Enhancing E-commerce Shipping Efficiency through Machine Learning Insights

Team members:

1. Fatima Aghapourasl
2. Kareem Albeetar
3. Chun Lam Cheung
4. Penelope De Freitas
5. Eltigani Hamadelniel Elfatih Hamadelniel
6. Yi Liu
7. Jeffrey Ching Lok Ng
8. Ivy Tsai

Table of Contents

Problem Statement

This report outlines the process that our team took to apply mathematical concepts and machine learning techniques to study **e-commerce shipping data**[1].

Our primary objectives were to:

- explore, preprocess, and clean the dataset
- create an experimentation plan (base, feature selection, and hypothesized model testing)
- build and tune the comparative models (Logistic Regression, Decision Tree, Random Forest, SGD, and SVM algorithms)
- evaluate and discuss our observations

The models were used to predict whether or not a product would **reach customers on time** (target) based on various features such as mode of shipment, weight, and cost of the product.

[1] Prachi Gopalani, n.d., E-Commerce Shipping Data <https://www.kaggle.com/datasets/prachi13/customer-analytics>

Experiment Methodology

Data Overview

An **e-commerce shipping dataset**, which can be accessed via Kaggle[1], was used for this project. The dataset comprised **10,999 observations** and **12 variables** ('ID', 'Warehouse_block', 'Mode_of_shipment', 'Customer_care_calls', 'Customer_rating', 'Cost_of_the_product', 'Prior_purchases', 'Product_importance', 'Gender', 'Discount_offered', 'Weight_in_grams', 'Reached_on_Time'). The chosen target variable was 'Reached_on_Time' due to its discrete representation (1 or 0).

Data Exploration, Preprocessing and Cleaning

Our group applied several techniques during this stage:

1. Examined the dataset for missing values: None were found.
2. Dropped irrelevant columns: The 'ID' column, which is essentially an identifier, was dropped since it does not provide meaningful information for predictions.
3. Encoded Categorical variables:
    - The categorical variable 'Warehouse_block' was one-hot encoded to convert them into a suitable format for machine learning models.
    - Ordinal encoding was applied to other categorical variables: 'Product_importance', 'Mode_of_shipment,' and 'Gender'.
4. Scaled features: Numerical features were scaled using Min-Max Scaler to ensure that all features contribute equally to the model.
5. Outliers: For handling outliers, we used boxplots (*Figure 1*) and z-score. We found two variables ('Prior_purchases' and 'Discount_offered') with outliers and took a decision to drop all observations corresponding to these outliers, since we did not have sufficient contextual information about their importance.

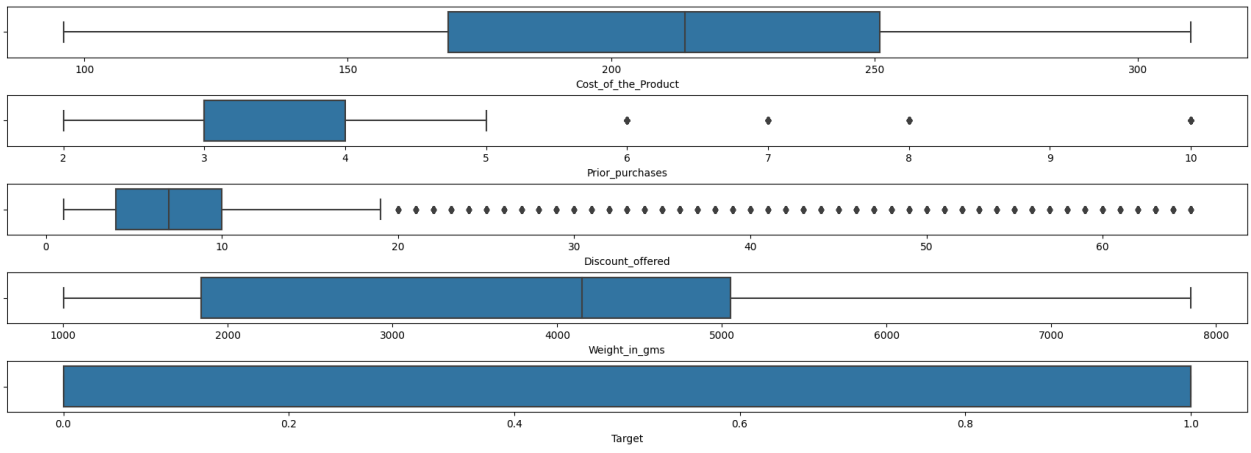

***Figure 1**. Outliers revealed in the dataset using box-plots*

6. Exploratory Data Analysis (EDA): Exploratory data analysis was conducted to understand the distribution of key features and relationships between variables. Histograms (*Figure 2*) and a correlation heatmap (*Figure 3*) were created to visualize the data.

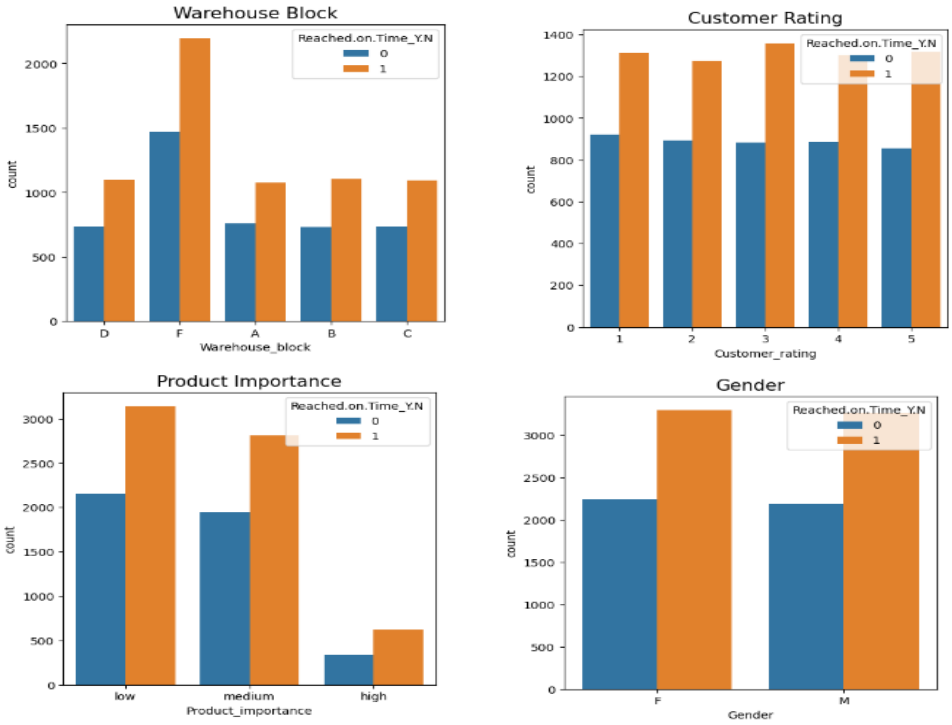

Figure 2. *Sample histograms showing variables with similar relationship pattern to the Target ('Reached_on_Time')*

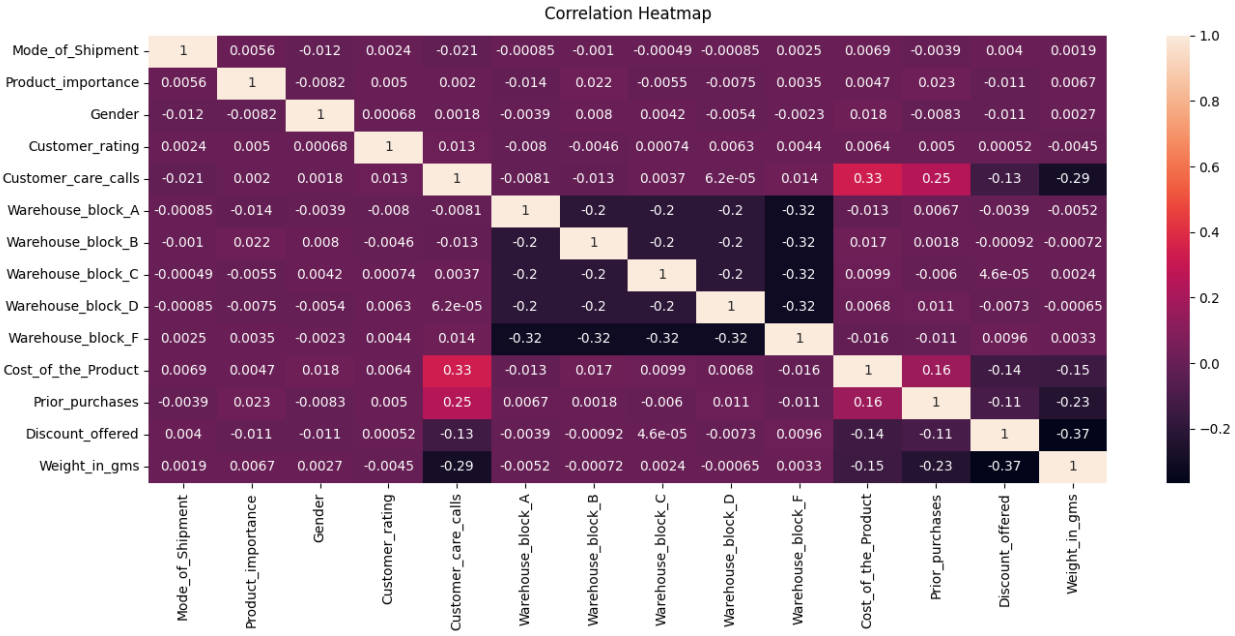

Figure 3. *Correlation Heatmap of the variables in the E-commerce dataset*

Comparative Models in the Study

For this study we worked with three models (see *Figure 5* for their specific features):

- **Base dataset**: comprised all features retained from the pre-processed stage (cleaned data)
- Algorithmic **Feature Selection model dataset**: retained only the top features (*Figure 4*) selected by Pearson, Chi-2, BFE, RFE, Random Forest, LightGBM

	Feature	Pearson	Chi-2	BFE	RFE	Random Forest	LightGBM	Total
1	Weight_in_gms	True	True	True	True	True	True	6
2	Cost_of_the_Product	True	True	True	True	True	True	6
3	Discount_offered	True	True	True	True	True	False	5
4	Warehouse_block_D	True	True	True	True	False	False	4
5	Warehouse_block_A	True	True	True	True	False	False	4
6	Product_importance	True	True	True	True	False	False	4
7	Prior_purchases	True	True	True	True	False	False	4
8	Mode_of_Shipment	True	True	True	True	False	False	4
9	Customer_care_calls	True	True	True	True	False	False	4
10	Warehouse_block_B	True	True	False	True	False	False	3
11	Gender	True	True	True	False	False	False	3
12	Customer_rating	True	True	False	True	False	False	3
13	Warehouse_block_F	False	False	True	True	False	False	2
14	Warehouse_block_C	False	False	True	False	False	False	1

Figure 4: *Top features ranked by several feature selection algorithms*

- **Hypothesized model dataset**: dropped features based graph analysis (*Figures 2 and 3*) and conscious reasoning (i.e., 'Gender', 'Product_importance', 'Customer_rating', 'Warehouse_block').

Features	**Base Model**	**Feature Selection model**	**Hypothesized model**
Warehouse_block (A, B, C, D, F) **Note:** Warehouse_block E does not exist.	✔	✔ Only warehouse_block (A, B, D) ✘ warehouse_block (C, F)	✘
Mode_of_Shipment	✔	✔	✔
Customer_care_calls	✔	✔	✔
Customer_rating	✔	✘	✘
Cost_of_the_Product	✔	✔	✔
Prior_purchases	✔	✔	✔
Product_importance	✔	✔	✘

Gender	✔	✔	✘
Discount_offered	✔	✔	✔
Weight_in_gms	✔	✔	✔

Figure 5: *A comparison of the features by each model*

Evaluation Metrics

- The primary evaluation metrics used were accuracy, F1-score, and the confusion matrix.
- ROC AUC score and feature importance were also evaluated in some of the classifiers.

Summary of Findings

In this section, we present and discuss the key results and observations from running our fine-tuned models: Logistic Regression, Decision Tree, Random Forest, SGD and SVM.

Logistic Regression

To start, we used the base model with the default parameters of the logistic regression model. In order to test both l1 and l2 regularization, we changed the solver to use liblinear. The rest of the default parameters were: C = 1 (Regularization term), penalty = l2. We then used GridSearchCV (GSCV) to tune the hyper parameters with the parameter list as shown below. The results were as follows:

Param List = C = 0.1, 0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75, 2 Penalty = l1, l2	Base Model	Base Model (GridSearchCV)
(Selected) Params	C = 1, Penalty = l2	C = 0.25 Penalty = l2
Number of Features	11	11
Accuracy Score	0.6387	0.6418
F1 Score (weighted)	0.6401	0.6418
Precision (weighted)	0.6421	0.6436
Recall (weighted)	0.6388	0.6418

Figure 6. *All scores of base model*

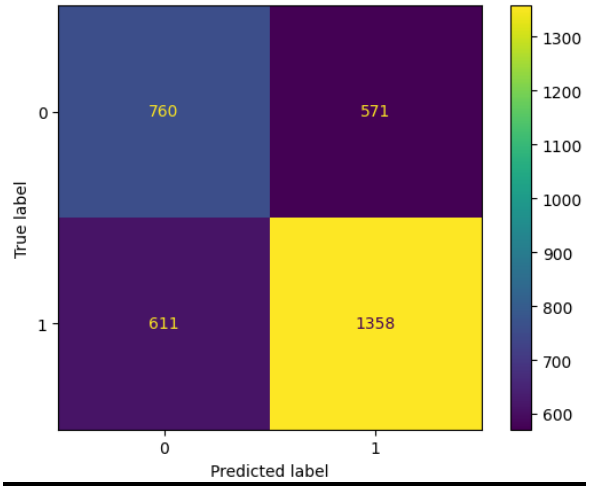

Figure 7. *Confusion Matrix of base model with GSCV*

These scores were used as a baseline to test how well the model performed versus our other models (feature selection and hypothesized). It was interesting to see that throughout all the different scoring metrics the base model performed around 64%.

Next we tested the feature selection model. We started with the base logistic regression model as seen above then used GridSearchCV to fine tune the hyperparameters. The best estimator parameters were C=0.25 and penalty=l2. The scores are shown here:

Param List = C = 0.1, 0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75, 2 Penalty = l1, l2	Algorithm Based Feature Selection Model	Algorithm Based Feature Selection Model (GridSearchCV)
(Selected) Params	C = 1, Penalty = l2	C = 1.25 Penalty = l2
Number of Features	11	11
Accuracy Score	0.6264	0.6264
F1 Score (weighted)	0.6287	0.6288
Precision (weighted)	0.6336	0.6338
Recall (weighted)	0.6264	0.6264

Figure 8. *All scores of Algorithm model*

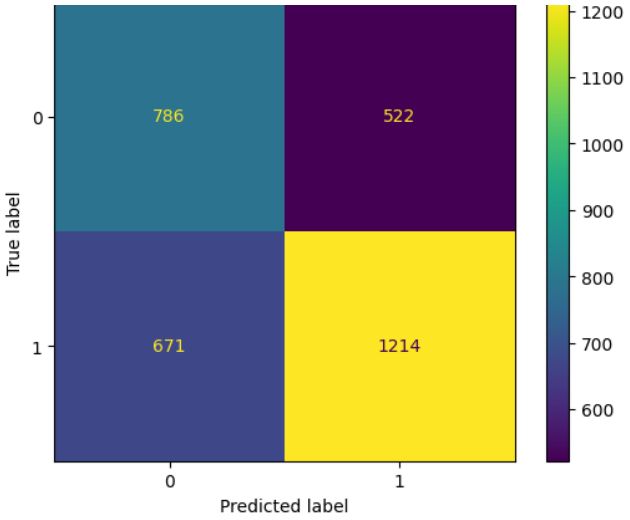

Figure 9. *Confusion Matrix of Algorithm model with GSCV*

As you can see the model's performance got worse overall, but did have a better time classifying true negatives. It seems that since the accuracy worsened it is underfitting now so the algorithm may not have chosen all the best features or the dataset does not have the right features to perform well.

Next we used the hypothesis based feature selection and then performed GSCV on it. The GSCV chose C=0.1 and penalty=l2 as the best hyper parameters. Here are the results:

Param List = C = 0.1, 0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75, 2 Penalty = l1, l2	Hypothesis Model	Hypothesis (GridSearchCV)
(Selected) Params	C = 1, Penalty = l2	C = 0.1 Penalty = l2
Number of Features	6	6
Accuracy Score	0.6273	0.6301
F1 Score (weighted)	0.6297	0.6311
Precision (weighted)	0.6347	0.6324
Recall (weighted)	0.6273	0.6301

Figure 10. *All scores of Hypothesis model*

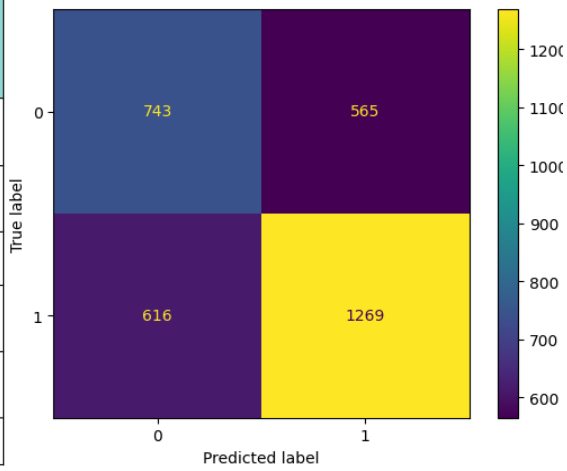

Figure 11. *Confusion Matrix of Hypothesis model with GSCV*

The hypothesis model actually beat the algorithmic feature selection model in all metrics, but still performed worse than the base model. This led us to believe that the model may be underfitting.

If we look at the coefficient weights of all the models, we see that every model is dominated by 2 features, 'Discount_offered' and 'Weight'.

	GridSearchCV (Base)	GridSearchCV (Algorithm)	GridSearchCV (Hypothesis)
Coefficients	array([[-0.67986412, 0.07207639, -0.30956858, -0.59510107, 0.06726261, 0.04928306, **5.35931639**, **-1.60595569**, 0.18402388, 0.15372334, 0.17328278, 0.08110743, 0.14497517, 0.03591874, 0.1422461 , 0.10678256]])	array([[**-1.69824634**, -0.35146997, **6.12109728**, -0.0550555 , 0.01728084, -0.02226871, -0.13604595, -0.67718927, 0.00752753, -0.67804766, 0.05887259]])	array([[0.03339014, -0.58967661, -0.29331266, -0.58422187, **4.09155515**, **-1.55534859**]])
Most Important Feats	1. Discount offered 2. Weight	1. Discount offered 2. Weight	1. Discount offered 2.Weight

Figure 12. *Coefficients of all the models*

To see if the model was really underfitting and not just a poor choice in features, we ran an exhaustive search for the best features and ended up with only marginally better scores of about 1%. This confirmed that Logistic Regression was underfitting the data.

Decision Tree

Applying the same approach as outlined above, we started with initializing two models with default parameters to measure the baseline performance of the decision tree algorithm on both datasets (feature selection and hypothesized) without any hyperparameter tuning.

Metrics	Feature Selection Model		Hypothesized Model	
	Training	Testing	Training	Testing
Accuracy (%)	100	63	100	64
F1 score	1	0.63	1	0.64
ROC score	1	0.62	1	0.62

Figure 13*. Feature selection vs. Hypothesized without tuning*

As observed from *figure 13*, we can interpret that both models produced the same score on both of the datasets thus showing that the difference in feature selection had no influence on the baseline performance of the models. Additionally, we can observe that both decision tree algorithms were overfitting to the training data and unable to generalize well to the testing data, resulting in significantly lower testing accuracy.

Parameter Grid				
Criterion	**Splitter**	**Min Samples Split**	**Max Features**	**Max Depth**
Gini	Best	2	None	None
Entropy	Random	5	Auto	3
Log loss	-	7	Sqrt	5

***Figure 14**. Hyperparameter grid for tuning the Decision Tree*

GridSearchCV was used to tune various parameters of the decision tree model namely: the function to measure the quality of the split (criterion), the strategy used to choose the split at each node (splitter), the maximum depth of the tree (max_depth), the minimum number of samples required to split an internal node (min_samples_split) and finally the number of features to consider when looking for the best split (max_features). Through this, we were able to test out a total of 270 different hyperparameter combinations for each dataset to examine the impact of different hyperparameters and features selected on the performance of the model.

Average Accuracy (%)	
Feature Selector	Hypothesis
64	64.5

***Figure 15**. Feature selector vs. hypothesis model accuracies*

Compiling all the results of these combinations together, we calculated the average accuracy score for all models trained on the feature selector dataset comparing them against those obtained with the hypothesis dataset. A slight improvement in the performance of the models, trained on the hypothesis dataset, can be observed in the table above.

Metric	Training	Testing
Accuracy (%)	68	66
F1 Score	0.67	0.65
ROC score	0.75	0.71
Best Parameters	**Criterion**: log_loss, **Splitter**: Best, **min_samples_split**:2,**max_features**: sqrt, **max_depth**:5	

***Figure 16**: Parameters and evaluation metrics for the best decision tree model*

After all our experiments our resultant best decision tree model had a depth of 5 and 53 internal nodes. We can observe that limiting the depth of the tree prevented the algorithm from overfitting to the training data thus allowing it to better generalize to the testing set resulting in a better performance overall in the testing dataset. Moreover, using log loss as the criterion function to measure the accuracy of predicted probabilities when selecting splits offered better performance when compared to the default Gini criterion technique.

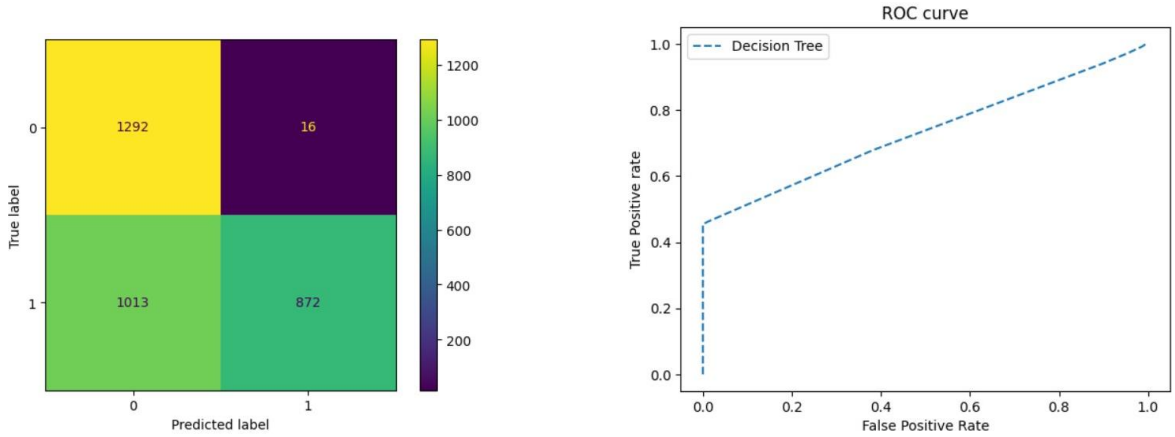

***Figure 17**: Confusion Matrix & ROC Curve for the best model*

Moreover, as observed in Figure 17 our best decision tree model has a significantly lower false positive rate and a higher false negative rate.

Random Forest

The RandomForestClassifier model was trained using our 3 datasets. Hyperparameters (number of estimators, max features, minimum samples split, max depth, Gini and entropy) were used to optimize performance.

Criterion	n_estimator	Min Samples Split	Max Features	Max Depth
Gini	100	2	None	3
Entropy	150	4	Auto	5
-	-	6	Sqrt	8

Figure 18. *Random Forest hyperparameters*

Metric	Training	Testing
Accuracy (%)	69	67
F1 Score	0.68	0.66
ROC score	0.79	0.73
Best Parameters	**Criterion**: Entropy, **n_estimators**: 100, **min_samples_split**:2, **max_depth**:5	

Figure 19. *Parameters and evaluation metrics for the best Random Forest model*

The model's accuracy on the test dataset was 67.8% with the feature selection tool and 67.9% with the hypothesized feature drop. This indicates that the hyperparameter tuning increased the model accuracy. Feature importance analysis was conducted to determine the most influential features in predicting customer ratings. The top features contributing to the model's predictions were 'Discount_offered', 'Weight_in_grams', 'Prior_purchases', 'Cost_of_the_product' and 'Customer_care_calls'.

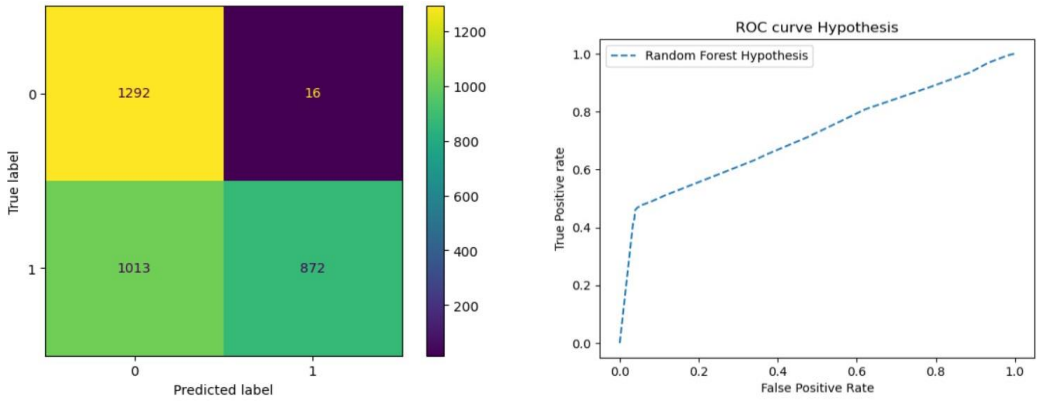

Figure 20. *Confusion Matrix & ROC Curve for the best Random Forest model*

Stochastic Gradient Descent (SGD)

GridSearchCV was used to tune the loss and alpha hyperparameters for the SGD Classifier. The loss parameter represents the loss function used during model training. As for alpha, it is the constant that multiplies the regularization term. Our group ran experiments using 'hinge', 'log_loss', and 'modified_huber' for the loss parameter, whereas for alpha we used values of 0.0001 and 0.001. All combinations of the tuned hyperparameters were then trained and tested on three datasets: base, feature selection, and hypothesis.

According to *Figure 21. SGD Results*, the best result was the hypothesized model with the best hyperparameter combination of loss: 'hinge' and alpha: 0.001. It had the highest accuracy of 65.64% and second best F1 score of 0.67 out of all three models.

The best hyperparameter combination - 'hinge' for loss and alpha: 0.001, was not surprising. With 'hinge' giving a linear SVM, it performed better than 'log_loss'- the logistic regression (this has been tested multiple times in logistic regression and SVM sections). As for 'modified_huber' it is normally best for outlier tolerance, but the outliers had been eliminated for all datasets; therefore, it was not suitable in this case. Last but not least, the higher the alpha value the stronger the regularization, thus the 0.001 parameter had better performances.

However, the feature selection model showed only slim differences in accuracy and F1 scores when compared to the hypothesized model. It is hard to determine that the methodology used for selecting features in the hypothesized dataset was better since there was barely any significant difference. We recommend that a better feature selection solution be constructed for accuracy and F1 score improvements.

Metrics	Base Model	Feature Selection Model	Hypothesized Model
Accuracy	61.23%	65.21%	65.64%
F1 score	0.6983	0.6645	0.6742
Classification Matrix	[[522 786] 452 1483]]	[[982 326] 785 1100]]	[[961 347] 750 1135]]
Best param_loss (hinge, log_loss, modified_huber)	log_loss	hinge	hinge
Best param_alpha (0.001, 0.0001)	0.001	0.001	0.001

Figure 21. *SGD Results*

Support Vector Machine (SVM)

This section shows the outcomes of using Support Vector Machines (SVMs) for classification tasks on the e-commerce shipping dataset. Initially, we used the Support Vector Classifier (SVC) with default hyperparameters for the 3 models (base, feature selection and hypothesized models), and then tuned them with GridSearchCV afterward.

According to *Figure 22*, the base has a better performance than feature selection models reached. On the other hand, the hypothesized model outperformed them, earning the greatest accuracy rate of 66.39%.

The F1 score varied significantly across the models. The feature selection model performed the best overall, with a 63% and 66% prediction rate for 1 and 0 classes. The hypothesized model, on the other hand, favored class 0 with a 70% prediction rate, while the basic model supported class 1 with a comparable 70% prediction rate. This observation underscores the different tendencies of these models towards class distribution.

The confusion matrix analysis demonstrated a trend toward fewer false positives for both the feature selection and the hypothesized model. This decrease indicates that the model's ability to correctly categorize cases belonging to the positive class improved.

Metrics	Base Model	Feature Selection Model	Hypothesized Model
Accuracy	65.89%	64.79%	66.39%
F1 score(on 0/1)	60% 70%	63% 66%	70% 62%
ROC AUC Score	71.81%	74.63%	74.63%
Confusion Matrix	[806, 502] [587, 1298]	[968, 340] [784, 1101]	[1241, 67] [1006, 879]

Figure 22. *SVC Results (base, feature selection and hypothesized models)*

When we looked at the ROC AUC scores, we saw that both the feature selection and the hypothesized model performed considerably better, with similar values that outperformed the base model. These measurements demonstrate the former two models' higher scores in the classification.

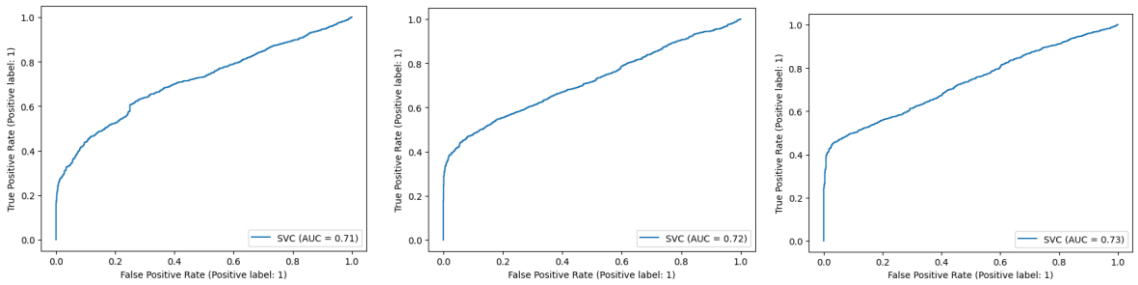

Figure 23. *SVC ROC AUC Curve from left to right (base, feature selection, hypothesized model from left to right)*

To further optimize the SVM models, we applied GridSearchCV to the hypothesized and the feature selection models. We tuned hyperparameters (see *Figure 24*) such as kernel type, gamma, C (regularization parameter), and class weight, which are around 210 candidates. The best parameters obtained are showcased in *Figure 25*.

```
params= {'C': [0.001,0.05 ,0.1, 0.5, 1, 10, 100 ],
         'kernel': ['linear', 'rbf','sigmoid'],
         'gamma':['scale', 'auto',1, 0.1, 0.01],
          'class_weight': [None, 'balanced'],
            'random_state': [42]}
```

Figure 24. *SVC hyperparameter in GridSearchCV (210 candidates)*

GridSearchCV	Feature selection	hypothesized
Parameters	C: 0.5 class_weight: None gamma: auto kernel: rbf	C: 10 class_weight: None gamma: scale kernel: rbf
Accuracy	65.78% (slightly improve)	66.86% (slightly improve)
Confusion Matrix	[966,342] [752,1133]	[1177,131] [941,944]
F1-score(on 0/1)	64% 67%	69% 64%
ROC_AUC_Score	72.3%	75.68%

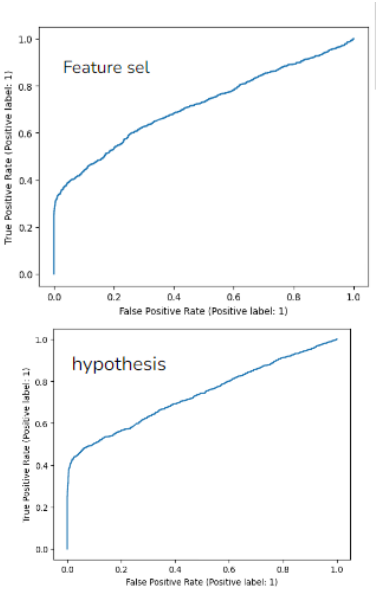

Figure 25. *SVC Best hyperparameters* ***Figure 26.*** *SVC after GridSearchCV ROC AUC curve*

Our research on SVM models revealed considerable disparities in their predictive abilities. With carefully chosen hyperparameters, the hypothesized model emerged as the most accurate classifier. We can see similar parameters (see *Figure 24*), such as kernel: ‘rbf’ and class weight: 'None' for the feature selection and hypothesized models. The 'rbf' kernel is particularly effective when the decision boundary is nonlinear, which makes sense in our dataset. A class weight of 'None' can mean the dataset is balanced. A large C parameter (low regularization) leads to a high-bias, low-variance model whereas a low C (high regularization) is just the opposite. According to *Figures 25 and 26*, these findings imply that, when properly set and optimized, SVMs can be useful tools for classification problems. Although our best results are somewhat poor, the hyperparameters can further be fine-tuned for predicting a lower false positive rate.

Conclusion

During model evaluations with the e-commerce dataset, our group's hypothesized model outperformed the other models (base and feature selection models) in most cases. Out of all of the classification algorithms used in this study, Random Forest provided the best accuracy (67.9%), whereas Logistic Regression yielded the worst results (63.01%). We also observed that the most significant features that emerged were the weight of shipment, cost of product, and discount offered.

We also noted that most of the models underfitted. Further, we believe that the e-commerce shipping dataset, although fairly larger than the project's requirements, was not representative of all e-commerce data; hence, our model cannot be generalized to the e-commerce market. We felt that in order to get a better model, we needed additional features in the dataset such as distance from shipment, estimated time, and stock levels.

For future enhancements to this project, our group will consider the following:

- Hyperparameter Tuning: through the exploration of additional hyperparameter settings to further optimize model performance.
- Feature Engineering: by considering additional techniques of using new features that may improve prediction accuracy.
- Data Collection or Using Additional features: related to customer interactions, estimated time of arrival, distance of travel, departure and arrival points may also enhance the model's predictive power.
- Continuous Monitoring and Updating: of the model to ensure its relevance over time.

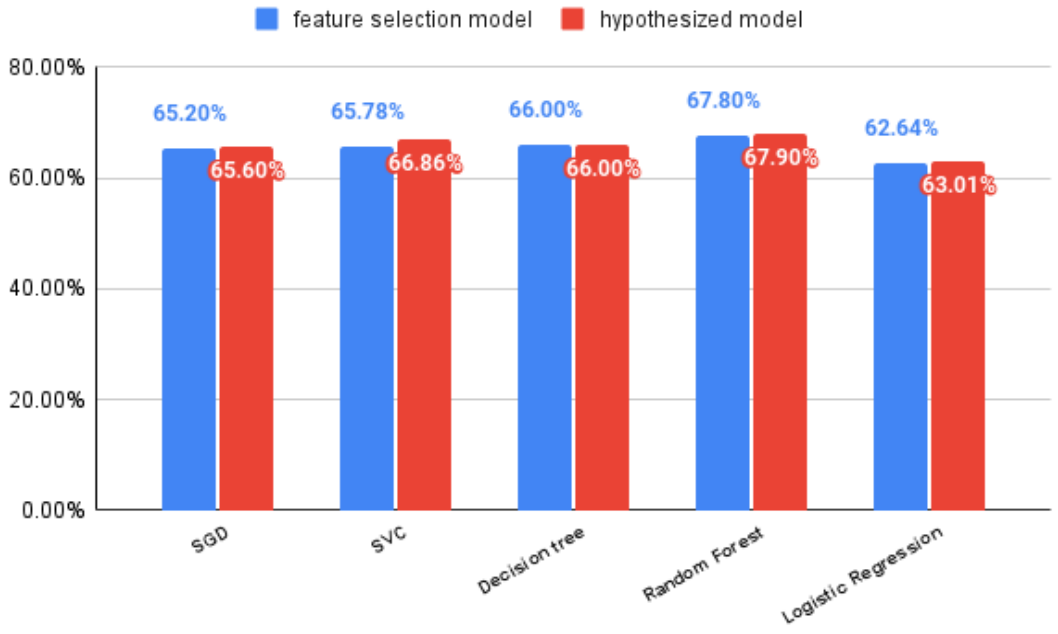

Figure 27. *Feature selection vs. Hypothesized model's accuracies*