E-commerce Shipping Prediction using Machine Learning

1. Introduction:

The e-commerce industry thrives on the promise of timely deliveries, yet the unpredictability of shipment arrival times can undermine customer satisfaction and trust. This project seeks to develop a robust predictive model to forecast whether an e-commerce shipment will be delivered on time. By leveraging historical shipment data and advanced machine learning techniques, we aim to provide reliable delivery estimates and optimize logistics operations.

1.1 Project Overviews

The project's primary goal is to enhance the reliability of e-commerce shipment deliveries through predictive analytics. Utilizing a dataset sourced from Kaggle, the project will analyze various factors that influence delivery times. Key phases include data preprocessing, feature engineering, model selection, and validation. By identifying and addressing data quality issues, we aim to build a dependable model that can accurately predict shipment delivery times.

The predictive model developed in this project will be integrated into the e-commerce platform, offering real-time delivery predictions. This will not only improve customer satisfaction by setting clear and realistic delivery expectations but also streamline logistics operations, reducing inefficiencies and associated costs.

1.2 Objectives

a. Develop an Accurate Predictive Model

The primary objective is to create a predictive model that can accurately forecast whether an e-commerce shipment will be delivered on time. This involves several critical steps:

- Data Exploration and Preprocessing: Cleaning the dataset, handling missing values.
- Feature Engineering: Identifying and creating meaningful features that enhance the predictive power of the model.
- Model Selection and Evaluation: Employing various machine learning algorithms such as logistic regression, decision trees, random forests, and gradient boosting to find the best-performing model. Model performance will be evaluated using metrics such as accuracy, precision, recall, and F1-score.

b. Improve Customer Satisfaction and Operational Efficiency

- Enhance Customer Trust: By providing accurate and reliable delivery estimates, the model will help build trust in the e-commerce platform, leading to increased customer satisfaction and loyalty.
- Optimize Logistics Operations: The insights gained from the predictive model will enable the e-commerce platform to address root causes of delays, optimize logistics operations, and reduce costs related to shipment delays and returns. This will result in a more efficient and cost-effective supply chain.

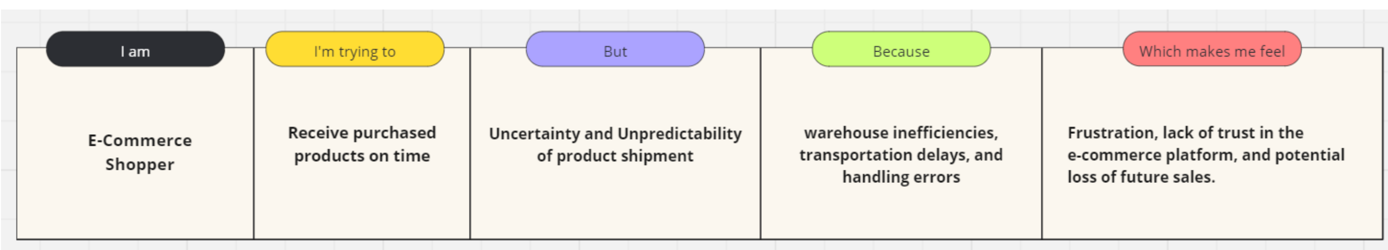
By achieving these objectives, the project will contribute to a more reliable and efficient e-commerce experience, benefiting both the customers and the platform.

2. Project Initialization and Planning Phase

2.1 Define Problem Statements:

As an e-commerce shopper, my primary expectation is to receive my purchased products on time, as promised by the platform. This assurance is critical for planning, especially when the items are gifts, essentials, or needed for specific occasions. However, this expectation is frequently unmet due to various issues such as warehouse inefficiencies, transportation problems, and handling errors. These delays and uncertainties disrupt the seamless shopping experience I seek and expect from an e-commerce platform.

As a result of these persistent delays and uncertainties, I experience significant frustration and decreased trust in the e-commerce platform. This erosion of trust not only affects my current shopping experience but also makes me hesitant to rely on the platform for future purchases. Concerns about potential impacts on my shopping decisions include the fear of late deliveries for critical items, leading me to consider alternative shopping options or platforms. This uncertainty and dissatisfaction could ultimately drive me away from the platform, affecting its customer retention and loyalty rates.



Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS	E-Commerce Shopper	Receive purchased products on time	Uncertainty and Unpredictability of product shipment	warehouse inefficiencies, transportation delays, and handling errors	Frustration, lack of trust in the e-commerce platform, and potential loss of future sales.

2.2 Project Proposal (Proposed Solution)

This project proposal presents a comprehensive solution to tackle a specific problem. The proposed solution elaborates on the methodology, key features, and necessary resources, encompassing hardware, software, and personnel requirements.

Project Overview	
Objective	The primary goal of this project is to develop a predictive model capable of accurately forecasting whether an e-commerce shipment will be delivered on time.
Scope	This project focuses on developing and validating a machine learning model using historical shipment data to predict on-time deliveries. It encompasses data preprocessing, feature engineering, model selection, and evaluation. The project is confined to the provided dataset and does not include real-time data integration or external factors beyond the dataset's scope.
Problem Statement	
Description	The issue at hand is the unpredictability of e-commerce shipment delivery times, leading to customer dissatisfaction and a loss of trust in the platform. Despite promises of timely delivery, various factors such as logistical inefficiencies, product characteristics, and shipment methods result in delays. This project aims to identify and analyze these factors to create a predictive model that accurately forecasts whether a shipment will be delivered on time, thereby improving the reliability of delivery estimates and enhancing overall customer satisfaction.
Impact	Addressing the problem of unpredictable e-commerce shipment delivery times will have significant positive implications. Accurate and reliable delivery predictions will enhance customer satisfaction by setting realistic expectations, thereby building trust in the e-commerce platform. This can lead to increased customer loyalty, positive reviews, and repeat business. Additionally, better predictability can optimize logistics operations, reduce costs associated with delays, and improve overall efficiency.
Proposed Solution	
Approach	The methodology for this project involves several key steps. Initially, data exploration and preprocessing will be conducted to clean the dataset and handle any missing values or inconsistencies. Feature engineering will be performed to create meaningful variables that could enhance the predictive power of the model. Various machine learning techniques, such as logistic

	regression, decision trees, random forests, and gradient boosting, will be employed to develop the predictive model. The dataset will be split into training and testing sets to evaluate model performance using accuracy, precision, recall, and F1-score metrics. Cross-validation techniques will be used to prevent overfitting. Finally, the best-performing model will be selected for integration into the e-commerce platform, providing real-time delivery predictions to improve customer satisfaction.
Key Features	The proposed solution is unique in its comprehensive approach to addressing the unpredictability of e-commerce shipment delivery times. By leveraging a combination of advanced machine learning techniques and thorough data analysis, the model will not only predict delivery times but also identify the key factors contributing to delays. This dual focus allows for more precise and actionable insights, enabling the e-commerce platform to address root causes and optimize logistics operations.

Resource Requirements

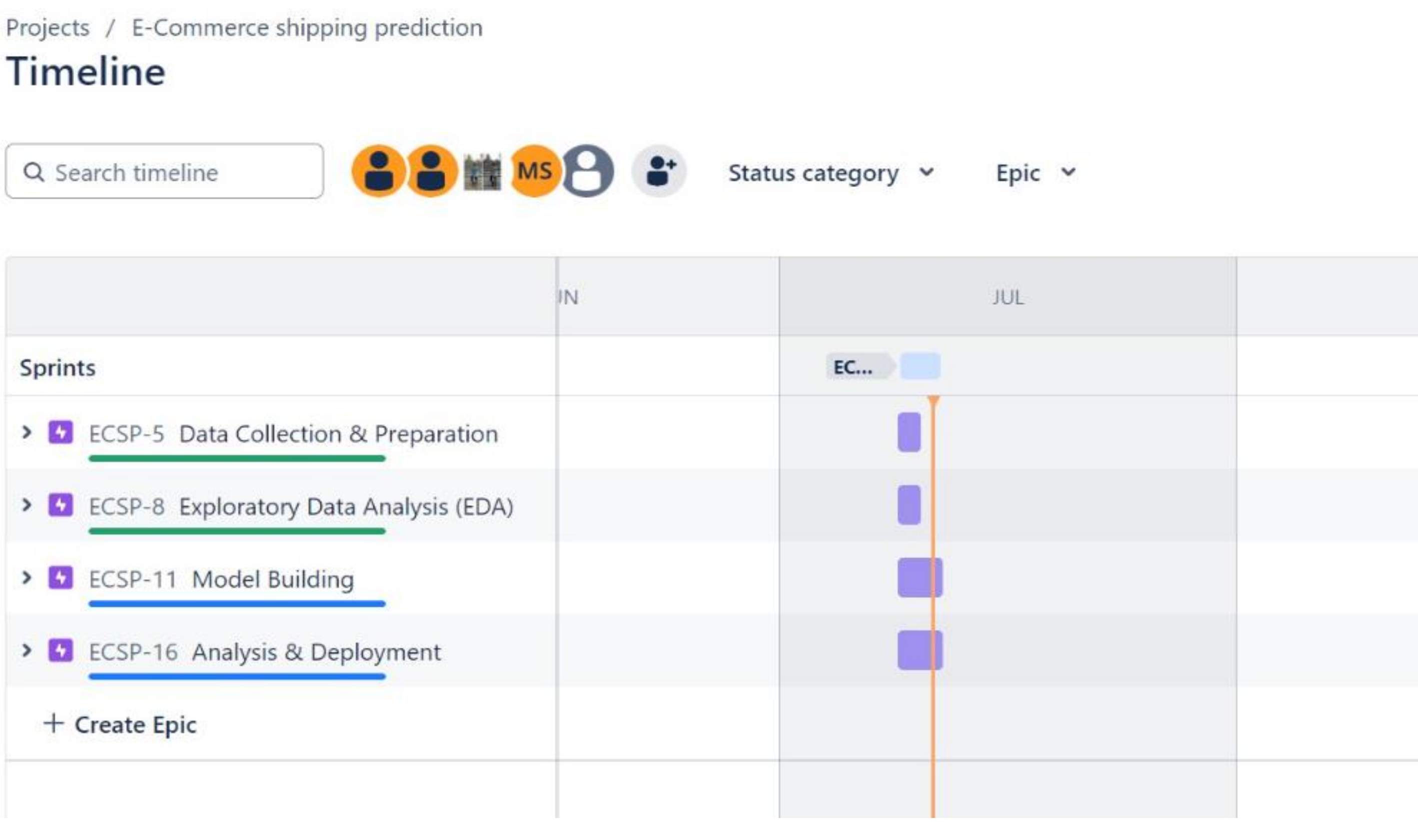
Resource Type	**Description**	**Specification/Allocation**
Hardware		
Computing Resources	CPU/GPU specifications, number of cores	Intel Core i7-12700H, NVIDIA GeForce RTX 3060 (6GB GDDR6)
Memory	RAM specifications	16 GB DDR5 RAM
Storage	Disk space for data, models, and logs	1 TB PCIe NVMe SSD
Software		
Frameworks	Python frameworks	Flask
Libraries	Additional libraries	scikit-learn, pandas, numpy, seaborn, pickle, sklearn, matplotlib, xgboost
Development Environment	IDE, version control	Jupyter Notebook, Git
Data		
Data	Source, size, format	Kaggle dataset (10999, 12), 430KB

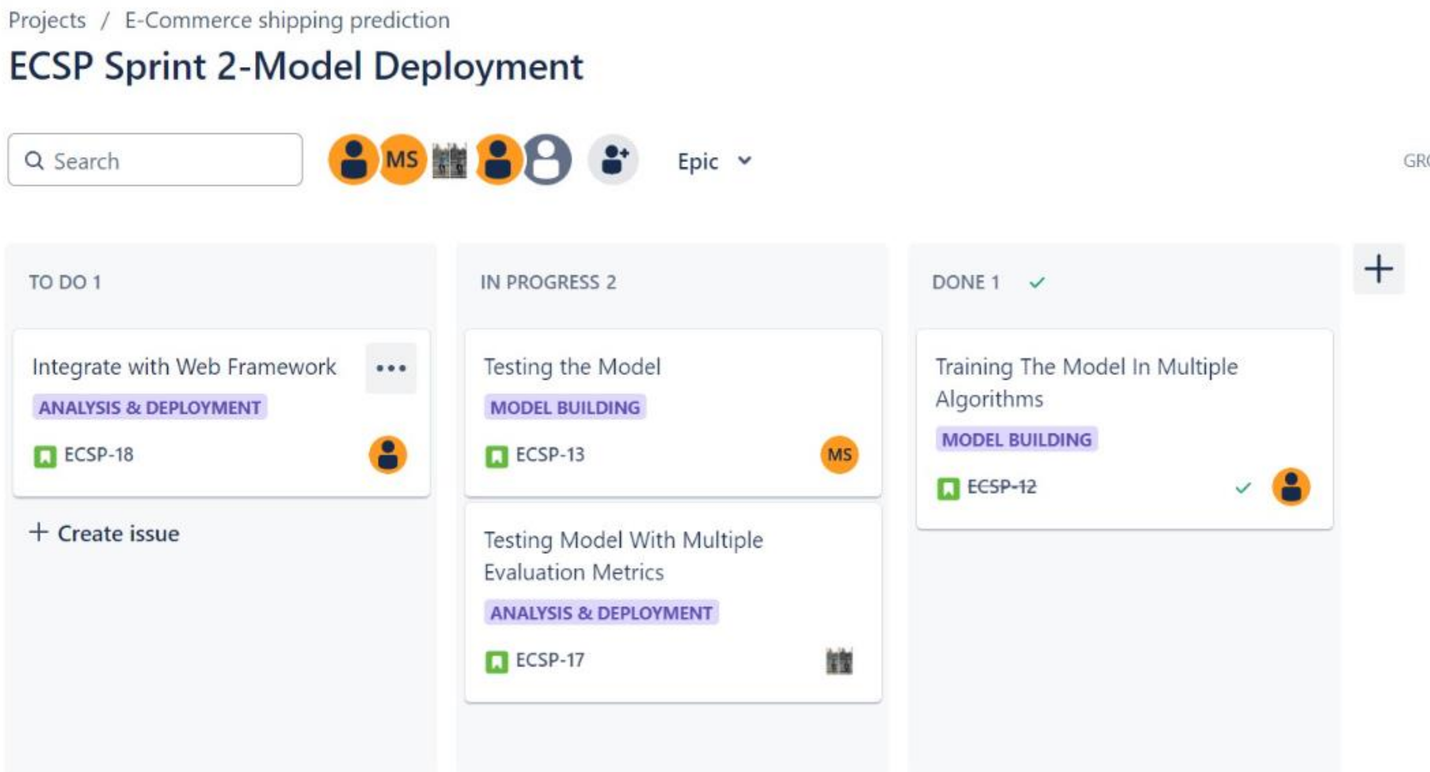
2.3 Initial Project Planning:

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Point	Priority	Team Members	Sprint Start Date	Sprint End Date
Sprint-1	Data Collection & Preparation	ECSP-6	Collecting Dataset	6	Medium	Madhu Varshini S	4/7/2024	9/7/2024
Sprint-1	Data Collection & Preparation	ECSP-7	Data Preparation	6	Medium	Ramakrishnan S	4/7/2024	9/7/2024
Sprint-1	Exploratory Data Analysis (EDA)	ECSP-9	Descriptive statistics	5	Low	Amithav Mrithyunjay	4/7/2024	9/7/2024
Sprint-1	Exploratory Data Analysis (EDA)	ECSP-10	Visual Analysis	7	Medium	Gowtham S	4/7/2024	9/7/2024
Sprint-2	Model Building	ECSP-12	Training The Model In Multiple Algorithms	10	High	Ramakrishnan S	9/7/2024	11/7/2024
Sprint-2	Model Building	ECSP-13	Testing the Model	10	High	Madhu Varshini S	9/7/2024	11/7/2024

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Point	Priority	Team Members	Sprint Start Date	Sprint End Date
Sprint-2	Analysis & Deployment	ECSP-17	Testing Model With Multiple Evaluation Metrics	10	High	Amithav Mrithyunjay	9/7/2024	11/7/2024
Sprint-2	Analysis & Deployment	ECSP-18	Integrate with Web Framework	10	High	Gowtham S	9/7/2024	11/7/2024

Screenshots:





3. Data Collection and Preprocessing Phase

3.1 Data Collection Plan & Raw Data Sources Identification

Optimizing our e-commerce shipment prediction project with a detailed Data Collection Plan and an in-depth Raw Data Sources report. This approach ensures precise data gathering and integrity, forming a robust base for accurate analysis and reliable predictive modeling. By focusing on well-curated historical shipment data, this project aims to enhance delivery reliability and customer satisfaction through informed decision-making and optimized logistics operations.

Data Collection Plan

Section	Description
Project Overview	This machine learning project aims to develop a predictive model that accurately forecasts whether e-commerce shipments will be delivered on time. By analysing historical shipment data and identifying key factors influencing delivery times, the project seeks to enhance delivery reliability and improve customer satisfaction. The ultimate objective is to provide a robust predictive tool that can be integrated into the e-commerce platform, offering customers transparent and reliable delivery estimates while optimizing logistics operations.
Data Collection Plan	The data is collected from the Skill Wallet platform, specifically from a dataset available on Kaggle.

Raw Data Sources Identified	The dataset used for model building contains 10,999 observations across 12 variables.

Raw Data Sources

Source Name	Description	Location/URL	Format	Size	Access Permissions
Dataset 1 - Skill Wallet platform	The dataset contains 10,999 observations across 12 variables, offering a comprehensive set of historical shipment data to analyze and identify key factors affecting delivery times.	https://www.kaggle.com/datasets/prachi13/customer-analytics?select=Train.csv	CSV	430 KB	Public

3.2 Data Quality Report Template

The Data Quality Report Template will provide a detailed summary of the data quality issues identified in the dataset, along with their severity levels and proposed resolution plans. This report aims to systematically identify and address data discrepancies, ensuring the dataset is clean and reliable for building a predictive model.

Data Source	Data Quality Issue	Severity	Resolution Plan
https://www.kaggle.com/datasets/prachi13/customer-analytics?select=Train.csv	Outliers in numerical features. Imbalanced target variable.	Moderate	Apply SMOTE to handle imbalanced data Use IQR methods to detect and cap outliers

3.3 Data Exploration and Preprocessing

Section	Description
Data Overview	

```
data.shape    data.ndim
```

```
(10999, 12)   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)
```

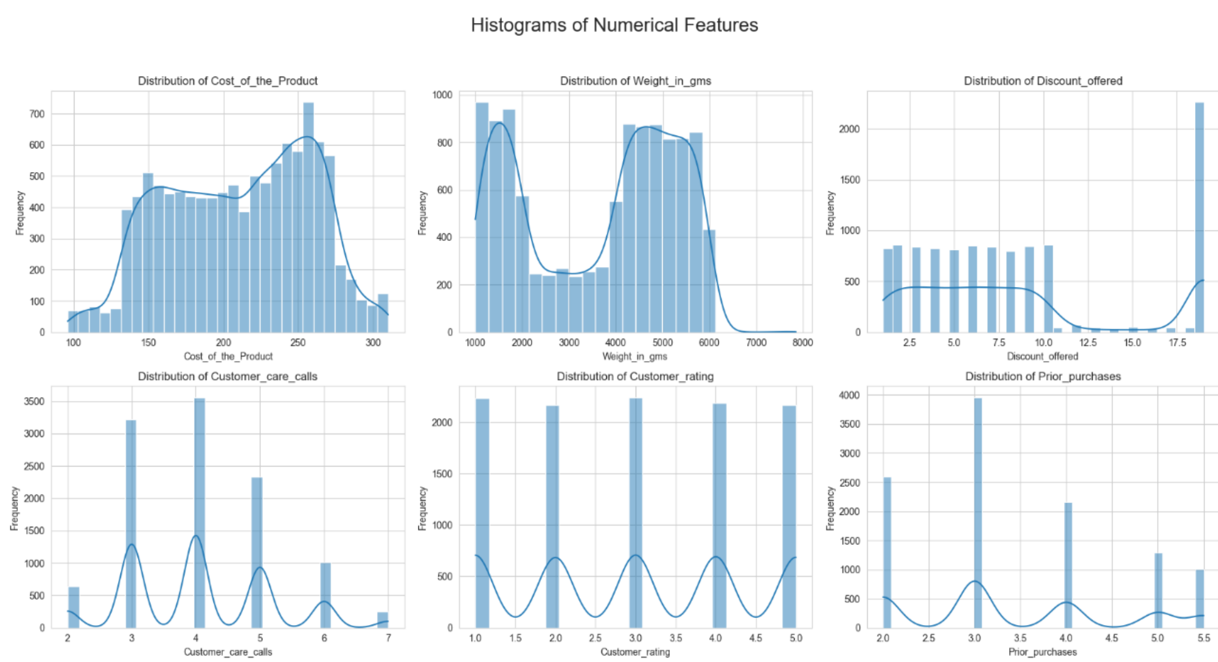
```
Warehouse_block           5
Mode_of_Shipment          3
Customer_care_calls       6
Customer_rating           5
Cost_of_the_Product     215
Prior_purchases           5
Product_importance        3
Gender                    2
Discount_offered         19
Weight_in_gms          4034
Reached.on.Time_Y.N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

```python
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()
```

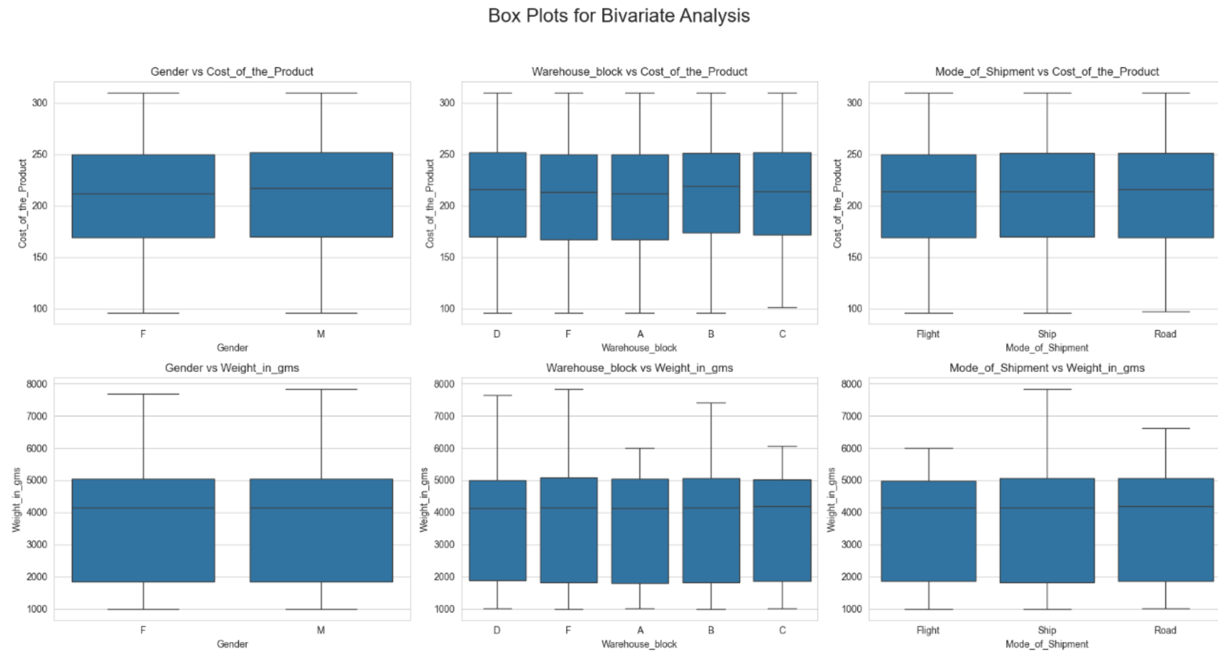


Bivariate Analysis

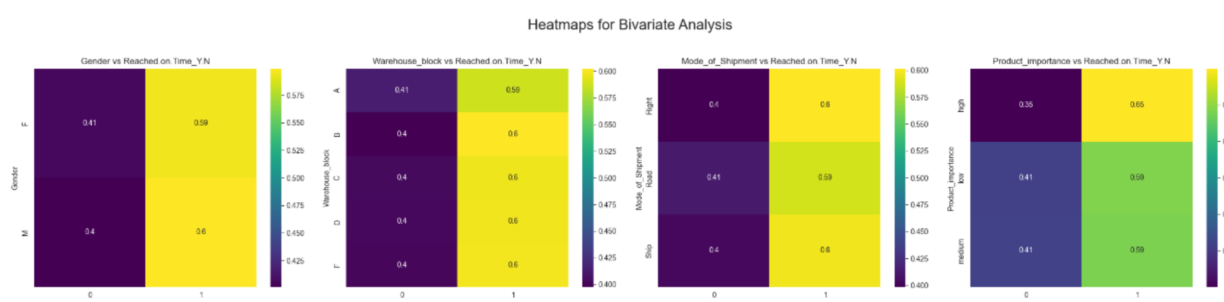
```
df.corr()
```

	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()
```



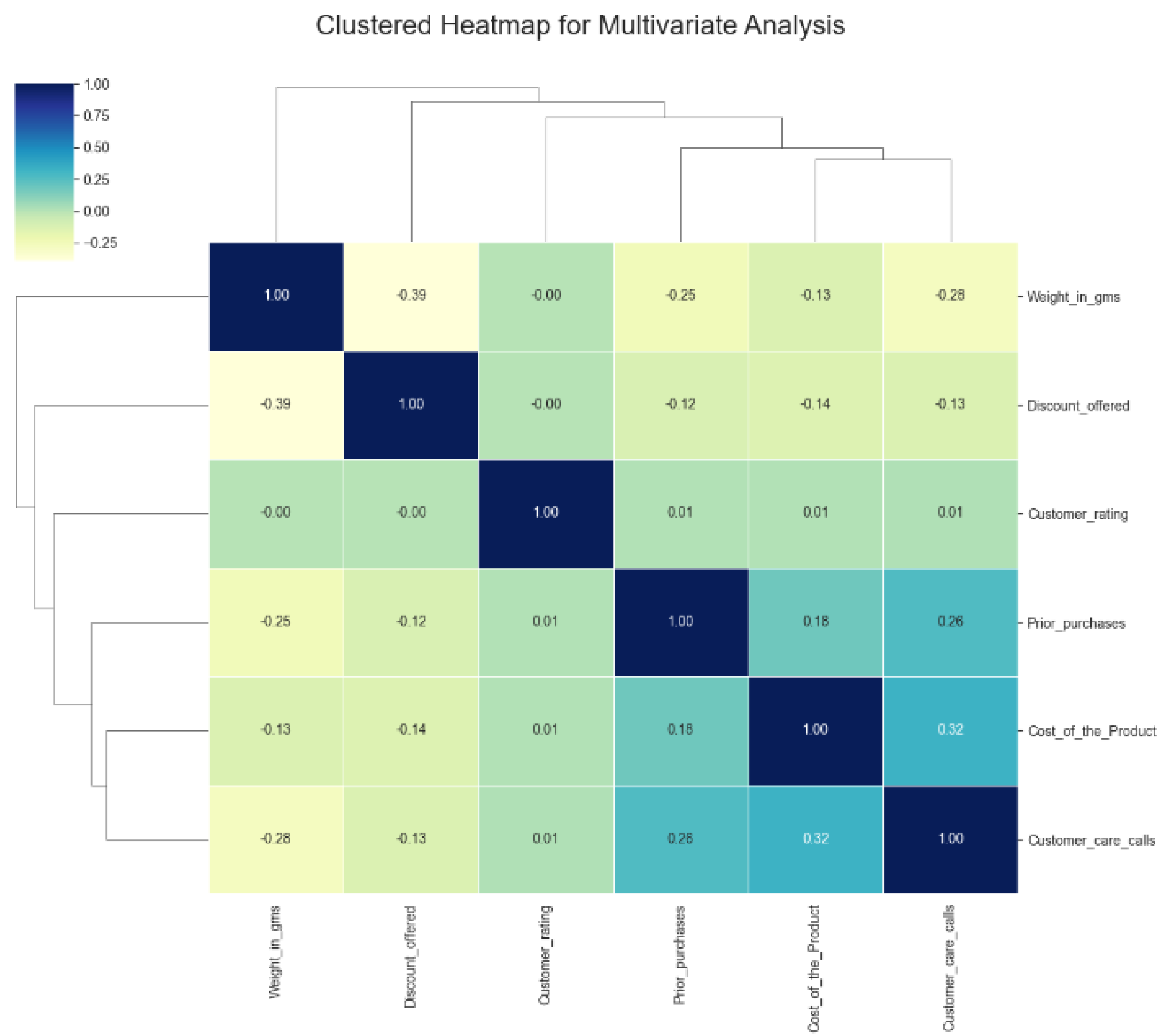

```
categorical_features = ['Gender', 'Warehouse_block', 'Mode_of_Shipment', 'Product_importance']
target_feature = 'Reached.on.Time_Y.N'
heatmaps_data = [
    pd.crosstab(df[cat_feature], df[target_feature], normalize='index')
    for cat_feature in categorical_features
]
fig, axes = plt.subplots(nrows=1, 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

```
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()
```

Outliers and Anomalies	<pre>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)</pre>

Data Preprocessing Code Screenshots	
Loading Data	<pre>data=pd.read_csv('Train.csv')</pre>

Handling Missing Data	<pre>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()</pre>

Data Transformation	<pre>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</pre>

	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 ]])
```

<table>
<tr><td>Feature Engineering</td><td>

```
df=data.drop(['ID'], axis=1 )
df.head()
```

	Warehouse_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	Cost_of_the_Product	Prior_purchases	Product_importance	Gender
0	D	Flight	4	2	177	3.0	low	F
1	F	Flight	4	5	216	2.0	low	M
2	A	Flight	2	2	183	4.0	low	M
3	B	Flight	3	3	176	4.0	medium	M
4	C	Flight	2	2	184	3.0	medium	F

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

</td></tr>
<tr><td>Save Processed Data</td><td>

```
data.to_csv('Train_cleaned.csv', index=False)
```

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

4. Model Development Phase

4.1 Feature Selection Report

Feature	Description	Selected (Yes/No)	Reasoning
ID	ID Number of Customers.	No	It is a unique identifier that does not contain any relevant information for predicting the delivery time. Including it would not contribute to the predictive power of the model and could introduce unnecessary noise.
Warehouse_block	The Company has a Warehouse divided into blocks such as A, B, C, D, and E.	Yes	The location within the warehouse can affect the speed of processing and dispatching shipments, impacting delivery times.
Mode_of_Shipment	The Company Ships the products in multiple ways such as Ship, Flight etc.,	Yes	Different shipment modes (Flight, Ship, Road) have varying transit times and reliability, making this a critical factor in predicting on-time delivery.

Customer_care_calls	The number of calls made for enquiry of the shipment.	Yes	The number of calls can indicate potential issues or delays, affecting the likelihood of on-time delivery.
Customer_rating	The company has rated from every customer.	Yes	Customer ratings may reflect past experiences and satisfaction levels, indirectly influencing the efficiency of handling and shipping processes.
Cost_of_the_Product	Cost of the Product in US Dollars.	Yes	Higher-cost products might receive priority handling and faster shipping options, impacting delivery times.
Prior_purchases	The Number of Prior Purchases.	Yes	A customer's history of purchases can influence the reliability and efficiency of the shipping process, as repeat customers may be prioritized.
Product_importance	The products are categorized into various parameters such as low, medium, and high.	Yes	The importance of the product (low, medium, high) can dictate the urgency and shipping method, affecting delivery speed.
Gender	Male and Female.	Yes	Gender of the customer might be correlated with specific delivery preferences, which could impact delivery times.
Discount_offered	Discount offered on that specific product.	Yes	Products with higher discounts might be shipped using slower methods to offset costs, affecting delivery times.
Weight_in_gms	Weight of the product in grams.	Yes	This can influence shipping method choices and transit times, impacting delivery accuracy.
Reached on Time Y.N	It is the target variable, where 1 Indicates that the product has NOT reached on time and 0 indicates it has reached on time.	Yes	The weight of the product can influence shipping method choices and transit times, impacting delivery accuracy.

4.2 Model Selection Report

Model	Description	Hyperparameters	Performance Metric (e.g., Accuracy, F1 Score)
Random Forest Classifier	This ensemble method combines multiple decision trees to improve accuracy and control over-fitting by averaging their predictions.	'classifier_max_depth': 10, 'classifier__min_samples_leaf': 1, 'classifier__min_samples_split': 2, 'classifier__n_estimators': 200	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
K-Nearest Neighbors Classifier	A simple, non-parametric method that classifies a data point based on the majority label among its k-nearest neighbors in the feature space.	'classifier_metric': 'euclidean', 'classifier__n_neighbors': 9, 'classifier__p': 1, 'classifier__weights': 'uniform'	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
Logistic Regression	A linear model used for binary classification that estimates the probability of a binary response based on one or more predictor variables using a logistic function.	'classifier__C': 0.01, 'classifier__max_iter': 100, 'classifier__penalty': 'l2', 'classifier__solver': 'liblinear'	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
XGB Classifier	An optimized gradient boosting library designed for speed and performance, which builds an ensemble of decision trees by sequentially minimizing a loss function.	'classifier__learning_rate': 0.01, 'classifier__max_depth': 5, 'classifier__n_estimators': 200, 'classifier__subsample': 0.7	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
Support Vector Classifier	A classifier that constructs a hyperplane in a high-dimensional space to separate different classes with maximum margin, often using kernel functions for non-linear separation.	'classifier__C': 10, 'classifier__gamma': 'auto', 'classifier__kernel': 'poly'	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

Decision Tree Classifier	A model that splits the data into subsets based on feature values, creating a tree structure where each leaf represents a class label and each node represents a decision rule.	'classifier__criterion': 'gini', 'classifier__max_depth': 10, 'classifier__min_samples_leaf': 4, 'classifier__min_samples_split': 2	Classification Report with Hyperparameter Tuning and SMOTE: precision recall f1-score support 0 0.57 0.89 0.70 1379 1 0.87 0.53 0.66 1921 accuracy 0.68 3300 macro avg 0.72 0.71 0.68 3300 weighted avg 0.74 0.68 0.67 3300
Naive Bayes Classifier	A probabilistic classifier based on Bayes' theorem, which assumes independence among features and calculates the probability of each class given the input features.	'classifier__var_smoothing': 1e-09	Classification Report with Hyperparameter Tuning and SMOTE: precision recall f1-score support 0 0.55 0.89 0.68 1379 1 0.86 0.48 0.62 1921 accuracy 0.65 3300 macro avg 0.70 0.68 0.65 3300 weighted avg 0.73 0.65 0.64 3300
AdaBoost Classifier	An ensemble method that combines multiple weak classifiers, typically decision trees, by weighting them according to their accuracy and iteratively improving the model.	'classifier__learning_rate': 1, 'classifier__n_estimators': 200	Classification Report with Hyperparameter Tuning and SMOTE: precision recall f1-score support 0 0.58 0.89 0.71 1379 1 0.87 0.54 0.67 1921 accuracy 0.69 3300 macro avg 0.73 0.72 0.69 3300 weighted avg 0.75 0.69 0.68 3300
Gradient Boost Classifier	An ensemble technique that builds models sequentially, each new model correcting the errors of the previous ones, and combines them to make a final prediction.	'classifier__learning_rate': 0.01, 'classifier__max_depth': 5, 'classifier__n_estimators': 200, 'classifier__subsample': 0.9	Classification Report with Hyperparameter Tuning and SMOTE: precision recall f1-score support 0 0.58 0.95 0.72 1379 1 0.93 0.51 0.66 1921 accuracy 0.69 3300 macro avg 0.76 0.73 0.69 3300 weighted avg 0.79 0.69 0.68 3300

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:

```
from sklearn.model_selection import train_test_split, GridSearchCV
from imblearn.over_sampling import SMOTE
from imblearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.preprocessing import LabelEncoder, StandardScaler
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=0)
```

Random Forest

```
from sklearn.ensemble import RandomForestClassifier
```

```
rf = RandomForestClassifier(n_estimators=200, criterion='entropy', random_state=56,max_depth=5)
rf.fit(x_train, y_train)
pred = rf.predict(x_test)
accuracy = accuracy_score(y_test, pred)

print(f"Accuracy without Hyperparameter Tuning and SMOTE: {accuracy:.6f}")
print("Classification Report without Hyperparameter Tuning and SMOTE:\n", classification_report(y_test, pred))
print("Confusion Matrix without Hyperparameter Tuning and SMOTE:\n", confusion_matrix(y_test, pred))
```

K Nearest Neighbors (KNN)

```
from sklearn.neighbors import KNeighborsClassifier
```

```
knn = KNeighborsClassifier(n_neighbors=10, weights='uniform', metric='minkowski', p=3)
knn.fit(x_train, y_train)
knn_pred = knn.predict(x_test)

print("Accuracy without Hyperparameter Tuning and SMOTE:", accuracy_score(y_test, knn_pred))
print("Classification Report without Hyperparameter Tuning and SMOTE:\n", classification_report(y_test, knn_pred))
print("Confusion Matrix without Hyperparameter Tuning and SMOTE:\n", confusion_matrix(y_test, knn_pred))
```

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
```

```
log_reg = LogisticRegression(solver='sag', penalty='l2', random_state=42)
log_reg.fit(x_train, y_train)
log_reg_pred = log_reg.predict(x_test)

print("Accuracy without Hyperparameter Tuning and SMOTE:", accuracy_score(y_test, log_reg_pred))
print("Classification Report without Hyperparameter Tuning and SMOTE:\n", classification_report(y_test, log_reg_pred))
print("Confusion Matrix without Hyperparameter Tuning and SMOTE:\n", confusion_matrix(y_test, log_reg_pred))
```

XGBoost

```
from xgboost import XGBClassifier
```

```
xgb = XGBClassifier(eval_metric='mlogloss', random_state=42)
xgb.fit(x_train, y_train)
xgb_pred = xgb.predict(x_test)
accuracy = accuracy_score(y_test, xgb_pred)

print(f"Accuracy without Hyperparameter Tuning and SMOTE: {accuracy:.6f}")
print("Classification Report without Hyperparameter Tuning and SMOTE:\n", classification_report(y_test, xgb_pred))
print("Confusion Matrix without Hyperparameter Tuning and SMOTE:\n", confusion_matrix(y_test, xgb_pred))
```

SVC

```
from sklearn.svm import SVC
```

```
svc = SVC(kernel='rbf', random_state=42)
svc.fit(x_train, y_train)
svc_pred = svc.predict(x_test)
accuracy = accuracy_score(y_test, svc_pred)

print(f"Accuracy without Hyperparameter Tuning and SMOTE: {accuracy:.6f}")
print("Classification Report without Hyperparameter Tuning and SMOTE:\n", classification_report(y_test, svc_pred))
print("Confusion Matrix without Hyperparameter Tuning and SMOTE:\n", confusion_matrix(y_test, svc_pred))
```

Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
```

```
dt = DecisionTreeClassifier(random_state=42)
dt.fit(x_train, y_train)
dt_pred = dt.predict(x_test)
accuracy = accuracy_score(y_test, dt_pred)

print(f"Accuracy without Hyperparameter Tuning and SMOTE: {accuracy:.6f}")
print("Classification Report without Hyperparameter Tuning and SMOTE:\n", classification_report(y_test, dt_pred))
print("Confusion Matrix without Hyperparameter Tuning and SMOTE:\n", confusion_matrix(y_test, dt_pred))
```

Naive Bayes

```
from sklearn.naive_bayes import GaussianNB
```

```
nb = GaussianNB()
nb.fit(x_train, y_train)
nb_pred = nb.predict(x_test)
accuracy = accuracy_score(y_test, nb_pred)

print(f"Accuracy without Hyperparameter Tuning and SMOTE: {accuracy:.6f}")
print("Classification Report without Hyperparameter Tuning and SMOTE:\n", classification_report(y_test, nb_pred))
print("Confusion Matrix without Hyperparameter Tuning and SMOTE:\n", confusion_matrix(y_test, nb_pred))
```

AdaBoost

```
from sklearn.ensemble import AdaBoostClassifier
```

```
ada = AdaBoostClassifier(random_state=42)
ada.fit(x_train, y_train)
ada_pred = ada.predict(x_test)
accuracy = accuracy_score(y_test, ada_pred)

print(f"Accuracy without Hyperparameter Tuning and SMOTE: {accuracy:.6f}")
print("Classification Report without Hyperparameter Tuning and SMOTE:\n", classification_report(y_test, ada_pred))
print("Confusion Matrix without Hyperparameter Tuning and SMOTE:\n", confusion_matrix(y_test, ada_pred))
```

Gradient Boosting

```
from sklearn.ensemble import GradientBoostingClassifier
```

```
gb = GradientBoostingClassifier(random_state=42)
gb.fit(x_train, y_train)
gb_pred = gb.predict(x_test)
accuracy = accuracy_score(y_test, gb_pred)

print(f"Accuracy without Hyperparameter Tuning and SMOTE: {accuracy:.6f}")
print("Classification Report without Hyperparameter Tuning and SMOTE:\n", classification_report(y_test, gb_pred))
print("Confusion Matrix without Hyperparameter Tuning and SMOTE:\n", confusion_matrix(y_test, gb_pred))
```

Model Validation and Evaluation Report

Model	Classification Report	Accuracy	Confusion Matrix
Random Forest Classifier	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	0.690000	Confusion Matrix without Hyperparameter Tuning and SMOTE: [[1250 129] [894 1027]]
K-Nearest Neighbors Classifier	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	0.666969	Confusion Matrix without Hyperparameter Tuning and SMOTE: [[1035 344] [755 1166]]
Logistic Regression	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	0.628484	Confusion Matrix without Hyperparameter Tuning and SMOTE: [[725 654] [572 1349]]
XGB Classifier	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	0.644545	Confusion Matrix without Hyperparameter Tuning and SMOTE: [[853 526] [647 1274]]
Support Vector Classifier	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	0.668788	Confusion Matrix without Hyperparameter Tuning and SMOTE: [[1129 250] [843 1078]]
Decision Tree Classifier	Classification Report without Hyperparameter Tuning and SMOTE: precision recall f1-score support 0 0.58 0.57 0.58 1379 1 0.70 0.70 0.70 1921 accuracy 0.65 3300 macro avg 0.64 0.64 0.64 3300 weighted avg 0.65 0.65 0.65 3300	0.647576	Confusion Matrix without Hyperparameter Tuning and SMOTE: [[787 592] [571 1350]]
Naive Bayes Classifier	Classification Report without Hyperparameter Tuning and SMOTE: precision recall f1-score support 0 0.55 0.78 0.65 1379 1 0.78 0.55 0.64 1921 accuracy 0.65 3300 macro avg 0.67 0.67 0.65 3300 weighted avg 0.68 0.65 0.65 3300	0.646364	Confusion Matrix without Hyperparameter Tuning and SMOTE: [[1079 300] [867 1054]]
Ada Boost Classifier	Classification Report without Hyperparameter Tuning and SMOTE: precision recall f1-score support 0 0.59 0.75 0.66 1379 1 0.78 0.63 0.69 1921 accuracy 0.68 3300 macro avg 0.69 0.69 0.68 3300 weighted avg 0.70 0.68 0.68 3300	0.679394	Confusion Matrix without Hyperparameter Tuning and SMOTE: [[1039 340] [718 1203]]
Gradient Boost Classifier	Classification Report without Hyperparameter Tuning and SMOTE: precision recall f1-score support 0 0.58 0.87 0.70 1379 1 0.85 0.55 0.67 1921 accuracy 0.68 3300 macro avg 0.72 0.71 0.68 3300 weighted avg 0.74 0.68 0.68 3300	0.683939	Confusion Matrix without Hyperparameter Tuning and SMOTE: [[1194 185] [858 1063]]

5. Model Optimization and Tuning

5.1 Hyperparameter Tuning

Model	Tuned Hyperparameters	Optimal Values

Random Forest Classifier	smote = SMOTE(random_state=42)
	classifier = RandomForestClassifier(n_estimators=200, criterion='entropy', random_state=56,max_depth=5)
	
	pipeline = Pipeline([
	    ('smote', smote),
	    ('classifier', classifier)
	])
	
	param_grid = {
	    'classifier__n_estimators': [100, 200, 300],
	    'classifier__max_depth': [None, 10, 20, 30],
	    'classifier__min_samples_split': [2, 5, 10],
	    'classifier__min_samples_leaf': [1, 2, 4]
	}
		Best Parameters: {'classifier__max_depth': 10, 'classifier__min_samples_leaf': 1, 'classifier__min_samples_sp
		ifier__n_estimators': 200}
		
		Best Cross-Validation Score: 0.6839847936339164
		Accuracy with Hyperparameter Tuning and SMOTE: 0.6893939393939394

K-Nearest Neighbors Classifier	smote = SMOTE(random_state=42)
	classifier = KNeighborsClassifier(n_neighbors=12, weights='uniform', metric='euclidean', p=5)
	
	pipeline = Pipeline([
	    ('smote', smote),
	    ('classifier', classifier)
	])
	
	param_grid = {
	    'classifier__n_neighbors': [3, 5, 7, 9],
	    'classifier__weights': ['uniform', 'distance'],
	    'classifier__metric': ['euclidean', 'manhattan', 'minkowski'],
	    'classifier__p': [1, 2]
	}
		Best Parameters: {'classifier__metric': 'euclidean', 'classifier__n_neighbors': 9, 'classifier__p': 1, 'clas
		s': 'uniform'}
		
		Best Cross-Validation Score: 0.6530739869775356
		Accuracy with Hyperparameter Tuning and SMOTE: 0.6633333333333333

Logistic Regression	smote = SMOTE(random_state=42)
	classifier = LogisticRegression(random_state=42)
	
	pipeline = Pipeline([
	    ('smote', smote),
	    ('classifier', classifier)
	])
	
	param_grid = {
	    'classifier__solver': ['liblinear', 'saga'],
	    'classifier__penalty': ['l1', 'l2'],
	    'classifier__C': [0.01, 0.1, 1, 10, 100],
	    'classifier__max_iter': [100, 200, 300]
	}
		Best Parameters: {'classifier__C': 0.01, 'classifier__max_iter': 100, 'classifier__penalty': 'l2', 'classifie
		blinear'}
		
		Best Cross-Validation Score: 0.6511227264283181
		Accuracy with Hyperparameter Tuning and SMOTE: 0.64363636

XGB Classifier	smote = SMOTE(random_state=42)
	classifier = XGBClassifier(eval_metric='mlogloss', random_state=42)
	
	pipeline = Pipeline([
	    ('smote', smote),
	    ('classifier', classifier)
	])
	
	param_grid = {
	    'classifier__n_estimators': [100, 200, 300],
	    'classifier__max_depth': [3, 5, 7],
	    'classifier__learning_rate': [0.01, 0.1, 0.2],
	    'classifier__subsample': [0.7, 0.8, 0.9]
	}
		Best Parameters: {'classifier__learning_rate': 0.01, 'classifier__max_depth': 5, 'classifier__n_estimators':
		r__subsample': 0.7}
		Best Cross-Validation Score: 0.6822955536990625
		Accuracy with Hyperparameter Tuning and SMOTE: 0.690606

Support Vector Classifier	smote = SMOTE(random_state=42)
	classifier = SVC(random_state=42)
	
	pipeline = Pipeline([
	    ('smote', smote),
	    ('classifier', classifier)
	])
	
	param_grid = {
	    'classifier__kernel': ['linear', 'poly', 'rbf', 'sigmoid'],
	    'classifier__C': [0.1, 1, 10, 100],
	    'classifier__gamma': ['scale', 'auto']
	}
		Best Parameters: {'classifier__C': 10, 'classifier__gamma': 'auto', 'classifier__kernel
		
		Best Cross-Validation Score: 0.6680090799389043
		Accuracy with Hyperparameter Tuning and SMOTE: 0.670303

Decision Tree Classifier	smote = SMOTE(random_state=42)
	classifier = DecisionTreeClassifier(random_state=42)
	
	pipeline = Pipeline([
	    ('smote', smote),
	    ('classifier', classifier)
	])
	
	param_grid = {
	    'classifier__criterion': ['gini', 'entropy'],
	    'classifier__max_depth': [None, 10, 20, 30, 40, 50],
	    'classifier__min_samples_split': [2, 5, 10],
	    'classifier__min_samples_leaf': [1, 2, 4]
	}
		Best Parameters: {'classifier__criterion': 'gini', 'classifier__max_depth': 10, 'classifier__min_samples_leaf'
		er__min_samples_split': 2}
		
		Best Cross-Validation Score: 0.6646317814738868
		Accuracy with Hyperparameter Tuning and SMOTE: 0.677576

Naive Bayes Classifier	smote = SMOTE(random_state=42)
	classifier = GaussianNB()
	
	pipeline = Pipeline([
	    ('smote', smote),
	    ('classifier', classifier)
	])
	
	param_grid = {
	    'classifier__var_smoothing': [1e-9, 1e-8, 1e-7]
	}
		Best Parameters: {'classifier__var_smoothing': 1e-
		
		Best Cross-Validation Score: 0.6559285967608465
		Accuracy with Hyperparameter Tuning and SMOTE: 0.6

Ada Boost Classifier	smote = SMOTE(random_state=42)
	classifier = AdaBoostClassifier(random_state=42)
	
	pipeline = Pipeline([
	    ('smote', smote),
	    ('classifier', classifier)
	])
	
	param_grid = {
	    'classifier__n_estimators': [50, 100, 200],
	    'classifier__learning_rate': [0.01, 0.1, 1]
	}
		Best Parameters: {'classifier__learning_rate': 1, 'classifier__n_estimator
		
		Best Cross-Validation Score: 0.6747627486223978
		Accuracy with Hyperparameter Tuning and SMOTE: 0.688485

Gradient Boost Classifier	smote = SMOTE(random_state=42)
	classifiergb = GradientBoostingClassifier(random_state=42)
	
	pipeline = Pipeline([
	    ('smote', smote),
	    ('classifier', classifiergb)
	])
	
	param_grid = {
	    'classifier__n_estimators': [100, 200, 300],
	    'classifier__learning_rate': [0.01, 0.1, 0.2],
	    'classifier__max_depth': [3, 5, 7],
	    'classifier__subsample': [0.7, 0.8, 0.9]
	}
		Best Parameters: {'classifier__learning_rate': 0.01, 'classifier__max_depth': 5, 'classifier__n_estimators':
		r__subsample': 0.9}
		
		Best Cross-Validation Score: 0.6825556315029999
		Accuracy with Hyperparameter Tuning and SMOTE: 0.692121

5.2 Performance Metrics Comparison Report

Model	Baseline Metric	Optimized Metric
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: 0.6893939393939 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]]
K-Nearest Neighbors Classifier	Accuracy without Hyperparameter Tuning and SMOTE: 0.666969696969696 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: 0.663333333333333 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]]
Logistic Regression	Accuracy without Hyperparameter Tuning and SMOTE: 0.628484848484848 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]]
XGB Classifier	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]]

Support Vector Classifier	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]]
Decision Tree Classifier	Accuracy without Hyperparameter Tuning and SMOTE: 0.647576 Classification Report without Hyperparameter Tuning and SMOTE precision recall f1-score support 0 0.58 0.57 0.58 1379 1 0.70 0.70 0.70 1921 accuracy 0.65 3300 macro avg 0.64 0.64 0.64 3300 weighted avg 0.65 0.65 0.65 3300 Confusion Matrix without Hyperparameter Tuning and SMOTE: [[787 592] [571 1350]]	Accuracy with Hyperparameter Tuning and SMOTE: 0.677576 Classification Report with Hyperparameter Tuning and SMOT precision recall f1-score support 0 0.57 0.89 0.70 1379 1 0.87 0.53 0.66 1921 accuracy 0.68 3300 macro avg 0.72 0.71 0.68 3300 weighted avg 0.74 0.68 0.67 3300 Confusion Matrix with Hyperparameter Tuning and SMOTE: [[1224 155] [909 1012]]
Naive Bayes Classifier	Accuracy without Hyperparameter Tuning and SMOTE: 0.646364 Classification Report without Hyperparameter Tuning and SMOTE precision recall f1-score support 0 0.55 0.78 0.65 1379 1 0.78 0.55 0.64 1921 accuracy 0.65 3300 macro avg 0.67 0.67 0.65 3300 weighted avg 0.68 0.65 0.65 3300 Confusion Matrix without Hyperparameter Tuning and SMOTE: [[1079 300] [867 1054]]	Accuracy with Hyperparameter Tuning and SMOTE: 0.650909 Classification Report with Hyperparameter Tuning and SMOTE precision recall f1-score support 0 0.55 0.89 0.68 1379 1 0.86 0.48 0.62 1921 accuracy 0.65 3300 macro avg 0.70 0.68 0.65 3300 weighted avg 0.73 0.65 0.64 3300 Confusion Matrix with Hyperparameter Tuning and SMOTE: [[1226 153] [999 922]]
Ada Boost Classifier	Accuracy without Hyperparameter Tuning and SMOTE: 0.679394 Classification Report without Hyperparameter Tuning and SMOTE precision recall f1-score support 0 0.59 0.75 0.66 1379 1 0.78 0.63 0.69 1921 accuracy 0.68 3300 macro avg 0.69 0.69 0.68 3300 weighted avg 0.70 0.68 0.68 3300 Confusion Matrix without Hyperparameter Tuning and SMOTE: [[1039 340] [718 1203]]	Accuracy with Hyperparameter Tuning and SMOTE: 0.688485 Classification Report with Hyperparameter Tuning and SMOT precision recall f1-score support 0 0.58 0.89 0.71 1379 1 0.87 0.54 0.67 1921 accuracy 0.69 3300 macro avg 0.73 0.72 0.69 3300 weighted avg 0.75 0.69 0.68 3300 Confusion Matrix with Hyperparameter Tuning and SMOTE: [[1229 150] [878 1043]]

<table>
<tr>
<td>Gradient Boost Classifier</td>
<td><pre>
Accuracy without Hyperparameter Tuning and SMOTE: 0.683939
Classification Report without Hyperparameter Tuning and SMOT
              precision   recall f1-score   support

           0   0.58   0.87   0.70   1379
           1   0.85   0.55   0.67   1921

    accuracy   0.68   3300
   macro avg   0.72   0.71   0.68   3300
weighted avg   0.74   0.68   0.68   3300

Confusion Matrix without Hyperparameter Tuning and SMOTE:
 [[1194 185]
 [858 1063]]
</pre></td>
<td><pre>
Accuracy with Hyperparameter Tuning and SMOTE: 0.692121

Classification Report with Hyperparameter Tuning and SMOT
              precision   recall f1-score   support

           0   0.58   0.95   0.72   1379
           1   0.93   0.51   0.66   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:
 [[1310   69]
 [947 974]]
</pre></td>
</tr>
</table>

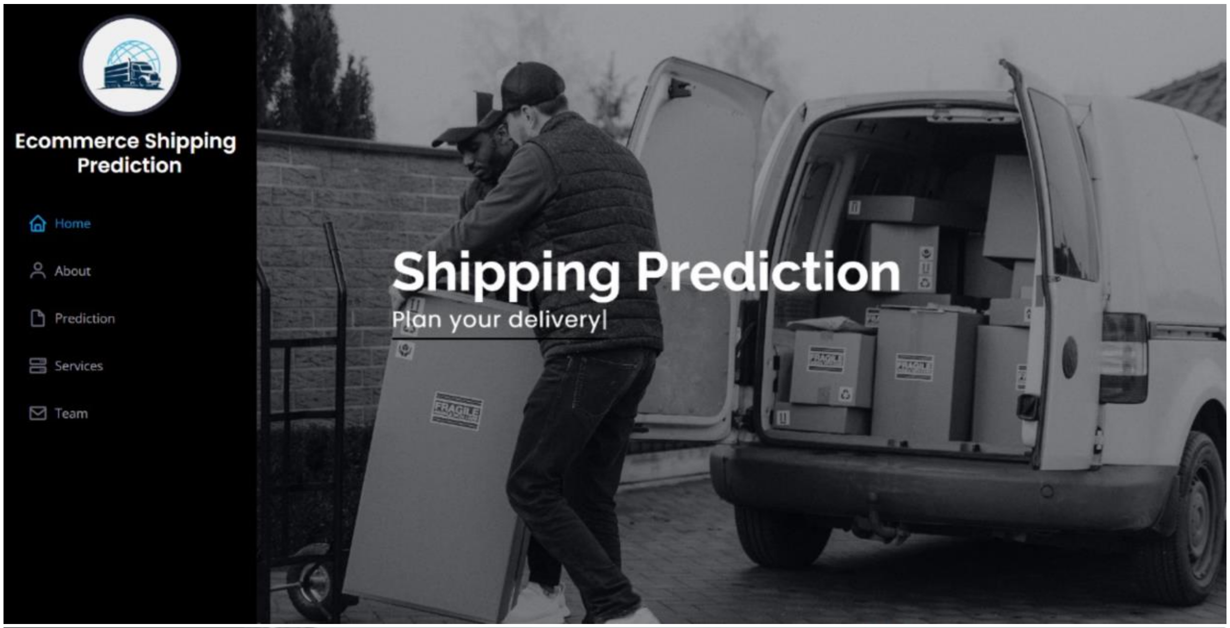
5.3 Final Model Selection

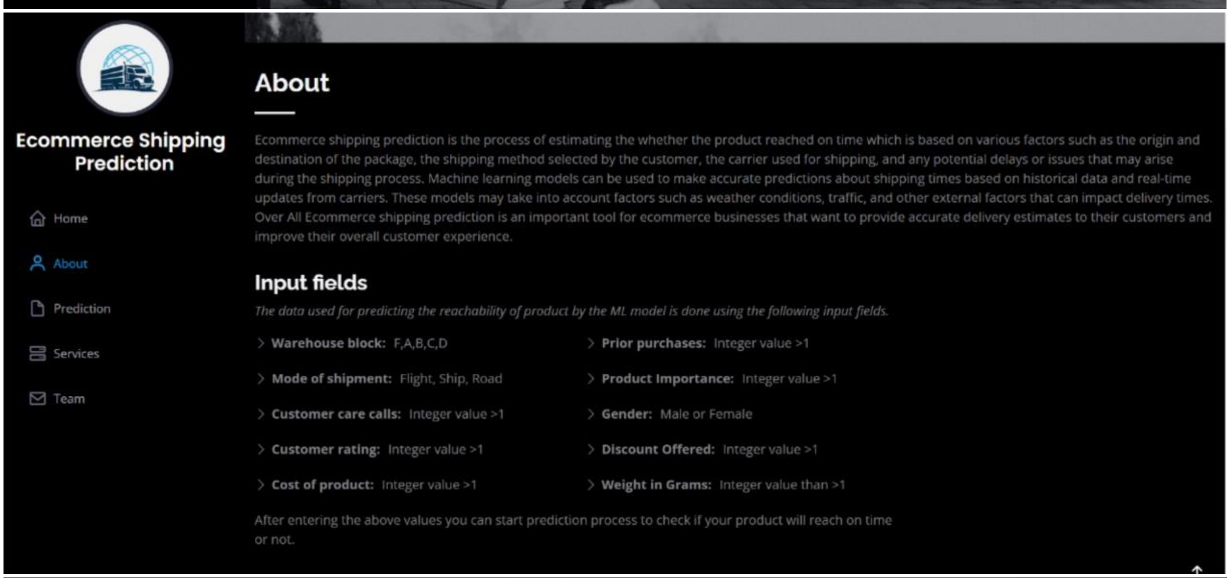
<table>
<tr><th>Final Model</th><th>Reasoning</th></tr>
<tr>
<td>Gradient Boost Classifier</td>
<td>Gradient Boosting was selected as the preferred algorithm for our classification task due to its superior accuracy in capturing complex patterns within the dataset. This algorithm excels in iteratively building an ensemble of weak learners, typically decision trees, where each subsequent tree corrects the errors of the previous ones. This boosting process focuses on improving the performance of areas where previous models underperformed, leading to a highly refined and accurate final model. Gradient Boosting's ability to effectively minimize errors and improve predictive power with each iteration makes it particularly suitable for datasets with intricate feature interactions and dependencies.

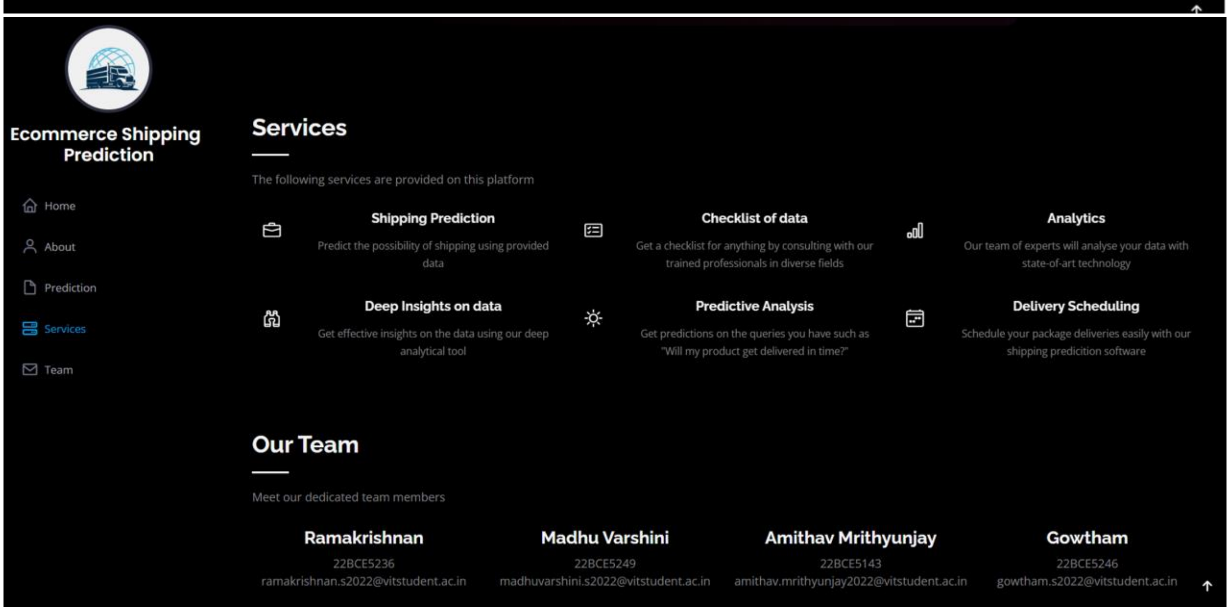
In contrast to other ensemble methods like Random Forest, XGBoost, and AdaBoost, Gradient Boosting's approach of sequentially improving weak learners allows it to model non-linear relationships more effectively. Random Forest, while robust due to its use of multiple trees and averaging, does not sequentially correct errors and may not capture complex patterns as efficiently. XGBoost, although similar to Gradient Boosting, uses different optimization techniques and regularization methods, which might not have been as well-suited to our specific data characteristics. AdaBoost, focusing on adjusting the weights of misclassified instances, may not achieve the same level of refinement in capturing intricate data patterns. Therefore, Gradient Boosting’s iterative and corrective approach provided a distinct advantage in modeling our dataset, leading to its selection for the highest accuracy performance.</td>
</tr>
</table>

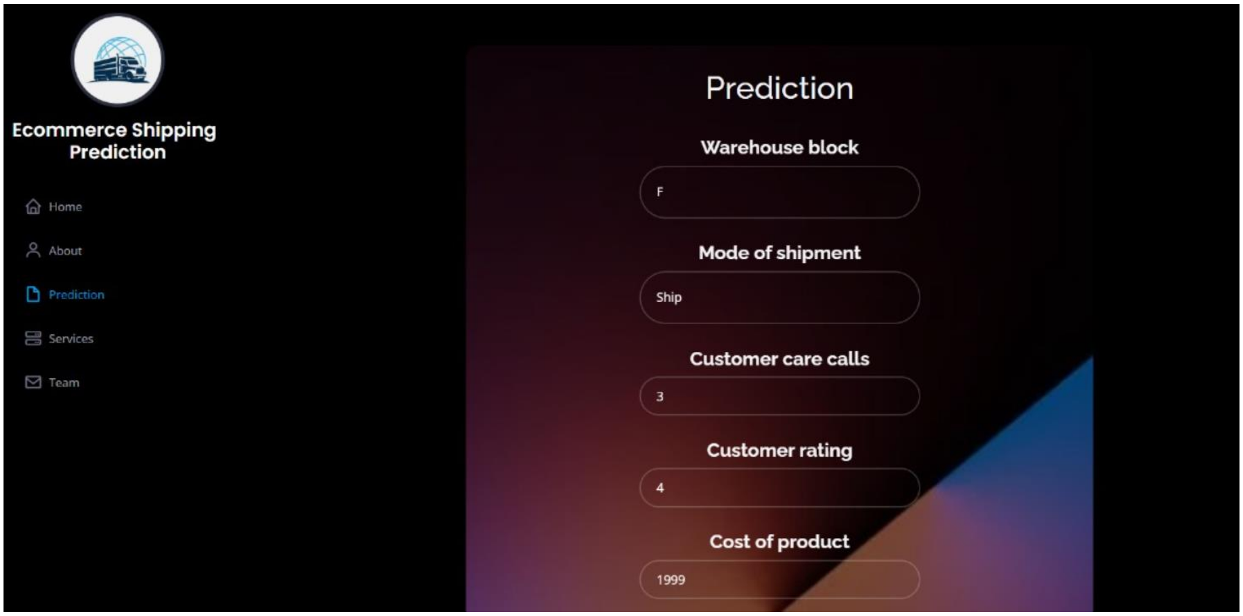
6. Results

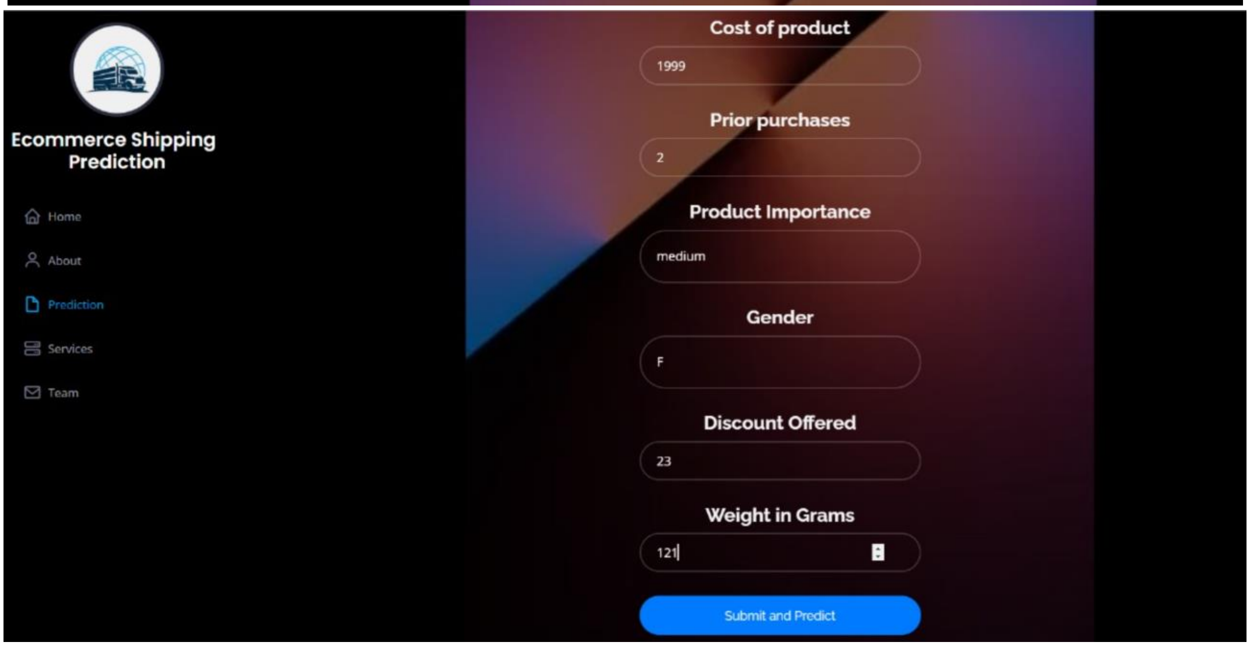
6.1 Output Screenshots

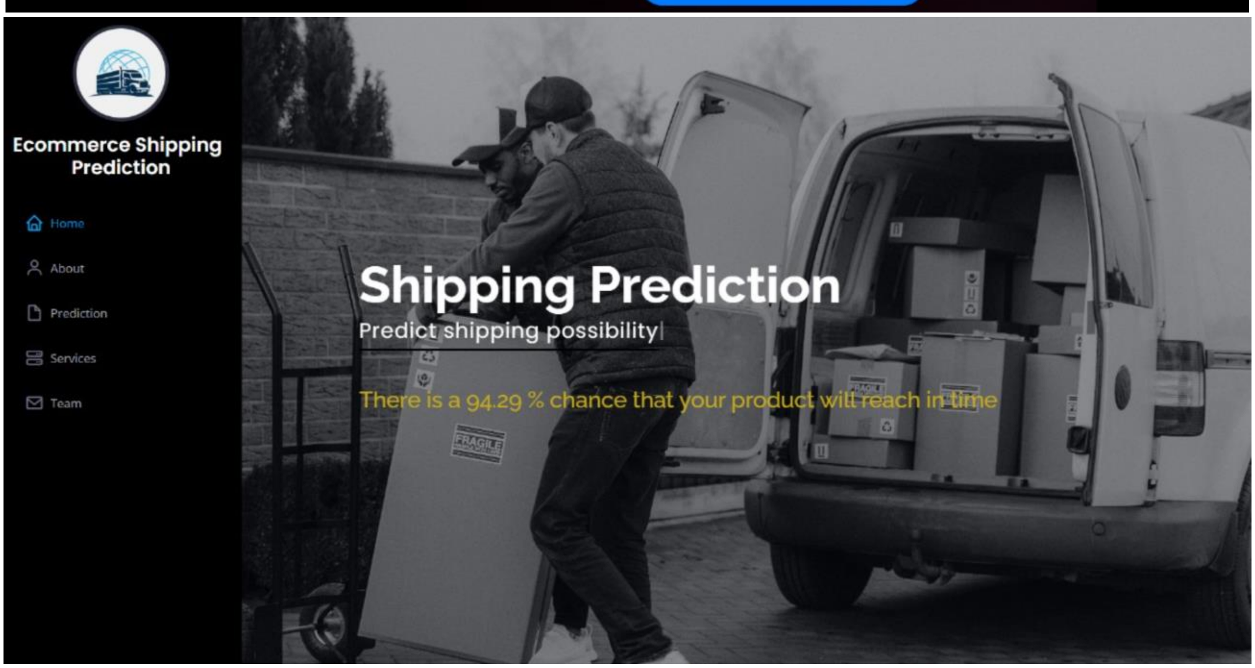







Ecommerce Shipping Prediction
Home
About
Prediction
Services
Team
Prediction
Warehouse block
F
Mode of shipment
Ship
Customer care calls
3
Customer rating
4
Cost of product
1999


Ecommerce Shipping Prediction
Home
About
Prediction
Services
Team
Cost of product
1999
Prior purchases
2
Product Importance
medium
Gender
F
Discount Offered
23
Weight in Grams
121
Submit and Predict


Ecommerce Shipping Prediction
Home
About
Prediction
Services
Team
Shipping Prediction
Predict shipping possibility
There is a 94.29 % chance that your product will reach in time

7. Advantages & Disadvantages

Advantages	Disadvantages
Improved Customer Satisfaction: Accurate delivery predictions enhance customer trust and satisfaction.	**Data Quality Issues**: The project depends on the quality and completeness of the historical data.
Operational Efficiency: Optimized logistics operations reduce delays and costs associated with returns.	**Complex Implementation**: Integrating the predictive model into the existing system can be complex and resource-intensive.
Increased Customer Retention: Reliable delivery estimates encourage repeat business and customer loyalty.	**Maintenance Requirements**: The model will need regular updates and maintenance to remain accurate.
Data-Driven Insights: Identifying key factors influencing delivery times helps improve other areas of the business.	**Initial Investment**: Developing and implementing the predictive model requires a significant upfront investment
Competitive Advantage: Offering reliable delivery predictions can differentiate the platform from competitors.	**Scalability Challenges**: Scaling the model to accommodate growing data and diverse products may present challenges.

8. Conclusion

In conclusion, developing a predictive model to forecast e-commerce shipment delivery times addresses a critical pain point for both customers and the e-commerce platform. The project aims to provide accurate and reliable delivery estimates by leveraging historical shipment data and advanced machine-learning techniques. This will significantly enhance customer satisfaction by setting clear expectations and reducing the frustration associated with delivery delays. Improved delivery predictions will also help build trust in the platform, increasing customer loyalty and retention.

Furthermore, the project offers substantial operational benefits. By identifying and analyzing the key factors that influence delivery times, the platform can optimize its logistics operations, reduce inefficiencies, and lower costs associated with delays and returns. The insights gained from this predictive model will enable more informed decision-making and strategic planning, providing a competitive edge in the highly dynamic e-commerce market. Overall, the successful implementation of this project promises to transform the e-commerce delivery experience, benefiting both customers and the platform in the long term.

9. Future Scope

The future scope of this project extends far beyond the initial implementation of the predictive model for e-commerce shipment delivery times. One significant avenue for future work is the integration of real-time data streams into the predictive model. By incorporating real-time tracking information, weather conditions, and traffic data, the model can provide even more accurate and dynamic delivery estimates. This real-time integration would allow the platform to adapt to changing conditions instantly, further enhancing delivery reliability and customer satisfaction.

Another promising area for future development is the application of advanced machine learning and artificial intelligence techniques. Techniques such as deep learning, reinforcement learning, and ensemble methods could be explored to improve the model's accuracy and robustness. Additionally, the use of explainable AI (XAI) could provide more transparent and interpretable predictions, helping stakeholders understand the factors influencing delivery times and making it easier to address specific issues.

Expanding the scope of the model to include international shipments is another potential direction. International deliveries come with their own set of challenges, including customs clearance, international logistics, and varying transportation regulations. By adapting the model to handle these complexities, the platform could offer reliable delivery predictions for global shipments, thus catering to a broader customer base and increasing its market reach.

Finally, the insights gained from this project could be leveraged to enhance other aspects of the e-commerce platform. For instance, predictive analytics could be applied to inventory management to anticipate stock levels based on expected delivery times and customer demand. Moreover, customer service operations could be improved by providing proactive updates and addressing potential delivery issues before they escalate. These extensions of the project would not only improve operational efficiency but also create a more seamless and satisfying customer experience overall.

10. Appendix

10.1 Source Code

https://drive.google.com/file/d/16vUvlLYC2-goe72DfYiCH9RYNP2qwa8S/view?usp=sharing

10.2 GitHub and Project Demo link

GitHub: https://github.com/srama-krishnan/E-Commerce-Shipping-Prediction-using-ML

Demo Link:
https://drive.google.com/file/d/1tfTaSbWJynJFaPVlTIwRco4olZu6U4FV/view?usp=sharing

Folder link: https://drive.google.com/drive/folders/1Avc5AK5SdVgViyBIxcQoDf-o0a-T-w-K?usp=drive_link