**E-Commerce Shipping Data Project**

Group 23

Chalida Naiyaporn

Li-Hsuan Lin

Yushan Yang

1. **Abstract**

Nowadays, more people tend to shop online so companies want to satisfy their customers by delivering their packages on time. Many factors may play a role affecting the package delivery. In this report, we utilized five popular supervised classification methods to classify whether a package is delivered on time based on other ten predictors. Their performance was compared along with the best tuning parameters by five-fold cross validation in each model. Finally, we picked AdaBoost as our best model considering its good performance in overall error without overfitting. We also computed the variable importance and found that two most important variables affecting the classification are the discount and the weight of packages.

1. **Problem and Motivation**

Due to the development of the internet and a broad spread of pandemic, the number of people using online shopping keeps increasing. Therefore, e-commerce has become an indispensable part of many companies. In between, package delivery is critical as people may be disappointed about the company if their package didn’t arrive on time. Our group is interested in studying the factors that affect the package delivery and wants to utilize them to classify the package delivery status. We also focus on the relationships among variables to see if there is any interesting finding. By doing those things we could give suggestions to the company in order to improve their delivery service.

1. **Data**

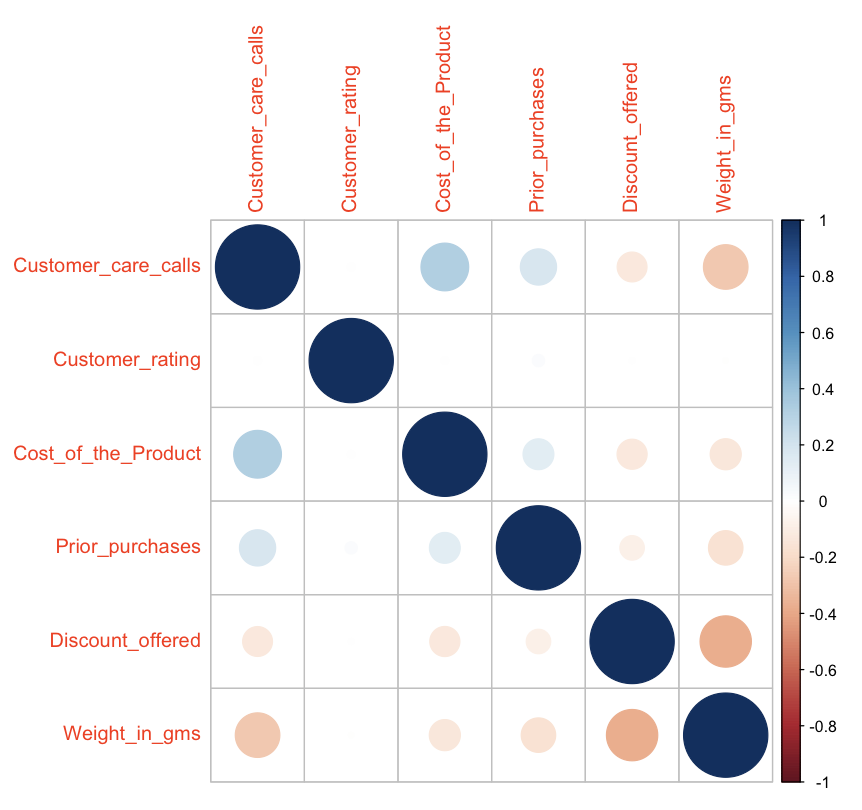
**3.1 Data Description**

The dataset we used is from an international company customer database in Kaggle [1] where there are 10999 observations in total. The response variable, Reached on time, contains 40% “Yes” class and 60% “No” class. Other ten variables were used as predictors, which are: Warehouse block (the warehouse of the company), Mode of shipment (ship, flight or road), Customer Care Calls (the number of calls made from enquiry or enquiry of the shipment), Customer rating (the rating of the company from customers), Cost of the product (the cost of the product in U.S. dollars), Prior purchases (the number of prior purchases), Product importance (low, medium and high categorized by the company), Gender (male or female), Discount offered (discount offered to the product) and Weight in gms (product weight in gms). Six of them are numerical variables, one is ordinal and the other three are categorical variables. Luckily, the response variable is not much imbalanced and there are no missing values so we directly used all rows for further data processing.

**3.2 Exploratory Data Analysis (EDA)**

In this part, we would like to analyze the relationship between features in the training dataset by visualizing before we fitted data in the model.

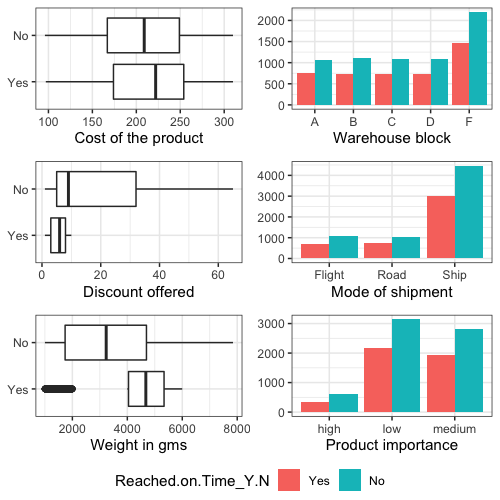
We started our analysis from numerical data by calculating the correlation matrix to see the relationship between numerical features.



**Figure 3.1** Correlation matrix between numeric features; customer care calls, customer rating, cost of the product, prior purchases, product importance, discount offered, and weight

According to the correlation matrix in Figure 3.1, we can see that the correlation is so weak. The weight in grams and discount offered are the most correlated features in different directions, -0.37, while cost of the product and customer care calls are the second highest correlation in the same directions around 0.32.

We have plotted the distribution of each numerical feature by boxplot and compared the distribution by class of response. We found three features that have different distributions between classes as shown in Figure 3.2 left column. The median cost of the product is slightly different between the two classes, while there is a higher number of packages reached on time than packages that did not reach on time. Moving on to discounts offered, we can see that the packages that delivered on time had a smaller range of discounts than packages that did not. The last boxplot clearly shows that weight in both classes are totally different. More than a half of data in class 1 (delivered on time) does not overlap with class 2 (did not reach on time).

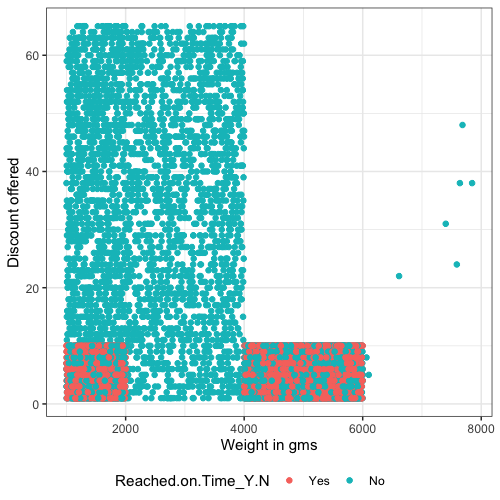


**Figure 3.2** Left column compared cost of the product, discount offered, and weight between classes. Right column showed histogram of warehouse block, mode of shipment, and product importance between levels and response.

For categorical and ordinal features as shown in the Figure 3.2 right column, we created histograms to compare the level of each feature by the response. To begin with the warehouse block, block F had more frequency than other blocks for both classes. While the most popular mode of shipment was by ship. The high product importance has the smallest frequency among all levels.

Then, we plotted scatter plots, as in Figure 3.3, between features and found that packages with less than 4,000 grams which were offered 10% discount tended to deliver late. In contrast, other pairs of scatter plots did not show an outstanding separation between two classes as discount and weight did.

After we analyzed the features, we assumed that our data may fit with the tree-based model because we can separate both classes by some variables.



**Figure 3.3** Scatter plot of discount offered against weight.

1. **Results**

**4.1 Methods**

We applied five supervised learning methods to do the classification. They are: Random Forest (RF), AdaBoost, K-nearest Neighbors (KNN), Support Vector Machine (SVM) and Logistic Regression (LR). RF and AdaBoost are both ensemble classifiers where RF reduces the correlation between trees by permuting some variables and Adaboost weights more on misclassified samples. We want to implement these two methods because firstly they are very stable compared to a single tree and they are usually good at classification problems containing categorical variables. KNN is a non-parametric approximation to the Bayes classifier. We want to implement this method due to its intuitive interpretation and no assumption on the data distribution. SVM is to identify a hyperplane with a large margin that attributes data to different classes in order to achieve a greater robustness to individual observations and have a better accuracy in classifying the rest of observations [2]. We make use of kernels to account for the possibility of nonlinear boundaries. Linear, polynomial and radical kernels were considered in this report. LR is an extension of linear regression models. To fit the model and find the coefficient estimates, we use the maximum likelihood method. Mathematically, the coefficient estimates are chosen to maximize the likelihood of predicted probabilities of the observed data. This method is relatively flexible with fewer assumptions about the data and we used the usual probability cutoff of 0.5.

**4.2 Data Preprocessing**

For the data preprocessing, we randomly sampled 70% of data in each response class as the training set and the rest as the test set. For RF, AdaBoost and LR, we didn’t do other data preprocessing procedures. As for KNN, because it is a distance-based classifier, we first turned the ordinal variable into the numerical variables and then did normalization and standardization on all numerical variables. One-hot encoding was performed on categorical variables to help calculate the distance. For SVM we also transferred the ordinal variables and did normalization and standardization on numerical variables.

**4.3 Parameters Tuning**

We tuned parameters in a method to attain the best model that accounts for both bias and variance. The reason is that if we tune the model too much, we tend to overfit the data. And if we barely tune the model, we underfit the data. Thus, tuning parameters in a model control its complexity. We conducted five-fold cross validation (CV) to choose the best set of tuning parameters that gives the lowest CV error. Table 4.1 shows the best tuning parameters for each method.

| **Methods** | **Best Tuning Parameters** | **CV Error** |
| --- | --- | --- |
| RF | mtry: 3, nodesize: 2, ntree: 5000 | 0.337 |
| Adaboost | N.tree: 250, interaction.depth = 2, shrinkage = 0.05 | 0.322 |
| KNN | K = 89 | 0.360 |
| SVM | Kernel: polynomial, cost = 0.1, degree = 2 | 0.357 |

**Table 4.1** Best tuning parameters for each model.

For RF, there are three tuning parameters: mtry (number of attributes used per splitting), nodesize (minimum size of terminal nodes) and ntree (number of trees). mtry in {1,3,5}, nodesize in {0.5,1,2} and ntree from 1000 to 6000 were tested. Without any constraint, mtry=1 was given as the best parameter, which does not make much sense for this model. So we finally fixed mtry as its default value 3 to search for the other two parameters.

AdaBoost also has three parameters to tune: N.tree (number of trees), interaction.depth (maximum depth of the tree) and shrinkage (learning rate for step size reduction). We tried N.tree in {100,250,500,1000,2000}, interaction.depth in {2,3} and shrinkage from 0.02 to 0.1. The best number of trees is 250, twenty times less than what RF gave, making the training faster.

For KNN, the best number of neighbors needs to be figured out and Figure 4.1 shows the result. As k increases, CV error first decreases and then goes up again, corresponding to the bias and variance trade-off.

In terms of SVM, there are three tuning parameters: kernel (approach to enlarge feature space), cost (budget for the amount of observation that can violate margin), degree (degree of the polynomial kernel) [2]. We try linear, polynomial, and radius kernels, and let degree be in {1,2,3,4} and cost be in {0.1,1,5,10}. The best set of parameters is that the kernel is polynomial with degree of 2 and cost is 0.1.



**Figure 4.1** Plot of CV error under different number of neighbors.

**4.4 Training and Test Error**

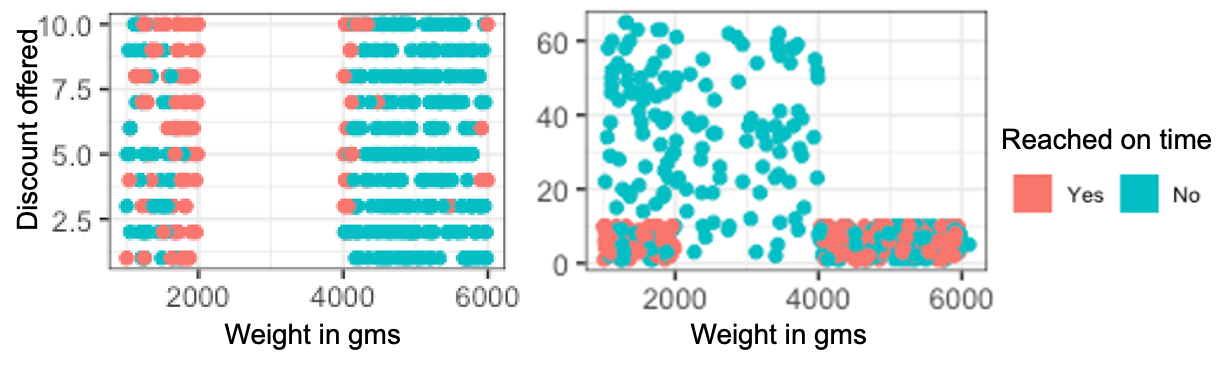
The training and test error along with the error in each class for five methods were summarized in Table 4.2. As we can see, RF performs perfectly on the training data with overall training error 0, indicating an overfitting. AdaBoost gives us the best overall test error 0.322 though this value is not that much smaller than what other methods provide. And LR gives the worst overall test error perhaps because it is a linear classifier so that does not work well in this dataset.

| **Methods** | **Training Error** | | | **Test Error** | | |
| --- | --- | --- | --- | --- | --- | --- |
| **Overall** | **Class “Yes”** | **Class “No”** | **Overall** | **Class “Yes”** | **Class “No”** |
| **RF** | 0 | 0 | 0 | 0.328 | 0.238 | 0.388 |
| **AdaBoost** | 0.305 | 0.160 | 0.403 | 0.322 | 0.186 | 0.414 |
| **KNN** | 0.319 | 0.189 | 0.406 | 0.341 | 0.240 | 0.409 |
| **SVM** | 0.336 | 0.219 | 0.437 | 0.341 | 0.230 | 0.441 |
| **LR** | 0.355 | 0.291 | 0.442 | 0.363 | 0.294 | 0.453 |

**Table 4.2** Training and test error for five methods in classifying package delivery status. (class “Yes”: packages delivered on time; class “No”: packages not delivered on time)

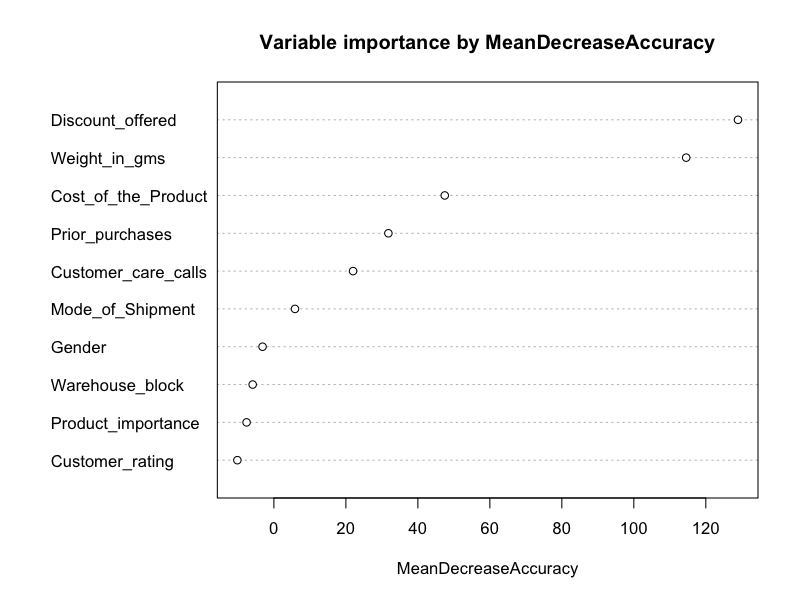
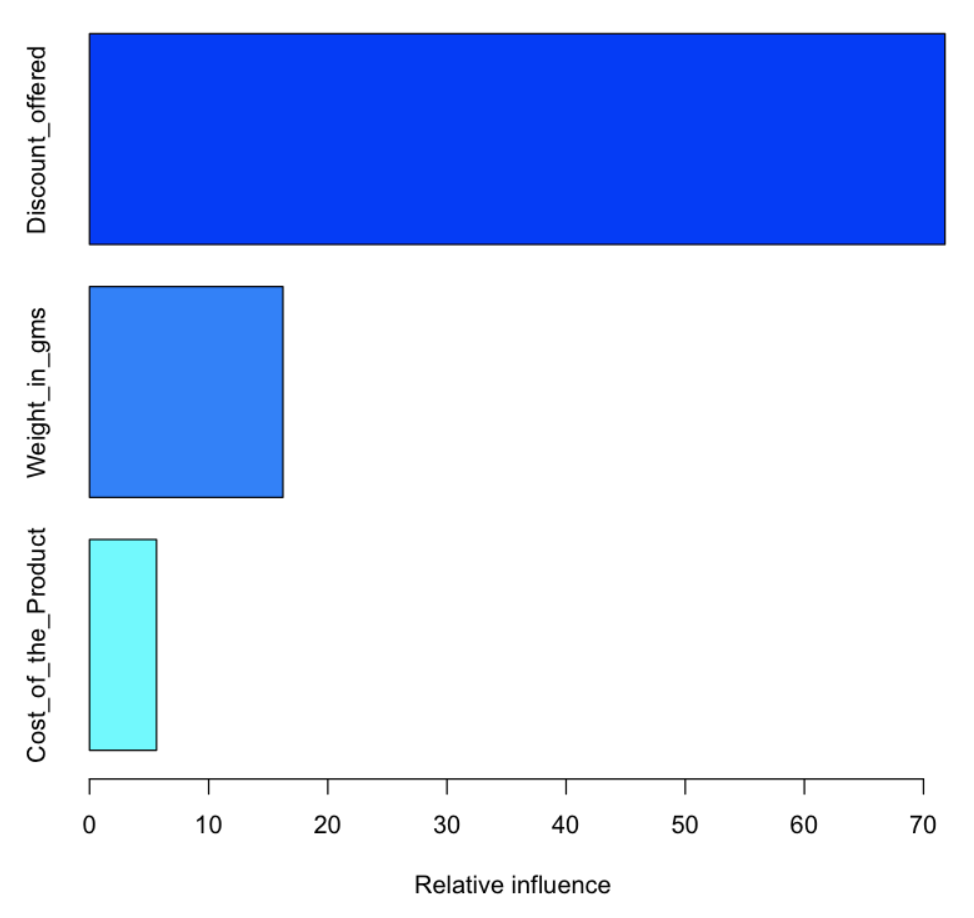
Another observation is that all methods have a worse classification performance on class “No”, which is not a good sign as we care more about the packages not delivered on time. The cause of different errors in two classes may depend on the distribution of data points in high dimension. As we can see from the scatter plot of misclassified samples compared to original samples projected on the most two important variables in Figure 4.2, all misclassified samples are from the mixture area of two classies. When weight in gms variable is over 4000, almost all misclassified samples are from class “No”. This is probably due to a less density of class “No” in that region, resulting in the data points more likely to be assigned as class “Yes”.

Considering the overall test error, overfitting issue and training time, AdaBoost was chosen as our best model for this dataset.

****

**Figure 4.2** Scatter plots of misclassified and the original samples projected on two dimensions (labeled with true class). Left: 500 misclassified points sampled from Adaboost method. Right: 500 original points (including correctly classified and misclassified) sampled from the overall dataset.

**4.5 Variable Importance**



**Figure 4.3** Variable importance plot calculated from AdaBoost (left) and RF (right).

We can use variance importance to assess how important each predictor is in terms of separating data with different classes. Specifically, at each split in each tree, the mean decrease accuracy is the importance measure attributed to the splitting predictor, and it is accumulated over all the trees [2].

As we can see from Figure 4.3, discount offer status (Yes v.s. No) and weight (gm) are distinctly important predictors for both the random forest and AdaBoost model. In other words, these two variables play a major role in separating the data between two classes (Reached on time vs Not Reached on time). Predictors such as cost of product, status of prior purchase (Yes v.s. No), and the number of customer care calls made also have an effect on predicting the outcome variable.

1. **Conclusions and Discussion**

Since we have analyzed the data in the EDA part, we found that our data does not have a strong correlation between features but it may fit with a tree-based model such as Random Forest and AdaBoost. Consequently, the best classifier for this analysis is AdaBoost with hyperparameters; number of trees = 250, interaction depth = 2, and shrinkage = 0.05. We derived feature importance from the AdaBoost model and we found that discount and weight are the top outstanding features. This followed our analysis in the EDA part that these two features can separate data between classes well.

To improve our analysis, we would like to collect more data to conduct this classification model because there is some lacking data in some levels and some length of values. Also, we would like to collect more features that reflected the delivery time such as weather, season, or gas price, etc.

1. **Contributions**

Yushan Yang: finish methods part in project proposal; write code for Random Forest, AdaBoost and KNN; write some code for exploratory data analysis; prepare figures/tables for presentation slides; finish abstract, problem and motivation, method description, data preprocessing and training and test error part in final report.

Li-Hsuan Lin: method description; parameter tuning part; variable importance part; piecing together appendix

Chalida: EDA, discussion and conclusion

We collaborated in a way that each one focused on part of the training and data analysis and then we discussed our individual results for further polish and modification.

1. **Reference**

[1] Kaggle e-commerce shipping dataset: <https://www.kaggle.com/datasets/prachi13/customer-analytics>

[2] James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: With Applications in R*

1. **Acknowledgements**

We thank the lab material of the course for helping us in the coding part.

1. **Appendix**

<https://github.com/yys1234/STATS503-project-group23>