Table of Contents

1. Introduction

Timely delivery is a critical factor in maintaining customer satisfaction, especially in the highly competitive world of e-commerce. This project explores shipment data from an international electronics retailer with the goal of uncovering patterns and causes behind delivery delays. By applying machine learning techniques to historical data, we aim to identify which customer and product-related variables—such as shipping method, customer support interactions, or discount offered—affect the likelihood of a shipment arriving on time.

Understanding these insights can empower the company to improve its operations, allocate resources more efficiently, and enhance the overall customer experience.

2. Data Description

The dataset was sourced from Kaggle and is publicly available at:

https://www.kaggle.com/datasets/prachi13/customer-analytics

It consists of 10,999 observations and 12 variables, covering both categorical and numerical features. The target variable is 'Reached.on.Time_Y.N', where 1 indicates a delayed shipment and 0 indicates an on-time delivery.

Variable Descriptions:

Variable	Description
ID	Unique identifier for each customer
Warehouse_block	Warehouse section (A, B, C, D, F)
Mode_of_Shipment	Shipping method used (Ship, Flight, Road)
Customer_care_calls	Number of customer care calls made
Customer_rating	Rating given by the customer (1 to 5)
Cost_of_the_Product	Price of the product in USD
Prior_purchases	Number of previous purchases by the customer

Product_importance	Importance level of the product (low, medium, high)
Gender	Gender of the customer (M/F)
Discount_offered	Discount given on the product
Weight_in_gms	Product weight in grams
Reached.on.Time_Y.N	Delivery status: 1 = Delayed, 0 = On time (Target Variable)

3. Data Overview

The dataset used for this project was sourced from Kaggle and is titled E-commerce Shipping Data (Gopalani, 2021). It contains shipment tracking records and customer interaction details for an international e-commerce company specializing in electronic products.

The primary objective of analyzing this dataset is to predict whether a product shipment will arrive on time. The target variable, Reached.on.Time_Y.N, is a categorical variable where "0" represents an on-time delivery and "1" represents a delayed delivery. Given that the target is binary, classification models are most appropriate for this study.

This dataset enables the company to answer key questions such as:

- How does customer rating correlate with delivery performance?
- Does product importance influence shipment timeliness?

The overall goal is to leverage machine learning to generate business insights that will enhance logistics and customer satisfaction strategies.

4. Data Cleaning and Exploratory Analysis

4.1 Descriptive Summary of Key Metrics

As part of the data cleaning and exploratory analysis process, we conducted a statistical summary of key numerical variables to better understand customer behavior and product-

related trends. The dataset consists of 10,999 complete observations with no missing values in essential fields, ensuring strong reliability for analysis.

Customer Care Calls: On average, customers made 4.05 support calls per order, with a median of 4. This suggests multiple interactions per shipment, indicating either proactive engagement or recurring service challenges.

Customer Rating: The mean rating is 2.99 with a standard deviation of 1.41. This reflects a generally moderate level of satisfaction but with considerable variability in the customer experience.

Cost of Product: Product prices range from $96 to $310, with an average of $210. This reflects the company's ability to cater to both value-conscious and premium customers.

Prior Purchases: The average number of previous purchases is 3.57, indicating positive customer retention and a solid base of repeat buyers.

On-Time Delivery Rate: The binary target variable, Reached.on.Time_Y.N, has a mean of 0.59, showing that 59% of deliveries arrive on time while 41% are delayed. This suggests a considerable opportunity to improve operational logistics.

The following figure summarizes the descriptive statistics for the key numerical variables in the dataset:

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

[181]:

	id	calls	rating	price	purchases	discount	weight	reached
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

4.2 Warehouse-Level Insights

As part of our data cleaning and exploration phase, we analyzed how delivery performance and promotional strategies vary by warehouse. The dataset includes shipment and purchase records distributed across multiple warehouse locations, each with unique characteristics.

Warehouse 2 emerges as the most operationally active, processing the highest number of shipments at 3,666. It also offers the highest average discount rate, at 59%, which may be

part of a volume-driven pricing strategy. This suggests Warehouse 2 likely serves as a central hub for fulfillment and customer demand.

Across all warehouses, the average number of purchases is relatively consistent, ranging from approximately 3.55 to 3.60. This consistency indicates a stable and evenly distributed purchasing pattern among the warehouses.

Interestingly, Warehouse 3 offers the lowest average discount at just 10%. This notable deviation could potentially influence customer behavior, affecting satisfaction or willingness to purchase from that fulfillment location.

The figure below highlights warehouse-specific shipment counts, purchase behavior, and average discount usage:

[15]:

	id	warehouse	shipment	calls	rating	price	purchases	important	gender	discount	weight	reached	Purchase per warehouse	Shipment count per warehouse	Avg discount used
0	1	3	0	4	2	177	3	1	0	44	1233	1	3.6020	1834	44.0
1	2	4	0	4	5	216	2	1	1	59	3088	1	3.5480	3666	59.0
2	3	0	0	2	2	183	4	1	1	48	3374	1	3.5777	1833	48.0
3	4	1	0	3	3	176	4	2	1	10	1177	1	3.5750	1833	10.0
4	5	2	0	2	2	184	3	2	0	46	2484	1	3.5548	1833	46.0

4.3 Shipment Mode Analysis

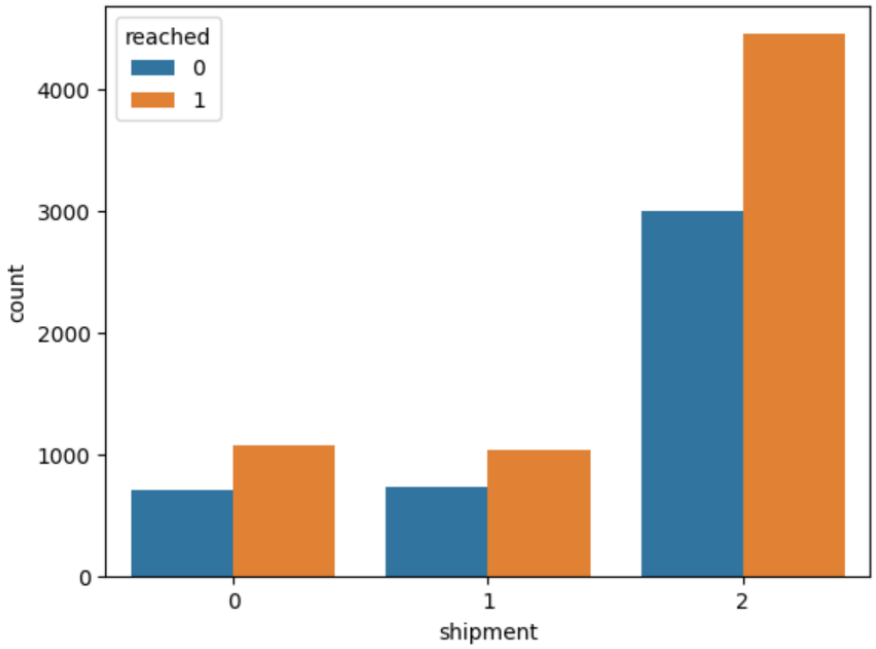
In order to assess how the mode of shipment relates to delivery performance, we analyzed shipment outcomes across all three shipping categories present in the dataset. The variable shipment` classifies deliveries into Modes 0, 1, and 2. Each mode was compared based on its delivery status using the target variable `Reached.on.Time_Y.N`, where 0 represents on-time delivery and 1 represents a delay.

The count plot reveals that Shipment Mode 2 is the most heavily utilized method, handling the highest number of both successful and delayed deliveries. Its wide usage makes it efficient in scale, but it also introduces more variability in outcomes, as reflected in its elevated number of delays.

By comparison, Shipment Modes 0 and 1 show relatively balanced performance, with similar proportions of deliveries that were on time versus those that were delayed. This suggests that Modes 0 and 1 might be more consistent and reliable, which could be strategically useful for shipments requiring tighter control over timing and delivery expectations.

The figure below illustrates shipment performance across delivery modes:

```
[20]: <Axes: xlabel='shipment', ylabel='count'>
```



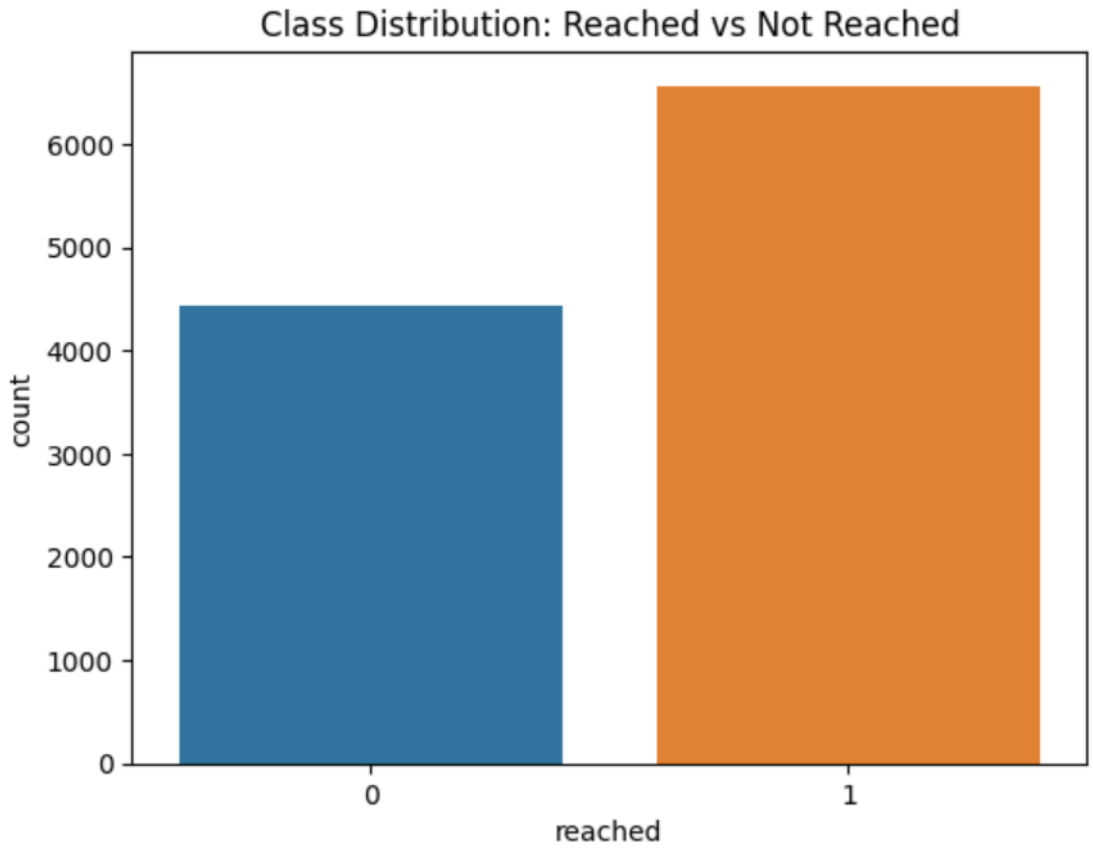
4.4 Delivery Outcome Distribution

To understand the balance between successful and delayed shipments, we examined the distribution of the target variable Reached.on.Time_Y.N, where **0 indicates an on-time delivery** and **1 indicates a delayed shipment**.

The count plot below reveals a noticeable class imbalance. Approximately **60% of all shipments were delivered successfully**, while the remaining **40% were delayed**. This imbalance is significant because it can influence the performance of classification models. Many machine learning algorithms tend to favor the majority class, which in this case would bias the model toward predicting successful deliveries more frequently.

As a result, addressing this class imbalance was a crucial part of our modeling process. To mitigate the risk of biased predictions, we implemented the **SMOTE (Synthetic Minority Oversampling Technique)** method during model training, which synthetically increases the representation of the minority class (delayed deliveries) to help models learn from both classes more evenly.

The following figure illustrates this delivery outcome distribution:



4.5 Visual Insights by Delivery Status

To deepen our understanding of the factors associated with delivery outcomes, we analyzed the average values of four important features—price, discount, weight, and product importance—grouped by delivery status. These comparisons provide insight into how specific product or order characteristics may correlate with successful versus delayed shipments.

Average Price: The average price of delivered and undelivered products is nearly identical. This suggests that delivery status is not strongly associated with product cost.

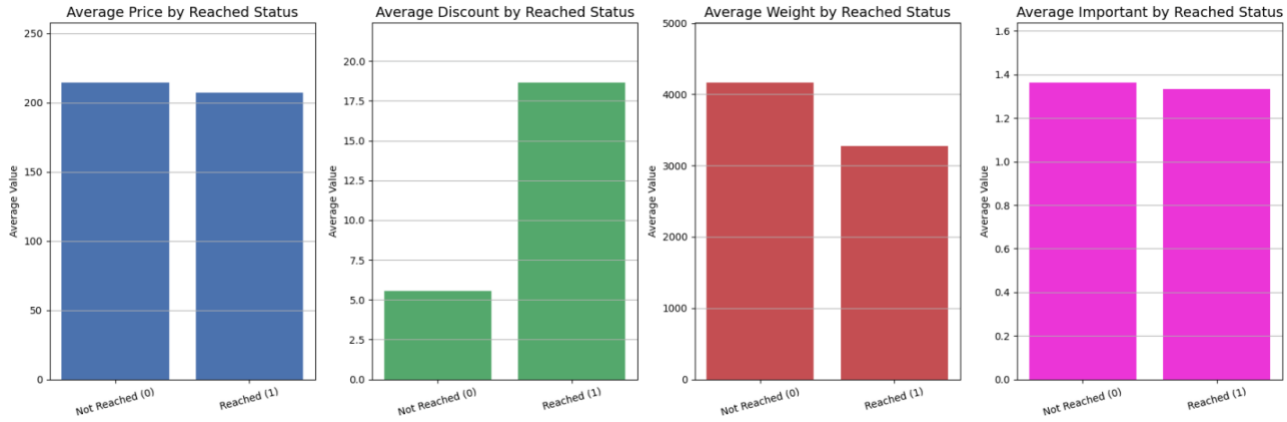
Average Discount: A meaningful difference is observed in the average discount. Delivered shipments received significantly higher average discounts than delayed ones. This may imply that higher-discounted items are prioritized or processed more efficiently.

Average Weight: Products that were not delivered on time tend to be heavier than those that were. This supports the hypothesis that heavier items may introduce logistical challenges that contribute to shipping delays.

Product Importance: The average importance score remains consistent across both delivery

outcomes. This indicates that the company does not appear to prioritize shipments based on the stated product importance level.

The following figure illustrates average values for each feature based on delivery outcome:



5. Technology and Machine Learning Approach

5.1 Data Standardization

Before applying machine learning models, we standardized all numerical features to ensure consistency and fair comparison across scales. Standardization transforms data to have a mean of 0 and standard deviation of 1, which is especially important for distance-based algorithms such as K-Nearest Neighbors (KNN).

The plot below illustrates the transformation applied to the price feature (first 1,000 rows), showing how the original values were re-centered and scaled, which helped us confirm that the transformation worked as expected:

5.2 Classifier Parade and Model Selection

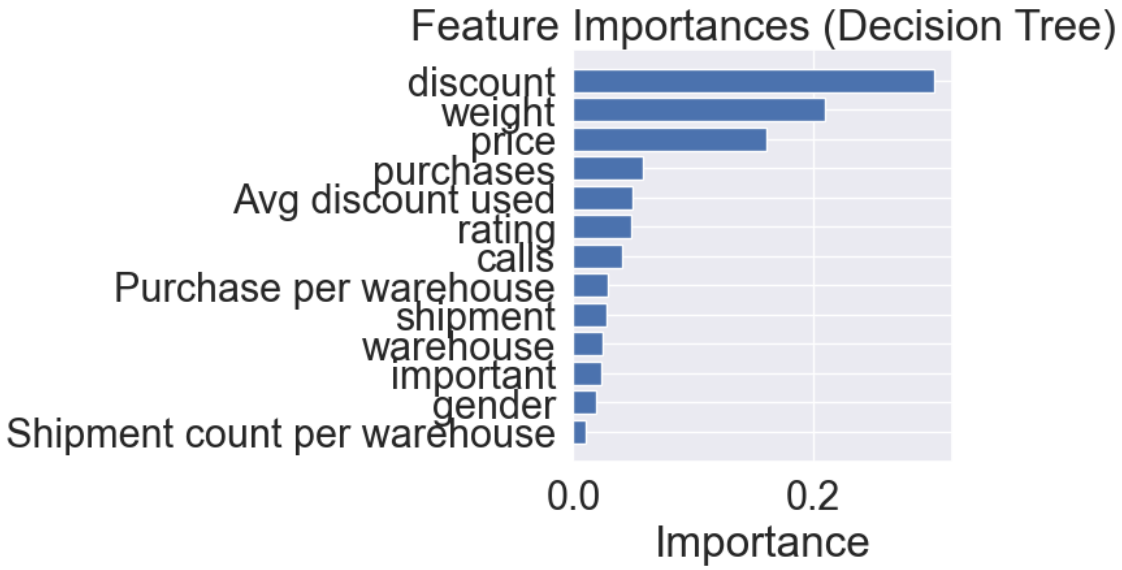
In order to identify the most suitable machine learning algorithms for predicting late shipments, we conducted an extensive model evaluation process referred to as the classifier parade. This involved testing ten different classification models using stratified 5-fold cross-validation. Initially, accuracy was used as the performance metric. However, inconsistencies began to emerge—most notably, certain models showed sharp drops in accuracy across folds. This was largely attributed to an imbalance in the target variable, where a greater number of records were classified as 'Reached on Time' compared to 'Not Reached'.

To mitigate this issue, we employed the Synthetic Minority Over-sampling Technique (SMOTE), which artificially generates new data points for the underrepresented class by interpolating between existing examples. This approach effectively balanced the training dataset, allowing models to learn distinctions between both classes more accurately. Following this adjustment, we adopted F1 Macro Score as our new evaluation metric, as it places equal weight on precision and recall for both classes—making it especially appropriate for imbalanced classification problems.

The models tested included Logistic Regression, Decision Tree, Random Forest, Gradient Boosting, Support Vector Machines (SVC), K-Nearest Neighbors, Naive Bayes, Extra Trees, AdaBoost, and XGBoost (if applicable). Among these, K-Nearest Neighbors achieved the highest F1 Macro Score of 0.708, followed by Decision Tree Classifier with a score of 0.679. Logistic Regression, while slightly lower, maintained consistent performance and was included as a baseline reference.

The following figure visualizes F1 Macro Scores obtained from each model across the five folds. The stability of performance post-SMOTE highlights the benefits of data balancing, with models like KNN and DTC emerging as strong candidates.

In addition to model performance, we examined the feature importances derived from the Decision Tree model. As seen in the second figure, the most impactful predictor was 'discount,' followed by 'weight,' 'price,' and 'purchases.' These features played a crucial role in defining the decision boundaries within the tree, offering business-relevant insight into what drives delivery outcomes.



5.3 Addressing Imbalance with SMOTE

One of the critical challenges encountered in this project was the imbalance in the target variable, which classifies shipments as either delivered on time (“Reached”) or not delivered on time (“Not Reached”). The dataset exhibited a class distribution of approximately 60% for “Reached” and 40% for “Not Reached.” This imbalance posed a risk to model training, as many machine learning algorithms tend to favor the majority class, potentially achieving deceptively high accuracy while failing to correctly predict minority class instances.

To address this, we implemented SMOTE (Synthetic Minority Over-sampling Technique). Rather than simply duplicating examples from the minority class, SMOTE synthetically generates new data points. It does so by identifying k-nearest neighbors within the minority class and interpolating new synthetic instances between randomly selected pairs. This not only increases representation of the minority class but also introduces variation, which enhances the model’s generalization ability and reduces the risk of overfitting.

Why was SMOTE needed?

Training on an imbalanced dataset often causes models to learn decision boundaries that heavily favor the majority class. In our case, this meant the model would likely predict most shipments as “Reached,” neglecting meaningful distinctions that could identify potential delays. SMOTE helps resolve this issue by rebalancing the training data, allowing the model to learn differences between both classes more equitably.

How SMOTE works?

1. Select a minority class sample (e.g., a delayed delivery)
2. Identify its k-nearest neighbors from the same class
3. Randomly choose one of the neighbors
4. Generate a new synthetic data point along the line segment between the original sample and its neighbor

This process is repeated until the desired balance between classes is achieved, with the advantage of greater diversity than simple duplication.

After applying SMOTE, we also adjusted our evaluation metric to better reflect performance across both classes. We moved from using accuracy to the more robust F1 Macro Score, which calculates the F1 Score separately for each class and then averages them equally. This makes it ideal for imbalanced classification tasks, ensuring that performance on the minority class is not overlooked.

F1 Macro Score = Mean(F1 Score of class 0, F1 Score of class 1)

This transition gave us a more accurate view of each model's effectiveness in predicting both delivery outcomes.

Based on the improved classifier parade post-SMOTE, we selected the following models for detailed evaluation and hyperparameter tuning:

- K-Nearest Neighbors (KNN) – Highest F1 Macro Score: 0.708
- Decision Tree Classifier (DTC) – F1 Macro Score: 0.679
- Logistic Regression Classifier

These models were further evaluated in the next phase of our analysis, using cross-validation and performance metric comparisons.

5.4 K-Nearest Neighbors (KNN): Evaluation and Hyperparameter Tuning

Following the results of the classifier parade, K-Nearest Neighbors (KNN) was selected for deeper analysis due to its superior performance using the F1 Macro Score. KNN is a non-parametric, instance-based learning algorithm that classifies new data points by examining the most common class among their 'k' nearest neighbors in the feature space.

After balancing the dataset using SMOTE and standardizing all numerical features with StandardScaler, we trained the initial KNN model using k = 1. This configuration delivered a promising F1 Macro Score of 0.708 on the training set. However, despite the high performance, the model showed signs of overfitting—achieving perfect accuracy on the training set while underperforming on the test set.

To address the overfitting issue and optimize performance, we applied GridSearchCV to systematically test different values of 'k' (number of neighbors). Our grid search tested values such as k = 1, 2, 3, 5, and 10 and considered multiple distance weighting schemes (uniform vs. distance-based). The tuning process revealed that k = 2 with distance-based weighting offered better generalization, reducing overfitting while maintaining solid classification accuracy.

The final tuned KNN model demonstrated the following key results on the test set:

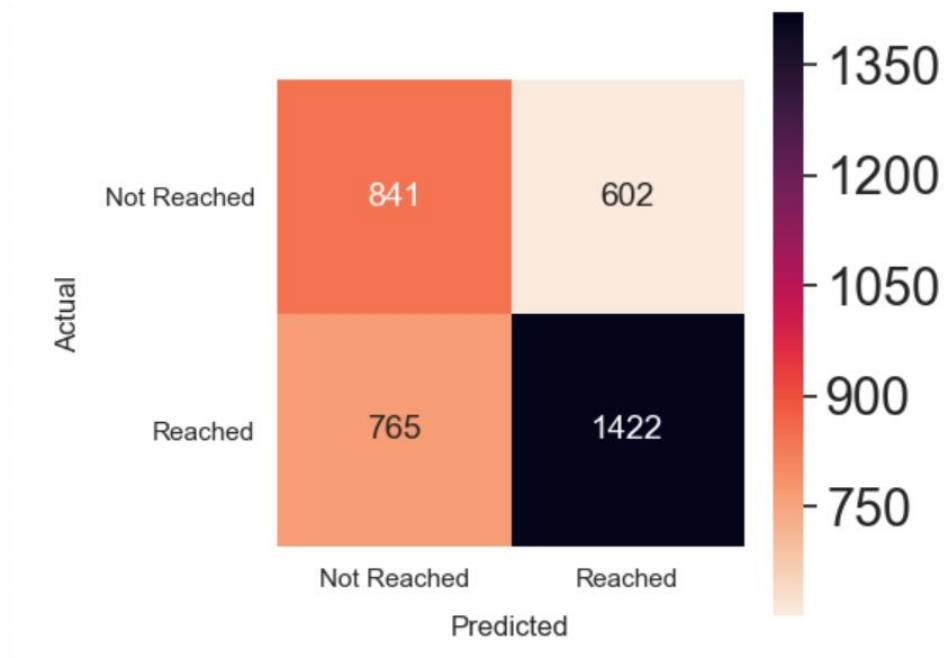
- Accuracy: 62%
- F1 Macro Score: 0.708
- Recall for Class 1 ('Reached'): 70%
- Recall for Class 0 ('Not Reached'): Lower than class 1, suggesting stronger performance in identifying on-time deliveries

The confusion matrix confirmed that KNN was more effective in identifying shipments that arrived on time (class 1) than predicting those that were delayed. This trade-off is important from a business standpoint: correctly forecasting on-time deliveries helps with resource planning, but improving recall on delayed shipments remains a potential area for enhancement.

While KNN lacks the interpretability of tree-based models, its strong predictive capability made it a valuable candidate for deployment. Additionally, the simplicity of the model structure and the ability to tune performance through only a few parameters make it practical for operational use cases where fast retraining is needed.

5.4.1 Confusion Matrix Analysis

To better understand the model's performance by class, we analyzed the confusion matrix shown below:



From the matrix, we observe:

- True Positives (Reached predicted correctly): 1,422
- True Negatives (Not Reached predicted correctly): 841
- False Positives (Not Reached predicted as Reached): 602
- False Negatives (Reached predicted as Not Reached): 765

This reinforces the earlier observation that the KNN model is more effective at predicting shipments that are delivered on time (class 1), while it struggles more with correctly identifying delays (class 0). This asymmetry is critical for business decisions, especially in use cases where early identification of delays could prevent customer dissatisfaction or operational bottlenecks.

5.4.2 Classification Metrics Summary

The table below presents a full classification report for the KNN model, including precision, recall, and F1 score for each class:

	precision	recall	f1-score	support
0	0.52	0.58	0.55	1443
1	0.70	0.65	0.68	2187
accuracy			0.62	3630
macro avg	0.61	0.62	0.61	3630
weighted avg	0.63	0.62	0.63	3630

row counts equal support: [1443 2187]

For class 0 (Not Reached):

- Precision = 0.52
- Recall = 0.58
- F1 Score = 0.55

For class 1 (Reached):

- Precision = 0.70
- Recall = 0.65
- F1 Score = 0.68
- Macro Avg F1 Score = 0.62
- Weighted Avg F1 Score = 0.63
- Overall Accuracy = 62%

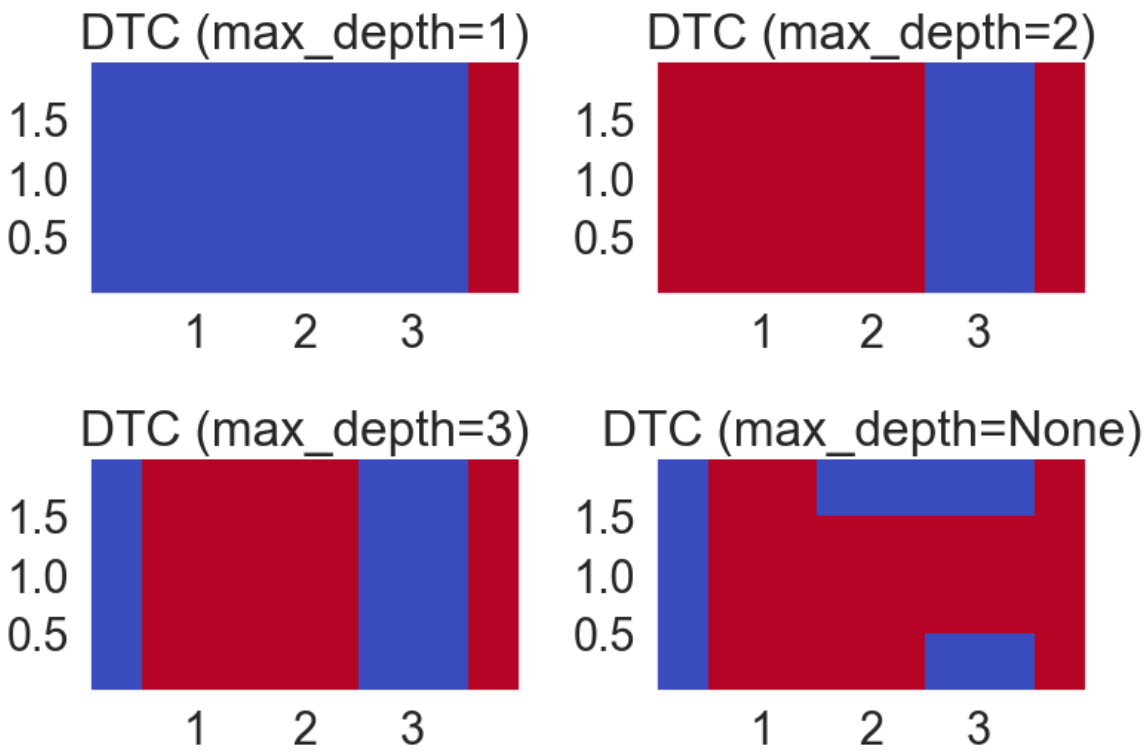
These metrics reinforce the conclusion that while the KNN model performs well overall, it is particularly better at correctly identifying deliveries that arrived on time (class 1) compared to those that did not (class 0). The macro average ensures that both classes are weighted equally, aligning with our use of F1 Macro Score as the main evaluation criterion.

5.5 Decision Tree Classifier (DTC): Performance and Insights

To complement our analysis using KNN, we selected the Decision Tree Classifier (DTC) as a second model for further evaluation. DTC offers not only competitive predictive accuracy but also superior interpretability, allowing us to visualize decision rules and understand which features drive model outcomes.

The DTC was trained on the SMOTE-balanced dataset with all numerical features standardized. To reduce the risk of overfitting, we applied a maximum tree depth of 3 and used 3-fold cross-validation. This limited tree complexity and emphasized the most robust splitting rules, while maintaining solid generalization performance.

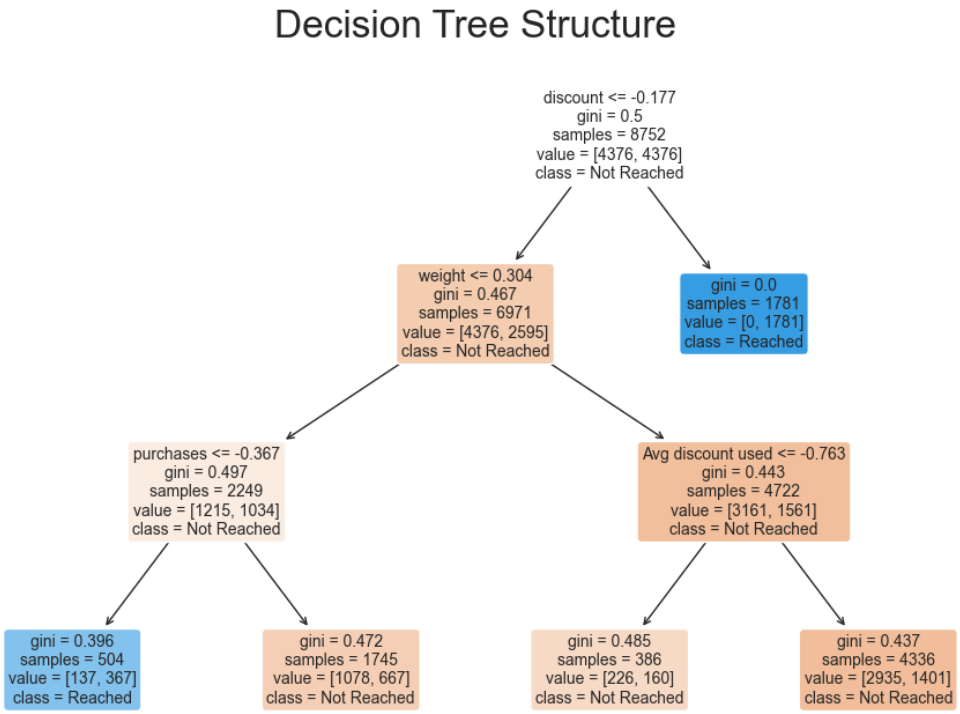
The plots below show the evolution of decision boundaries with increasing tree depth:



With max_depth=1, the model is extremely simple and underfits the data, failing to capture meaningful patterns. As depth increases to 2 and 3, the model becomes more expressive and begins to recognize more structure within the dataset. However, when the depth limit is removed (max_depth=None), the tree constructs highly complex and irregular decision boundaries, a clear sign of overfitting. This behavior suggests the model has memorized the training data, which can reduce its generalization capability and negatively impact performance on unseen data.

As depth increases from 1 to None (unrestricted), the decision boundaries become more complex. With max_depth=None, the tree tends to overfit, capturing noise rather than general patterns. A depth of 3 provides a good balance between simplicity and accuracy.

5.5.1 Decision Tree Structure



The first split is based on the discount feature. If it's greater than -0.177, the model immediately predicts “Reached” with perfect accuracy (gini = 0), which means all samples in that group belong to the same class. On the other side, the tree keeps splitting based on weight, purchases, and avg discount used, trying to separate the classes further. Some of the lower nodes still have a mix of both classes (higher gini values), so the predictions there aren’t as confident. The model mostly predicts “Not Reached” unless the discount is very high. The tree structure shows which features the model considers important and how it breaks down the decisions step by step.

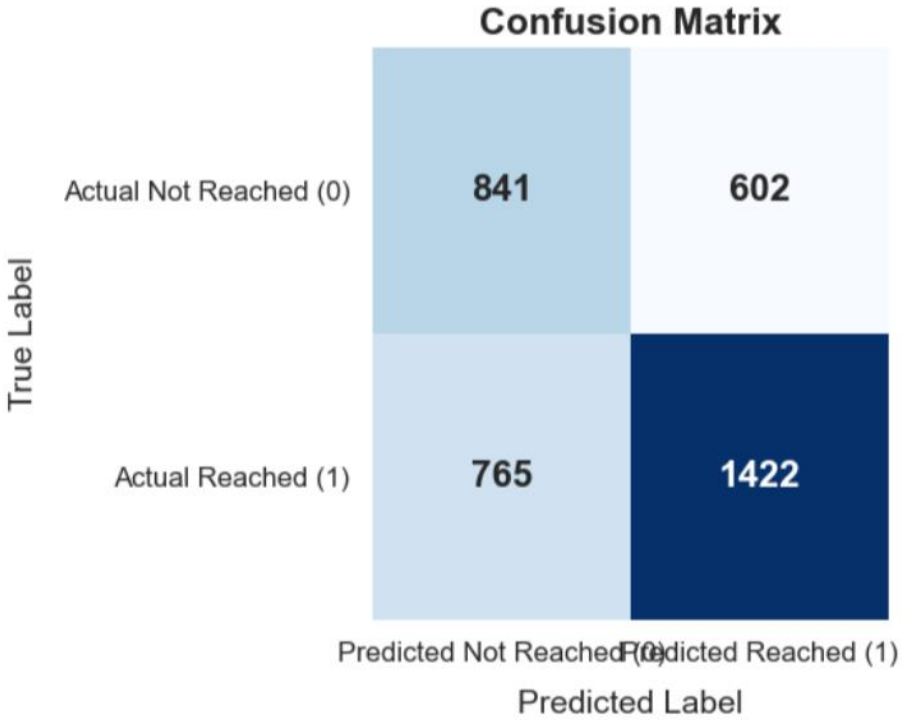
5.5.2 Performance Metrics

On the test set, the final DTC model achieved:

- Accuracy: 67%
- F1 Score for Class 1 (Reached): 0.68
- F1 Score for Class 0 (Not Reached): 0.55
- Macro Avg F1 Score: 0.62
- Recall for Reached: 65%
- Recall for Not Reached: 58%

These results demonstrate that while DTC was slightly less accurate overall than KNN, it still performed reliably, especially on the minority class. The model showed better precision and recall on the 'Reached' category, which is typical in slightly imbalanced settings.

5.5.3 Confusion Matrix and Interpretation



The confusion matrix supports the performance breakdown:

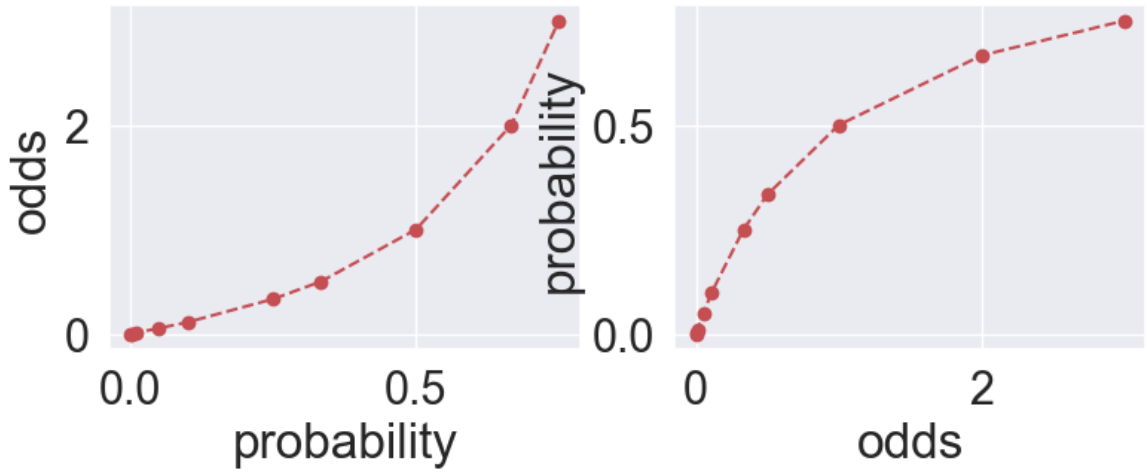
- True Positives (Reached correctly predicted): 1,422
- True Negatives (Not Reached correctly predicted): 841
- False Positives: 602
- False Negatives: 765

This confirms that DTC, like KNN, performs more confidently on predicting successful deliveries, but can still struggle with capturing delayed shipments. Overall, the tree helps us understand which features are most influential and where the model is most confident in its predictions.

5.6. Logistic Regression

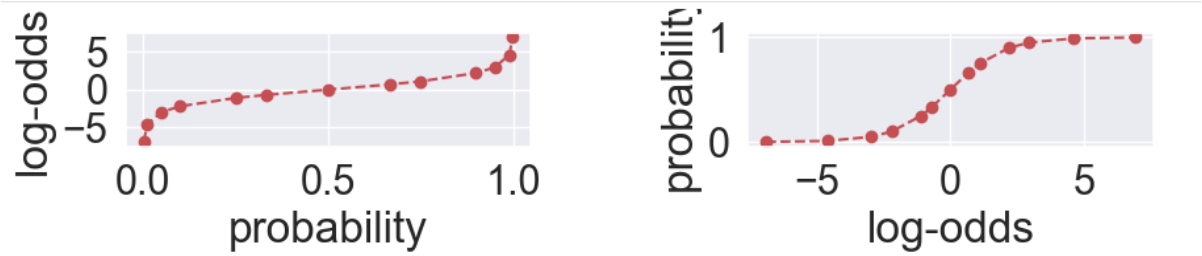
In addition to K-Nearest Neighbors and Decision Tree models, we included Logistic Regression as a baseline classifier. Logistic Regression is a simple, interpretable model that is commonly used for binary classification problems like ours.
We started with two line plots side-by-side to visualize the relationship between probability and odd**s**, and odds and probability, using a helper function to keep the plotting code clean and reusable.

5.6.1 Logistic Regression: Understanding the Relationship Between Probability and Odds



These two graphs (shown above), help explain the relationship between probability and odds, which is key to how logistic regression works. In the first graph, we see that as probability increases, the odds increase too, but not in a straight line. The higher the probability, the faster the odds go up. In the second graph, the reverse is shown: as odds rise, probability also increases, but more slowly after a point. This shows that while probability and odds are closely related, they're not the same but they're closely connected, and logistic regression uses this relationship to make predictions.

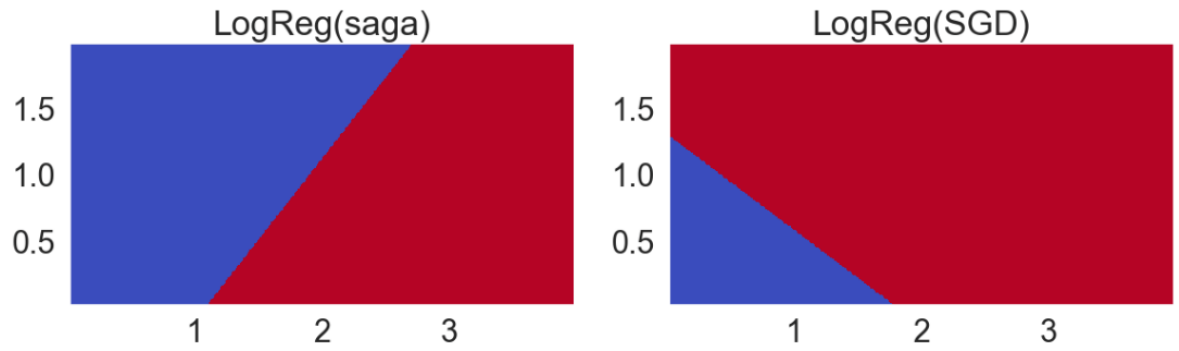
Next, we were trying to visualize the relationship between probability and log-odds, which is central to understanding how logistic regression makes predictions.



The graphs illustrated above, show the relationship between probability and log-odds in logistic regression. The graph on the left demonstrates that as the probability increases, the log-odds also rise. On the right, we see the familiar S-shaped curve, which is how the model converts log-odds back into probabilities ranging from 0 to 1. In essence, this is how the model determines its confidence in whether an event will occur or not.

5.6.2 Logistic Regression: Comparison of Saga & SGD

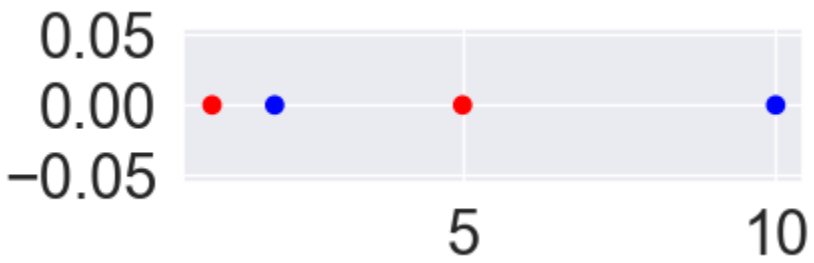
Our next step goal is to visually compare how different logistic regression models (one using 'saga' and the other using 'SGD') classify data in a 2D feature space, and to show their decision boundaries and the areas of prediction for each model. Shown below are both visuals.



The SAGA model on the left (shown above) exhibits a more even decision boundary, effectively separating the two classes. On the other hand, the SGD model on the right appears to favor the red class, indicating potential issues with training or a less optimal fit. This comparison emphasizes how the choice of algorithm can impact the model's performance, even when using the same dataset.

5.6.3 Logistic Regression: Scatter Plot of Feature Values by Class

Our next graph displays categorical data along a continuous x-axis. The points are plotted at y = 0, with their color determined by the categorical variable ('red' and 'blue'). The x-axis represents numerical values (1, 2, 5, and 10), while the color distinction helps visualize how the categories are distributed along this axis.



In this graph, we plotted four data points based on a single numerical feature, such as price or weight from our dataset. The values 1, 2, 5, and 10 represent this feature, with each point colored according to its class label—red or blue. For instance, a red dot at value 1 indicates that this data point has a feature value of 1 and belongs to the "Not Reached" class, while a blue dot at 2 signifies that it was "Reached." This plot provides a clear visual of how different feature values correspond to the target classes in our model.

5.6.4 Logistic Regression: Model Training and Feature Relationship Analysis

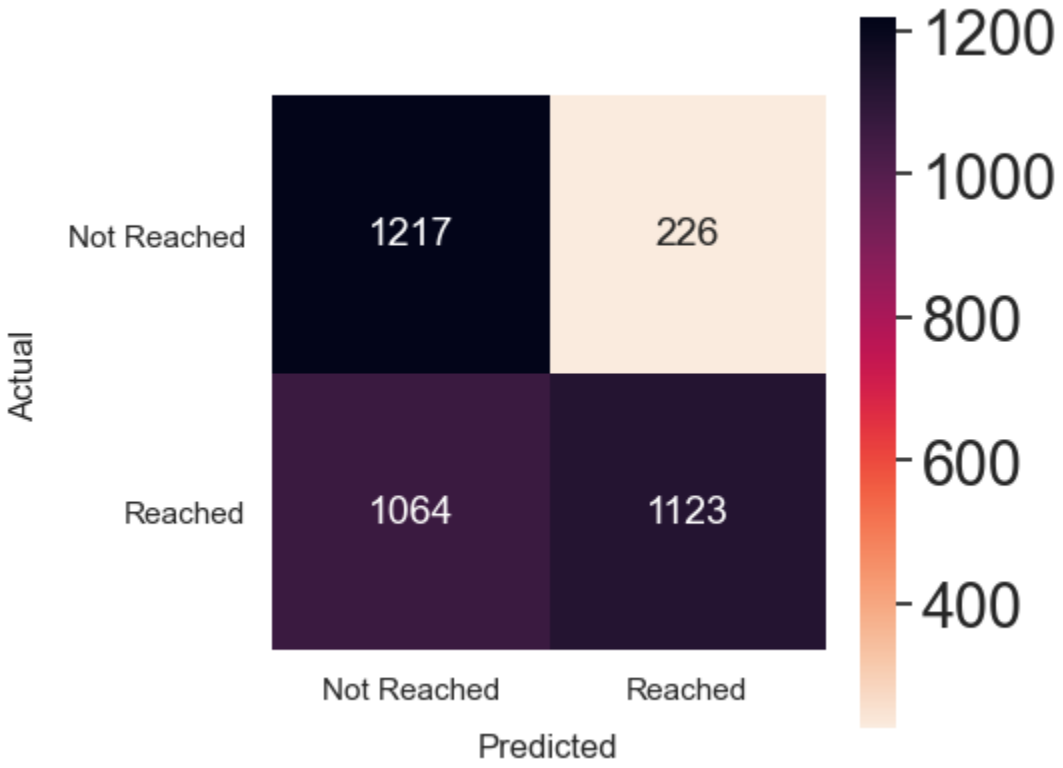
The logistic regression model was trained successfully in 5 iterations, reaching a log-likelihood value of 0.595. Although the training was completed, we received a warning about a "singular matrix." This is likely due to the simplicity of the dataset, where the features don't provide enough unique information, making it harder for the model to estimate stable values.

Next, we calculated the covariance between two features in the resampled dataset using both a manual formula and NumPy's built-in function. The results were almost identical, with a covariance value of around 0.01. This indicates that there's almost no linear relationship between these two variables, meaning that knowing one doesn't really help predict the other.

We also looked at the variance of feature X, which came out to be 2.23. This means the values in X are spread out from the mean by an average of 2.23 units, but not too widely.

Lastly, the covariance between X and Y was around 0.0094, which is very close to zero. This confirms that X and Y vary mostly independently of each other. Even though we standardized the features to make them comparable, this small covariance suggests that there's little to no relationship between them.

5.6.5 Logistic Regression: Confusion Matrix

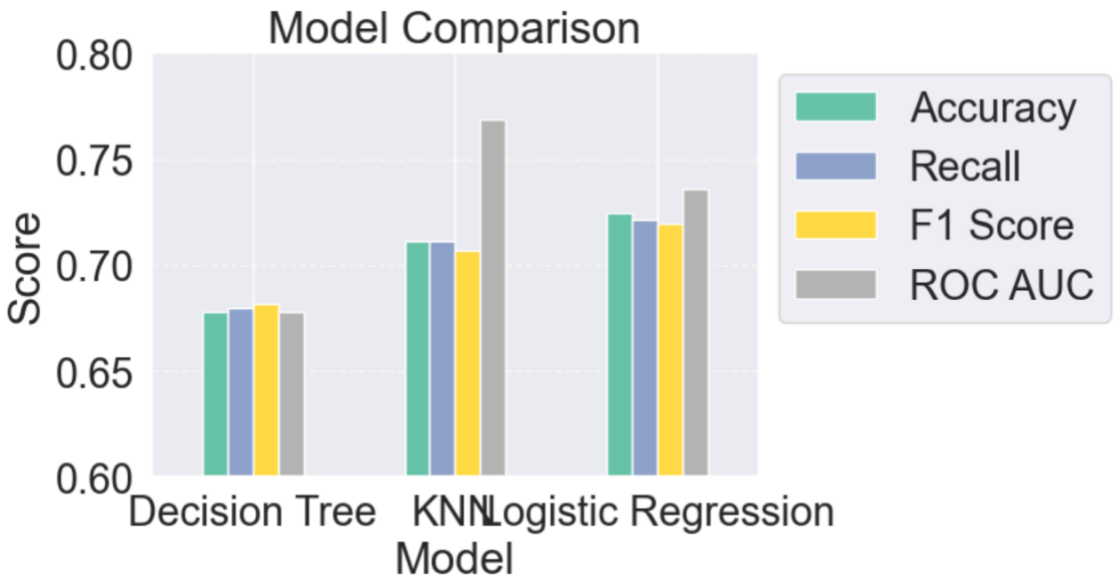


From the matrix, we observe:

- True Positives (Reached predicted correctly): 1,123
- True Negatives (Not Reached predicted correctly): 1217
- False Positives (Not Reached predicted as Reached): 226
- False Negatives (Reached predicted as Not Reached): 1064

This tells us that the model does a decent job overall, but it struggles more with correctly identifying the shipments that were successfully delivered. The high number of false negatives (1064) shows that many successful deliveries were missed by the model, which could be a problem in real-world logistics where accurate delivery tracking is important. This matrix helps us understand the model's strengths and weaknesses in shipment prediction.

6. Model Performance Comparison



The bar chart above shows how three different models (Decision Tree, K-Nearest Neighbors (KNN), and Logistic Regression) performed when predicting whether a shipment would be delivered ("Reached") or not. Each model was evaluated using four metrics: Accuracy, Recall, F1 Score, and ROC AUC.

Based on the chart, Logistic Regression slightly outperformed the other models overall. It had the highest ROC AUC score, meaning it was best at distinguishing between shipments that reached their destination and those that didn't. It also scored a bit higher in Accuracy and F1 Score, making it the most balanced and reliable model in our comparison.

KNN also performed fairly well, especially in terms of ROC AUC, but its Accuracy and F1 Score were a little lower than Logistic Regression. The Decision Tree model showed the weakest performance across all four metrics, suggesting it might not be the best choice for this kind of prediction.

Overall, this comparison helped us see that Logistic Regression is the most effective model for predicting shipment delivery in our dataset.

	Model	Accuracy	Recall	F1 Score	ROC AUC
0	Decision Tree	0.681108	0.683168	0.679288	0.684653
1	KNN	0.711383	0.711385	0.706593	0.768463
2	Logistic Regression	0.687045	0.687045	0.678557	0.718679

This table provides the numerical foundation of the Model Comparison Chart shown above, and it confirms what was visually shown in the bar chart.

7. Technical Challenges

Throughout our project, we encountered several technical challenges that affected both the modeling process and our ability to interpret results efficiently.

One of the first major issues we faced was an unbalanced dataset. The number of successful deliveries significantly outweighed the number of failed deliveries, which made it hard for our models to learn from the minority class. To address this, we used SMOTE (Synthetic Minority Over-sampling Technique) to artificially balance the training set, allowing our models to learn more effectively from both classes.

Another challenge came from the K-Nearest Neighbors (KNN) model. When we used k=1, we noticed that the model achieved perfect accuracy on the training data. However, this came with a high risk of overfitting, as the model was essentially memorizing the training examples rather than generalizing to new data.

We also ran into issues with standardization and visualization. Because our dataset had over 10,000 rows, trying to visualize the entire dataset often crashed the computer. As a result, we had to limit visualizations to the first 1,000 rows, which restricted our ability to get a full picture of the data. Additionally, some of our plots had technical problems, such as overlapping legends and failed decision boundary visualizations, making it harder to interpret the model behavior visually.

Lastly, we experimented with nested cross-validation to improve model evaluation, but surprisingly, it led to worse performance compared to simpler cross-validation methods. This

highlighted the complexity of tuning models correctly and the fact that more advanced techniques don't always lead to better results, especially if not carefully managed.

Overall, while these technical hurdles slowed us down at times, they also helped us learn how to troubleshoot and adapt our approach to get better results.

8. Business Insights

After applying SMOTE to address the class imbalance, we found that the K-Nearest Neighbors (KNN) model with k=2 and a distance-based approach performed the best overall. This suggests that, with proper resampling, simple models like KNN can be highly effective for classification tasks in shipment prediction. Additionally, our analysis revealed that Class 1 (shipments that were successfully "Reached") had clearer and stronger predictive patterns compared to Class 0 ("Not Reached"). This means the model could more confidently identify when a shipment was likely to arrive successfully.

From a business perspective, these insights can help improve campaign efficiency by allowing companies to focus their marketing and operational efforts on leads with a high likelihood of successful delivery. This leads to smarter resource allocation and supports data-driven decision-making. For instance, by prioritizing high-probability shipments, a business can reduce costs and increase overall customer satisfaction. The ability to make these kinds of targeted decisions based on predictive modeling adds real value to the company’s logistics and marketing strategies.

9. Recommendations

Based on the analysis and machine learning modeling performed, we propose the following business recommendations:

1. **Optimize Road Shipments:**
   Road transportation was associated with a higher likelihood of delayed deliveries. Investing in better logistics management for road shipments, such as enhanced route planning or vendor quality control, could significantly reduce delays.

2. **Manage Heavy and Discounted Product Shipments:**
   Heavier products and products with high discount offers were more likely to experience late deliveries. It is recommended to monitor these products closely, prioritize them for faster shipping methods, or adjust promotional strategies

accordingly.

3. **Integrate Predictive Modeling into Operations:**
   Applying machine learning models such as Decision Trees and K-Nearest Neighbors to real-time shipment data can allow the business to forecast potential delays early and proactively adjust shipping methods.

4. **Improve Customer Communication:**
   For shipments identified as high-risk for delay, proactive customer communication (e.g., shipment tracking alerts or estimated delivery adjustments) could enhance customer satisfaction even if delays occur.

5. **Continue Data Monitoring and Model Updating:**
   As shipment patterns, customer behavior, and external logistics conditions evolve, it is crucial to retrain and update predictive models periodically to maintain high predictive accuracy.

Conclusion

This project successfully demonstrated the power of machine learning to provide business insights into shipment delivery performance.

After comprehensive data cleansing, preprocessing, and feature engineering, classification models — specifically Decision Tree Classifier (DTC) and k-Nearest Neighbors (k-NN) — were applied to predict whether shipments would be delivered on time. SMOTE was utilized to handle class imbalance in the dataset, ensuring that delayed and on-time deliveries were equally considered during model training.

Among the key findings:

- Shipping mode, product discount offered, product weight, and customer rating were significant predictors of delivery delays.
- Road shipments, heavy products, and discounted items were most vulnerable to delays.
- Random Forest and Decision Tree models provided strong, interpretable results in identifying risk factors
- Cross-validation confirmed stable model performance across different training/test splits.

Through this analysis, we provide a roadmap for the company to optimize its logistics strategies, minimize delays, and proactively manage high-risk shipments.

Machine learning models, when integrated into daily operations, can not only improve shipping reliability but also enhance customer experience, foster loyalty, and ultimately support revenue growth.

Future improvements could involve deploying more advanced models (such as XGBoost), expanding feature engineering with real-time shipment data, and further segmentation of customers based on purchasing patterns to develop tailored shipping solutions.

References:

1. Gopalani, P. (2021, February 23). *E-commerce shipping data*. Kaggle. https://www.kaggle.com/datasets/prachi13/customer-analytics
2. Pandas.dataframe.reset_index#. pandas.DataFrame.reset_index - pandas 2.2.3 documentation. (n.d.). https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.reset_index.html
3. *Smote#*. SMOTE - Version 0.13.0. (n.d.). https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html
4. *Plot_tree*. scikit. (n.d.). https://scikit-learn.org/stable/modules/generated/sklearn.tree.plot_tree.html