THE UNIVERSITY OF SYDNEY

Olist: E-Commerce Analytics

Predicting Order Delivery Time

Group 14

SIDs:

530519221
530062808
530116042
530524786.

1. Executive Summary

Within the ecommerce business landscape, consumers are increasingly expecting and prioritising brands that deliver orders by the promised date. The rise of machine learning provides an opportunity for businesses to improve the accuracy of estimated delivery dates, which is crucial in managing expectations and in ensuring satisfied customers.

Olist is a Brazilian ecommerce marketplace that has collected and provided a comprehensive dataset capturing approximately 100,000 orders from 2016 to 2018. This report aims to analyse this given dataset to reveal insights into what affects delivery time and how such insights can be leveraged to predict the number of delivery days from order purchase timestamp to actual order delivered customer date.

This report will:

- Provide an Exploratory Data Analysis to reveal patterns and relationships within the data.
- Discuss the Feature Engineering techniques used to transform raw data into meaningful features.
- Demonstrate the model experimentation process.
- Recommend and justify the implementation of a LightGBM model to address this business problem.
- Discuss a strategy to deploy our machine learning solution.

2. Problem Formulation & Business Understanding

2.1. Business problem definition

A common challenge faced by ecommerce businesses is ensuring orders are delivered to customers by the promised date. Research has shown that delivery performance impacts review scores, whereby customers provide lower ratings to orders delivered late and higher ratings for orders delivered early (Jain, 2022).

Given historical data about orders made at Olist, the business question that machine learning has the potential to solve is what factors affect the number of delivery days and how can we leverage this information to provide a more accurate estimation of the delivery date.

The outcome of the machine learning model is to predict the number of delivery days between the order purchase date and the actual delivery date. Given that the target variable is a continuous, numerical value, this business problem will be a supervised regression task.

An example of a business use case is when the business is experiencing an increasing number of orders being delivered late, resulting in a high volume of customer complaints. In this situation, using machine learning can benefit both the customers and the business, whereby:

- Accurate prediction of order delivery time can help provide a more realistic estimate to the customer.
- Businesses can understand and proactively address the factors that affect delivery time.

Therefore, measurable value is created through improving on-time delivery rates, increasing customer satisfaction scores and reducing churn when customers receive their orders within the promised timeframe.

2.2. Justification for machine learning

There are three main reasons to use machine learning for this problem:

1. Machine learning has the capability to make predictions based on historical data.
2. Machine learning can effectively process large datasets to make predictions. The given Olist dataset consists of around 100,000 data points. This vast amount of available data makes it impractical to use traditional analytics, which are not designed to handle a large scale of big data (Infomineo, 2024).
3. Machine learning can help uncover hidden patterns or relationships within the data, enabling businesses to understand the important factors that contribute to the order delivery time.

Subsequently, the potential benefits of applying machine learning include:

1. Improving effectiveness of decisions by allowing businesses to leverage the predictions made by machine learning models to inform strategies for optimising order delivery time. For example, machine learning can reveal potential factors contributing to delays. In response, businesses then know what actions to take to address the issue and minimise its negative impact.
2. Enhancing the customer journey experience through providing accurate and reliable predictions about the delivery dates, which is essential in managing customer expectations. When customers receive their order by the promised date, they are more likely to be satisfied, hence boosting company reputation.

2.3. Business success metrics

Business metrics provide insight into the business impact of the model, which in this case includes:

1. **On-Time Delivery (OTD)**, which measures the percentage of orders fulfilled by the expected delivery date (Lopienski, 2024).
2. **Customer satisfaction score (CSAT)**, which can be typically measured on a scale of 1 to 5. The score can tell businesses how happy their customers are with the order fulfillment process.

To evaluate the predictive performance of the model, the **Root Mean Square Error (RMSE)** is used, which measures the difference between the predicted and actual delivery time value. The aim is to find the model that provides the smallest RMSE on test data.

Both types of metrics help to evaluate the success of machine learning models. They also reflect broader strategic business goals of:

1. Reducing customer churn by improving their experience with the company.
2. Increasing sales and revenue by attracting new and existing customers who are inclined to buy from businesses with a shorter delivery time.

Therefore, having an accurate estimated delivery date will be essential in achieving these goals.

3. Exploratory Data Analysis (EDA)

3.1. Overview of the training dataset

3.1.1. Target Variable Y - Number of Delivery Days

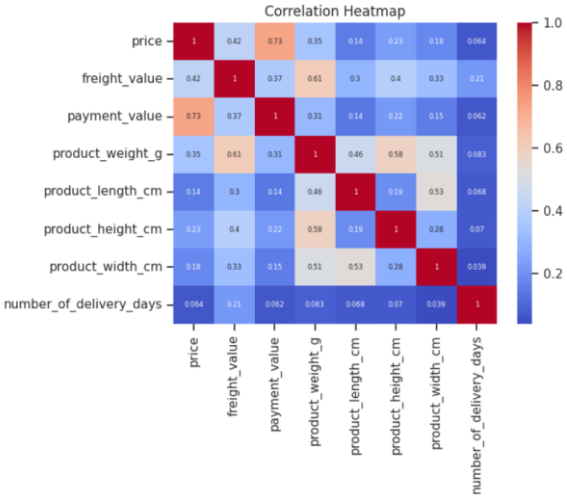
The exploratory analysis of Number of Delivery Days in the dataset of 79,233 observations reveals that the average delivery time is approximately 12 days. Key statistics include a lower quartile (Q1) of 6 days, a median of 10 days, and an upper quartile (Q3) of 15 days, indicating that 50% of observations fall between 6 and 15 days of delivery. The standard deviation is 9.35, reflecting considerable variability in the data. As shown in Figure 1, the distribution analysis shows a right-skewed pattern, with histograms indicating that most orders took around 12 days to be delivered. This skewness suggests a relatively small number of orders require exceptionally long delivery time, which extends the distribution's tail to the right.

Figure 1: Distribution of Delivery Time

3.1.2. Continuous Predictors

In our dataset, the continuous predictors include: price, freight value, payment value, product weight, length, height and width.

Figure 2: Correlation heatmap between continuous predictors



From the correlation heatmap in Figure 2, we observe that the target variable shows the strongest correlation with freight value but overall has a weak correlation with all numeric predictors. This suggests that linear relationships are minimal and non-linear models may be more appropriate. Moreover, some predictors are highly correlated with each other, including:

- Price and Payment (0.73)
- Product Height and Product Weight (0.58)

These correlations could introduce multicollinearity and should be monitored in model training. Although no single feature has strong predictive power individually, variables like freight_value and product_weight_g show relatively higher correlations with delivery days and may still contribute meaningfully when combined in a multivariate model.

Our exploratory analysis of continuous predictors revealed several important insights. We observed potential outliers in payment_value, which may require further treatment. Several numerical features, including product dimensions and prices, exhibited skewed distributions, suggesting the need for transformation to improve model performance. Visualisations also indicated possible non-linear relationships between some predictors and the target variable, number_of_delivery_days. The correlation heatmap highlighted potential interaction effects among product_length_cm, product_height_cm, and product_width_cm. Among all features, freight_value showed the strongest linear correlation with the target ($r = 0.214$).

3.4.3. Categorical Predictors

Categorical predictors in our dataset include: customer cities and states, seller cities and states, product category name, and payment type. An important takeaway from our analysis is Phi-K Correlation between Customer City vs Delivery Time was 0.67 (highest). This suggests that depending on where the customer lives, this can impact the delivery time.

3.4.4. Datetime Predictors

We decomposed purchase timestamp and shipping limit dates into hours of day, day of week, and month. Notably, these predictors exhibited a moderate Phi-K correlation (~0.22) between the month of the year and delivery duration, as shown in Figure 3. This suggests potential seasonal influences on delivery times, such as increased delays during peak shopping periods or holidays, which may need further investigation.

Figure 3: Number of Delivery Days by Purchase Timestamp Month

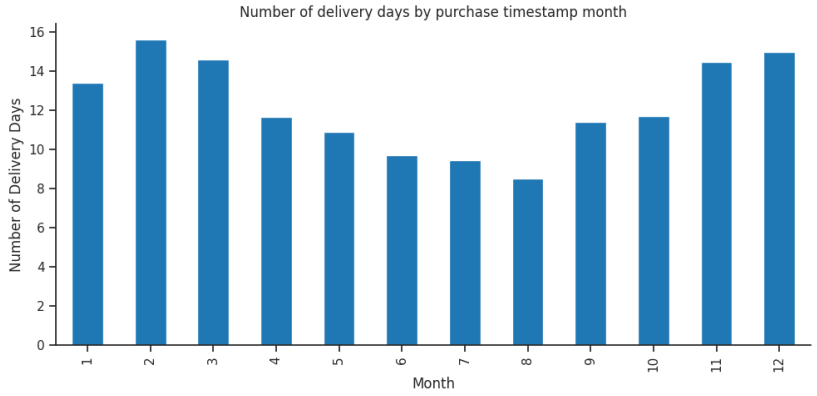
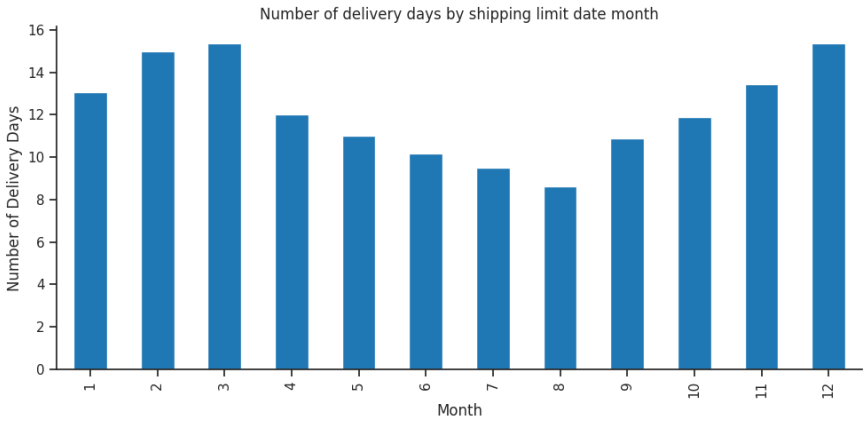


Figure 4: Number of Delivery Days by Shipping Limit Date Month



3.4.6. Geographical Predictors

We explored geographical patterns using customer and seller coordinates provided via longitude and latitude. These maps provide insight into potential delivery challenges associated with regional distances, remote areas, or clustering around urban centers. For example, greater distances between customers and sellers may lead to longer delivery times, thus customer-seller distance might be a potential predictor in our model to improve delivery time estimation.

3.4.7. ID Predictors

Analysis revealed that certain sellers and products are associated with consistently longer delivery times. This suggests a potential influence from product characteristics, such as size, weight, or category (e.g., bulky or fragile items), as well as fulfillment behavior specific to certain sellers, indicating that interaction effects involving these IDs may be valuable in feature engineering.

4. Feature Engineering

4.1. Created Features

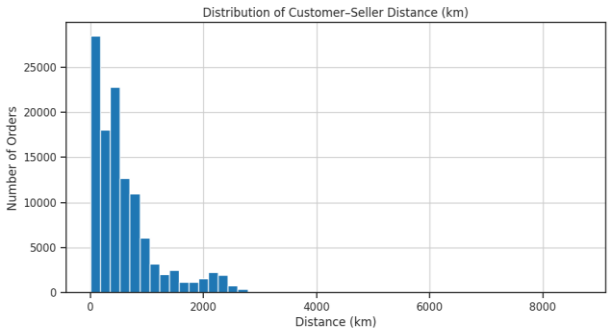
4.1.1. Created Feature 1: Distance between Customer and Seller

The first created feature is customer_seller_distance_km. As discussed earlier in part 3.4.6., this feature is potentially useful for predicting delivery durations, as longer distances may influence logistics times and customer satisfaction. This variable represents the great-circle distance (in kilometres) between the latitude and longitude coordinates of the customer and the seller. The coordinates were first obtained by mapping the zip code prefixes of customers and sellers to their average latitude and longitude using the olist_geolocation_dataset. These averages were then merged into the main dataset. To calculate the actual distance, we applied the Haversine formula (r = 6371 km, which is Earth's radius):

$$\text{customer\_seller\_distance\_km} = 2r \cdot arcsin\left(\sqrt{sin^2\left(\frac{\Delta lat}{2}\right) + cos(lat_1)cos(lat_2)sin^2\left(\frac{\Delta lng}{2}\right)}\right)$$

The regression plot with the target variable shows a steeper line, indicating a potential relationship, though the patterns in the data are not immediately obvious. The correlation coefficient of 0.4 further suggests that this variable could be a moderate predictor of the delivery time. The distribution graph and regression plot is shown as follow:

Figures 5 & 6 Histogram and Regression Plot vs.Target Variable of Customer-Seller Distance



However, we identified a few outliers that appear to skew the model training and hinder generalization. In particular, the regression plot revealed four outliers with distance larger than 4000 km but unusually short delivery time. These outliers do not reflect the typical relationship between distance and delivery duration, and their presence may distort the model's understanding of this pattern. To mitigate their influence, we removed these points by introducing a capped variable, customer_seller_distance_km_capped, which excludes extreme distance values beyond a reasonable threshold.

4.1.2. Created Feature 2: Square of Distance between Customer and Seller

Given the non-linear relationship between the distance and delivery time, we tried incorporating the square of the distance to better capture this relationship, and it did improve our model as shown in Appendix A.

4.1.3. Created Feature 3: Product Size

As discussed in Section 3.4.7, bulkier product dimensions may contribute to longer delivery times due to increased logistical complexity. To better capture this relationship, we engineered interaction features based on the product's physical dimensions - specifically, the length, height, and width. This

included both the three-dimensional interaction term (length × height × width) and all possible two-dimensional combinations. Compared to the original individual predictors, these interaction terms show stronger correlations with the target variable. Notably, the three-dimensional interaction feature exhibits the highest correlation, with both the Pearson and Phi-k coefficients approximately equal to 0.1. Given its improved predictive potential, this feature will be retained for model training and evaluation in later stages.

4.1.4. Other Created Features:

There are 5 other created features in the dataset (Appendix __). Firstly, two features are created - Sellers Process Time and Seller Shipping Delay - due to the fact that delivery time is affected by the process of the seller, and some sellers may be faster or slower than others (as discussed in part 3.4.7). Secondly, Number of Items in Order, which means if an order has a larger quantity of product, it might take longer to collect the product from sellers and distribute to customers. Thirdly, Average Delivery Time by Product, as discussed in part 3.4.7, some products were observed to have longer delivery time than others. Lastly, Order Approval Processing Time, as the longer the time to process an order for approval, the longer the delivery time.

4.2. Feature transformations & engineering techniques

A key feature transformation that we performed was the encoding of categorical variables. The predictors Payment Type, Product Category Name English, Customer State and Seller State were encoded for our Multiple Linear Regression, Generative Additive Models, LightGBM model and Deep Learning models.

By encoding the categorical variables, we were able to include them into the MLR, which cannot naturally handle non-numerical values. We performed dummy encoding for the MLR model to avoid the issue of perfect multicollinearity. By including these categorical variables into our baseline MLR model, there was a significant increase in the r-squared value, and decrease in the MAE and rMSE values.

Evaluation Metric	MLR without categorical variables	MLR with categorical variables
RMSE	8.288	8.056
MAE	5.166	4.978
R^2	0.217	0.260

However, a trade-off associated with the encoding of these categorical variables, was the resulting high cardinality. Customer State, Seller State and Product Category Name had high numbers of categories, at 27, 22 and 71 respectively. However, we determined that compared to other categorical variables such as customer city or seller_id, our chosen variables had relatively low numbers of categories and therefore decided on dummy and one-hot-encoding as our approaches. We also wanted to avoid the issue of target leakage which would have occurred if we chose an alternative approach of target encoding.

4.3. Handling missing values & outliers

4.3.1. First Data Cleaning

To construct a modeling-ready dataset for predicting delivery duration, we performed a multi-step merge across several tables in the Olist E-commerce dataset from the beginning. The process began with the core orders table, which was incrementally joined with other relevant tables using key identifiers such as order_id, product_id, and seller_id, allowing us to gather all essential customer, product, order, review, and logistics information into a single dataset.

After merging, we conducted an initial data cleaning phase to ensure the dataset was well-suited for our prediction task. Firstly, we dropped the original Portuguese column product_category_name, since we had already merged in its English-translated counterpart. Secondly, columns such as review_comment_title and review_comment_message were removed, which contain free-text data with high rates of missing values and are not directly useful for predicting delivery time, especially since customer reviews occur after the delivery event. Thirdly, we identified and corrected typographical errors in column names, such as product_description_lenght. Fourthly, all timestamp fields were converted from string to proper datetime format to enable accurate time-based computations and feature engineering.

A critical part of this preprocessing stage involved creating the target variable: number of delivery days. We calculated this by subtracting order_purchase_timestamp (the time the order was placed) from order_delivered_customer_date (the time the customer received the order). Prior to calculating this difference, we validated the timestamp values to check for anomalies - such as incorrect years, months, or out-of-range time entries - but found that the values were consistent with the expected dataset time range, confirming their reliability for target variable construction.

4.3.2. Data Splitting

We then adopted a two step split approach to the fully merged dataset, which resulted in 80,926 observations in the training set, and 34,683 observations in each validation and test set. To avoid data leakage, EDA was performed exclusively on the training set. This ensures that insights and transformations do not incorporate information from validation or test sets, which are strictly reserved for hyperparameter tuning and final performance evaluation.

4.3.3. Second Data Cleaning

Our data cleaning focused on two key areas: handling missing values and preventing data leakage, both essential for ensuring model reliability.

We removed 1,648 observations missing number_of_delivery_days, which resulted from undelivered orders with statuses like shipped, cancelled, or processing. As these orders lacked order_delivered_customer_date, they could not contribute to our target variable and were excluded. The order_status column was also dropped, as it held no further predictive value. One row contained missing product dimensions and weight, linked to a product purchased only once. As this was isolated and random, we removed the row. For geolocation data, we imputed missing latitude and longitude values using city-level medians. However, 34 entries lacked matching cities in the geolocation dataset and were removed.

To avoid data leakage, we dropped columns unavailable at prediction time, including order_delivered_customer_date and all review-related fields (e.g., review_score, review_comment_message). These features, occurring post-delivery, could otherwise introduce unintended bias.

4.4. Feature selection process

Before creating our models, we identified that including any of the predictors in the Olist_order_reviews Dataset in our model would all cause data leakage. Since our business problem aims to replace the current estimated delivery date, including order_estimated_delivery_date in our predictions would be counterintuitive and we therefore decided to exclude it. Review data is typically only available after the customers have received their purchase, therefore using any of these variables for our delivery days prediction, would result in an overconfident model that would underperform when implemented due to the issue of data leakage.

When creating our baseline model we performed variable selection on our MLR models. We began by including all numerical variables and our created features Customer Seller Distance and Customer Seller Distance ^ 2. We then performed backwards and forwards selection to identify which variables contributed most to the r-squared value and those which contributed the least, some even decreasing it. Using the results of the forwards and backwards selection, correlation scores with the target variable and some trial and error, we identified that the subset of numerical variables that resulted in the lowest rMSE and MSE values were customer_seller_distance_km_capped,, average_seller_process_time_capped, order_purchase_timestamp_day and customer_seller_distance_km_capped_squared order_purchase_timestamp_month. This feature selection process improved the evaluation metrics significantly as seen in the table below.

	RMSE	MAE	R^2
MLR (all numerical predictors)	8.733	5.381	0.137
MLR (selected numerical predictors)	8.288	5.166	0.217
Baseline Model - MLR (selected numerical and categorical predictors)	8.056	4.978	0.260
Ridge (selected numerical & categorical)	8.056	4.977	0.260

For the GAMs model we used a similar method of basing our feature selection process on the forward and backward selection performed on the numerical features. However, for the random forest models, all the features were included, since a characteristic of the model itself is to choose the most significant predictors. We created feature importance graphs based on the results of our random forest models and this guided the feature selection process for the gradient boosting methods, including Light GBM.

5. Recommended Models & Justification

5.1. Baseline Model

The baseline model we chose was a multiple linear regression with the following variables:

- Customer Seller Distance
- Customer Seller Distance Squared
- Average Seller Process Time
- Order Purchase Timestamp Day
- Order Purchase Timestamp Month
- Payment Type
- Seller State
- Customer State
- Product Category Name English

This was selected as the baseline model, as it had the best rMSE and MAE values at 8.056 and 4.978 respectively, out of all the other MLR models after careful feature selection. Since interpretability is not a key requirement of our model, the number of predictors, which increases significantly when encoding the categorical variables, means other simpler models weren't favoured over our chosen baseline.

	RMSE	MAE	R^2
Baseline Model - MLR (selected numerical and categorical predictors)	8.056	4.978	0.260

5.2. Best Single Model

The single model that provided the best performance was LightGBM, which included the following variables:

- Freight Value
- Payment Type
- Customer Seller Distance
- Customer City
- Customer State
- Seller City
- Seller State
- Payment Type
- Order Quantity
- Product Size
- Order Purchase Timestamp Month
- Shipping Limit Date Month
- Order Approved At Month
- Order Delivered Carrier Date Month
- Average Seller Process Time
- Average Delivery Time by Product

During model experimentation, we discovered that "Order Approved At Month" and "Order Delivered Carrier Date Month" significantly improved model performance, suggesting that they provide valuable information for predictions.

This improvement is further justified from a technical perspective, whereby the evaluation metrics showed the lowest error on validation dataset compared to other models.

RMSE	MAE	R^2

6.758	3.814	0.480

The increase in R^2 suggested an improvement in the model's ability to explain the variation in the number of delivery days by the inputs.

5.3. Model Stack

Choosing one single model might have certain limitations, such as different models capture different patterns in data or model ranking can change with small shifts in test data **(article)**. Therefore, to further improve prediction performance, we conducted a Model Stacking approach to leverage models' complementary strengths. Since Model Stacking is most effective when the base models are different, we choose stacking Multiple Linear Regression (best in capturing linear relationship, Appendix A), and LightGBM (for best performer in non-linear relationship, Appendix A).

Given the huge computational cost that each single model already had, instead of stacking models using model averaging with k-fold cross-validation, Blending method was preferred. This approach is a more simplified stacking that fits the meta-learner on validation test predictions. The result is in the table below:

RMSE	MAE	R^2
6.7612	3.8047	0.48

5.4. Recommended model for deployment

After evaluating multiple models, we recommend LightGBM as the most suitable single model for deployment. It offers a strong balance between accuracy, efficiency, and ease of integration, outperforming simpler models like Multiple Linear Regression in predictive accuracy. Moreover, LightGBM especially attractive is its relatively low computational cost compared to model stacking, and its ability to handle large datasets efficiently without requiring a complex deployment pipeline. However, one limitation of this model is that it only works post-purchase due to variables such as order approval and shipping timestamps, which will be discussed in detail in part 8.1.

Model stacking, on the other hand, did not perform better than LightGBM, and also introduced additional system complexity and maintenance challenges. Therefore, we recommend LightGBM as the preferred model for initial deployment, especially when considering resource constraints and the need for stable, scalable performance.

6. Model Evaluation & Performance Analysis

6.1. Performance Evaluation of the Three Recommended Models

To quantify model performance, three metrics were used: Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R^2 score. The results are shown in the next section - 6.2.

6.2. Robustness and generalisation analysis

	Validation			Test		
	RMSE	**MAE**	**R^2**	**RMSE**	**MAE**	**R^2**
MLR	8.056	4.978	0.260	8.056	4.978	0.260
LightGBM	6.758	3.814	0.480	7.886	5.518	0.292
Stacked Model	6.761	3.805	0.480	7.873	5.409	0.295

Table _. Results of Recommended Models (Validation and Test Sets)

To evaluate the robustness and generalisability of each model, their performance was compared across the validation and test datasets using RMSE, MAE, and R^2 metrics.
The MLR model shows identical performance on both validation and test sets (RMSE = 8.056, MAE = 4.978, R^2 = 0.260). This suggests that the model generalises consistently. However, its predictive power is weaker compared to the other two models, as indicated by its RMSE and MAE values.

In contrast, both LightGBM and the Stacked Model perform better on the validation set. From validation to test, the RMSE for both models increases by approximately 1.1, MAE rises by around 1.6, and R^2 declines by roughly 0.18. This drop suggests a potential overfitting issue, where the model appears to capture validation-specific noise rather than generalisable trends. Despite this, the Stacked Model consistently achieves the lowest RMSE and MAE on both validation and test sets, making it a stronger candidate for deployment in practical scenarios.

6.3. Error analysis

Error analysis was conducted to examine the nature of the residuals. All three models exhibit a centralised error distribution with a slight positive skew, indicating occasional underestimation of delivery time.

- **Distribution of Errors**: The residual distribution for the stacked model is the sharpest among all three, highly centralised near zero, suggesting the lowest average prediction error.
- **Q-Q Plot**: The residuals for the three models are approximately normally distributed in the middle. However, at both ends of the plot, the points curve away from the red line, especially the upper tail. This suggests that there exists an impact of outliers or there is still an underlying pattern that the model cannot capture.
- **Errors vs Predicted**: For LightGBM and the Stacked Model, more errors are clustered around 0. Their underestimation is more localised, with fewer but larger individual misses. On the other hand, MLR's error is more systematic.
- **Boxplot**: LightGBM and the Stack Model's boxplot exhibit fewer outliers than MLR, demonstrating better handling of potential variance.

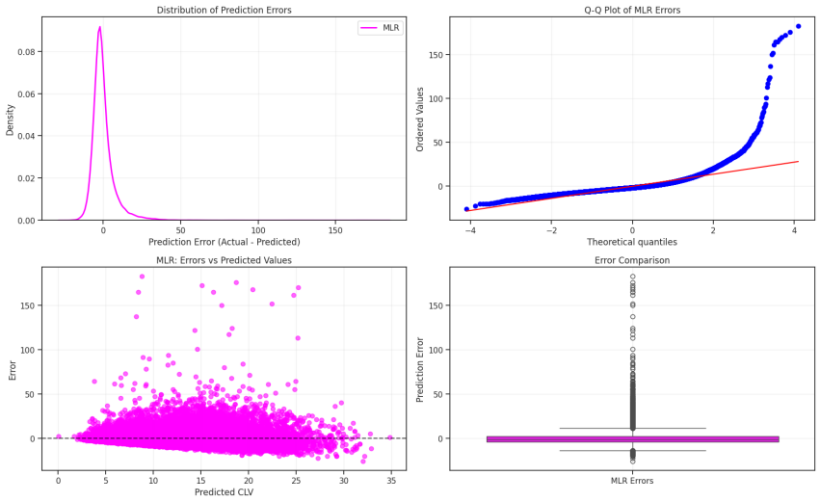


Figure 6. MLR Error Analysis

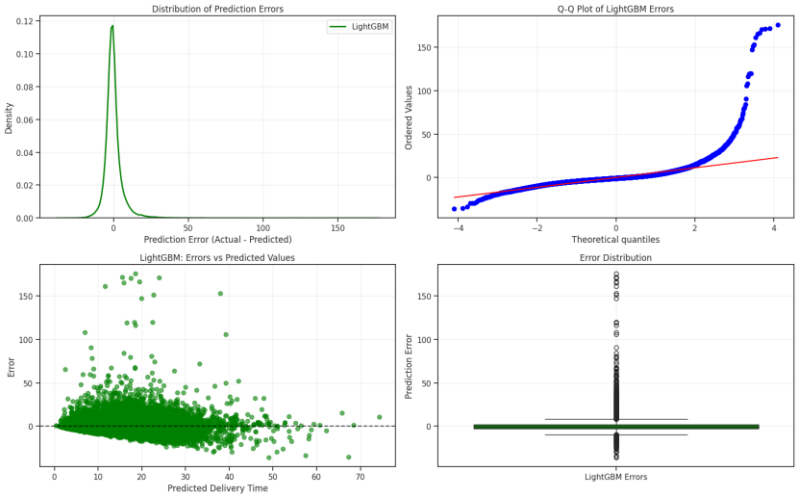


Figure 7. LightGBM Error Analysis

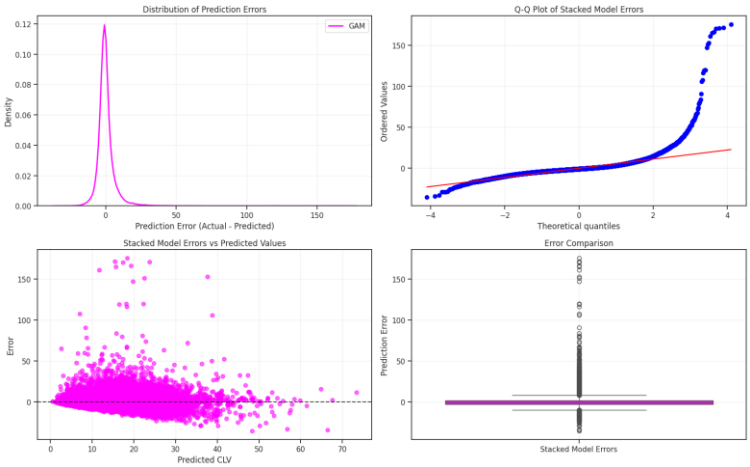


Figure 8. Stacked Model Error Analysis

7. Business Insights

7.1. Key business insights

An insight that was revealed by the analysis is that customer-to-seller distance is one of the most influential predictors of delivery duration, as captured by the engineered feature “customer_seller_distance_km”. It is evident that longer distances were strongly associated with extended delivery times. For example, customer_seller_distance_km was a selected predictor from backward selection, and contributes significantly to gradient boosting models, as seen in their feature importance graphs. This insight suggests that regional optimisation of inventory or promotion of closer sellers could yield improvements in delivery efficiency.

Another insight was that another variable that contributed much to the prediction is “average_seller_process_time”. This is because some sellers might tend to take longer to process orders, resulting in a longer delivery timeframe, as discussed in 3.4.7.

7.2. Actionable recommendations

One recommendation by this report is to promote sellers with faster processing times. While the platform cannot force sellers to process orders more quickly, it can encourage behavioural change through incentives. Olist can highlight or badge "fast-processing" sellers, prioritise their listings in search results, or include them in promotional campaigns. Over time, such visibility-driven rewards may create competitive pressure and encourage slower sellers to improve operational efficiency.

The second recommendation is to inform customers about expected delivery durations in real-time, ensuring transparency. The platform could display estimated delivery ranges on product pages based on predictive models. This can help set realistic expectations and improve customer satisfaction as they are proactively being notified about potential delays for bulky or distant shipments.

The last recommendation is to suggest sellers or products based on location. Although geographic distance is not alterable, its negative impact can be reduced by improving recommendation systems. For example, the platform can suggest similar products from sellers geographically closer to the customer. This ensures that customer preferences are still met, while potentially reducing delivery durations.

7.3. Limitations of insights and assumptions

While this analysis provides actionable insights into the drivers of delivery duration, it is important to acknowledge the underlying assumptions and practical limitations that may impact the applicability of the findings.

A core assumption that motivated the selection of this prediction problem is that delivery duration directly influences customer satisfaction, loyalty, and lifetime value (CLV). The premise is that faster deliveries result in more satisfied customers, who are in turn more likely to repurchase and maintain a long-term relationship with the platform.

Another assumption is that logistics and delivery carriers are reliable and consistent across transactions. In practice, factors such as courier performance and regional logistics infrastructure

could contribute to delivery times but are not captured in the available features. This might limit the model's ability to fully explain delays or inform carrier selection strategies.

From a limitation standpoint, Olist lacks control over which sellers join the marketplace, meaning that it is not feasible to selectively retain only high-performing sellers based on processing speed. However, the platform can promote or incentivise fast-processing sellers as mentioned in the previous section. These indirect levers could help shift seller behaviour over time.

Furthermore, the most influential feature, customer-to-seller distance, presents a practical constraint. While it is a strong predictor of delivery duration, it is largely outside the platform's control once a customer selects a seller. Unlike features that can be engineered or influenced (e.g., seller process time), geographic distance is fixed. Olist could explore strategic warehousing or region-based recommendations to partially mitigate this issue, but direct intervention is limited.

8. Business Implementation & Impact

8.1. Deployment strategy

For **short-term strategy**, LightGBM is more accurate because it uses extra information like when the order is approved or when the seller ships the product. However, since the model uses unseen variables at checkout, LightGBM cannot be used for real-time customer-facing predictions. Instead, it is better suited for supporting Logistics Operations and Customer Service teams by refining delivery timelines after the purchase. It can also be used to trigger proactive alerts in case of delays, as Caruelle et al. (2023) found that even when delays occur, timely notifications can help maintain customer satisfaction. In addition to the Logistics and Customer Service teams, the Engineering team would manage system integration, while the Data Science team would be responsible for regular model updates and monitoring.

To ensure the solution evolves with Olist's needs, we also propose a **long-term strategy**: integrating a stacked model, combining the strengths of both LightGBM and Multiple Linear Regression (MLR).

In this setup, MLR would provide an initial estimate at checkout using only features available at the time of purchase - ideal for fast, real-time delivery estimates. It can be deployed on Olist's website to replace the business's current rule-based estimated delivery time shown to customers at checkout. As shown in Figure __, the model outperforms Olist's existing method by producing lower RMSE scores and a more centered, consistent residual distribution. Since it requires very little computing power, it can give results immediately. Therefore, Customer Service and UX/Product Teams would perform front-end deployment and integration, while the Data Science team would keep and manage the model in the long-term.

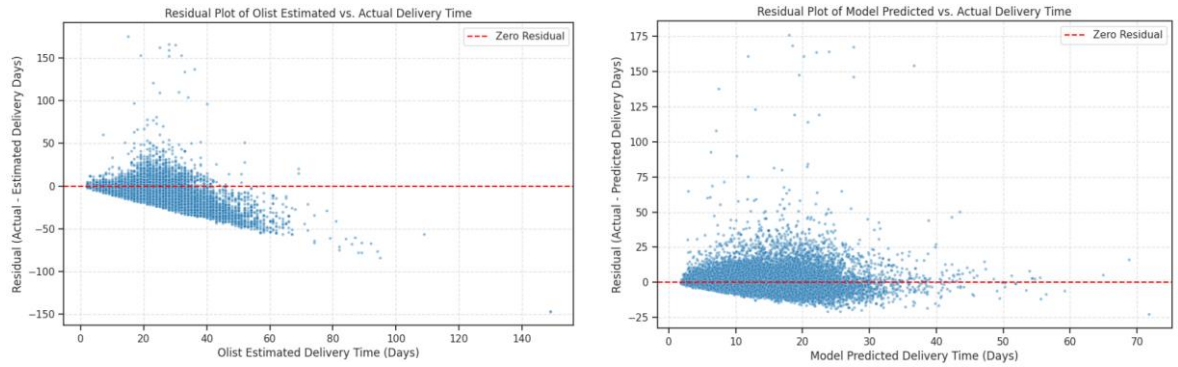


Figure 9. Residual Plots: Olist Baseline vs. Model Prediction Accuracy

Once post-purchase data becomes available (e.g., order_approved_at, order_delivered_carrier_date), LightGBM would generate a refined prediction based on a more complete picture. The final estimate would be the output of a meta-model that intelligently combines both predictions.

Although more complex, this two-phase system offers the best of both worlds: speed and simplicity for customers, and accuracy and adaptability for internal operations. The Engineering team would manage the API and data flow integration, while Data Science would oversee model orchestration and retraining. Customer Service and Logistics teams could then use updated predictions for proactive communication and planning.

Given Olist's scale and technical capability, implementing this stacked approach is feasible in the long term and would provide a more resilient and flexible solution.

8.2. Risks, challenges, and mitigation strategies

Even with a strong model, there are challenges to consider. We also propose some strategies below to mitigate the challenges below:

8.2.1. Model Drift

Given that our dataset is from 2016 - 2018, a key risk is model drift, which may reduce accuracy over time as business and consumer behavior changes. Concept drift happens when the way certain features relate to delivery time starts to change. For example, if sellers begin shipping orders more quickly because of better logistics, or if Olist launches new same-day delivery options, the model's original patterns may no longer apply. Data drift is when the input data itself starts to look different from what the model was trained on. For instance, if a product category, like home appliances, suddenly becomes more popular, the overall delivery pattern would shift. Therefore, there is a need for a regular retraining schedule using recent data (e.g., monthly or quarterly), along with drift detection tools to monitor changes in feature distributions and model error rates.

8.2.2. Operational Challenges

Running complex models, especially stacked models, requires sophisticated infrastructure. It can be challenging to keep data flows synchronized, track multiple model versions, and ensure everything runs smoothly without interruptions. To manage this, we recommend starting with the simpler model to ensure stability, then gradually introducing the more advanced models as the system matures. This approach reduces risk and allows the team to adapt to increasing complexity. Moreover, we need to assign the responsibilities clearly, such as the Engineering team ensures system reliability, the Data Science team oversees model performance and updates, and the Operations team applies the predictions in daily workflows.

8.2.3. Ethical Concerns

a) Environmental Impact of Over-Optimization:

Using an optimized model to improve delivery time can unintentionally put pressure on the logistics system to meet tight deadlines. As a result, it may encourage less sustainable delivery practices - such as last-minute air shipments or under-filled delivery routes - which can increase carbon emissions and packaging waste (Das, 2024).

We suggest two practical solutions to address this issue. First, the model can be designed to think about the environmental impact of fast deliveries. For example, it can detect which delivery routes produce more emissions and flag them inside the system, helping the business make more eco-friendly decisions. Second, we recommend giving customers the option to choose between faster and more eco-friendly delivery options at checkout. For instance, Olist could display: "Fast (2-3 days)" or "Eco (4-6 days, lower impact)." This not only promotes transparency but also encourages more sustainable choices without compromising user experience.

b) Customer Trust:

Delivery estimates have a direct impact on customer satisfaction. If the estimated date changes after the order is placed, or if the prediction feels random or unclear, it can lead to confusion, frustration, and a loss of trust in the platform. These issues can also increase pressure on customer service teams, as more customers reach out for clarification or to report concerns.

To build trust, first, Olist should show a time range instead of a fixed date, such as “Estimated delivery: 3-5 business days” to set more realistic expectations and reduce disappointment. Adding a short note like “Based on your location and item size” can also help customers understand the estimate without needing full model details. Second, if the delivery estimate changes, customers should receive a clear, friendly update. Personalizing these messages based on order type or customer behavior can even make them feel more relevant and considerate (Pappas et al., 2017).

8.3. Monitoring & continuous improvement

8.3.1 How the model’s performance will be tracked over time

The model’s performance will be tracked over time, by recalculating the rMSE and MAE evaluation metrics at regular intervals after deployment. During the days after implementation, these metrics should be calculated hourly, to ensure implementation was successful and allow for any required adjustments to be made. After the first few weeks monitoring should be reduced to daily, and a manual weekly review should identify if there are any unexpected issues, determine possible causes and possible corrective features.

8.3.2 Key monitoring metrics

The key metrics that will be used to monitor the model after deployment are the rMSE and the MAE values. The MAE value is the preferred metric due to the nature of the dataset and presence of outliers that may disproportionately affect the rMSE value. Some key business impact indicators include any significant changes in review score, CLV, and other customer satisfaction indicators.

8.3.3 Plan for model retraining or refinement

Concept and data drift are the types of model drift that are likely to require retraining and refinement of the model. If there is an issue of concept drift, where the underlying relationship between the predictors and delivery days changes, the model would have to be retrained to capture these new relationships. For example, if new carriers are used by the sellers, which make the customer_seller_distance a less important predictor in the number of delivery days, then new models should be trained and tested to determine which new model best captures this new relationship between the predictors and number of delivery days.

If there is an issue of data drift, where the distribution of input data changes then the model would have to be refined to incorporate this new data. For example, if a new seller is added to the platform, since they weren’t part of the training dataset, there could be an issue of data drift if not addressed. Therefore the model would have to be retrained on a new dataset, which includes the new seller, to ensure that the model’s performance isn't affected.

8.4. Business impact

By providing more accurate estimates of delivery time, the recommended model can help the business improve customer satisfaction, contribute to a higher customer lifetime value (CLV), reduce uncertainty, and enhance transparency in delivery communications.

According to research, with real-time inventory availability and advanced technology, brands can enhance the accuracy of estimated delivery dates, boosting conversion by 7%, and increasing customer satisfaction by as much as 5 Net Promoter Score (NPS), further amplifying CLV (Zach, 2024). Additionally, delivery time promise policy can improve the sale volumes by 6.1%, demonstrating the importance of providing an accurate delivery timeframe (Salari et al., 2020).

While MLR offers consistency, its limited accuracy restricts its utility in real-time systems. LightGBM and the Stacked Model offer richer insights and better performance, making them more aligned with data-driven, customer-centric business strategies. Assuming a 5% increase in CLV, and an average customer value of $200, the potential uplift in lifetime value across a customer base of 50,000 could amount to $500,000 in additional long-term revenue. Additionally, if we assume the platform averages between 40,000 to 66,000 orders per month, a 3 ~ 5% uplift in conversion could result in an additional 2,000 ~ 3,300 orders monthly, translating to significant revenue growth. Hence, by deploying the recommended model, it is expected to lead to an increase in revenue in the long term, benefiting businesses.

9. Reference List

Caruelle, D., Lervik-Olsen, L., & Gustafsson, A. (2023). The Clock Is ticking—Or Is it? Customer Satisfaction Response to Waiting Shorter vs. Longer than Expected during a Service Encounter. *Journal of Retailing*, *99*(2), 247–264. https://doi.org/10.1016/j.jretai.2023.03.003

Das, B. (2024, September 8). *The Unsustainable Race to Faster Delivery: How Lightning-Speed Promises Are Ruining Our Society and the Earth*. Linkedin.com. https://www.linkedin.com/pulse/unsustainable-race-faster-delivery-how-promises-ruining-bimal-das-ikvjf/

Infomineo. (2024, September 20). *Big Data Analytics Versus Traditional Data Analytics: A Comprehensive Overview*. https://infomineo.com/data-analytics/big-data-analytics-versus-traditional-data-analytics/

Jain, S., & Singh, R. (2022). Impact of delivery performance on online review ratings: the role of temporal distance of ratings. *Journal of Market Analytics*, *11*(2), 149–159. https://doi.org/10.1057/s41270-022-00168-5

Lopienski, K. (2024, December 31). *On-Time Delivery in 2025: KPIs, Challenges, and How to Optimise Your OTD Strategy*. ShipBob. https://www.shipbob.com/au/blog/on-time-delivery/

Pappas, I. O., Kourouthanassis, P. E., Giannakos, M. N., & Chrissikopoulos, V. (2017). Sense and sensibility in personalized e-commerce: How emotions rebalance the purchase intentions of persuaded customers. *Psychology & Marketing*, *34*(10), 972–986. https://doi.org/10.1002/mar.21036

Pavlyshenko, B. (2018, August 1). *Using Stacking Approaches for Machine Learning Models*. IEEE Xplore. https://doi.org/10.1109/DSMP.2018.8478522

Sahai, A., Machiraju, V., Sayal, M., van Moorsel, A., & Casati, F. (2002). Automated SLA Monitoring for Web Services. *Lecture Notes in Computer Science*, 28–41. https://doi.org/10.1007/3-540-36110-3_6

Salari, N., Liu, S., & Shen, Z.-J. M. (2020, September 28). *Real-Time Delivery Time Forecasting and Promising in Online Retailing: When Will Your Package Arrive?* Papers.ssrn.com. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3701337

Zach Zalowitz. (2024, June 6). *The Importance of Accurate Estimated Delivery Dates (EDD) - Fluent Commerce*. Fluent Commerce. https://fluentcommerce.com/resources/blog/the-importance-of-accurate-estimated-delivery-dates/

10. Appendix

Appendix A: Model development & experimentation

Overview of model development approach

Modelling strategy

The general approach to model development was starting with basic regression methods, including simple linear regression and multiple linear regression models. Then, we looked at increasingly complex models including Generalised Additive Models (GAMs), Random Forests, Gradient Boosting and Model Stacking. The goal of the model experimentation process was to create a model that provides the best generalisation ability.

Our modelling strategy evolved during experimentation through iterative refinement, whereby we started with building models using raw features. Then over time, as our understanding of the business problem increased, we started engineering and including more informative features to boost model performance. In the end, we had a wide range of models, in which we then selected the best model in each family and performed hyperparameter tuning to further optimise their performance.

Types of models tested

The following table summarises the types of models tested and why they were chosen.

Type of model	Justification
Multiple Linear Regression	Multiple Linear Regression allows us to understand the linear relationship between the number of delivery days and two or more predictors.
Generalised Additive Models (GAMs)	GAMs extend upon linear regression methods by allowing us to capture non linear relationships in the data.
Random Forest	Random Forest is a tree based method that allows us to understand feature importance, providing valuable business insights into what factors are the most important in affecting the number of delivery days.
Gradient Boosting	Gradient Boosting models, including XGBoost, CatBoost and LightGBM, brings advantages of minimal pre-processing of the data, enabling handling of missing values that were present in the dataset. They also outperform other models in capturing complex non linear relationships within structured data.
Deep Learning	Deep Learning can enhance the predictive accuracy by automatically learning features from the data and capturing complex patterns in the data.

Model evaluation and selection strategy

After fitting each model to the training set, their performance was evaluated based on predictive accuracy on the validation dataset, whereby we calculated the RMSE, MAE and R^2. By using the same metrics for evaluation, we were able to compare the generalisation ability of different models.

In addition to comparing the evaluation metrics, we also assessed and selected the best model by balancing the trade-offs between computational speed and accuracy. Furthermore, while interpretability can provide insights into what factors affect delivery time, the primary goal was to recommend a model with the highest accuracy.

Baseline model

The selected baseline model was a multiple linear regression with the following variables:

- Customer Seller Distance
- Customer Seller Distance Squared
- Average Seller Process Time
- Order Purchase Timestamp Day
- Order Purchase Timestamp Month
- Payment Type
- Seller State
- Customer State
- Product Category Name English

This was selected as the baseline model because it had the best RMSE and MAE values at 8.056 and 4.978 respectively, out of all the other MLR models after careful feature selection. Since interpretability is not a key requirement of our model, the number of predictors, which increases significantly when encoding the categorical variables, means other simple models were not favoured over our chosen baseline.

The initial performance metrics of the baseline model were:

	RMSE	MAE	R^2
Baseline Model - MLR (selected numerical and categorical predictors)	8.056	4.978	0.260

While the baseline model was a good starting point, it was unable to capture non-linear and more complex relationships between predictors and the Number of Delivery Days. This led to us creating more complex models including GAMs, Random Forests, Deep Learning and Model Stacking.

Model experimentation & refinement process

1. Baseline Model - Multiple Linear Regression

When creating Multiple Linear Regression Models, there were 3 iterations in total that led to the development of our final baseline model.

In the first MLR model, we decided to include all the numerical variables. While there was a risk of overfitting, we wanted to see what the prediction performance metrics would be as it would provide a good starting point to refine our model. This initial model included the following variables:

- Customer Seller Distance
- Customer Seller Distance Squared
- Product Size
- Average Seller Process Time
- Order Quantity
- Average Delivery Time by Product
- Price
- Freight Value
- Payment Value
- Product Weight
- Payment Installments
- Payment Sequential
- Product Name Length
- Product Photos Quantity
- Product Description Length
- Order Purchase Timestamp Hour
- Order Purchase Timestamp Day
- Order Purchase Timestamp Month
- Shipping Limit Date Hour
- Shipping Limit Date Day
- Shipping Limit Date Month

We then performed feature selection, by performing backward and forward selection and using trial and error to determine which subset of the numerical predictors resulted in the best performance. The second iteration of our MLR contained only a few selected numerical variables, which resulted in significantly better performance. This model included the following predictors:

- Customer Seller Distance
- Customer Seller Distance Squared
- Average Seller Process Time
- Order Purchase Timestamp Day
- Order Purchase Timestamp Month

To create our baseline model we encoded and included categorical variables. This further helped improve the performance metrics and revealed that while the additional numerical predictors had resulted in overfitting, the categorical predictors improved how well the model generalised. Our final baseline model included the following predictors:

- Customer Seller Distance
- Customer Seller Distance Squared
- Average Seller Process Time
- Order Purchase Timestamp Day
- Order Purchase Timestamp Month
- Payment Type
- Seller State
- Customer State
- Product Category Name English

The improvement in the performance metrics, RMSE, MAE and R^2 can be seen in the table below.

	RMSE	MAE	R^2
MLR (all numerical predictors)	8.733	5.381	0.137
MLR (selected numerical predictors)	8.288	5.166	0.217
Baseline Model - MLR (selected numerical and categorical predictors)	8.056	4.978	0.260

2. Generalized Additive Models (GAMs)

Given the non-linear relationship between several predictors and the target variable, we explored Generalized Additive Models (GAMs) as a suitable modeling approach.

We began with a simple spline-based GAM using a single predictor: the distance between customer and seller. Through manual tuning, we adjusted the degrees and number of knots, observing that increasing model complexity led to slight improvements on the training set but diminishing returns on the validation set. To avoid overfitting, we selected a degree of 3 with 5 knots, which resulted in RMSE = 8.5210, MAE = 5.4110, and R^2 = 0.1736.

In the second iteration, we replaced the regular spline with a natural spline using the same predictor and complexity settings. This adjustment provided slightly better performance (RMSE = 8.5012, MAE = 5.3944, R^2 = 0.1776). This is due to the fact that natural splines handle outliers and boundary behavior more effectively.

Building on this, we introduced additional predictors. These were chosen based on our earlier EDA and matched those used in the linear regression model, even the encoded categorical variables. This multivariate GAM showed further improvement, with RMSE = 8.32, MAE = 5.08, and R^2 = 0.214.

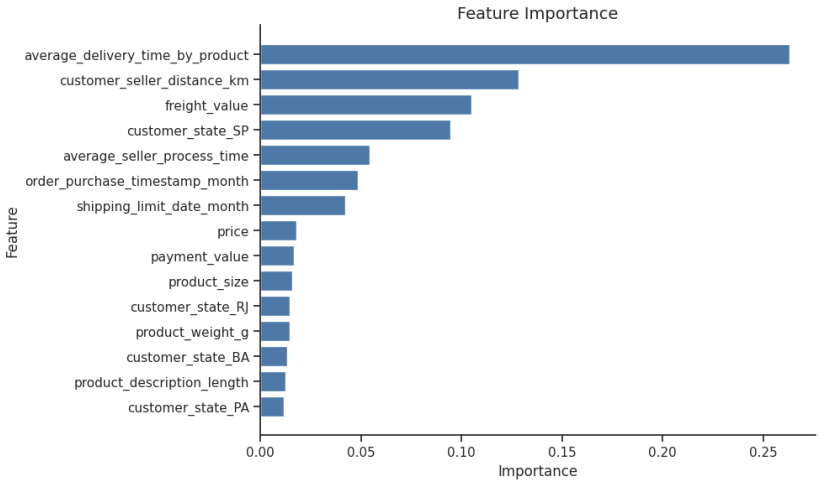
In the third iteration, to optimize performance and reduce overfitting, we used Optuna to tune key hyperparameters, including smoothing penalties (lambdas) for both spline and linear terms. The tuned model performed better across all metrics, with RMSE = 8.2, MAE = 4.92, and R^2 = 0.23. This final model delivered the best results so far, while still ensuring interpretability and avoiding overfitting. Moreover, through each iteration, we observed steady performance gains and developed a model that captured complex patterns while remaining robust and explainable.

3. Random Forest

The numerical predictors selected in random forest include: customer_seller_distance_km, product_size, average_seller_process_time, order_quantity, average_delivery_time_by_product, price, freight_value, payment_value, product_weight_g, payment_installments, payment_sequential, product_name_length, product_photos_qty, product_description_length, order_purchase_timestamp_month, shipping_limit_date_month. The one-hot encoded categorical variables included are: customer_state, seller_state, product_category_name_english, payment_type.

In its initial configuration, the model was set with a maximum of 5 features considered at each split and a minimum of 5 samples required to be at a leaf node. This yielded an MAE of 4.700, RMSE of 7.755, and R^2 of 0.317. The most influential feature demonstrated in Feature Importance is average_delivery_time_by_product (0.26), followed by customer_seller_distance (0.13) and freight_value (0.10).

Figure 10. Feature Importance



To enhance model performance, hyperparameter tuning was conducted using Optuna. The tuning process identified maximum depth of 15, minimum sample per leaf of 5, and maximum features of 15 to be the best combination. The optimised random forest demonstrated marginal performance improvements, with an MAE of 4.556, RMSE of 7.689, and R^2 of 0.327. Notably, the top three features remained consistent, reinforcing their importance across random forests.

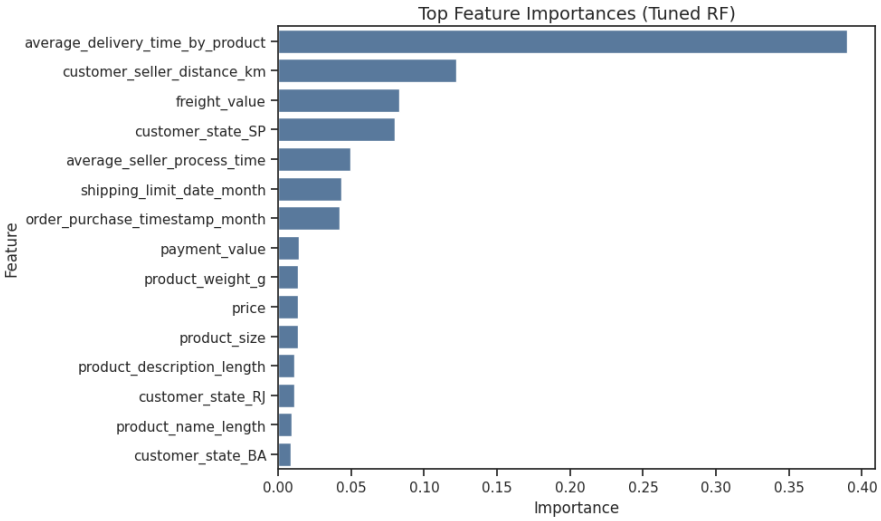


Figure 11. Feature Importance by Tuned RF

These findings underscore the effectiveness of Random Forest in handling structured, tabular data with mixed feature types. Additionally, hyperparameter tuning achieved incremental gains, which can translate to better forecasting accuracy in real-world applications.

4. Gradient Boosting

During experimentation of gradient boosting models, it was discovered that including variables about Order Approved At and Order Delivered Carrier Date significantly improved model performance. Therefore, for each gradient boosting model tested, these variables were included.

Three types of gradient boosting models were tested:

1. **XGBoost** - we first considered the numerical variables that were assessed as the most important from the Random Forest.
2. **CatBoost** - in addition to numerical variables, we wanted to include categorical variables to further model performance. CatBoost provides the advantage of automatically encoding listed categorical variables, therefore, minimising the need for preprocessing.
3. **LightGBM** - we created a LightGBM model based on default settings and using the same variables from the CatBoost model.

The following table summarises the evaluation metrics for each gradient boosting model:

	RMSE	MAE	R^2
XGBoost (default)	7.762	4.705	0.314
CatBoost (default)	7.274	3.916	0.398
LightGBM (default)	7.030	4.087	0.438

The table shows that LightGBM provided the best prediction performance, whereby the predictors included are:

- Freight Value
- Payment Type
- Customer Seller Distance
- Customer City

- Customer State
- Seller City
- Seller State
- Payment Type
- Order Quantity
- Product Size
- Order Purchase Timestamp Month
- Shipping Limit Date Month
- Average Seller Process Time
- Average Delivery Time by Product

Subsequently, hyperparameter tuning through cross validation was implemented to optimise the LightGBM model's parameters. The tuning result showed that the optimal number of trees is 5000, suggesting that more trees provided better accuracy.

A LightGBM model was fitted using the optimised hyperparameters, which provided better performance metrics compared to the default LightGBM, as shown in the table below:

	RMSE	MAE	R^2
LightGBM (default)	7.030	4.087	0.438
LightGBM (optimised)	6.758	3.814	0.480

What can be learnt through gradient boosting model experimentation is that not only does LightGBM provide the fastest processing time, it also yields the highest accuracy performance out of all the models tested thus far.

5. Deep Learning Model - Simple Multilayer Perceptron (MLP)

The initial MLP included numerical predictors (price, freight_value, payment_value, product_weight_g, product_size, average_seller_process_time, cusomter_seller_distance_km) and categorical variables that have been one-hot encoded (customer_state, seller_state, product_category_name_english, payment_type). The first 8 epochs are shown as follow:

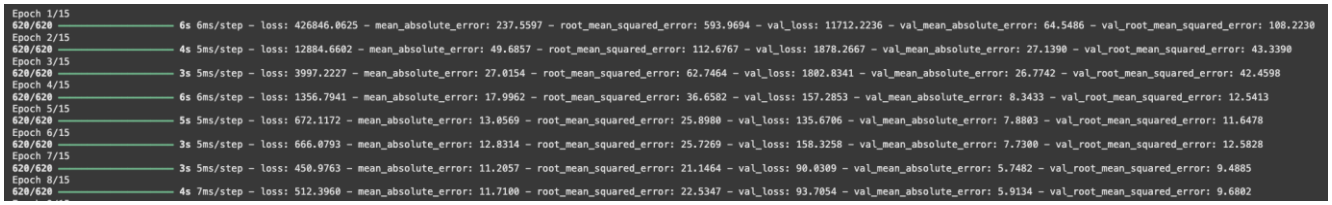


Figure 12. Deep Learning Epochs

In the first iteration, the validation MAE was 64.549, and the RMSE was 108.223. As training progressed through the first four iterations, performance significantly improved, with iteration 4 reaching a validation MAE of 8.343 and RMSE of 12.540. The model achieved its best performance in iteration 7, where the MAE dropped to 5.748 and RMSE to 9.489.

This report then applied standardisation scaling to the numerical predictors to improve learning efficiency and prediction accuracy. The first five iterations are demonstrated as follow:

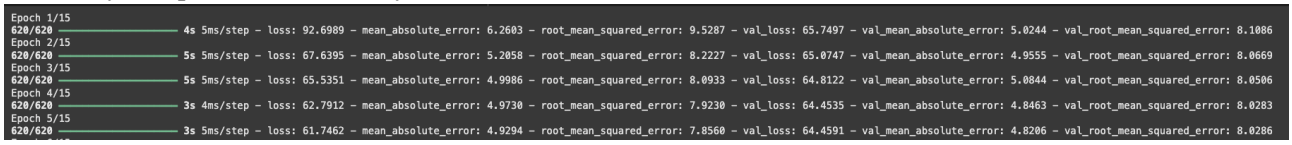


Figure 13. Deep Learning Iterations

The effect of standardisation was immediately observable. The standardised numerical predictors significantly decreased the time to achieve a reasonable outcome. In the very first epoch post-scaling, the validation MAE fell to 5.024 and RMSE to 8.109, already surpassing the best performance achieved by the non-scaled model. Continued training further refined the results, with the fifth iteration yielding the lowest MAE of 4.821 and RMSE of 8.029.

This enhancement demonstrates the importance of appropriate feature scaling in deep learning models. It not only accelerates convergence but also improves generalisation performance.

6. Model Stacking - Blending

For model stacking, this report used the MLR and LightGBM as the selected models, since this combination captures both linear and non-linear effects. The method used to stack is “blending”, which is to aggregate the validation results of these two models linearly. The result for model stacking yielded an MAE of 3.805, RMSE of 6.761, and R^2 of 0.480. Interestingly, the model performance did not improve that much from the LightGBM alone.

Feature Engineering & Selection

Feature selection significantly improved the performance for our initial MLR models. In our first model we included all the numerical predictors. However, since we jumped from simple linear regression to including many predictors, we wanted to ensure there wasn’t a risk of overfitting. Therefore, we decided to perform backward and forward selection to determine which predictors contributed most of the model’s performance. The backward selection determined that the Product Size, Product Photos Quantity and Shipping Limit Date Day, reduced the model fit and we therefore excluded these variables.

```
adj_r2 if all variables included: 0.511893
deleting product_size increases adj_r2 from 0.511893 to 0.511897
deleting product_photos_qty increases adj_r2 from 0.511897 to 0.511901
deleting shipping_limit_date_day increases adj_r2 from 0.511901 to 0.511904
final model is number_of_delivery_days ~ product_weight_g + average_seller_process_time_capped +
```

Figure 14. Backward Selection

Based on the order in which the forward selection included the other numerical predictors into the model, we conducted some trial and error to determine the best subset of numerical features for our MLR model.

```
Adding average_delivery_time_by_product increases adj_r2 from 0.000000 to 0.453545
Adding customer_seller_distance_km_capped increases adj_r2 from 0.453545 to 0.505375
Adding customer_seller_distance_km_capped_squared increases adj_r2 from 0.505375 to 0.508856
Adding order_purchase_timestamp_month increases adj_r2 from 0.508856 to 0.509768
Adding average_seller_process_time_capped increases adj_r2 from 0.509768 to 0.510356
Adding order_purchase_timestamp_day increases adj_r2 from 0.510356 to 0.510720
Adding payment_value increases adj_r2 from 0.510720 to 0.511066
Adding shipping_limit_date_hour increases adj_r2 from 0.511066 to 0.511387
Adding freight_value increases adj_r2 from 0.511387 to 0.511577
Adding order_purchase_timestamp_hour increases adj_r2 from 0.511577 to 0.511688
Adding payment_installments increases adj_r2 from 0.511688 to 0.511743
Adding product_weight_g increases adj_r2 from 0.511743 to 0.511783
Adding product_description_length increases adj_r2 from 0.511783 to 0.511820
Adding order_quantity increases adj_r2 from 0.511820 to 0.511862
Adding price increases adj_r2 from 0.511862 to 0.511882
Adding shipping_limit_date_month increases adj_r2 from 0.511882 to 0.511899
Adding product_name_length increases adj_r2 from 0.511899 to 0.511902
Adding payment_sequential increases adj_r2 from 0.511902 to 0.511904
Final model is number_of_delivery_days ~ average_delivery_time_by_product + customer_seller_distance_km_capped +
```

Figure 15. Forward Selection

This feature selection process resulted in a significant increase in r^2 value, indicating better model fit. The decrease in rMSE and MAE values, suggests there was an issue of overfitting in our previous model and by decreasing the number of predictors, we improved its generalisation ability.

	RMSE	MAE	R ^ 2
MLR (all numerical predictors)	8.733	5.381	0.137
MLR (selected numerical predictors)	8.288	5.166	0.217

Hyperparameter Tuning & Model Optimisation

To further improve the predictive performance of the Simple Multilayer Perceptron (MLP), this report applied standardisation scaling to the numerical predictors. Standardisation transforms features to have a mean of 0 and a standard deviation of 1, ensuring that each numerical variable contributes equally during gradient-based learning. This is especially important for neural networks, where differing magnitudes across features can slow convergence and lead to suboptimal weights.

	Before Hyperparameter Tuning		After Hyperparameter Tuning	
	MAE	RMSE	MAE	RMSE
1st Iteration	64.549	108.223	5.024	8.109
4th Iteration	8.343	12.540	4.846	8.028
Best Iteration	5.748	9.489	4.821	8.029

The impact of tuning was significant. Before applying standardization, the best validation results from the initial MLP model achieved a MAE of 5.748 and RMSE of 9.489. After standardization, the improved MLP achieved a lower MAE of 4.821 and RMSE of 8.029, representing a 16.5% reduction in MAE and a 15.4% reduction in RMSE. Notably, the optimised model reached this performance in fewer epochs, suggesting faster convergence and greater efficiency. These improvements reflect the importance of automated hyperparameter tuning for enhancing model accuracy, generalisability, and training stability.

Conclusion & Final Model Recommendations

The following table summarises the evaluation metrics from each model created:

	RMSE	MAE	R^2
MLR	8.056	4.978	0.260
GAMs (optimised)	8.203	4.893	0.239
Random Forest	7.689	4.556	0.327
XGBoost (default)	7.762	4.705	0.314
CatBoost (default)	7.274	3.916	0.398
LightGBM (optimised)	6.758	3.814	0.480
MLP (optimised)	8.029	4.821	-
Model Stacking	6.761	3.805	0.480

Summary of Models Explored
Although each model offered unique strengths, the decision was based on a combination of predictive performance, interpretability, and alignment with business objectives:

- MLR: Only consider linear relationships, which might be simplifying complex real-world scenarios.
- GAMs: Valuable for interpretation and exploratory analysis, but insufficient in predictive accuracy for deployment.
- Random Forest: Useful for feature selection and quick prototyping, but less scalable and interpretable.
- MLP: High potential, but lacked interpretability and required longer development cycles.
- LightGBM: Excellent standalone performer with balanced accuracy and training time.
- Stacked Model: Demonstrated superior predictive ability and consistent generalisation, making it the most reliable and robust model for business use.

Justify the Final Model Recommendations
Based on comprehensive evaluation, this report recommends LightGBM as the most suitable single model for deployment. It offers a strong balance between accuracy, efficiency, and ease of integration. Moreover, LightGBM has relatively low computational cost compared to model stacking, and can handle large datasets without requiring a complex deployment pipeline.
Model stacking did not perform better than LightGBM, and also introduced additional system complexity and maintenance challenges. Therefore, we recommend LightGBM as the preferred model for initial deployment, especially when considering resource constraints and the need for stable, scalable performance.