
mean	5500.00000	2.333394	1.516865	4.054459	2.990545	210.196836	3.567597
std	3175.28214	1.490726	0.756894	1.141490	1.413603	48.063272	1.522860
min	1.00000	0.000000	0.000000	2.000000	1.000000	96.000000	2.000000
25%	2750.50000	1.000000	1.000000	3.000000	2.000000	169.000000	3.000000
50%	5500.00000	3.000000	2.000000	4.000000	3.000000	214.000000	3.000000
75%	8249.50000	4.000000	2.000000	5.000000	4.000000	251.000000	4.000000
max	10999.00000	4.000000	2.000000	7.000000	5.000000	310.000000	10.000000

In [13]:

```
#Visualizing Outliers
c=0
plt.figure(figsize=(18,10))
for i in data.drop(columns=['ID','Warehouse_block','Mode_of_Shipment','Product_importance','Gender','Reached.on.Time_Y.N'
    if str(data[i].dtype)=='object':
        continue
    plt.subplot(2,3,c+1)
    plt.boxplot(data[i])
    plt.title(i)
    c+=1
plt.show()
```



In [14]:
```
categorical_attributes = []
numerical_attributes = []

for col in data.columns[1:-1]:
    if data[col].dtype == 'object':
        categorical_attributes.append(col)
    else:
        numerical_attributes.append(col)
print(f"Categorical attributes: {categorical_attributes}\nNumerical attributes: {numerical_attributes}")
```

```
Categorical attributes: []
Numerical attributes: ['Warehouse_block', 'Mode_of_Shipment', 'Customer_care_calls', 'Customer_rating', 'Cost_of_the_Produ
ct', 'Prior_purchases', 'Product_importance', 'Gender', 'Discount_offered', 'Weight_in_gms']
```

In [15]:
```
label_map={}
for i in data.columns:
    if str(data[i].dtype) is 'object':
        temp={}
        cats=data[i].unique()
        for index in range(len(cats)):
            temp[cats[index]]=index
        label_map[i]=temp
        #labeling
        data[i]=data[i].map(temp)
label_map
```

Out[15]:
```
{}
```

In [16]:

```
def check_outliers(arr):
    Q1=np.percentile(arr,25,interpolation='midpoint')
    Q3=np.percentile(arr,75,interpolation='midpoint')
    IQR=Q3-Q1
    upper=Q3+1.5*IQR
    upper_arr=np.array(arr>=upper)
    print('    ',len(upper_arr[upper_arr==True]),'are over the upper bound:',upper)
    lower=Q1-1.5*IQR
    lower_arr=np.array(arr<=lower)
    print('    ',len(lower_arr[lower_arr==True]),'are less than the lower bound:',lower)
for i in data.drop(columns=['ID','Warehouse_block','Mode_of_Shipment','Product_importance','Gender','Reached.on.Time_Y.N'
    if str(data[i].dtype)=='object':
        continue
    print(i)
    check_outliers(data[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
```

In [17]:

```
x=data.drop(columns=['ID','Reached.on.Time_Y.N'])
y=data['Reached.on.Time_Y.N']
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=1234,shuffle=True)
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(8799, 10)
(2200, 10)
(8799,)
(2200,)
```

In [18]:

```
from sklearn.preprocessing import StandardScaler
scale=StandardScaler()
xnorm_train = scale.fit_transform(x_train)
xnorm_test = scale.fit_transform(x_test)
```

In [19]:

```
from sklearn.preprocessing import MinMaxScaler
norm=MinMaxScaler()
x=norm.fit_transform(x)
x
```

```
Out[19]: array([[0.75      , 0.        , 0.4       , ..., 0.        , 0.671875  ,
         0.03389335],
        [1.        , 0.        , 0.4       , ..., 1.        , 0.90625   ,
         0.30489408],
        [0.        , 0.        , 0.        , ..., 1.        , 0.734375  ,
         0.34667641],
        ...,
        [0.5       , 1.        , 0.6       , ..., 0.        , 0.046875  ,
         0.02249817],
        [1.        , 1.        , 0.6       , ..., 1.        , 0.015625  ,
         0.03053324],
        [0.75      , 1.        , 0.        , ..., 0.        , 0.078125  ,
         0.09320672]])
```

```
In [20]:
x=data.drop(columns=['ID','Reached.on.Time_Y.N'])
y=data['Reached.on.Time_Y.N']
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=1234,shuffle=True)
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(8799, 10)
(2200, 10)
(8799,)
(2200,)
```

```
In [22]:
from sklearn import svm
from sklearn.linear_model import LogisticRegression, LogisticRegressionCV, RidgeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from xgboost import XGBClassifier
from sklearn.preprocessing import Normalizer
from sklearn.metrics import accuracy_score, f1_score, recall_score, precision_score, confusion_matrix

def model_evaluation(x_train,y_train,x_test,y_test):
    lr=LogisticRegression(random_state=1234)
    lr.fit(x_train,y_train)
    print('LOGISTIC REGRESSION')
    print('Train Score:',lr.score(x_train,y_train))
    print('Test Score:',lr.score(x_test,y_test))
    print()

    lcv=LogisticRegressionCV(random_state=1234)
    lcv.fit(x_train,y_train)
    print('LOGISTIC REGRESSION CV')
    print('Train Score:',lcv.score(x_train,y_train))
    print('Test Score:',lcv.score(x_test,y_test))
    print()

    xgb=XGBClassifier(random_state=1234)
    xgb.fit(x_train,y_train)
    print('XGBOOST')
    print('Train Score:',xgb.score(x_train,y_train))
    print('Test Score:',xgb.score(x_test,y_test))
    print()
```

```python
    rc=RidgeClassifier(random_state=1234)
    rc.fit(x_train,y_train)
    print('RIDGE CLASSIFIER')
    print('Train Score:',rc.score(x_train,y_train))
    print('Test Score:',rc.score(x_test,y_test))
    print()

    kn=KNeighborsClassifier()
    kn.fit(x_train,y_train)
    print('K NEIGHBORS CLASSIFIER')
    print('Train Score:',kn.score(x_train,y_train))
    print('Test Score:',kn.score(x_test,y_test))
    print()

    rf=RandomForestClassifier(random_state=1234)
    rf.fit(x_train,y_train)
    print('RANDOM FOREST CLASSIFIER')
    print('Train Score:',rf.score(x_train,y_train))
    print('Test Score:',rf.score(x_test,y_test))
    print()

    svc=svm.SVC(random_state=1234)
    svc.fit(x_train,y_train)
    print('SVM CLASSIFIER')
    print('Train Score:',svc.score(x_train,y_train))
    print('Test Score:',svc.score(x_test,y_test))
    print()

    return lr,lcv,xgb,rc,kn,rf,svc
```

In [23]:

```python
lr,lcv,xgb,rc,kn,rf,svc = model_evaluation(xnorm_train,y_train,xnorm_test,y_test)
```

```
LOGISTIC REGRESSION
Train Score: 0.6416638254347085
Test Score: 0.6404545454545455
```

```
LOGISTIC REGRESSION CV
Train Score: 0.6446187066712127
Test Score: 0.6386363636363637

XGBOOST
Train Score: 0.9136265484714172
Test Score: 0.6463636363636364

RIDGE CLASSIFIER
Train Score: 0.6529151039890897
Test Score: 0.6468181818181818

K NEIGHBORS CLASSIFIER
Train Score: 0.7734969882941243
Test Score: 0.634090909090909

RANDOM FOREST CLASSIFIER
Train Score: 1.0
Test Score: 0.6522727272727272

SVM CLASSIFIER
Train Score: 0.7053074212978747
Test Score: 0.6636363636363637
```

In [24]:

```
def eval(name,model):
    y_pred=model.predict(xnorm_test)
    result=[]
    result.append(name)
    result.append("{:.2f}".format(accuracy_score(y_test,y_pred)*100))
    result.append("{:.2f}".format(f1_score(y_test,y_pred)*100))
    result.append("{:.2f}".format(recall_score(y_test,y_pred)*100))
    result.append("{:.2f}".format(precision_score(y_test,y_pred)*100))
    return result
model_list={
    'logistic regression':lr,
    'logistic regression CV':lcv,
    'XGBoost':xgb,
    'ridge classifier':rf,
    'knn':kn,
    'random forest':rf,
    'support vector classifier':svc
}
model_eval_info=[]
for i in model_list.keys():
    model_eval_info.append(eval(i,model_list[i]))
model_eval_info=pd.DataFrame(model_eval_info,columns=['Name','Accuracy','F1_score','Recall','Precision'])
model_eval_info.to_csv('model_eval.csv')
model_eval_info
```

Out[24]:

	Name	Accuracy	F1_score	Recall	Precision
0	logistic regression	64.05	69.61	69.48	69.75
1	logistic regression CV	63.86	70.32	72.24	68.51
2	XGBoost	64.64	70.42	71.01	69.83
3	ridge classifier	65.23	68.76	64.57	73.54
4	knn	63.41	68.71	67.79	69.66
5	random forest	65.23	68.76	64.57	73.54
6	support vector classifier	66.36	66.24	55.67	81.76

In [25]:

```
comp_data=pd.read_csv('model_eval.csv')
comp_data
```

Out[25]:

	Unnamed: 0	Name	Accuracy	F1_score	Recall	Precision
0	0	logistic regression	64.05	69.61	69.48	69.75
1	1	logistic regression CV	63.86	70.32	72.24	68.51
2	2	XGBoost	64.64	70.42	71.01	69.83
3	3	ridge classifier	65.23	68.76	64.57	73.54
4	4	knn	63.41	68.71	67.79	69.66
5	5	random forest	65.23	68.76	64.57	73.54
6	6	support vector classifier	66.36	66.24	55.67	81.76

In [26]:

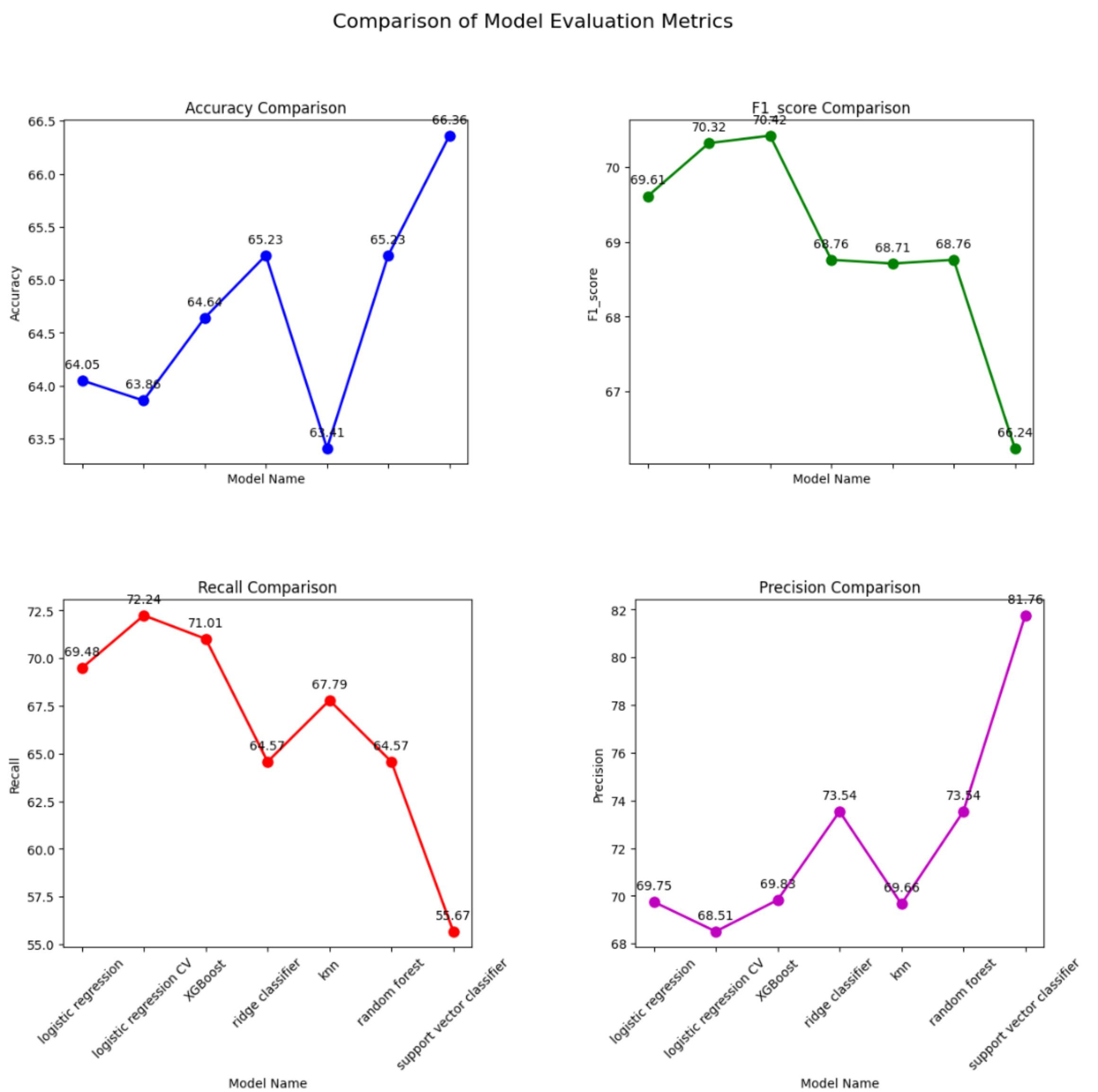
```python
metrics = ['Accuracy', 'F1_score', 'Recall', 'Precision']
colors = ['b', 'g', 'r', 'm']

# Set plot size and style
fig, axs = plt.subplots(2, 2, figsize=(14, 12), sharex=True)
plt.subplots_adjust(hspace=0.4, wspace=0.4)

# Plot each metric
for i, metric in enumerate(metrics):
    ax = axs[i//2, i%2]
    ax.plot(comp_data['Name'], comp_data[metric], marker='o', color=colors[i], linestyle='-', linewidth=2, markersize=8)
    ax.set_title(f'{metric} Comparison')
    ax.set_xlabel('Model Name')
    ax.set_ylabel(metric)
    ax.tick_params(axis='x', rotation=45)
    for j in range(len(comp_data)):
        ax.annotate(f'{comp_data[metric][j]:.2f}', (comp_data['Name'][j], comp_data[metric][j]), textcoords="offset point

# Set the main title
fig.suptitle('Comparison of Model Evaluation Metrics', fontsize=16)

# Show the plot
plt.show()
```



In [28]:

```
import pandas as pd
from sklearn.preprocessing import scale
from sklearn.neighbors import KNeighborsClassifier
import pickle

# Load dataset
ecomm = pd.read_csv("Train.csv")

# Rename columns
cols=[]
for i in ecomm.columns[1:-1]:
    i = i.lower()
    cols.append(i);
cols = ['ID'] +  cols
cols.append('arrival')
ecomm.columns = cols

# Data preprocessing
ecomm['gender'] = ecomm.gender.map({'F':0, 'M':1})
ecomm['customer_rating'] = ecomm['customer_rating'].map({5:0, 4:0, 3:0, 2:0, 1:1})
dummy = pd.DataFrame(pd.get_dummies(ecomm[['warehouse_block', 'mode_of_shipment','product_importance']]))
ecomm1 = pd.DataFrame(scale(ecomm[['cost_of_the_product','weight_in_gms','discount_offered']]),
                      columns=['cost_of_the_product','weight_in_gms','discount_offered'])
ecomm_final = pd.concat([ecomm1,dummy,ecomm[['customer_care_calls', 'prior_purchases','gender', 'arrival','customer_rati
                        axis=1)

# Split data into output and input
X = ecomm_final.iloc[:,:-1] # inputs
Y = ecomm_final['customer_rating'] # outputs

# Model building
KNN_model = KNeighborsClassifier(n_neighbors=11, metric='euclidean')
KNN_model.fit(X, Y)

# Save the model
filename = 'final.pkl'
pickle.dump(KNN_model, open(filename, 'wb'))
```

10.2. GitHub & Project Demo Link :

- Github Link : https://github.com/snehakavitake/ecommerce-shipping-prediction
- Project Demo Link
  https://drive.google.com/file/d/11In8cP1H4p452vTEdrAD1Z7kZRxNR-YT/view