Ecommerce Shipping Prediction using Machine Learning

1. Introduction

Ecommerce shipping prediction is the process of estimating that, whether the product reached on time or not. which is based on various factors such as the origin and destination of the package, the shipping method selected by the customer, the carrier used for shipping, and any potential delays or issues that may arise during the shipping process. Machine learning models can be used to make accurate predictions about shipping times based on historical data and real-time updates from carriers. Overall Ecommerce shipping prediction is an important tool for ecommerce businesses that want to provide accurate delivery estimates to their customers and improve their overall customer experience.

1.1. Project overviews

The Ecommerce Shipping Prediction project aims to develop a machine learning model that accurately predicts whether an ecommerce package will be delivered on time. This project uses a dataset with various shipment-related factors, which includes warehouse block, mode of shipment, customer care calls, customer rating, cost of the product, prior purchases, product importance, gender, discount offered, and weight in grams. By leveraging these features, the model provides accurate delivery estimates, enhancing customer satisfaction.

1.2. Objectives

The primary objective of this project is to develop a machine learning model which predicts whether the product will reach on time based on various factors, by providing accurate delivery estimates, project aims to improve the overall customer experience in the ecommerce industry.

By training a Machine Learning model we can estimate delivery time as on time or not, and web deploying it helps people to use this application. This project aims to complete the above mentioned case in series of milestones,

Now we can get into the project Milestones and their respective completion of work in detail,

Milestone 1:

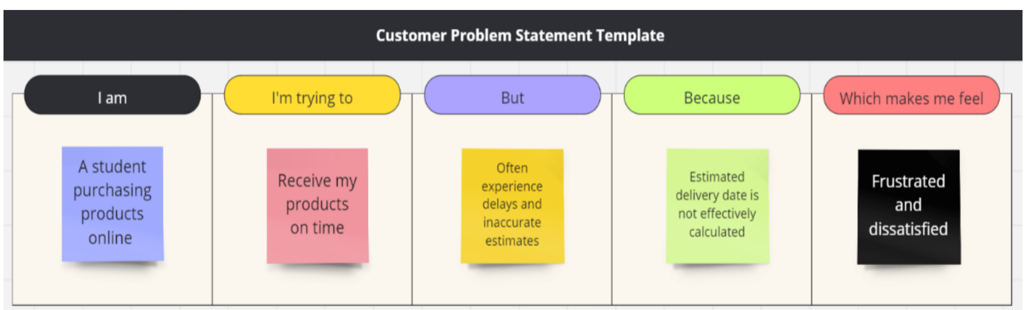
2. Project Initialization and Planning Phase

Project initialization and planning phase, gives an idea and plan, about the project regarding how to complete by making us to create a problem statement, project proposed solution and a planning report using sprints, epics and stories using JIRA.

2.1. Define Problem Statement

Problem Statement:

Customers often experience delays and inaccurate delivery estimates for their online purchases, leading to frustration and dissatisfaction about the delivery estimation time. Current systems fail to consider dynamic factors like shipment mode, warehouse block, customer care interactions, product importance, and real-time conditions, resulting in unreliable delivery predictions, which are becoming the pain points of current customers. This impacts customer trust and loyalty, and harms the business's reputation.



By developing a Machine Learning model using data on shipment details, customer interactions, product attributes, and more, we aim to provide accurate delivery time predictions. This will enhance customer satisfaction, reduce support costs, and improve overall business performance.

2.2. Project Proposal (Proposed Solution)

The project aims to provide significant benefits to the e-commerce business and its customers, fostering a more reliable, efficient, and customer-friendly shipping process. By leveraging advanced machine learning techniques, the project seeks to enhance the reliability and precision of shipping time predictions, ultimately leading to improved customer satisfaction.

Refer the Template Screenshot provided:

Proposed Solution	
Approach	Employing machine learning classification techniques to analyse and predict creditworthiness, creating a dynamic and adaptable shipping. Classification Algorithms should be used, chose best model and thereby saving the model and deploying the model through flask
Key Features	-Implementation of a machine learning-based credit assessment model. -Flask deployment for easy access. -Order-Related Features. -Customer-Related Features. -Shipping Carrier Features.

Resource Requirements

Resource Type	**Description**	**Specification/Allocation**
Hardware		
Computing Resources	CPU/GPU specifications, number of cores	2 x NVIDIA V100 GPUs
Memory	RAM specifications	8 GB
Storage	Disk space for data, models, and logs	1 TB SSD
Software		
Frameworks	Python frameworks	Flask
Libraries	Additional libraries	Pandas, NumPy, Matplotlib, Seaborn, Scikit-learn
Development Environment	IDE, version control	Jupyter Notebooks, PyCharm, or VS Code.
Data		
Data	Source, size, format	Kaggle dataset URL: https://www.kaggle.com/datasets/prachi13/customer-analytics?select=Train.csv Size: 124KB Rows: 10999, Columns: 12

2.3. Initial Project Planning

Planning the sprints(schedule) to complete the project in time, we used JIRA to plan the epics and user stories and allocated respective works to the team.

Refer the Template Screenshot provided:

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	ESPUML-2	Specify The Business Problem	2	Medium	Srikar	6/7/2024	7/7/2024
Sprint-1	Define Problem / Problem Understanding	ESPUML-3	Business Requirements	1	Low	Kesava	6/7/2024	7/7/2024
Sprint-1	Define Problem / Problem Understanding	ESPUML-4	Literature Survey	1	Low	Kesava	6/7/2024	7/7/2024
Sprint-1	Define Problem / Problem Understanding	ESPUML-5	Social or Business Impact	2	Medium	Sai Viswanadh	6/7/2024	7/7/2024

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members	Sprint Start Date	Sprint End Date (Planned)
Sprint-2	Data Collection & Preparation	ESPUML-7	Collect the Dataset	2	Medium	Narendra	7/7/2024	8/7/2024
Sprint-2	Data Collection & Preparation	ESPUML-8	Data Preparation	2	Medium	Srikar	7/7/2024	8/7/2024
Sprint-5	Project Demonstration & Documentation	ESPUML-18	Project Demonstration & Documentation	3	High	Kesava & Sai Viswanadh	7/7/2024	12/7/2024
Sprint-2	Exploratory Data Analysis	ESPUML-10	Descriptive Statistics	1	Low	Narendra	7/7/2024	8/7/2024
Sprint-2	Exploratory Data Analysis	ESPUML-11	Visual Analysis	2	Medium	Sai Viswanadh	7/7/2024	8/7/2024
Sprint-3	Model Building	ESPUML-13	Training The Model In Multiple Algorithms	3	High	Srikar & Kesava	8/7/2024	10/7/2024
Sprint-3	Model Building	ESPUML-14	Testing The Model	2	Medium	Narendra	8/7/2024	10/7/2024
Sprint-3	Performance Testing & Hyperparameter Tuning	ESPUML-16	Testing Model With Multiple Evaluation Metrics	2	Medium	Kesava	8/7/2024	10/7/2024
Sprint-4	Model Deployment	ESPUML-19	Save The Best Model	1	Low	Kesava	10/7/2024	12/7/2024
Sprint-4	Model Deployment	ESPUML-20	Integrate With Web Framework	3	High	Srikar & Sai Viswanadh & Narenda	10/7/2024	12/7/2024

Milestone 2:

3. Data Collection and Preprocessing Phase

This phase 'Data Collection and Preprocessing' involves executing a plan to gather relevant 'Ecommerce Shipping' dataset from Kaggle, ensuring data quality through verification and addressing missing values. Preprocessing tasks include cleaning, encoding, handling imbalance and organizing the dataset for subsequent exploratory analysis and machine learning model development.

3.1. Data Collection Plan and Raw Data Sources Identified

In this subphase, we planned to collect the publicly available dataset in Kaggle,

Refer the Template Screenshot provided:

Data Collection Plan Template

Section	Description
Project Overview	The project estimates that, whether the product reached on time, which is based on various factors such as warehouse, the shipping method, the carrier used for shipping, and any potential delays or issues that may arise during the shipping process. This is Machine Learning Classification approach.
Data Collection Plan	From Kaggle- "Ecommerce Shipping data" the dataset is going to be collected.
Raw Data Sources Identified	Name of the Dataset in Kaggle: E-Commerce Shipping Data, File Size: 124KB The collected data contains variables such as
	Mode_of_Shipment, Cost_of_the_Product, Product_importance, Weight_in_gms, Reached.on.Time_Y.N etc

Raw Data Sources Template

Source Name	Description	Location/URL	Format	Size	Access Permissions
Dataset 1	The dataset includes shipment details like Mode_of_Shipment, Cost_of_the_Product, Product_importance, Weight_in_gms, Reached.on.Time_Y.N etc	https://www.kaggle.com/datasets/prachi13/customer-analytics?select=Train.csv	CSV	124 KB	Public

3.2. Data Quality Report

In this phase we looked at the dataset and analysed the issues in it, our dataset is a large with 11,000 rows and 12 columns as features. There are outliers and imbalance on target column.

Refer the Template Screenshot provided:

Data Quality Report Template

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

Data Source	Data Quality Issue	Severity	Resolution Plan
Source Name: Kaggle	It is a publicly available dataset in Kaggle of size 124KB, The dataset contains many Features to predict on, and also have many outliers in two columns (Features), and data is imbalance on target Feature because the number of 1's, 0's are different. The dataset has 11000 Rows which is huge.	Moderate	We are going, Find the outliers and replace them with Mean value of the column, because removing them causing 2000 data points loss and remaining 8 columns (Features) which are valuable for prediction are removing because of just 2 columns. Data Imbalance is to be solved by using SMOTE over sampling.

3.3. **Data Exploration and Preprocessing**

Here we have done preprocessing the dataset with Descriptive and Visual analysis, replaced outliers with mean value of columns, used SMOTE for balancing the dataset with respect to Target column.

Refer the Template Screenshot provided:

Data Exploration and Preprocessing Template

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

Section	Description
Data Overview	Looked at the dataset for its shape, info and description of basic statistics of the features.

```
[4]: df.shape

[4]: (10999, 12)

[5]: df.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
```

```
[7]: df.describe()
```

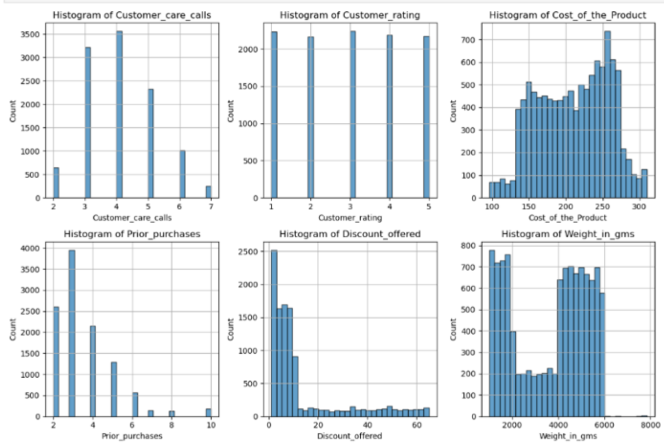
	ID	Warehouse_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	Cost_
count	10999.00000	10999.000000	10999.000000	10999.000000	10999.000000	
mean	5500.00000	2.333394	1.516865	4.054459	2.990545	
std	3175.28214	1.490726	0.756894	1.141490	1.413603	
min	1.00000	0.000000	0.000000	2.000000	1.000000	
25%	2750.50000	1.000000	1.000000	3.000000	2.000000	
50%	5500.00000	3.000000	2.000000	4.000000	3.000000	
75%	8249.50000	4.000000	2.000000	5.000000	4.000000	
max	10999.00000	4.000000	2.000000	7.000000	5.000000	

<table>
<tr>
<td>Univariate Analysis</td>
<td>

It is the single to single feature analysis, Used Histograms for Numerical Features and Count Plot for categorical Features with seaborn and matplotlyb libraries.

Univariate

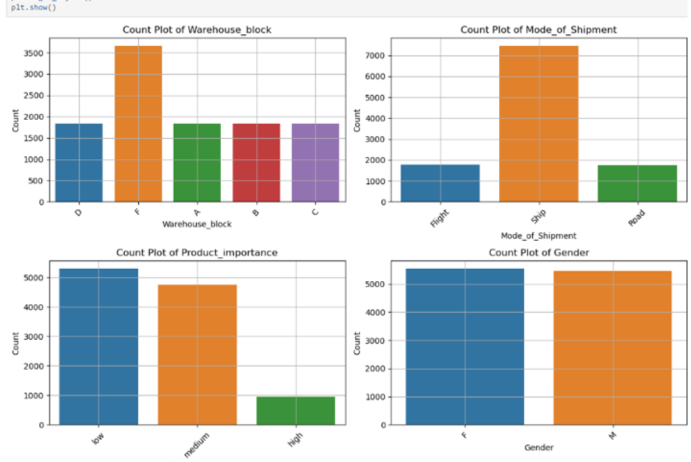
```
[8]: # List of numerical columns
numerical_columns = ['Customer_care_calls', 'Customer_rating', 'Cost_of_the_Product', 'Prior_purchases', 'Discount_offered', 'Weight_in_gms']

plt.figure(figsize=(12, 8))
for i, col in enumerate(numerical_columns):
    plt.subplot(2, 3, i + 1)
    plt.hist(df[col], bins=30, edgecolor='k', alpha=0.7)
    plt.xlabel(col)
    plt.ylabel('Count')
    plt.title(f'Histogram of {col}')
    plt.grid(True)

plt.tight_layout()
plt.show()
```



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

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

```
categorical_cols = df.select_dtypes(include=['object']).columns

plt.figure(figsize=(12, 8))
for i, col in enumerate(categorical_cols):
    plt.subplot(2, 2, i + 1)
    sns.countplot(x=col, data=df)
    plt.xlabel(col)
    plt.ylabel('Count')
    plt.title(f'Count Plot of {col}')
    plt.xticks(rotation=45)
    plt.grid(True)

plt.tight_layout()
plt.show()
```



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

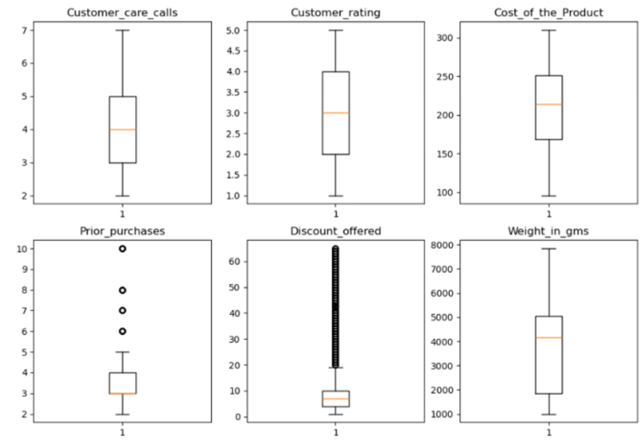
<table>
<tr>
<td>Bivariate Analysis</td>
<td>
Used Boxplots for Bivariate analysis, and also to check for outliers.

Bivariate Analysis

<pre>
[10]: # Plotting box plots for numerical features
plt.figure(figsize=(12, 8))

for i, column in enumerate(numerical_columns, 1):
    plt.subplot(2, 3, i)
    plt.boxplot(df[column])
    plt.title(column)

plt.show()
</pre>

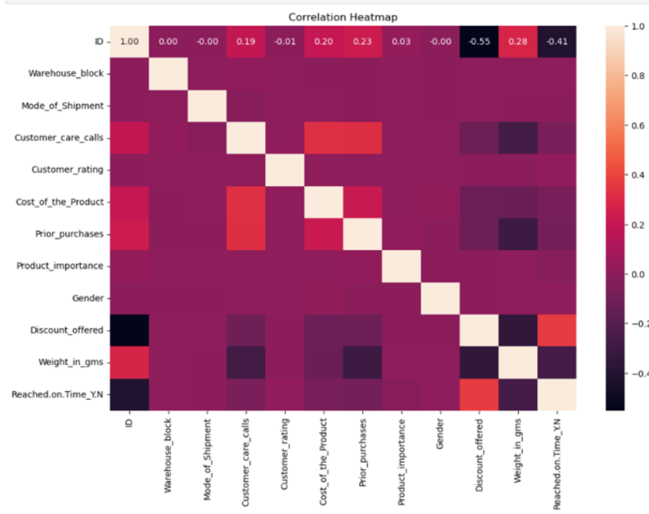


</td>
</tr>
</table>

<table>
<tr>
<td>Multivariate Analysis</td>
<td>
Used Heatmap which is the best way for multivariate analysis, it is plotted based on correlation values between each Feature.
-Due to some version issues the numbers are not getting to every cell.

<pre>
[16]: corr_matrix = df.corr()

plt.figure(figsize=(12, 8))
sns.heatmap(corr_matrix, annot=True, fmt=".2f")
plt.title('Correlation Heatmap')
plt.show()
</pre>



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

Outliers and Anomalies

Found the outliers and replaced them with Mean value of the column, because removing them causing 2000 data points loss, and remaining 8 columns (Features) which are valuable for prediction are removing because of just 2 columns.

```python
# 'Prior_purchases', 'Discount_offered

def remove_outliers(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    mean_value = df[column].mean()

    # Replace outliers with the mean
    df.loc[(df[column] < lower_bound) | (df[column] > upper_bound), column] = mean_value

    return df

df = remove_outliers(df, 'Prior_purchases')
df = remove_outliers(df, 'Discount_offered')
```

Data Preprocessing Code Screenshots

Loading Data

With pandas loaded the dataset downloaded from Kaggle.

```python
df=pd.read_csv('Train.csv')
```

```python
df.head()
```

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

Handling Missing Data

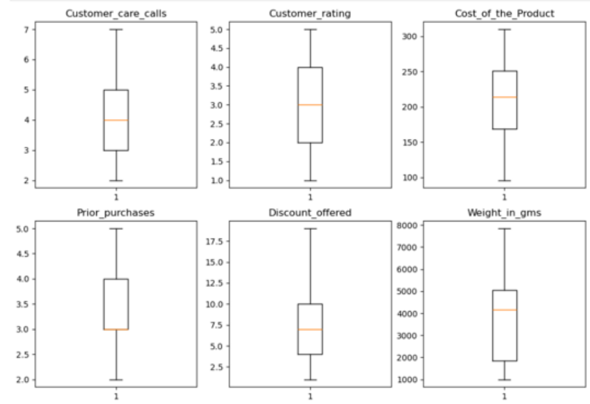
There are no Missing Values in the dataset.

```python
6]: df.isna().sum()
```

```
6]: 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
```

After removing

```
# checking the removal
plt.figure(figsize=(12, 8))

for i, column in enumerate(numerical_columns, 1):
    plt.subplot(2, 3, i)
    plt.boxplot(df[column])
    plt.title(column)

plt.show()
```



Data Transformation

Used Label Encoding to transform Categorical features, and Standard Scaler is used to scale the values.

Encoding

```
le=LabelEncoder()
df.Product_importance=le.fit_transform(df.Product_importance)
df.Gender=le.fit_transform(df.Gender)
df.Mode_of_Shipment=le.fit_transform(df.Mode_of_Shipment)
df.Warehouse_block=le.fit_transform(df.Warehouse_block)
```

```
df.head()
```

	ID	Warehouse_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	Cost_of_the_Product	Prior_purchases	Product_impc
0	1	3	0	4	2	177	3.0	
1	2	4	0	4	5	216	2.0	
2	3	0	0	2	2	183	4.0	
3	4	1	0	3	3	176	4.0	
4	5	2	0	2	2	184	3.0	

Scaling the data

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

pkl.dump(sc,open("Ecommerce.pkl",'wb'))
```

Feature Engineering

Just removed the ID column which has no use in predicting the target feature(‘Reached on time’)

```
# Removing id column and making x,y data

x=df.drop(columns=['ID','Reached.on.Time_Y.N'],axis=1)    # id wont effect
y=df['Reached.on.Time_Y.N']
```

Save Processed Data

We can save the processed data with the code,

```
x.to_csv('preprocessed_data.csv')
```

Milestone 3:

4. **Model Development Phase**

In this milestone, we have done the feature selection (which features are useful for prediction) in the dataset and trained different models and validated with metrics.

4.1. **Feature Selection Report**

There are 12 Features (columns) including one target column in the dataset. We validated the importance of each feature and created a template,

Refer the Template Screenshot provided:

Feature Selection Report Template

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
Warehouse_block	Which warehouse the product is stored before delivery.	Yes	Provides data and operational insights to predicts ecommerce shipment time.
Mode_of_Shipment	Mode of transport of the product- ship, flight, by road.	Yes	Different modes have different average travel times and reliability.
Customer_care_calls	The number of calls made for enquiry of the shipment.	Yes	Provides insights on delivery issues, resolution times to predict shipment time.

Customer_rating	Customer ratings, 1 is the low (Worst), 5 is the high (Best).	Yes	Indicates the service quality which could be related to shipment time.
Cost_of_the_Product	Cost of the Product in US Dollars.	Yes	Influence shipment priority, handling care, and shipping method.
Prior_purchases	The Number of Prior Purchase.	Yes	Indicates customer behavior like preferred delivery method etc.
Product_importance	Categorized the product with parameter such as low, medium, high priorities.	Yes	Influence shipping priority and method by considering how critical the product is to timely delivery.
Gender	Customer gender.	Yes	Gender-based preferences or purchasing behaviors help in personalized delivery options
Discount_offered	Discount offered on that specific product.	Yes	Affect the volume of orders and processing delays.
Weight_in_gms	Weight of the product in grams.	Yes	Affects shipping cost, and handling requirements and improves accuracy of shipment prediction.
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	History of shipments reaching on time is crucial for predicting future delivery times.
ID	Customer Id	No	Id of the customer has no impact on prediction.

4.2. **Model Selection Report**

We used different Classification Algorithms in Machine Learning to train the model, and evaluated the models with performance metrics, which is Accuracy Score.

Refer the Template Screenshot provided:

Model Selection Report

In the forthcoming Model Selection Report, various models will be outlined, detailing their descriptions, hyperparameters, and performance metrics, including Accuracy or F1 Score. This comprehensive report will provide insights into the chosen models and their effectiveness.

Model Selection Report:

Model	Description	Hyperparameters	Performance Metric (e.g., Accuracy, F1 Score)
Logistic Regression	It is a statistical method primarily used for binary classification tasks. The model uses a logistic function (also known as the sigmoid function) to map the linear combination of features to a probability score between 0 and 1.	–	Accuracy Score = 67%
Random Forest	It is an ensemble learning technique based on decision trees. It combines multiple decision trees to improve overall accuracy and reduce overfitting. Each tree in the forest is trained on random subset of features. The final prediction is obtained by aggregating the predictions of individual trees. Random Forest handles non-linear relationships well and is robust against noisy data.	Criterion= 'entropy'	Accuracy Score = 73%

Decision Tree	A hierarchical model. It resembles a flowchart, with nodes representing decision stages. Internal nodes correspond to attribute tests, and leaf nodes indicate class labels. Decision trees are interpretable and widely used for classification and regression tasks.	Criterion= 'entropy'	Accuracy Score = 70%
K-Nearest Neighbours	k-NN is an instance-based learning algorithm. Given a new data point, it identifies the k nearest neighbors (based on a distance metric like Euclidean distance) and assigns the majority class label among those neighbors.	–	Accuracy Score = 70%
SVM(Support Vector Machine)	SVM creates a hyperplane to separate data into classes. It maximizes the margin between these classes, aiming to find the best separation boundary. It works well for both linearly separable and non-linearly separable data.	–	Accuracy Score = 73%
XG Boost	XGBoost is a powerful gradient boosting algorithm that builds an ensemble of weak learners, typically decision trees. During training, it optimizes a loss function by iteratively adding trees to minimize the error.	–	Accuracy Score = 72%

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

Here we trained the models on training data and validated them using testing data with Evaluation metrics such as classification report and confusion matrix.

Refer the Template Screenshot provided:

Initial Model Training Code:

```
#logistic regression
lr=LogisticRegression()
lr.fit(x_train,y_train)
```

```
LogisticRegression
LogisticRegression()
```

```
#random forest
rf=RandomForestClassifier(criterion='entropy',random_state=1)
rf.fit(x_train,y_train)
```

```
RandomForestClassifier
RandomForestClassifier(criterion='entropy', random_state=1)
```

```
#decision tree
dt=DecisionTreeClassifier(criterion='entropy',random_state=0)
dt.fit(x_train,y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

```
#KNN
knn=KNeighborsClassifier()
knn.fit(x_train, y_train)
```

```
KNeighborsClassifier
KNeighborsClassifier()
```

```
#SVM
model= SVC()
model.fit(x_train,y_train)
```

```
SVC
SVC()
```

```
#XG Boost
xg=xgb.XGBClassifier()
xg.fit(x_train,y_train)
```

```
XGBClassifier
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=None, n_jobs=None,
              num_parallel_tree=None, random_state=None, ...)
```

<table>
<tr><th>Model</th><th>Classification Report</th><th>Accuracy</th><th>Confusion Matrix</th></tr>
<tr><td>LogisticRegression</td><td>print(classification_report(y_test,ypred))
precision recall f1-score support
0 0.66 0.72 0.69 1321
1 0.69 0.62 0.65 1305
accuracy 0.67 2626
macro avg 0.67 0.67 0.67 2626
weighted avg 0.67 0.67 0.67 2626</td><td>67%</td><td>print(confusion_matrix(y_test,ypred))
[[955 366]
[497 808]]</td></tr>
<tr><td>Random Forest</td><td>print(classification_report(y_test,ypred1))
precision recall f1-score support
0 0.69 0.86 0.77 1321
1 0.81 0.60 0.69 1305
accuracy 0.73 2626
macro avg 0.75 0.73 0.73 2626
weighted avg 0.75 0.73 0.73 2626</td><td>73%</td><td>print(confusion_matrix(y_test,ypred1))
[[1141 180]
[520 785]]</td></tr>
<tr><td>Decision Tree</td><td>print(classification_report(y_test,ypred2))
precision recall f1-score support
0 0.71 0.70 0.71 1321
1 0.70 0.71 0.71 1305
accuracy 0.71 2626
macro avg 0.71 0.71 0.71 2626
weighted avg 0.71 0.71 0.71 2626</td><td>71%</td><td>print(confusion_matrix(y_test,ypred2))
[[927 394]
[376 929]]</td></tr>
<tr><td>KNN</td><td>print(classification_report(y_test,ypred3))
precision recall f1-score support
0 0.68 0.78 0.72 1321
1 0.73 0.62 0.67 1305
accuracy 0.70 2626
macro avg 0.71 0.70 0.70 2626
weighted avg 0.70 0.70 0.70 2626</td><td>70%</td><td>print(confusion_matrix(y_test,ypred3))
[[1028 293]
[494 811]]</td></tr>
<tr><td>SVM</td><td>print(classification_report(y_test,ypred4))
precision recall f1-score support
0 0.66 0.94 0.78 1321
1 0.89 0.51 0.65 1305
accuracy 0.73 2626
macro avg 0.78 0.72 0.71 2626
weighted avg 0.78 0.73 0.71 2626</td><td>73%</td><td>print(confusion_matrix(y_test,ypred4))
[[1243 78]
[642 663]]</td></tr>
<tr><td>XG Boost</td><td>print(classification_report(y_test,ypred5))
precision recall f1-score support
0 0.70 0.78 0.74 1321
1 0.75 0.65 0.70 1305
accuracy 0.72 2626
macro avg 0.72 0.72 0.72 2626
weighted avg 0.72 0.72 0.72 2626</td><td>72%</td><td>print(confusion_matrix(y_test,ypred5))
[[1035 286]
[451 854]]</td></tr>
</table>

Milestone 4:

5. **Model Optimization and Tuning Phase**

We had done Hyperparameter Tuning for the best 3 models, selected based on Accuracy Score, compared performance metrics and finally selected the best model based on its performance for the project.

5.1. **Hyperparameter Tuning Documentation**

Selected 3 models for Hyperparameter Tuning, evaluated based on Accuracy score.

Refer the Template Screenshot provided:

Model	Tuned Hyperparameters	Optimal Values

SVM(Support Vector Machine)	Kernel, C, gamma.
	
	parameters={
	'kernel':['rbf'],
	'C':[0.1,0.01],
	'gamma':[0.01,0.0001]
	}
		Accuracy = 70%
		
		print(fit1.best_estimator_,fit1.best_params_,fit1.best_score_)
		SVC(C=0.1, gamma=0.01) {'C': 0.1, 'gamma': 0.01, 'kernel': 'rbf'} 0.6996190476190476

Random Forest	n_estimators, criterion, max_depth, max_features
	
	param_grid = {
	'n_estimators': [200, 300, 500],
	'criterion': ['entropy'],
	'max_depth': [8,9],
	'max_features': ['log2','sqrt']
	}
		Accuracy = 74%
		
		print(fit2.best_estimator_,fit2.best_params_,fit2.best_score_)
		RandomForestClassifier(criterion='entropy', max_depth=9, max_features='log2',
		n_estimators=500, random_state=1) {'criterion': 'entropy', 'max_depth': 9, 'max_features': 'log2', 'n_estimators': 500} 0.7364761904761905

XG Boost	min_child_weight, gamma, colsample_bytree, max_depth
	
	params = {
	'min_child_weight': [10, 20],
	'gamma': [1.5, 2.0, 2.5],
	'colsample_bytree': [0.6, 0.8, 0.9],
	'max_depth': [4, 5, 6]
	}
		Accuracy = 72%
		
		print(fit3.best_estimator_,fit3.best_params_,fit3.best_score_)
		XGBClassifier(base_score=None, booster=None, callbacks=None,
		colsample_bylevel=None, colsample_bynode=None,
		colsample_bytree=0.6, device=None, early_stopping_rounds=None,
		enable_categorical=False, eval_metric=None, feature_types=None,
		gamma=2.5, grow_policy=None, importance_type=None,
		interaction_constraints=None, learning_rate=0.5, max_bin=None,
		max_cat_threshold=None, max_cat_to_onehot=None,
		max_delta_step=None, max_depth=5, max_leaves=None,
		min_child_weight=10, missing=nan, monotone_constraints=None,
		multi_strategy=None, n_estimators=100, n_jobs=None, nthread=3,
		num_parallel_tree=None, ...) {'colsample_bytree': 0.6, 'gamma': 2.5, 'max_depth': 5, 'min_child_weight': 10} 0.7291428571428572

5.2. Performance Metrics Comparison Report

Compared before and after Hyperparameter Tuning - Accuracy score for the 3 models selected.

Refer the Template Screenshot provided:

<table>
<tr>
<td>Random forest</td>
<td><pre>
ypred1=rf.predict(x_test)
print(classification_report(y_test,ypred1))

              precision   recall f1-score   support

           0   0.69   0.86   0.77   1321
           1   0.81   0.62   0.70   1305

    accuracy   0.74   2626
   macro avg   0.75   0.74   0.73   2626
weighted avg   0.75   0.74   0.73   2626

[30]:
print(confusion_matrix(y_test,ypred1))
[[1130 191]
 [501 804]]
</pre></td>
<td><pre>
print(fit2.best_estimator_,fit2.best_params_,fit2.best_score_)
RandomForestClassifier(criterion='entropy', max_depth=9, max_features='log
2',
                       n_estimators=500, random_state=1) {'criterion': 'entr
opy', 'max_depth': 9, 'max_features': 'log2', 'n_estimators': 500} 0.7364761
904761905
</pre></td>
</tr>
<tr>
<td>XG boost</td>
<td><pre>
[41]:
ypred5=xg.predict(x_test)
print(classification_report(y_test,ypred5))

              precision   recall f1-score   support

           0   0.70   0.80   0.75   1321
           1   0.76   0.66   0.70   1305

    accuracy   0.73   2626
   macro avg   0.73   0.73   0.73   2626
weighted avg   0.73   0.73   0.73   2626

[42]:
print(confusion_matrix(y_test,ypred5))
[[1052 269]
 [449 856]]
</pre></td>
<td><pre>
print(fit3.best_estimator_,fit3.best_params_,fit3.best_score_)
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=0.6, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=Non
e,
              gamma=2.5, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.5, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=5, max_leaves=None,
              min_child_weight=10, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=100, n_jobs=None, nthread=3,
              num_parallel_tree=None, ...) {'colsample_bytree': 0.6, 'gamm
a': 2.5, 'max_depth': 5, 'min_child_weight': 10} 0.7291428571428572
</pre></td>
</tr>
<tr>
<td>SVM</td>
<td><pre>
[38]:
ypred4=model.predict(x_test)
print(classification_report(y_test,ypred4))

              precision   recall f1-score   support

           0   0.66   0.94   0.77   1321
           1   0.89   0.51   0.65   1305

    accuracy   0.73   2626
   macro avg   0.77   0.72   0.71   2626
weighted avg   0.77   0.73   0.71   2626

[39]:
print(confusion_matrix(y_test,ypred4))
[[1238   83]
 [638 667]]
</pre></td>
<td><pre>
print(fit1.best_estimator_,fit1.best_params_,fit1.best_score_)
SVC(C=0.1, gamma=0.01) {'C': 0.1, 'gamma': 0.01, 'kernel': 'rbf'} 0.69961904
76190476
</pre></td>
</tr>
</table>

5.3. Final Model Selection Justification

Selected the Best model out of the three based on its performance, Accuracy score and its abilities to manage different aspects for this project.

Refer the Template Screenshot provided:

Final Model	Reasoning
Random Forest	This model was chosen for its superior performance, achieving an accuracy of 74%, the highest among all evaluated models. Random Forest is renowned for its robustness and ability to handle large datasets with high dimensionality. It operates by constructing multiple decision trees during training and outputting the mode of the classes for classification tasks, ensuring improved accuracy and reduced overfitting.

6. **Results**

Saved the best model using pickle library and tested the model with some unknown input, to check whether the model is classifying or not.

6.1. **Output Screenshots**

'1' refers the product will not reach on time, '0' refers product reaches on time.

Ecommerce Testing File

Predicting output for Random Forest

chosen for high accuracy of approx 74

```
[1]: import pickle
     import numpy as np
     import warnings
     warnings.filterwarnings('ignore')
```

```
[2]: # Loading the model

     model = pickle.load(open('Ecommerce_RF74_model.pkl', 'rb'))
     scaler = pickle.load(open("EcommerceScaler.pkl","rb"))
```

```
[3]: model.predict(scaler.transform([[1,0,4,4,200,2,3,1,12.22,2333]]))

t[3]: array([1], dtype=int64)
```

```
[4]: model.predict(scaler.transform([[3,2,3,3,203,2,1,1,9,5733]]))

t[4]: array([0], dtype=int64)
```

We had done the Flask deployment locally of the website, in which there is a form to fill the necessary details (features), and this filled details are passed to the saved model file to predict the result.

The website is done to have good User Interface, having separate sections, data need to be filled in predict section, result will be displayed on predict section itself.

Flask folder structure:

```
Name
> static
> templates
  app.py
  Ecommerce_RF74_model.pkl
  Ecommerce_training_File.ipynb
  EcommerceScaler.pkl
  Train.csv
```

Static folder contains- CSS styles, JS, pictures any images used for the website.

Templates contains- index.html file.

App.py file contains the code to load the files and deploy the website. Make predictions from pkl file and give result.

app.py

```
from flask import Flask, request, render_template
import pickle
import numpy as np
import warnings
warnings.filterwarnings("ignore")

# Initializing
app = Flask(__name__)

# Loading models
model = pickle.load(open('Ecommerce_RF74_model.pkl', 'rb'))
scaler = pickle.load(open("EcommerceScaler.pkl","rb"))

# home route
@app.route('/')
def home():
    return render_template('index.html')

# predict route
@app.route('/predict', methods=['POST'])
def predict():
    if request.method == 'POST':
        # Extract data from form
        data = [
            request.form['Warehouse_block'],
            request.form['Mode_of_Shipment'],
            request.form['Customer_care_calls'],
            request.form['Customer_rating'],
            request.form['Cost_of_the_Product'],
            request.form['Prior_purchases'],
            request.form['Product_importance'],
            request.form['Gender'],
            request.form['Discount_offered'],
            request.form['Weight_in_gms']
        ]

        # Convert data to numpy array and reshape for scaler
        data_array = np.array(data, dtype=float).reshape(1, -1)

        # Scale data
        scaled_data = scaler.transform(data_array)
        prediction = model.predict(scaled_data)

        # prediction
        result = 'Your Product will reach On Time' if prediction == 0 else 'Your product will get Delayed'
        return render_template('index.html', prediction_text=result)

if __name__ == '__main__':
    app.run(debug=False)
```

Running the code generates the address to type in the browser,

Deployed result: Local deployment,

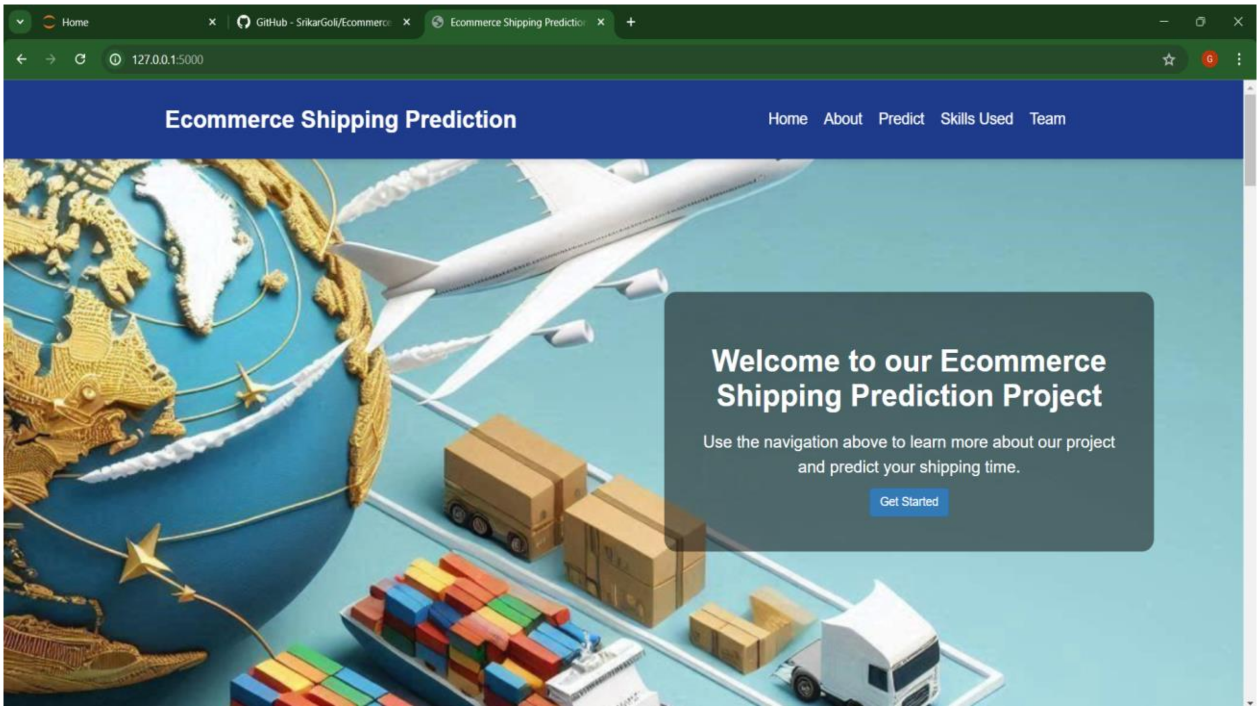
Console 1/A

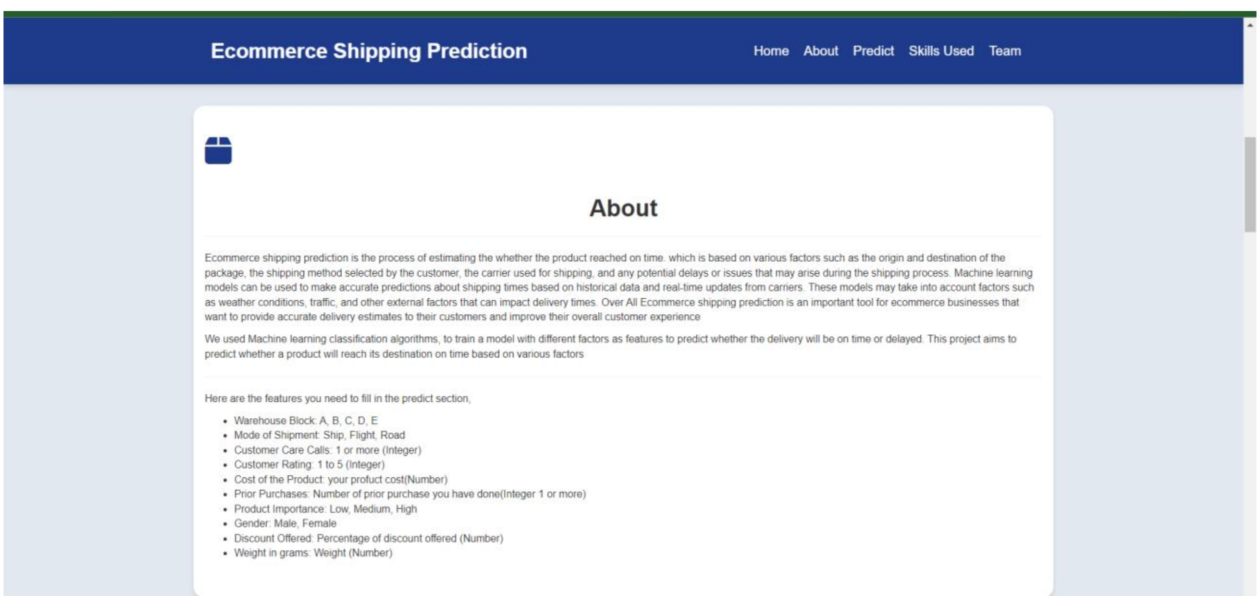
```
Python 3.11.7 | packaged by Anaconda, Inc. | (main, Dec 15 2023, 18:05:47) [MSC v.1916 64 bit
(AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 8.20.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/srika/Ecommerce Shipping Prediction Using Machine Learning/5. Project
Executable Files/Ecommerce_Flask_folder/app.py', wdir='C:/Users/srika/Ecommerce Shipping
Prediction Using Machine Learning/5. Project Executable Files/Ecommerce_Flask_folder')
 * Serving Flask app 'app'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production
WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

If we go to the following Http address (http://127.0.0.1:5000) in the browser, we get

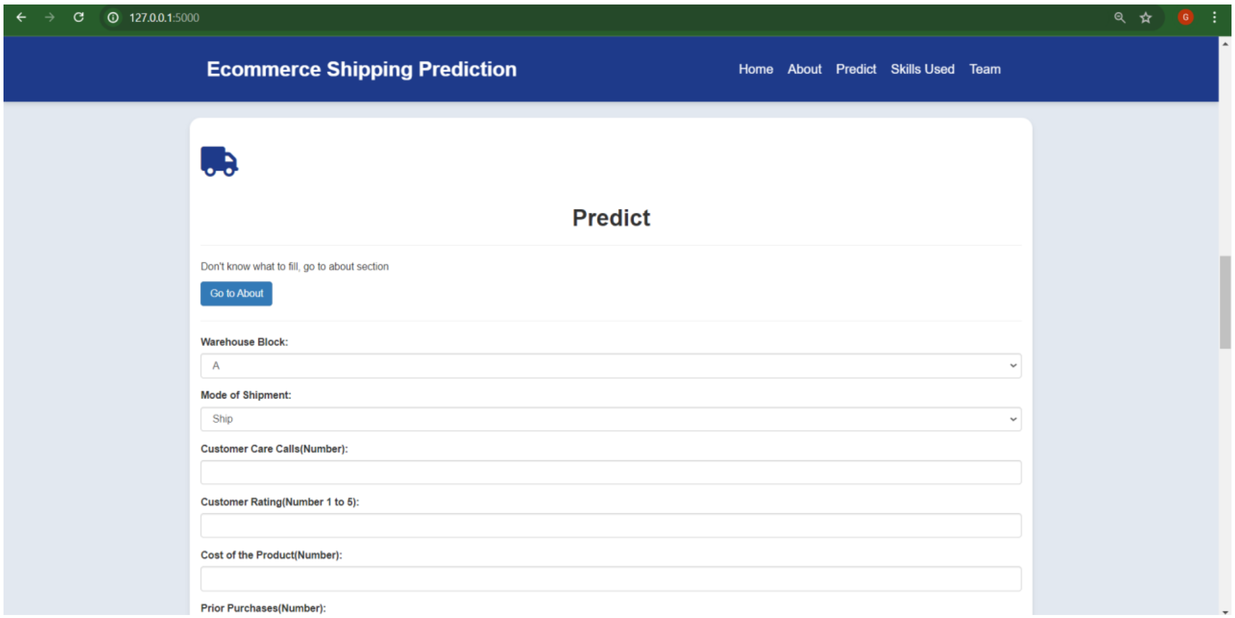
This website "Ecommerce shipping website" is designed with good UI, and visually appealing from scratch, to give good user experience.

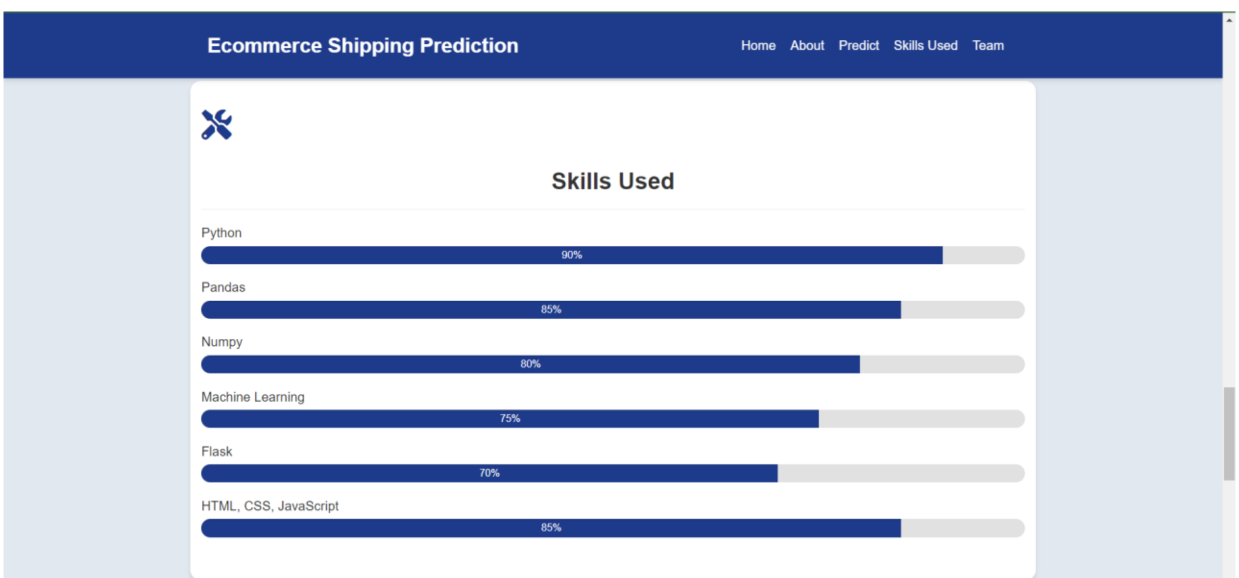
This can be integrated in any of the platform and make changes to use for more applications.

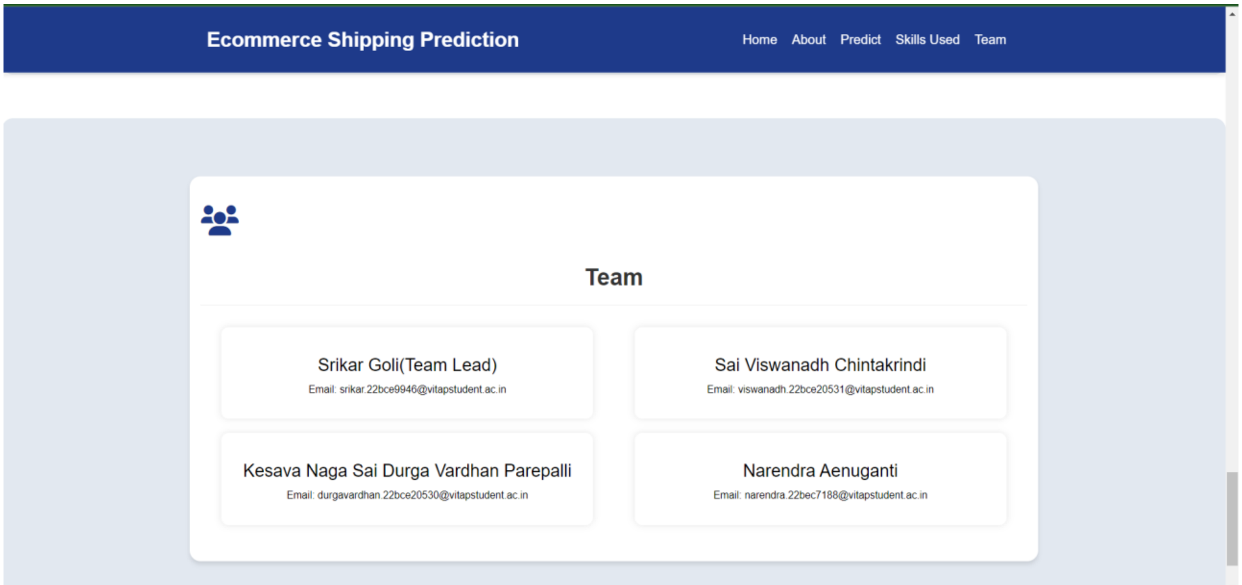




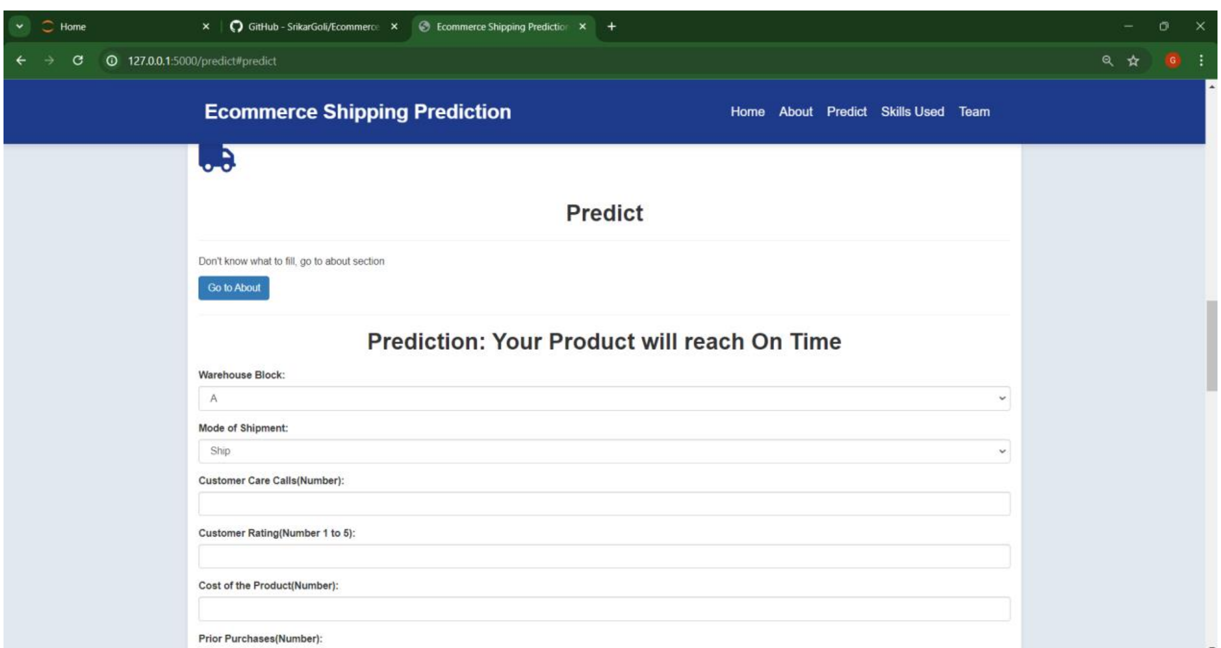
Predict section: Where we need to fill the details to predict the delivery on time or delayed






Ecommerce Shipping Prediction
Home
About
Predict
Skills Used
Team
Team
Srikar Goli(Team Lead)
Email: srikar.22bce9946@vitapstudent.ac.in
Sai Viswanadh Chintakrindi
Email: viswanadh.22bce20531@vitapstudent.ac.in
Kesava Naga Sai Durga Vardhan Parepalli
Email: durgavardhan.22bce20530@vitapstudent.ac.in
Narendra Aenuganti
Email: narendra.22bec7188@vitapstudent.ac.in

Prediction:


Home
GitHub - SrikarGoli/Ecommerce
Ecommerce Shipping Prediction
127.0.0.1:5000/predict#predict
Ecommerce Shipping Prediction
Home
About
Predict
Skills Used
Team
Predict
Don't know what to fill, go to about section
Go to About
Prediction: Your Product will reach On Time
Warehouse Block:
A
Mode of Shipment:
Ship
Customer Care Calls(Number):
Customer Rating(Number 1 to 5):
Cost of the Product(Number):
Prior Purchases(Number):

7. Advantages & Disadvantages

Advantages:

1) By providing accurate delivery estimates, customers can plan their schedules better, reducing frustration caused by delayed deliveries.

2) This model helps in optimizing logistics and supply chain operations by predicting potential delays and allowing proactive measures to be taken.

3) Efficient shipping methods can be chosen based on predictions, potentially lowering shipping costs and improving overall profitability.

4) This model can be adapted for various industries, such as food delivery, courier services, and supply chain management, increasing its applicability and value. Can also expand using Deep Learning.

5) With integration of real-time data like traffic and weather, predictions can be continuously updated, ensuring high accuracy under varying conditions.

6) With the help of this website developed, customer have good user experience to use the 'Ecommerce shipping prediction' and good future scope to integrate in all kinds of business.

Disadvantages:

1) The accuracy of predictions heavily depends on the quality and quantity of historical and real-time data available. Incomplete or inaccurate data can lead to erroneous predictions.

2) Integrating various data sources (traffic, weather, carrier updates) and ensuring their seamless operation can be technically challenging.

3) Unpredictable external factors such as sudden natural disasters, political unrest, or unforeseen events can still impact delivery times despite accurate predictions.

4) Some users might be resistant to relying on automated predictions and prefer traditional methods, especially if the system initially produces any inaccurate predictions.

5) Overfitting or not maintaining the data up-to date may cause predictions erroneous.

8. Conclusion

The Ecommerce Shipping Prediction project is a practical application of machine learning to address a common challenge in logistics and supply chain management. By leveraging historical data and real-time updates, the model provides accurate predictions on whether a product will reach its destination on time or delayed. This capability can significantly enhance customer satisfaction, operational efficiency, and cost-effectiveness for ecommerce businesses.

The project highlights the importance of data quality and integration. Despite the challenges associated with data dependency, complexity, and maintenance, the benefits of implementing such a predictive system are very useful. This project sets the stage for future enhancements and adaptations across various industries, using machine learning in optimizing logistics and improving customer experiences.

9. Future Scope

The future scope of the Ecommerce Shipping Prediction project is extensive and promising. Key areas for enhancements are integrating real-time data, such as traffic and weather updates, to improve prediction accuracy. Expanding the model to other industries like food delivery and courier services can boost efficiency and customer satisfaction across various applications. Advanced machine learning and deep learning techniques can further enhance model performance.

Incorporating user feedback for continuous improvement and developing personalized predictions, integrating the model with popular ecommerce platforms and developing a mobile app for real-time tracking, adapting the model for international use by considering customs and cross-border logistics can broaden its applicability in global shipping scenarios. These advancements collectively enhance the model's robustness, versatility, and overall efficiency in logistics and delivery services.

10. Appendix

10.1. Source Code

The code which we have done,

```
# Regular EDA and plotting libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
# for encoding the data
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
# for scaling
```

```
from sklearn.preprocessing import StandardScaler
# for balancing the dataset
from imblearn.over_sampling import SMOTE
# To save the model
import pickle as pkl
#Models
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
import xgboost as xgb
# Model evaluators and splitting
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, f1_score, confusion_matrix, classification_report
# for hyperparameter tuning
from sklearn.model_selection import GridSearchCV
import warnings
warnings.filterwarnings("ignore")

df=pd.read_csv('Train.csv')
df.head()      # Quick view of the dataset

df.shape
df.info()
df.isna().sum()

# Univariate analysis
numerical_columns = ['Customer_care_calls', 'Customer_rating', 'Cost_of_the_Product',
'Prior_purchases', 'Discount_offered', 'Weight_in_gms']

plt.figure(figsize=(12, 8))
for i, col in enumerate(numerical_columns):
    plt.subplot(2, 3, i + 1)
    plt.hist(df[col], bins=30, edgecolor='k', alpha=0.7)
    plt.xlabel(col)
    plt.ylabel('Count')
    plt.title(f'Histogram of {col}')
    plt.grid(True)

plt.tight_layout()
plt.show()
categorical_cols = df.select_dtypes(include=['object']).columns

plt.figure(figsize=(12, 8))
for i, col in enumerate(categorical_cols):
    plt.subplot(2, 2, i + 1)
    sns.countplot(x=col, data=df)
    plt.xlabel(col)
    plt.ylabel('Count')
    plt.title(f'Count Plot of {col}')
    plt.xticks(rotation=45)
    plt.grid(True)

plt.tight_layout()
plt.show()

# Plotting box plots for bivariate analysis
plt.figure(figsize=(12, 8))

for i, column in enumerate(numerical_columns, 1):
    plt.subplot(2, 3, i)
    plt.boxplot(df[column])
    plt.title(column)

plt.show()
```

```
# Removing outliers

def remove_outliers(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    mean_value = df[column].mean()

    # Replace outliers with the mean
    df.loc[(df[column] < lower_bound) | (df[column] > upper_bound), column] = mean_value

    return df

df = remove_outliers(df, 'Prior_purchases')
df = remove_outliers(df, 'Discount_offered')
# checking the removal
plt.figure(figsize=(12, 8))

for i, column in enumerate(numerical_columns, 1):
    plt.subplot(2, 3, i)
    plt.boxplot(df[column])
    plt.title(column)

plt.show()
# Encoding
le=LabelEncoder()
df.Product_importance=le.fit_transform(df.Product_importance)
df.Gender=le.fit_transform(df.Gender)
df.Mode_of_Shipment=le.fit_transform(df.Mode_of_Shipment)
df.Warehouse_block=le.fit_transform(df.Warehouse_block)

# Multivariate analysis
corr_matrix = df.corr()

plt.figure(figsize=(12, 8))
sns.heatmap(corr_matrix, annot=True, fmt=".2f")
plt.title('Correlation Heatmap')
plt.show()

# descriptive analysis
df.describe()

# Removing id column and making x,y data

x=df.drop(columns=['ID','Reached.on.Time_Y.N'],axis=1)   # id wont effect
y=df['Reached.on.Time_Y.N']

# balancing
smote=SMOTE()
x,y=smote.fit_resample(x,y)  # Using smote to balance data set with respect to target column because
of huge difference

print(x.shape)
print(y.value_counts())

# scaling
sc=StandardScaler()
x=pd.DataFrame(sc.fit_transform(x))

pkl.dump(sc,open("EcommerceScaler.pkl",'wb'))

# splitting
x_train, x_test, y_train, y_test=train_test_split(x,y,train_size=0.80,random_state=42)

# Initial Training
# Logistic Regression
```

```
lr=LogisticRegression()
lr.fit(x_train,y_train)
ypred=lr.predict(x_test)
print(classification_report(y_test,ypred))
print(confusion_matrix(y_test,ypred))
# Random Forest
rf=RandomForestClassifier(criterion='entropy',random_state=1)
rf.fit(x_train,y_train)
ypred1=rf.predict(x_test)
print(classification_report(y_test,ypred1))
print(confusion_matrix(y_test,ypred1))
# Decision Tree
dt=DecisionTreeClassifier(criterion='entropy',random_state=0)
dt.fit(x_train,y_train)
ypred2=dt.predict(x_test)
print(classification_report(y_test,ypred2))
print(confusion_matrix(y_test,ypred2))
# KNN
knn=KNeighborsClassifier()
knn.fit(x_train, y_train)
ypred3=knn.predict(x_test)
print(classification_report(y_test,ypred3))
print(confusion_matrix(y_test,ypred3))
# SVM
model= SVC()
model.fit(x_train,y_train)
ypred4=model.predict(x_test)
print(classification_report(y_test,ypred4))
print(confusion_matrix(y_test,ypred4))
# Xg Boost
xg=xgb.XGBClassifier()
xg.fit(x_train,y_train)
ypred5=xg.predict(x_test)
print(classification_report(y_test,ypred5))
print(confusion_matrix(y_test,ypred5))

# Hyperparameter Tuning
# Random forest
param_grid = {
    'n_estimators': [200, 300, 500],
    'criterion': ['entropy'],
    'max_depth': [8,9],
    'max_features': ['log2','sqrt']
}
fit2=GridSearchCV(estimator=rf, param_grid=param_grid, scoring='accuracy', n_jobs=-1, verbose=3)
fit2.fit(x_train, y_train)
print(fit2.best_estimator_,fit2.best_params_,fit2.best_score_)
# SVM
parameters={
    'kernel':['rbf'],
    'C':[0.1,0.01],
    'gamma':[0.01,0.0001]
}
fit1 = GridSearchCV(SVC(), param_grid=parameters,scoring='accuracy', n_jobs=-1,verbose=3)
fit1.fit(x_train, y_train)
print(fit1.best_estimator_,fit1.best_params_,fit1.best_score_)
# Xg Boost
params = {
    'min_child_weight': [10, 20],
    'gamma': [1.5, 2.0, 2.5],
    'colsample_bytree': [0.6, 0.8, 0.9],
    'max_depth': [4, 5, 6]
}

xgbc = xgb.XGBClassifier(learning_rate=0.5,
                n_estimators=100,
                objective='binary:logistic',
                nthread=3)
```

```
fit3 = GridSearchCV(xgbc,param_grid=params,cv=5,refit=True,scoring='accuracy',n_jobs=-1,verbose=3)
fit3.fit(x_train, y_train)
print(fit3.best_estimator_,fit3.best_params_,fit3.best_score_)

# Saving the best model
pkl.dump(fit2,open('Ecommerce_RF74_model.pkl','wb'))
```

10.2. GitHub & Project Demo Link

GitHub link: https://github.com/SrikarGoli/Ecommerce-Shipping-Prediction-Using-Machine-Learning

Project Demo Link: https://drive.google.com/file/d/12vlvUYEsU5TOe4dtgZ49-qT7FlxpCLYA/view