Late Delivery Prediction in Product Delivery Using Binary Classification Approach.

A Project Report

Presented to

Dr. Joanna Mills Flemming

STAT 5620 – Data Analysis

Faculty of Science

Dalhousie University

By

Shaikh Mushfikur Rahman - B00908273

Abstract

The aim of this paper is to develop a binary classification model for late delivery prediction and identify the key factors that are responsible for late delivery. The 'Olist' dataset provided by the largest departmental store of Brazil is used for model building purposes. Although this paper followed the CRISP-DM(apa) development cycle to analyze the dataset, the general flow of the paper will follow the steps shown in the following table of contents (*Data Science Process Alliance*, n.d.).

TABLE OF CONTENTS

KEY WORDS

Late Delivery Prediction, Ecommerce, Supply Chain, Logistic Partner, Binary Classification, Payment Type, Shipping Delay

INTRODUCTION

The overall goal of this project is to analyze the Brazilian E-Commerce Public Dataset by Olist identify the key factors that are responsible for late delivery as well as predict the late delivery using a binary classification approach.

For this project, I use the Python programming language with Jupyter Notebook which is a web based interactive environment exclusively for Data Science related projects (*Jupyter*, n.d.).

To make my analysis consistent and persistent, I followed the CRISP DM process to accomplish my project goal. The CRISP DM process that I have taken to accomplish my project is given below.

1. Project goal understanding.
2. Data exploration and preparations.
3. Model development.
4. Evaluation of project outcomes.

Data Exploration

For the analyzing purpose, the Brazilian E-Commerce Public Dataset by Olist dataset is used exclusively. The dataset is provided by the well known Brazilian e-commerce Olist. The dataset was taken from Kaggle(apa) Olist is a well known Brazilian e-commerce platform. It connects small businesses all over Brazil and the sellers can sell their goods by using the Olist platform and also ship them directly to the customers using Olist logistics partners. The Olist store seller gets notified after each purchase to fulfill their order. Once the estimated delivery date is due the customer gets a satisfaction survey by email where they could share their experience by writing down some comments. As it is real commercial data, references of the companies and partners in the review text section are replaced with the names of Game of Thrones' great houses (apa). The dataset contains information of 100k orders ranging from 2016 to 2018 and 52 features. One of the very important characteristics of this dataset is viewing an order from multiple dimensions, such as order status, payment, price and freight performance to customer location, product attributes and finally reviews given by customers. The dataset is divided into nine files and each file represents some unique information regarding the datasets such as customers information, order item information, order review information etc. The following Figure1 describes how the chunk of the dataset connects to each other.

This dataset contains more than 100K instances and 52 features. The actual dataset is divided into nine datasets for better understanding and organization. Not all datasets meet my analyzing criteria, So I have chosen the following datasets for my analyzing purposes. All the datasets are in csv format. I am going to give a brief introduction of each chunk of the dataset that I have chosen to better understand the datasets.

Dataset Descriptions

Order Dataset

This is the core dataset. From each order you might find all other information.

Features	Feature Descriptions
order_id	Unique identifier of the order.
customer_id	Key to the customer dataset. Each order has a unique customer_id.
order_status	Reference to the order status (delivered, shipped, etc).
order_purchase_timestamp	Shows the purchase timestamp.
order_approved_at	Shows the payment approval timestamp.
order_delivered_carrier_date	Shows the order posting timestamp. When it was handled to the logistic partner.
order_delivered_customer_date	Shows the actual order delivery date to the customer.
order_estimated_delivery_date	Shows the estimated delivery date that was informed to customer at the purchase moment.

Customer Dataset

This dataset has information about the customer and its location. Use it to identify unique customers in the orders dataset and to find the orders delivery location.

Features	Feature Descriptions
customer_id	Key to the orders dataset. Each order has a unique customer_id
customer_unique_id	Unique identifier of a customer.
customer_zip_code_prefix	First five digits of customer zip code
customer_city	Customer city name
customer_state	Customer state name

Order Items Dataset

This dataset includes data about the items purchased within each order.

Features	Feature Descriptions
order_id	Order unique identifier
order_item_id	Sequential number identifying number of items included in the same order.
product_id	Product unique identifier
seller_id	Seller unique identifier
shipping_limit_date	Shows the seller shipping limit date for handling the order over to the logistic partner.
price	Item price
freight_value	Item freight value item (if an order has more than one item the freight value is splitted between items)

Products Dataset

This dataset includes data about the products sold by Olist.

Features	Feature Descriptions
product_id	Unique product identifier
product_category_name	Root category of product, in Portuguese.

product_name_lenght	Number of characters extracted from the product name.
product_description_lenght	Number of characters extracted from the product description.
product_weight_g	Product weight measured in grams.
product_length_cm	Product length measured in centimeters.
product_height_cm	Product height measured in centimeters.
product_width_cm	Product width measured in centimeters.

Sellers Dataset

This dataset includes data about the sellers that fulfilled orders made at Olist. Use it to find the seller location and to identify which seller fulfilled each product.

Features	Feature Descriptions
seller_id	Seller unique identifier
seller_zip_code_prefix	First 5 digits of seller zip code
seller_city	Seller city name
seller_state	Seller state

Order Payments Dataset

This dataset includes data about the orders payment option

Features	Feature Descriptions

order_id	Unique identifier of an order.
payment_sequential	A customer may pay an order with more than one payment method. If he does so, a sequence will be created to
payment_type	Method of payment chosen by the customer.
payment_installments	Number of installments chosen by the customer.
payment_value	Transaction value.

Merge DataSets:

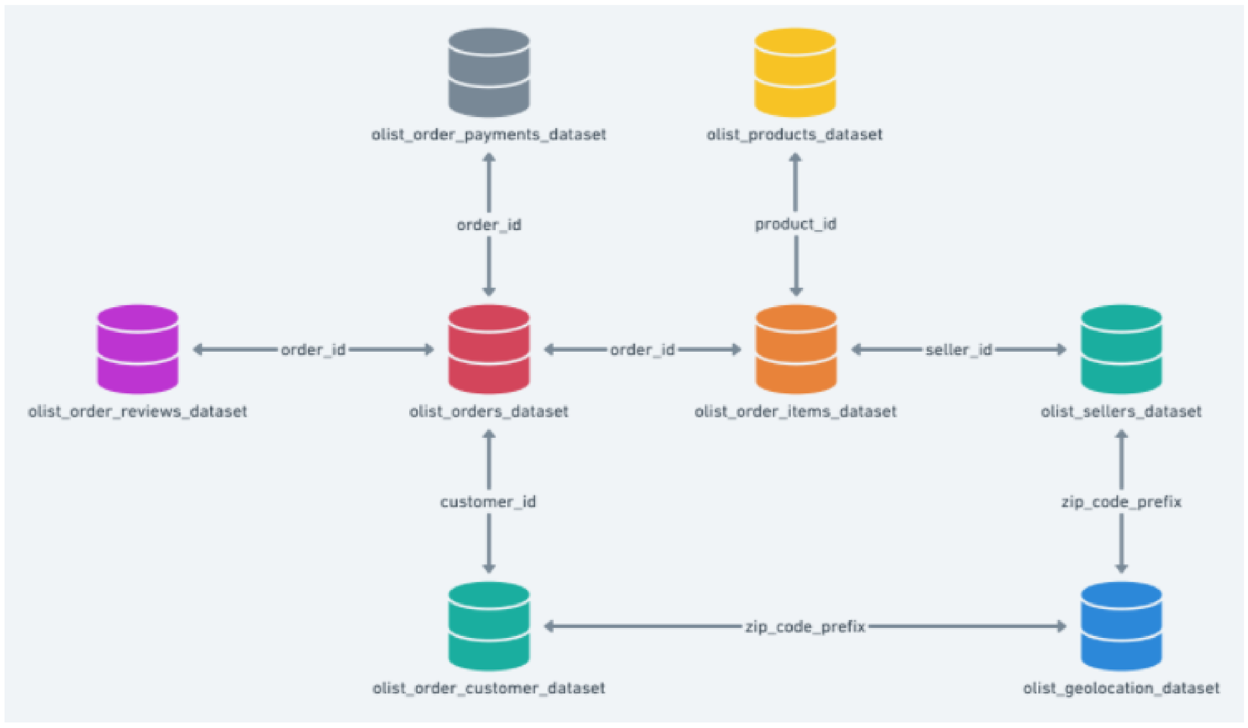
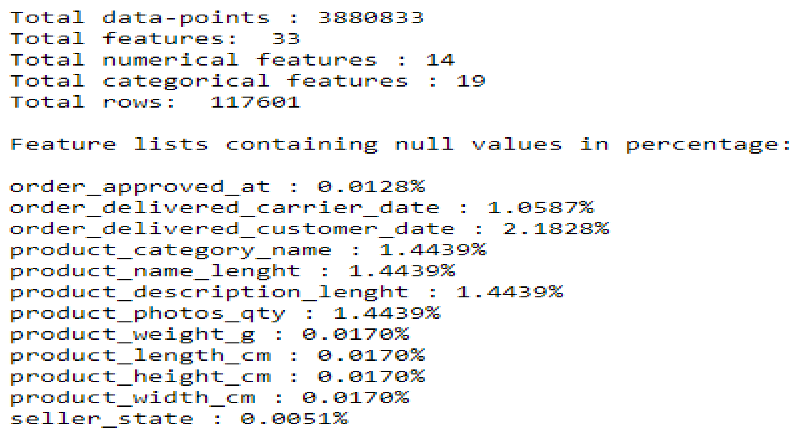


Figure 1. The file distribution of the Olist dataset (Magioli, n.d.)

There are a total six chunks of datasets. So before proceeding further, I merge all the datasets together to make my data analysis a bit easier. I followed the Figure_1 schema to merge my datasets. After merging the datasets a quick data exploration is done and the outcome is given below.

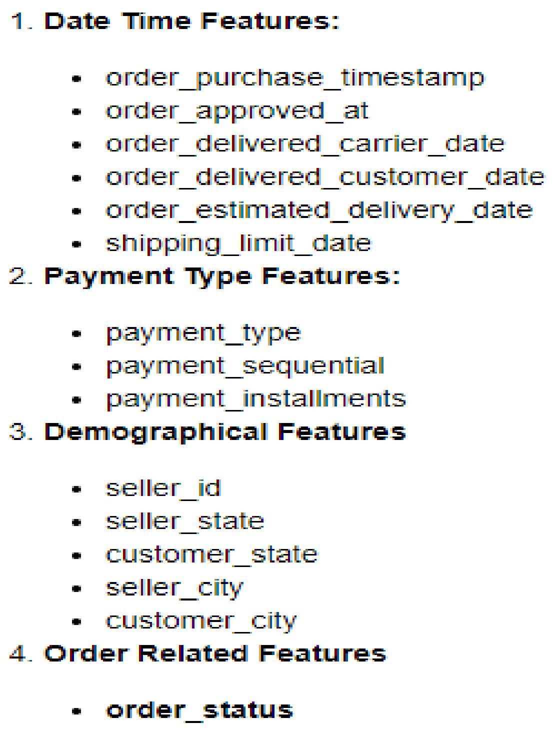


```
Total data-points : 3880833
Total features:  33
Total numerical features : 14
Total categorical features : 19
Total rows:  117601

Feature lists containing null values in percentage:

order_approved_at : 0.0128%
order_delivered_carrier_date : 1.0587%
order_delivered_customer_date : 2.1828%
product_category_name : 1.4439%
product_name_lenght : 1.4439%
product_description_lenght : 1.4439%
product_photos_qty : 1.4439%
product_weight_g : 0.0170%
product_length_cm : 0.0170%
product_height_cm : 0.0170%
product_width_cm : 0.0170%
seller_state : 0.0051%
```

Figure 2. Data Explorations

Relevant Features

There are a total 33 features in this joined dataset while the total number of rows is 117601. The number of null values is not that high, which is a good indicator of the usability of this dataset. There are a total 33 features in this dataset and not all features are important to my classification context. For simplicity purpose, I pick up the most relevant features and divide the features into four categories as indicated in Figure 3.



1. **Date Time Features:**
    - order_purchase_timestamp
    - order_approved_at
    - order_delivered_carrier_date
    - order_delivered_customer_date
    - order_estimated_delivery_date
    - shipping_limit_date
2. **Payment Type Features:**
    - payment_type
    - payment_sequential
    - payment_installments
3. **Demographical Features**
    - seller_id
    - seller_state
    - customer_state
    - seller_city
    - customer_city
4. **Order Related Features**
    - **order_status**

Figure 3. Relevant features.

In Figure 3, it lists all the relevant features based on categories. Here, one of the most important features is order_status which refers to the current status of the orders, such as delivered, shipped etc.

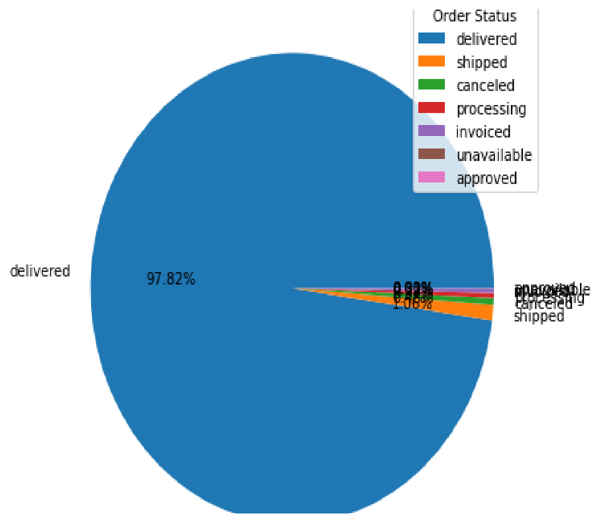


Figure 4. Order Status

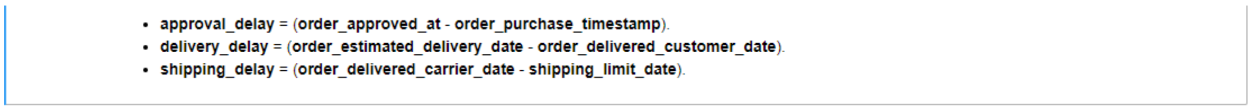
In figure 4, it seems that order status has a total of 7 types of features. Those are delivered, shipped, canceled, processed, invoiced, unavailable and approved. Here around 97.82% of orders are in delivered status and the rest are others. As my project goal is to predict the timely delivery of the orders, I remove the rest of the orders except delivered orders.

Remove Null Values

Total number of data points in this merge data set is 3880833(Figure 2) while the total number of null data points is around 6623, which is quite lower compared with the total number of data points. So I removed the null rows to get a precise investigation.

Date Time Feature Analysis

There are a total of six time-related features (Figure 3) in this dataset and all are in a string format. For analyzing and modeling purposes, I convert all the time-related features into datetime features and get their intervals (Figure 5). For example, delivery_delay is the time difference of the actual delivery time and expected delivery date (Figure 5). I create those date time interval features to see how much impact those date time features have on the status of the delayed delivery.



- **approval_delay** = (**order_approved_at** - **order_purchase_timestamp**).
- **delivery_delay** = (**order_estimated_delivery_date** - **order_delivered_customer_date**).
- **shipping_delay** = (**order_delivered_carrier_date** - **shipping_limit_date**).

Figure 5. Date Time Features.

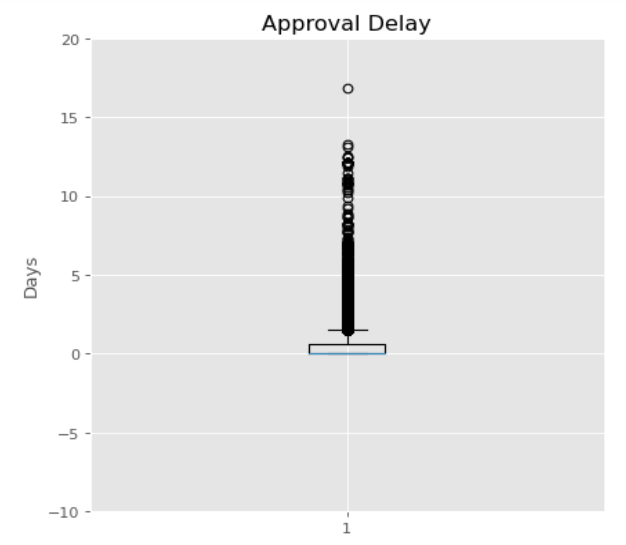


Figure 6. Approval Delay

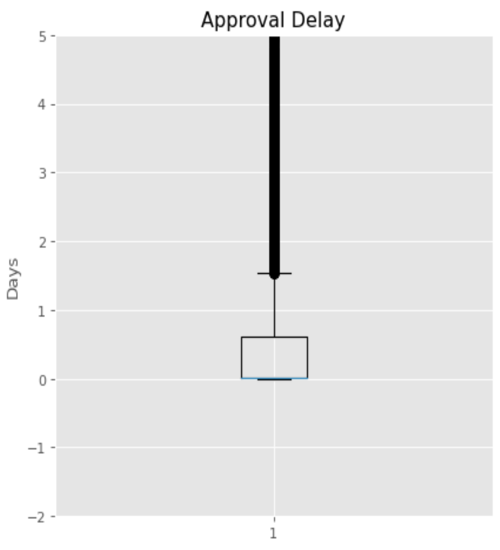


Figure 7. Approval Delay (Zoomed)

Approval Delay is the time taken to approve the order after purchasing it by the customer. It seems that most of the orders around 75% of the orders are approved within one day, while around 24% of the order takes more than one day to be approved. Here are some outliers and those tooks around more than 15 days to be approved (Figure 6 && Figur 7).

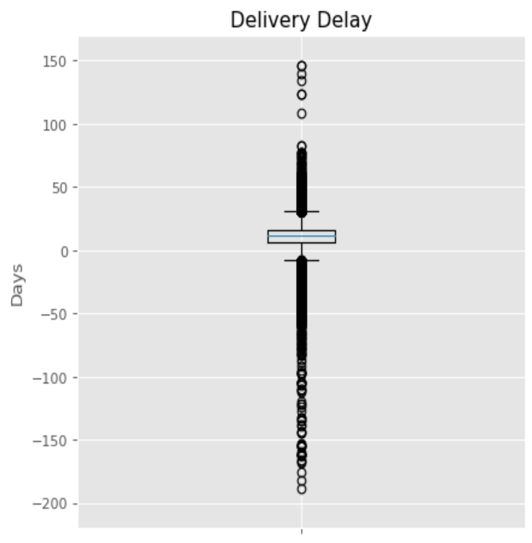


Figure 8. Delivery Delay

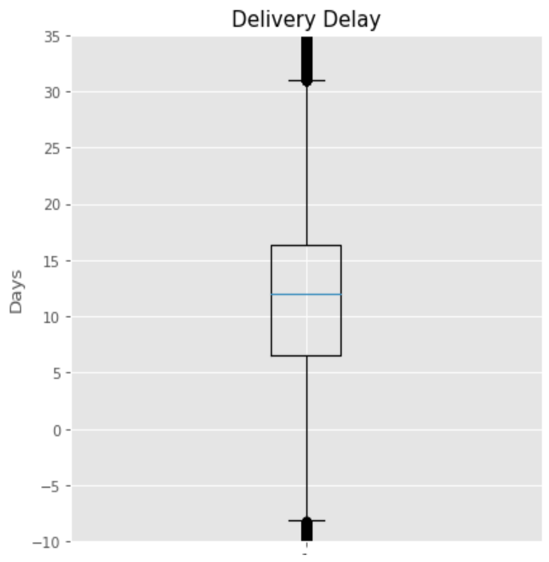


Figure 9. Delivery Delay (Zoomed)

Delivery Delay is the time difference between the estimated delivery time and the actual delivery time **(order_estimated_delivery_date - order_delivered_customer_date)**. I subtract the actual delivery time from the estimated delivery time, that means if the **order_delivered_customer_date** is greater than the **order_estimated_delivery_date**, it gives a negative value. Otherwise it will give a positive value. On Figure 8, the smallest number is -200, that means some deliveries took around 200 days after the deadline to reach their final destinations. In Figure 9, the Q1 and Q3 are slightly higher 5 days and 15 days respectively. So it is clearly evident that around 50% of the delivery takes around 6-16 days to reach its destinations. The maximum of the boxplot is a little bit higher than 30 days. That means around 24% of orders reach their destination before the deadline. as, delivery_delay = (order_estimated_delivery_date - order_delivered_customer_date)

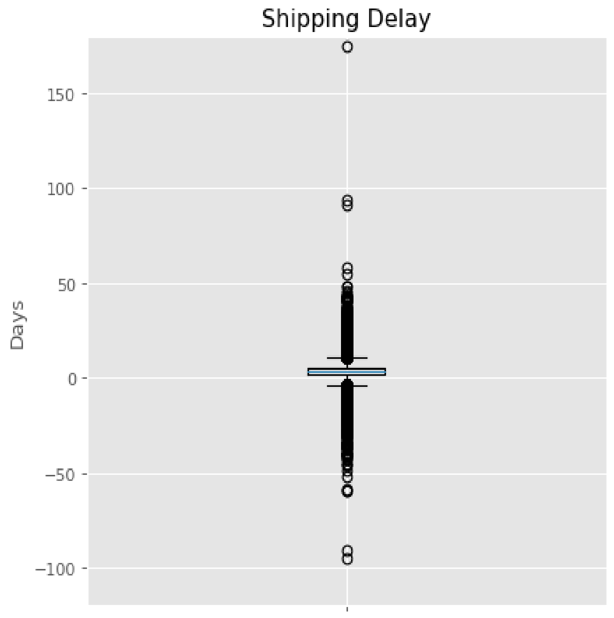


Figure 10. Shipping Delay

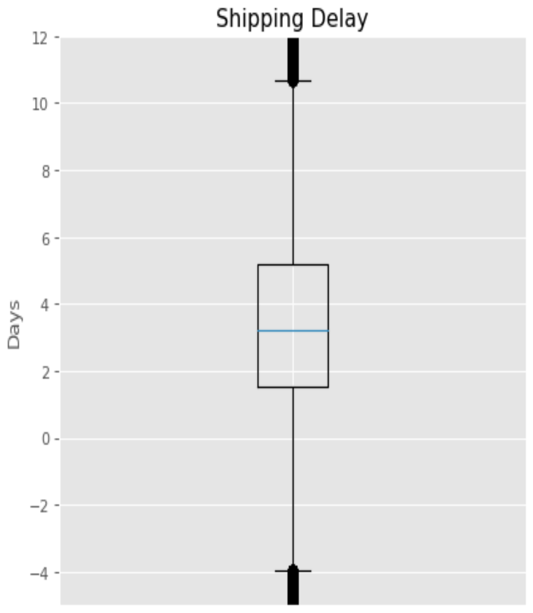


Figure 11. Shipping Delay (Zoom)

Shipping Delay is the time difference between the actual shipping handled time to the logistic partner and the expected time to handle to the logistic partner (**shipping_limit_date - order_delivered_carrier_date**). As I subtracted the actual carrier delivery time to the logistic partner from the shipping limit date, that means if the **order_delivered_carrier_date** is greater than the **shipping_limit_date**, it gives a negative value. Otherwise it will give a positive value. After analyzing the boxplot from Figure 10, it is evident that shipping_delay has some very large outliers. Those are -100 on the lower side and more than 150 on the upper side. From Figure 11, Q1 is slightly below 2 and Q3 is between 4 and 6. So it can be said that around 50% of the sellers handle their product to their logistic partners within day_1 to day_6 after the confirmation of the orders. The minimum is 4, which means around 24% of the sellers take an extra 1 to 4 days after the shipment_deadline to handle their product to the logistics partners.

Detecting and Remove Outliers

I consider 15 as an outlier in terms of approval_delay (Figure 6) while, in terms of delivery_delay, I am considering 50 as an outlier on the upper side and -150 as an outlier on the lower side. On the other hand, in terms of shipping_delay, I am considering -50 and 40 are outliers. So I will remove them from my code before jumping into the next section.

Key Feature - Late Delivery

Late Delivery(late_delivery) is my key feature and derives this feature from Delivery Delay (Figure 5). I have considered all delivery_delay values less than zero as late delivery and greater than zero as timely delivery. I plotted a pie chart to find out the ratio of the ratio of the time delivery and late delivery.

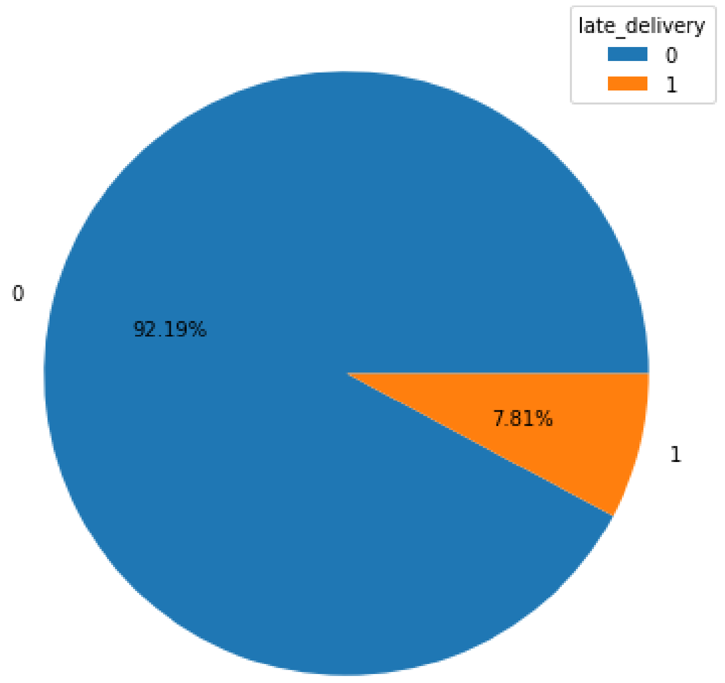


Figure 12. Late Delivery

From Figure 12, the blue color refers to the timely delivery, while the yellow color refers to the late delivery. Here, around 92.19% orders are timely delivered to the customers, while 7.81% of the orders are late delivered.

Impact of Late Shipping Delivery to the Logistic Partner

I derive Late Shipping Delivery(late_shipping_delivery) from the Shipping Delay feature (Figure 5). In Figure 12, if the seller timely handles the product to the logistic partner then it is zero, and if the seller lately handles the product to the logistic partner then it is one. I plotted a pie chart to find out the ratio of the timely handled product to the logistic partner by seller.

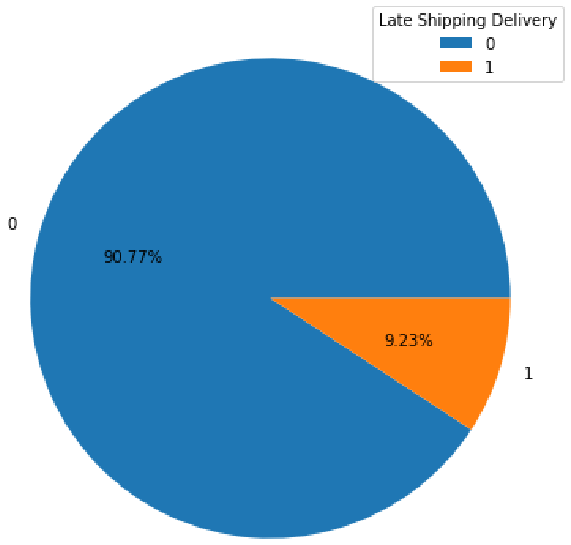
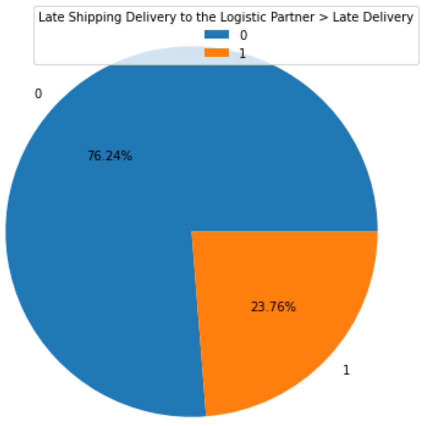


Figure 13. Late Shipping Delivery to the Logistic Partner.

After analyzing the pie chart from Figure 13, it is clear that around 90.77% of the seller's handle their product to the logistic partner before deadline, while the rest of the 9.23% delayed to handle their product to their logistic partner.



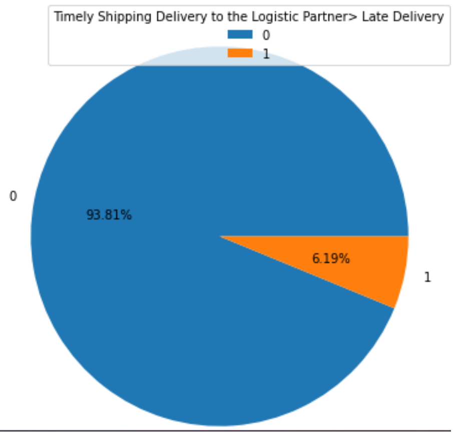


Figure 14. Late Shipping to the Logistic Partner **Figure 15. Timely Shipping Delivery to the Logistic partner**

In the above pie chart, Figure 14, I analyze the late_delivery status of the product, when the seller handles their product to their logistic partner after the deadline. The pie chart from Figure 14, is a clear indication that, around 23.76% of the product is delivered late to the final destination when the seller hands the product to their logistic partner after the deadline. On the other hand, from Figure 15, I analyze the late_delivery status of the product, when the seller handles their product before the deadline to their logistic partner. In Figure 15, the late_delivery status of the product is very low, which is around 6.19% and around 93.81% of the product reaches its final destination before the deadline.

Demographic Feature Analysis

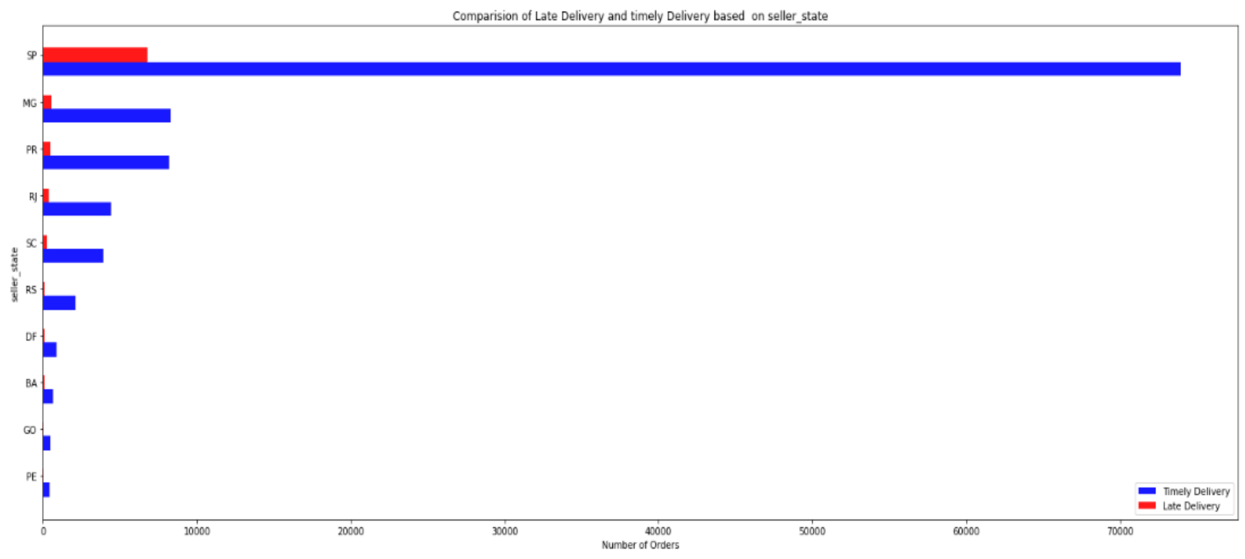


Figure 16. Comparison of Late and Timely Delivery on Seller State

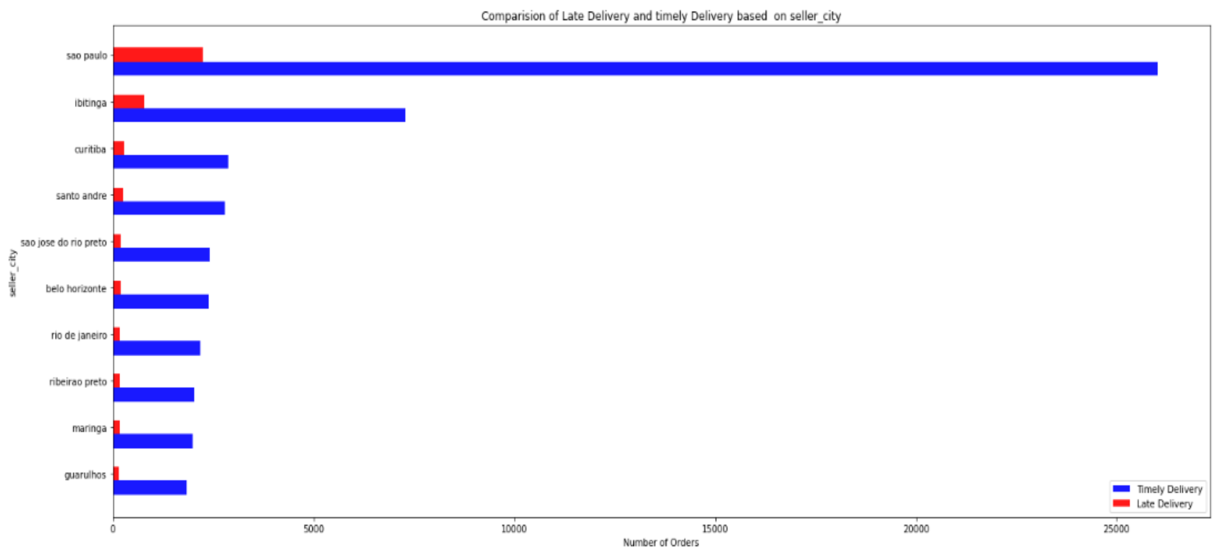


Figure 17. Comparison of Late and Timely Delivery in Seller City

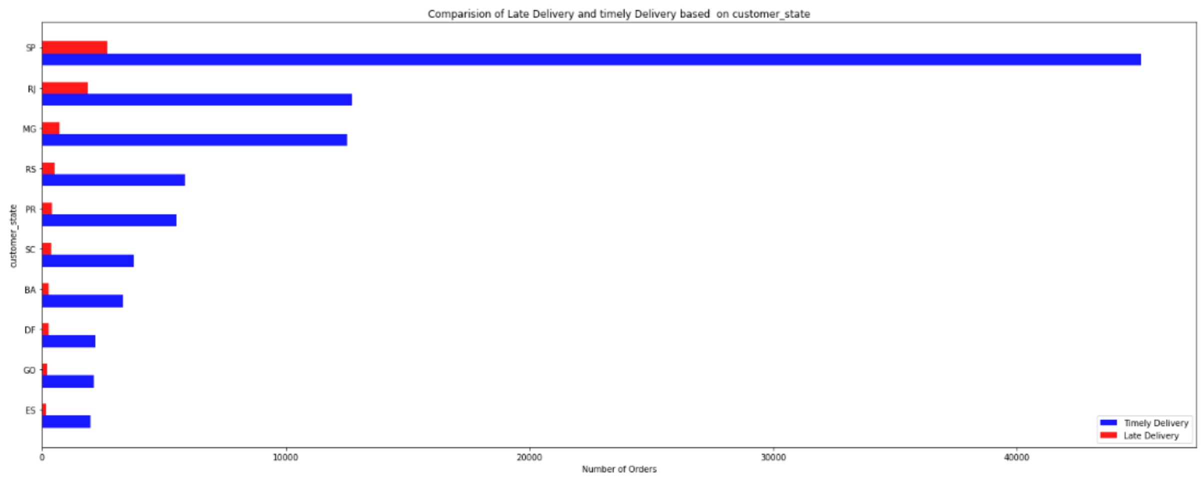


Figure 18. Comparison of Late and Timely Delivery in Customer State

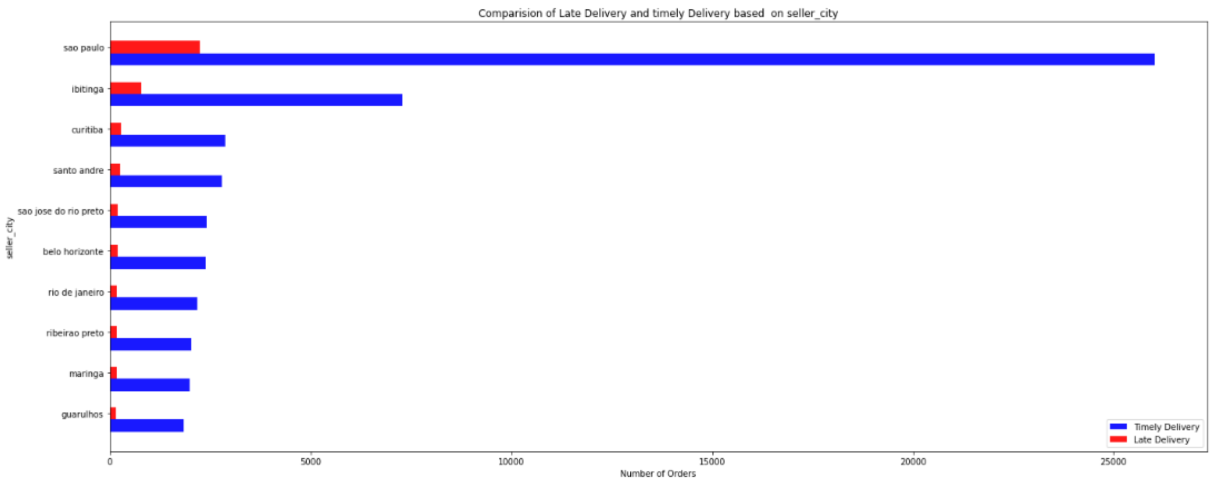


Figure 19. Comparison of Late and Timely Delivery in Customer City

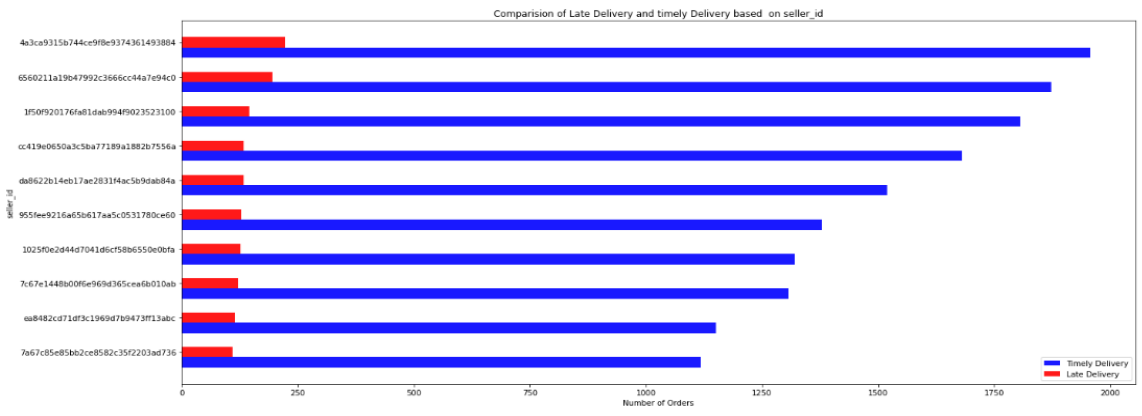


Figure 20. Comparison of Late and Timely Delivery in Customer City

In Figure 16,17,18,19 I plotted the top ten seller_state, seller_city, customer_state, customer_city where most of the order comes from and found out one common pattern, that is, the number of late delivery increases as the number of orders increases. As an example, In Figure 17, most of the orders came from Sao Paulo, so the occurrence of the late delivery is comparatively high there. On the other hand, the least number of late deliveries occurs in the Guarulhos (Figure 17) as the total number of orders is relatively low there. The Olist provides each seller a unique id and I plot the seller_id, who's getting most of the orders since 2016 to 2018 (Figure 20) and it follows the usual trends, that is the number of late deliveries increases as the total number of orders increases.

Payment Type Feature Analysis

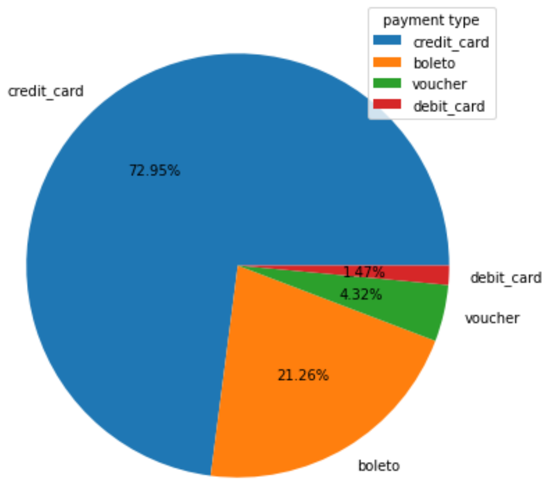


Figure 21. Payment Type on Late Delivery

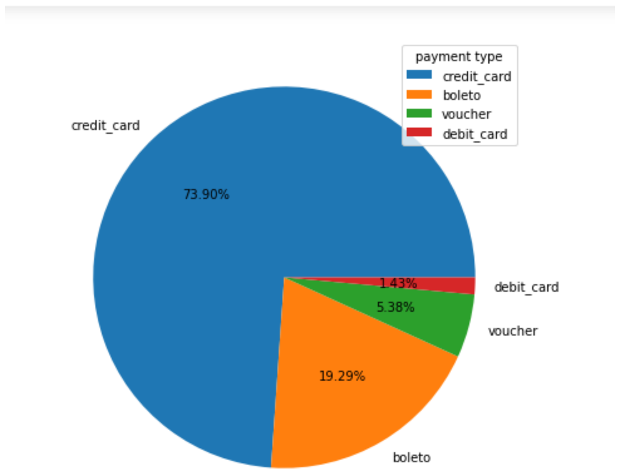


Figure 22. Payment Type on Timely Delivery

I plotted two different pie charts to find out impacts of payment type on the late delivery status. In Figure 21, I plotted a pie chart based on payment type where all the orders lately arrived to their customers and in Figure 22, all the orders timely arrived to their customers. My observation is the use of voucher and credit_cart is slightly higher on timely delivery while the use of boleto and debit_card is higher on late delivery.

Methods

Pickup Relevant Features

My merge dataset contains around 33 features and not all features are relevant for late delivery predictions. Such as some of the features containing product specific information such as product image, descriptions. I excluded those features for modeling purposes. I have chosen the following images based on my domain knowledge and the visualizations that I have drawn to find out important features.

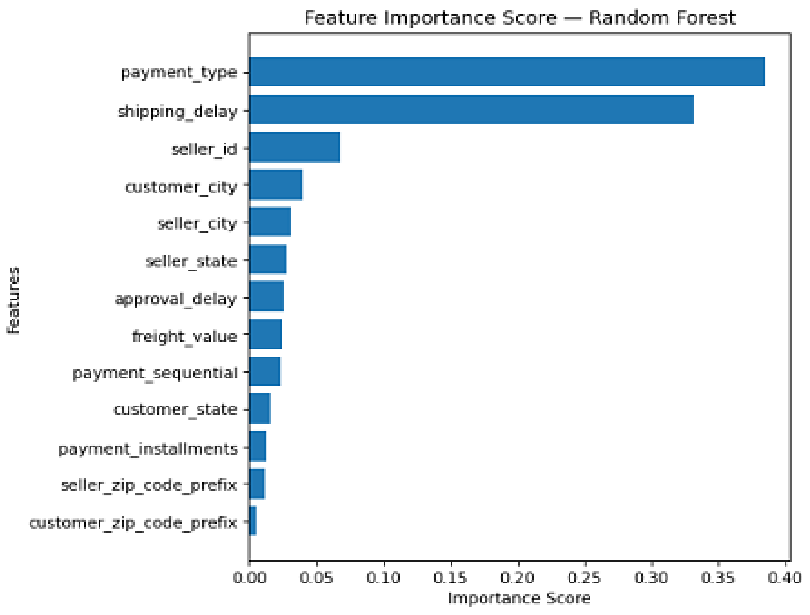
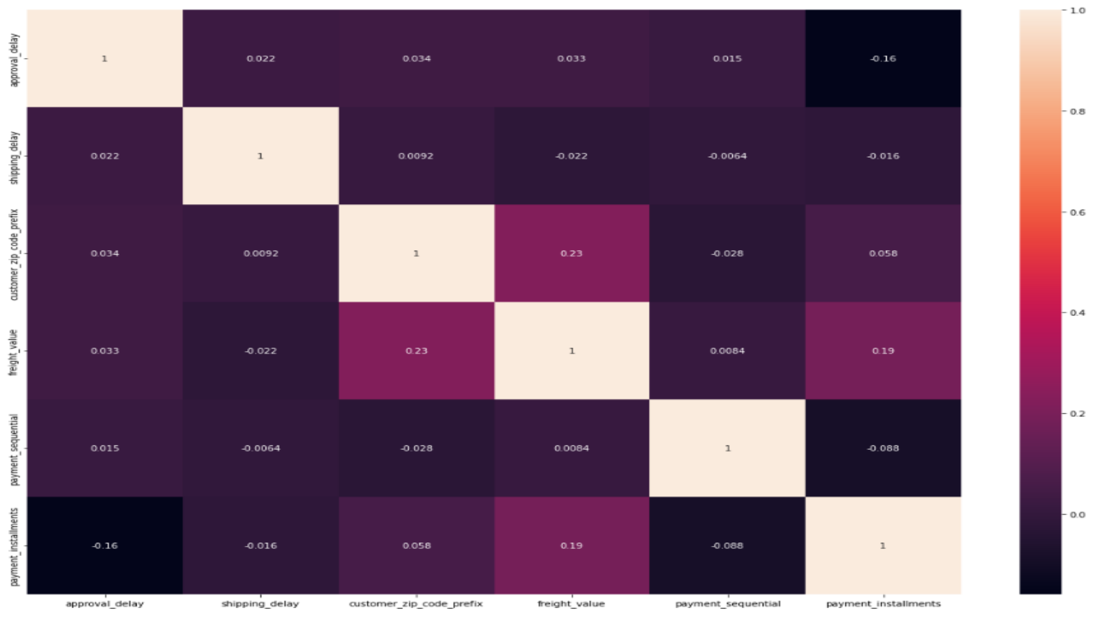


Figure 23. Important Features For Modeling.

Check Highly Correlated Features



	approval_delay	shipping_delay	customer_zip_code_prefix	freight_value	payment_sequential	payment_installments
approval_delay	1	0.022	0.034	0.033	0.015	-0.16
shipping_delay	0.022	1	0.0092	-0.022	-0.0064	-0.016
customer_zip_code_prefix	0.034	0.0092	1	0.23	-0.028	0.058
freight_value	0.033	-0.022	0.23	1	0.0084	0.19
payment_sequential	0.015	-0.0064	-0.028	0.0084	1	-0.088
payment_installments	-0.16	-0.016	0.058	0.19	-0.088	1

Figure 24. Feature Correlation

I drew a heatmap to see the correlation between the numerical features that I have chosen for modeling purposes (Figure 24) and find out that some features are slightly correlated. but their correlation is between -.2 to .2 (Figure 24). As their correlation is not that high, so I keep all the selected numerical features for my modeling purposes.

Leave One Out Encoding

My modeling features contain some categorical values such as seller_state, seller_city etc. So I need to encode those features before feeding my model. For encoding purposes, I have chosen the Leave One Out Encoding technique. Leave One Out Encoding usually calculates the mean of the target features for all the records containing the same value for the categorical feature (P, 2018).

Undersampling

As my project goal is to correctly identify the late delivery of the orders from timely delivery and my dataset is highly unbalanced (Figure 12), So I need to do some undersampling before modeling. For undersampling I have chosen the Near-Miss algorithm where I kept all the examples from the minority class, and selected examples from majority classes (Brownlee, 2020). Before undersampling there were a total 113202 rows in the data set and after undersampling it comes to a total 17684 rows.

Split the Dataset for Training and Testing Purpose

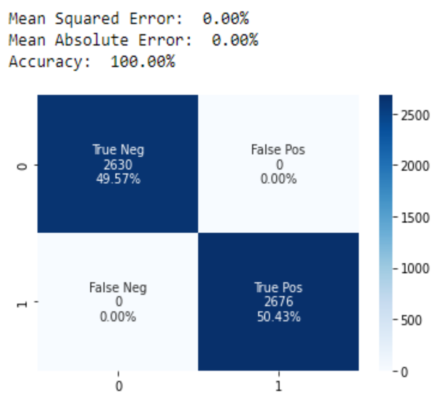
Before modeling, I splitted the dataset into training and testing sets. I kept the training and testing ratio as .7:.3.

Modeling

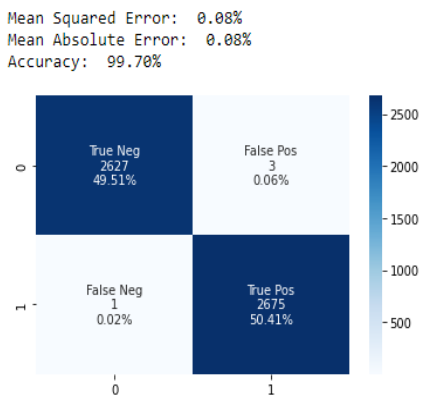
CRT and RandomForest from sklearn libraries have been used for modeling purposes (Navlani, 2018).

Results

I draw two confusion matrices based on the predicted results on both models (T, 2019).



```
Mean Squared Error:  0.00%
Mean Absolute Error:  0.00%
Accuracy:  100.00%
```



```
Mean Squared Error:  0.08%
Mean Absolute Error:  0.08%
Accuracy:  99.70%
```

Figure 25. Confusion Matrix on CRT **Figure 26. Confusion Matrix on Random Forest**

The accuracy on CRT is 100% while the accuracy on the Randomforest is 99.70%. In RandomForest the True Negative and True Positive rate is around 49.51% and 50.41% respectively while the total number of False Positives is only 3 which is around .06%. And the total number of False Negatives is only 1 which is only .02%. So it can be said that the model performance is very good.

Conclusion

Based on the Olist Dataset analysis I have some findings and suggestions, those are given below.

- Most of the late delivery occurs due to the seller side and their logistics partners. Whenever sellers lately deliver their products to the logistic partner after the shipping deadline, a large number of late deliveries occours (Figure 14).
- However, a quite good number of late deliveries happened which is around 6.19% (Figure 15), even though the seller handed their product to their logistic partner before the deadline. So the Olist need to think about their logistic partners.
- Payment types have shown a very little impact on the late delivery performance. Most of the fastest transactions happened whenever the customer used a credit card or voucher. So the Olist may encourage their customers to use more credit cards for payment.
- The delivery performance decreases as the total number of orders increases from a particular state or city or even a specific seller(Figure 16-18). So Olist needs to give extra attention from the city or state where most of the orders are coming from.

Appendix

Git and Github have been used throughout this project to track the progress. The code source of my github account is given below.

Src: https://github.com/smrahman0009/STAT_5620

References

Brownlee, J. (2020, January 20). *Undersampling Algorithms for Imbalanced Classification*. Machine Learning Mastery. Retrieved April 19, 2022, from https://machinelearningmastery.com/undersampling-algorithms-for-imbalanced-classification/

Data Science Process Alliance. (n.d.). What Is CRISP DM? Retrieved April 18, 2022, from https://www.datascience-pm.com/crisp-dm-2/

Jupyter. (n.d.). Jupyter. Retrieved April 18, 2022, from https://jupyter.org/

Magioli, F., & D. (n.d.). *Brazilian E-Commerce Public Dataset by Olist*. Kaggle. Retrieved April 18, 2022, from https://www.kaggle.com/datasets/olistbr/brazilian-ecommerce

Navlani, A. (2018, December 28). *Decision Tree Classification in Python Tutorial*. Datacamp. Retrieved April 19, 2022, from https://www.datacamp.com/community/tutorials/decision-tree-classification-python

P, P. (2018, June 18). *Leave One Out Encoding for Categorical Feature Variables on Spark*. Mawazo. Retrieved April 19, 2022, from https://pkghosh.wordpress.com/2018/06/18/leave-one-out-encoding-for-categorical-feature-variables-on-spark/

T, D. (2019, July 25). *Confusion Matrix Visualization*. Medium. Retrieved April 19, 2022, from https://medium.com/@dtuk81/confusion-matrix-visualization-fc31e3f30fea