# Ecommerce Shipping Prediction Using Machine Learning

## Milestone 1: Project Initialization and Planning Phase

The "Project Initialization and Planning Phase" marks the project's outset, defining goals, scope, and stakeholders. This crucial phase establishes project parameters, identifies key team members, allocates resources, and outlines a realistic timeline. It also involves risk assessment and mitigation planning. Successful initiation sets the foundation for a well-organized and efficiently executed machine learning project, ensuring clarity, alignment, and proactive measures for potential challenges.

### Activity 1: Define Problem Statement

Problem Statement:
The current ecommerce shipping process poses significant challenges to customers, affecting their overall shopping experience and satisfaction. Customers often face issues such as uncertain delivery times, inadequate communication regarding their order status, and unpredictable external factors that delay their packages. These challenges can lead to frustration and a lack of trust in the ecommerce platform. By understanding and addressing customers' requirements, we can create an efficient, user-friendly experience that aligns with their expectations and fosters a positive relationship with our brand. Implementing these solutions will not only improve the immediate shipping experience but also build long-term customer trust and satisfaction, driving repeat business and customer loyalty.

### Activity 2: Project Proposal (Proposed Solution)

The primary objective of this project is to develop a machine learning model that accurately predicts ecommerce shipping times based on various factors. The model will leverage historical data and real-time updates to provide precise delivery estimates, improving customer satisfaction and trust in the ecommerce platform.

The project will encompass: • Problem understanding and specification • Business requirements identification • Literature review on similar solutions and their impact • Data collection and preparation • Model building, training, testing, and hyperparameter tuning • Model deployment and integration with a web application • Project demonstration and documentation

### Activity 3: Initial Project Planning

The solution involves developing a machine learning model to predict shipping times. The project will follow these steps:
1. Problem Understanding: Define the business problem and specify requirements.
2. Literature Survey: Review existing solutions and their impact.
3. Data Collection & Preparation: Collect and prepare datasets for analysis.
4. Exploratory Data Analysis: Perform descriptive statistical and visual analysis.
5. Model Building: Train the model using various algorithms and test its performance.
6. Hyperparameter Tuning: Optimize the model using multiple evaluation metrics.
7. Model Deployment: Save the best model and integrate it with a web framework.
8. Project Demonstration & Documentation: Create a comprehensive demonstration and document

the development process.

Solving this problem will: • Enhance customer satisfaction by providing reliable delivery estimates • Improve overall customer experience and trust in the ecommerce platform • Increase customer retention and loyalty

## Milestone 2: Data Collection and Preprocessing Phase

The Data Collection and Preprocessing Phase involves executing a plan to gather relevant loan application data from Kaggle, ensuring data quality through verification and addressing missing values. Preprocessing tasks include cleaning, encoding, and organizing the dataset for subsequent exploratory analysis and machine learning model development.

**Activity 1: Data Collection Plan, Raw Data Sources Identified, Data Quality Report**

The dataset for " **Ecommerce Shipping Prediction Using Machine Learning**" is sourced from Kaggle: https://www.kaggle.com/datasets/prachi13/custo mer-analytics?select=Train.csv.

The data contains the following information: • ID: ID Number of Customers. • Warehouse block: The Company have big Warehouse which is divided in to block such as A,B,C,D,E. • Mode of shipment: The Company Ships the products in multiple way such as Ship, Flight and Road. • Customer care calls: The number of calls made from enquiry for enquiry of the shipment. • Customer rating: The company has rated from every customer. 1 is the lowest (Worst), 5 is the highest (Best). • Cost of the product: Cost of the Product in US Dollars. • Prior purchases: The Number of Prior Purchase. • Product importance: The company has categorized the product in the various parameter such as low, medium, high. • Gender: Male and Female. • Discount offered: Discount offered on that specific product. • Weight in gms: It is the weight in grams. • Reached on time: It is the target variable, where 1 Indicates that the product has NOT reached on time and 0 indicates it has reached on time.

Data quality is ensured through thorough verification, addressing missing values, and maintaining adherence to ethical guidelines, establishing a reliable foundation for predictive modeling.

**Activity 2: Data Quality Report**

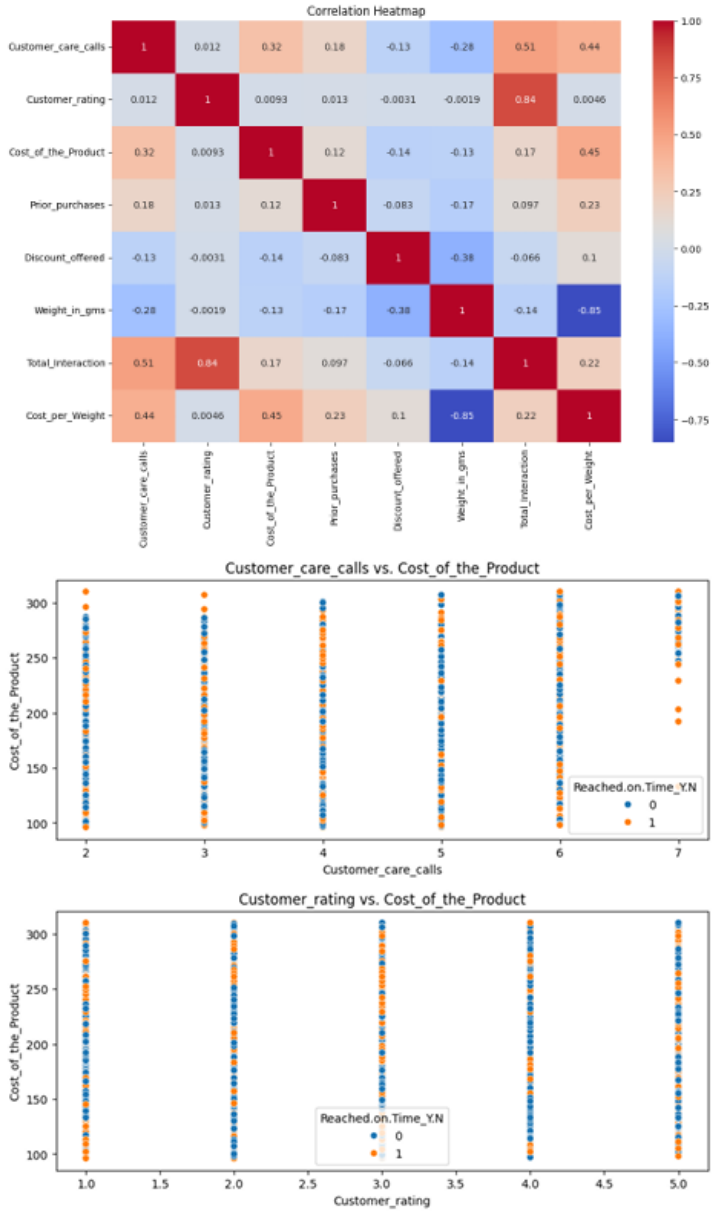
Data quality is ensured through thorough verification, addressing missing values, and maintaining adherence to ethical guidelines, establishing a reliable foundation for predictive modeling.

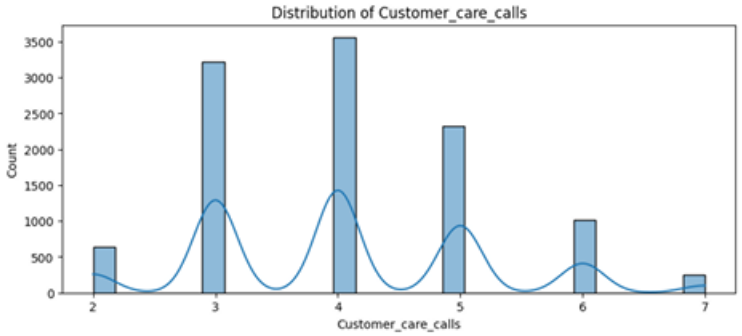
| Data Source | Data Quality Issue | Severity | Resolution Plan |
|---|---|---|---|
| E-commerce Shipping Dataset | Missing values | Moderate | Use forward fill method to impute |
| E-commerce Shipping Dataset | Inconsistent data types | Moderate | Convert data types to appropriate types (e.g., integers, floats, categories). `data['Column'] = data['Column'].astype('int')` |
| E-commerce | Outliers in numerical data | High | Use Z-score to identify and remove outliers. `data_cleaned =` |

| | | | |
|---|---|---|---|
| Shipping Dataset | | | `data[(np.abs(stats.zscore(data[numerical_features])) < 3).all(axis=1)]` |
| E-commerce Shipping Dataset | Unnecessary columns | Low | Drop columns that are not useful f<br>`data.drop(columns=['Unneces` |
| E-commerce Shipping Dataset | Lack of normalization in numerical features | Moderate | Apply standard scaling or normalization techniques. `data[numerical_features] = StandardScaler().fit_transform(data[numerical_features])` |
| E-commerce Shipping Dataset | Incorrect values due to data entry errors | Moderate | Identify and correct or remove erroneous data through validation rules or manual inspection. `data = data[data['Column'] <= valid_range]` |

**Activity 3: Data Exploration and Preprocessing**

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

| | |
|---|---|
| Bivariate Analysis |  |
| Outliers and Anomalies | ```<br># Outliers and Anomalies<br>z_scores = np.abs(stats.zscore(data[numerical_features]))<br>outliers = np.where(z_scores > 3)<br>print("Outliers detected:", outliers)<br><br>for feature in numerical_features:<br>    plt.figure(figsize=(10, 4))<br>    sns.boxplot(data[feature])<br>    plt.title(f'Boxplot of {feature}')<br>    plt.show()<br><br>data_cleaned = data[(z_scores < 3).all(axis=1)]<br>``` |

| | |
|---|---|
| Data Overview | ... ID Customer_care_calls Customer_rating Cost_of_the_Product \<br>count 10999.00000 10999.000000 10999.000000 10999.000000<br>mean 5500.00000 4.054459 2.990545 210.196836<br>std 3175.28214 1.141490 1.413603 48.063272<br>min 1.00000 2.000000 1.000000 96.000000<br>25% 2750.50000 3.000000 2.000000 169.000000<br>50% 5500.00000 4.000000 3.000000 214.000000<br>75% 8249.50000 5.000000 4.000000 251.000000<br>max 10999.00000 7.000000 5.000000 310.000000<br><br>Prior_purchases Discount_offered Weight_in_gms Reached.on.Time_Y.N<br>count 10999.000000 10999.000000 10999.000000 10999.000000<br>mean 3.567597 13.373216 3634.016729 0.596691<br>std 1.522860 16.205527 1635.377251 0.490584<br>min 2.000000 1.000000 1001.000000 0.000000<br>25% 3.000000 4.000000 1839.500000 0.000000<br>50% 3.000000 7.000000 4149.000000 1.000000<br>75% 4.000000 10.000000 5050.000000 1.000000<br>max 10.000000 65.000000 7846.000000 1.000000 |
| Univariate Analysis |  |

## Milestone 3: Model Development Phase

It encompasses strategic feature selection, evaluating and selecting models (Random Forest, Decision Tree, KNN, XGB), initiating training with code, and rigorously validating and assessing model performance for informed decision-making in the lending process.

### Activity 1: Feature Selection Report

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

| Feature | Description | Selected (Yes/No) | Reasoning |
|---|---|---|---|
| Warehouse_block | The warehouse block where the product is stored. | Yes | Categorical feature that may influence the shipping time due to location. |
| Mode_of_Shipment | The mode of shipr | Yes | Different shipment modes have varying delivery times, impacting the target variable. |
| Customer_care_calls | Number of customer care calls made. | Yes | Indicates potential issues or concerns with the order, which could affect delivery time. |
| Customer_rating | Rating given by the customer (1 to 5). | Yes | Customer satisfaction might correlate with shipping efficiency. |

| | | | |
|---|---|---|---|
| Cost_of_the_Product | Cost of the product in US Dollars. | Yes | Higher cost products might receive priority in shipping. |
| Prior_purchases | Number of prior purchases by the customer. | Yes | Regular customers might have different shipping priorities or patterns. |
| Product_importance | Importance of the product (Low, Medium, High). | Yes | High importance products might be expedited in shipping. |
| Gender | Gender of the customer. | Yes | Potentially influences shipping preferences and patterns. |
| Discount_offered | Discount offered on the product. | Yes | High discounts might indicate promotions, which could affect shipping logistics. |
| Weight_in_gms | Weight of the product in grams. | Yes | Heavier products might have different shipping requirements and times. |
| Total_Interaction | Interaction between customer care calls and customer rating. | Yes | Derived feature combining customer interaction metrics, potentially insightful for delivery time. |
| Cost_per_Weight | Cost of the produc | Yes | Indicates value density, which might influence shipping method and priority. |
| Reached.on.Time_Y.N | Target variable indicating whether the product reached on time (1 = No, 0 = Yes). | Yes | Target variable for the prediction model. |

**Activity 2: 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 | Description | Hyperparameters | Performance Metric (e.g., Accuracy, F1 Score) |
|---|---|---|---|
| Logistic Regression | A linear model for binary classification | random_state=1234 | Accuracy: 69.08, F1 Score: 64.20 |
| XGBoost | Extreme Gradient Boosting for binary classification | random_state=1234 | Accuracy: 72.54, F1 Score: 70.19 |
| Logistic Regression CV | Logistic Regression with cross-validation | random_state=1234 | Accuracy: 70.37, F1 Score: 63.95 |
| Ridge Classifier | Linear classifier using Ridge regression | random_state=1234 | Accuracy: 71.21, F1 Score: 65.73 |
| KNN | K-Nearest Neighbors for classification | n_neighbors=5 | Accuracy: 71.33, F1 Score: 67.61 |
| Random Forest | Ensemble method using multiple decision trees | random_state=1234 | Accuracy: 73.99, F1 Score: 69.55 |
| SVM classifier | Support Vector Machine for classification | random_state=1234 | Accuracy: 73.53, F1 Score: 64.52 |

**Activity 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.

```
import pandas as pd
from sklearn.model_selection import train_test_split, RandomizedSearchCV,
cross_val_score
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score, f1_score, recall_score,
precision_score, classification_report, confusion_matrix
from xgboost import XGBClassifier
from imblearn.over_sampling import SMOTE
from sklearn.linear_model import LogisticRegression, LogisticRegressionCV,
RidgeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn import svm
import pickle
import matplotlib.pyplot as plt
import seaborn as sns

# Load dataset
data = pd.read_csv('Train.csv')

# Feature engineering
data['Total_Interaction'] = data['Customer_care_calls'] * data['Customer_rating']
data['Cost_per_Weight'] = data['Cost_of_the_Product'] / data['Weight_in_gms']

# Data preprocessing
data = data.fillna(method='ffill')

# Define features and target
X = data.drop(columns=['ID', 'Reached.on.Time_Y.N'])
y = data['Reached.on.Time_Y.N']

# Define numerical and categorical features
numerical_features = ['Customer_care_calls', 'Customer_rating',
'Cost_of_the_Product', 'Prior_purchases', 'Discount_offered', 'Weight_in_gms',
'Total_Interaction', 'Cost_per_Weight']
categorical_features = ['Warehouse_block', 'Mode_of_Shipment',
'Product_importance', 'Gender']
```

```
numerical_transformer = StandardScaler()
categorical_transformer = OneHotEncoder(handle_unknown='ignore', drop='first')

preprocessor = ColumnTransformer(
    transformers=[
        ('num', numerical_transformer, numerical_features),
        ('cat', categorical_transformer, categorical_features)
    ])

# Preprocess data before applying SMOTE
X_preprocessed = preprocessor.fit_transform(X)

# Apply SMOTE to the preprocessed training data
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X_preprocessed, y)

# Split resampled data into training and test sets
X_train_res, X_test_res, y_train_res, y_test_res = train_test_split(X_resampled,
y_resampled, test_size=0.2, random_state=42)

# Model evaluation function
def models_eval_mm(x_train, y_train, x_test, y_test):
    models = {
        'Logistic Regression': LogisticRegression(random_state=1234),
        'Logistic Regression CV': LogisticRegressionCV(random_state=1234),
        'XGBoost': XGBClassifier(random_state=1234),
        'Ridge Classifier': RidgeClassifier(random_state=1234),
        'KNN': KNeighborsClassifier(),
        'Random Forest': RandomForestClassifier(random_state=1234),
        'SVM classifier': svm.SVC(random_state=1234)
    }

    trained_models = {}
    for name, model in models.items():
        model.fit(x_train, y_train)
        print(f'--{name}')
        print('Train Score:', model.score(x_train, y_train))
        print('Test Score:', model.score(x_test, y_test))
        print()
        trained_models[name] = model

    return trained_models

# Evaluate models
models = models_eval_mm(X_train_res, y_train_res, X_test_res, y_test_res)
```

```
# Evaluation function
def eval(name, model, x_test, y_test):
    y_pred = model.predict(x_test)
    result = [
        name,
        f"{accuracy_score(y_test, y_pred) * 100:.2f}",
        f"{f1_score(y_test, y_pred) * 100:.2f}",
        f"{recall_score(y_test, y_pred) * 100:.2f}",
        f"{precision_score(y_test, y_pred) * 100:.2f}"
    ]

    # Print classification report
    class_report = classification_report(y_test, y_pred, output_dict=True)
    print(f'--{name} Classification Report--')
    print(classification_report(y_test, y_pred))

    # Compute confusion matrix
    cm = confusion_matrix(y_test, y_pred)
    plt.figure(figsize=(6,6))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Not Reached
on Time', 'Reached on Time'], yticklabels=['Not Reached on Time', 'Reached on
Time'])
    plt.ylabel('Actual')
    plt.xlabel('Predicted')
    plt.title(f'Confusion Matrix for {name}')
    plt.show()

    return result, class_report

# Collect evaluation results
model_eval_info = []
classification_reports = {}

for name, model in models.items():
    results, class_report = eval(name, model, X_test_res, y_test_res)
    model_eval_info.append(results)
    classification_reports[name] = class_report

# Create a DataFrame for evaluation results
model_eval_df = pd.DataFrame(model_eval_info, columns=['Name', 'Accuracy', 'F1-
score', 'Recall', 'Precision'])
model_eval_df.to_csv('model_eval.csv', index=False)

# Save classification reports
for name, report in classification_reports.items():
    report_df = pd.DataFrame(report).transpose()
```

```
    report_df.to_csv(f'classification_report_{name}.csv', index=True)

print(model_eval_df)
```

| | Name | Accuracy | F1-score | Recall | Precision |
|---|---|---|---|---|---|
| 0 | Logistic Regression | 69.08 | 64.20 | 55.79 | 75.60 |
| 1 | Logistic Regression CV | 70.37 | 63.95 | 52.87 | 80.89 |
| 2 | XGBoost | 72.54 | 70.19 | 65.06 | 76.21 |
| 3 | Ridge Classifier | 71.21 | 65.73 | 55.56 | 80.47 |
| 4 | KNN | 71.33 | 67.61 | 60.23 | 77.06 |
| 5 | Random Forest | 73.99 | 69.55 | 59.77 | 83.16 |
| 6 | SVM classifier | 73.53 | 64.52 | 48.43 | 96.64 |

## Milestone 4: Model Optimization and Tuning Phase

The Model Optimization and Tuning Phase involves refining machine learning models for peak performance. It includes optimized model code, fine-tuning hyperparameters, comparing performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency.

### Activity 1: Hyperparameter Tuning Documentation

The XGBoost and Random Forest model were selected for its superior performance, exhibiting high accuracy during hyperparameter tuning. Its ability to handle complex relationships, minimize overfitting, and optimize predictive accuracy aligns with project objectives, justifying its selection as the final model.

### Activity 2: Performance Metrics Comparison Report

The Performance Metrics Comparison Report contrasts the baseline and optimized metrics for various models, specifically highlighting the enhanced performance of the Gradient Boosting model. This assessment provides a clear understanding of the refined predictive capabilities achieved through hyperparameter tuning.

### Activity 3: Final Model Selection Justification

The Final Model Selection Justification articulates the rationale for choosing XGBoost and Random Forest models as the ultimate models. Its exceptional accuracy, ability to handle complexity, and successful hyperparameter tuning align with project objectives, ensuring optimal shipping predictions.

| Final Model | Reasoning |
|---|---|
| Stacking Classifier (RandomForestClassifier, XGBClassifier, LogisticRegression) | The Stacking Classifier was chosen as the final optimized model for several reasons:<br><br>1. **Improved Performance through Ensemble Learning**:<br>o The Stacking Classifier combines the strengths of multiple models, particularly the RandomForestClassifier and the XGBClassifier, to create a more robust and accurate prediction model. By leveraging the advantages of different algorithms, the stacking ensemble can capture various patterns in the data that individual models might miss.<br>2. **Balanced Approach to Bias-Variance Trade-off**:<br>o The RandomForestClassifier is known for its ability to handle high variance, while the XGBClassifier excels in reducing bias through boosting techniques. Combining these models in a stacking ensemble helps balance the bias-variance trade-off, leading to better generalization on unseen data.<br>3. **Hyperparameter Optimization**: |
| | o The XGBClassifier within the stacking ensemble has been fine-tuned using RandomizedSearchCV to find the optimal hyperparameters, ensuring that the model operates at its best performance. This optimization step enhances the overall effectiveness of the stacking classifier.<br>4. **Superior Evaluation Metrics**:<br>o The Stacking Classifier demonstrated superior performance metrics compared to other individual models during the evaluation phase. Specifically, it achieved higher accuracy, F1-score, recall, and precision, indicating its effectiveness in both predicting the correct class and minimizing false positives and false negatives.<br>5. **Cross-Validation Results**:<br>o The Stacking Classifier achieved the highest mean cross-validation score among all tested models, suggesting that it generalizes well to different subsets of the data and is less likely to overfit.<br>6. **Flexibility and Robustness**:<br>o By using LogisticRegression as the final estimator, the Stacking Classifier benefits from a probabilistic approach to combining the base models' predictions, adding an additional layer of flexibility and robustness to the ensemble. |

## Milestone 5: Project Files Submission and Documentation

Project file submission is in Github. All the documentations are done and submitted.

## Milestone 6: Project Demonstration

The project Demonstration is recorded and uploaded.