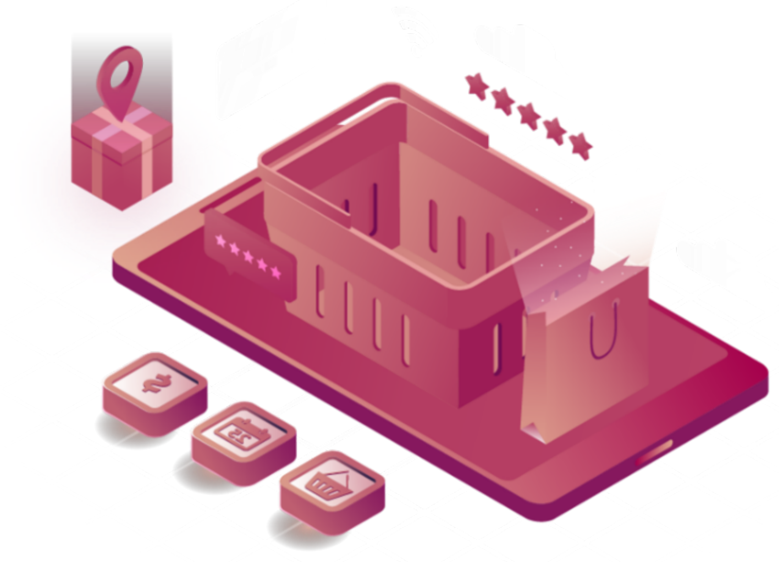


COM6003 - Final Project Report

Brazilian E-Commerce Public Dataset by Olist

Module code: COM6003

Module name: Artificial Intelligence and Machine Learning

Group[2] member:

P233351 Qiu Yi

Project Introduction

1. Background

We used the Brazilian E-Commerce Public Dataset by Olist data source, found at: https://www.kaggle.com/datasets/olistbr/brazilian-ecommerce/data.

This data belongs to Olist and was released by them.

Olist is a Brazilian e-commerce platform that connects SMEs to end customers, similar to Taobao or Amazon. It was founded in 2015 and became a unicorn in 2021. It also wishes to expand out of Brazil into all of Latin America (Crunchbase, n.d.; McCarthy, 2021). The data spans from 15 September 2016 to 29 August 2018.

2. Objectives

I. For this project we assumed the role of consultants to the retailer. Our goal is to help Olist discover what their customers care about and what impacts customer satisfaction so that it can be improved.

II. First, we used Natural Language Processing (NLP) to get insights from customers reviews. NLP was chosen as it can help Olist better understand consumer feedback and evaluation. Allowing Olist to improve products and services, enhance user experience, and increase user satisfaction, which will increase sales and customer loyalty.

III. Next, we applied the Natural Language Toolkit (NLTK) method and the TF-IDF method for comparison. We also tried with the Recurrent Neural Network (RNN) method.

IV. Later we used time series and geographical distribution and keyword analysis to dive deep analyze the data.

V. As we will see, NLP identified delays in delivery as related to dissatisfaction, so our other goal was to learn what affects a review's score.

VI. Finally, we generate a predictive model with XGboost.

3. Dataset

The entire dataset consists of 8 CSV files with 50 features and 1,551,016 records in total. There were no duplicates, but some values were missing. If more than half of the data in a column was missing, the column was dropped.

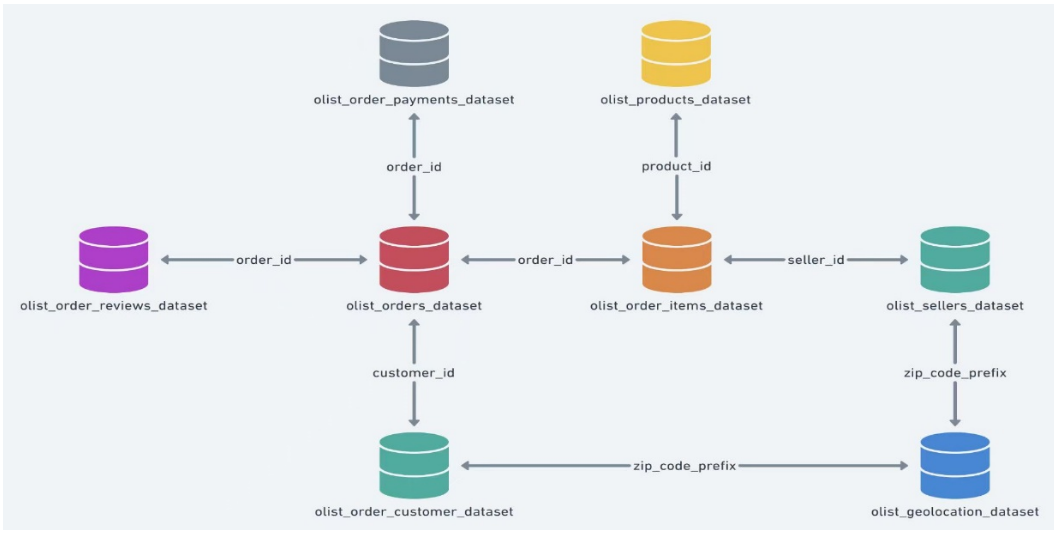
The data includes information about:

- Orders, payment, and delivery related dates.
- Customer and seller related information, including their location.
- Products related information such as category, size, and weight.
- Customers' reviews in text for and as a 1-5 score.
- And more.

Basic information of original files

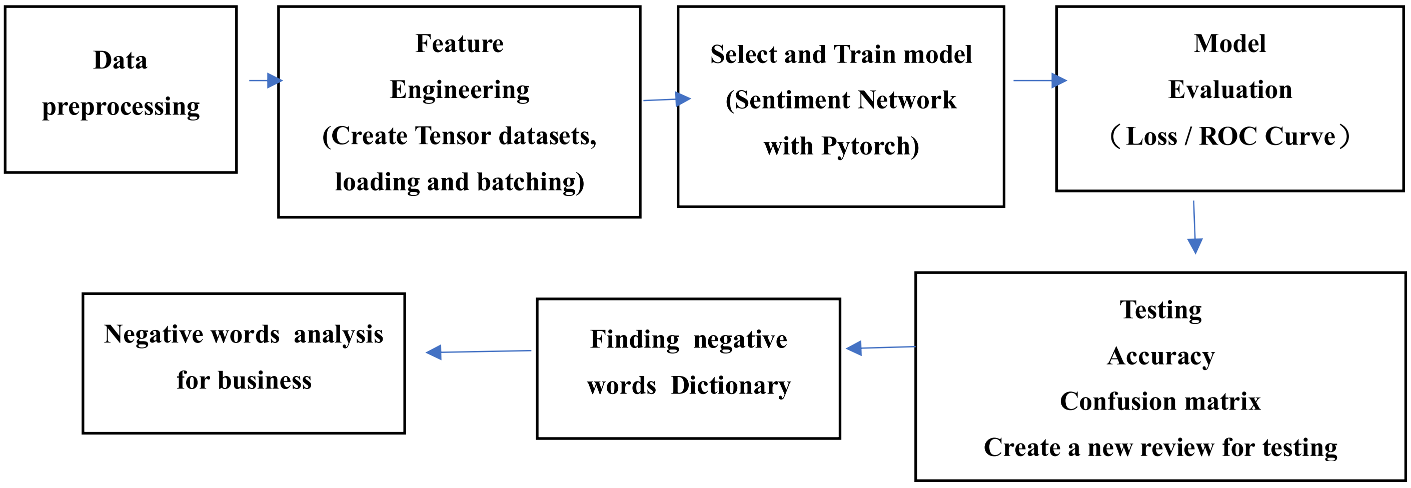
	dataset	nrows	ncols	names_of_null_cols	num_null_cols
0	products_df	32340	9		0
1	order_items_df	112650	7		0
2	olist_orders_df	99441	8	order_approved_at, order_delivered_carrier_dat...	3
3	order_reviews_df	100000	7	review_comment_title, review_comment_message	2
4	olist_customers_df	99441	5		0
5	order_payments_df	103886	5		0
6	sellers_df	3095	4		0
7	geolocation_df	1000163	5		0

Logic and connections for all files



4. Work Flow

Following Amazon's Machine Learning Pipeline, we will next describe data augmentation which includes evaluating the dataset, preprocessing, feature engineering, selecting and training the chosen model, evaluating the model and, if necessary, finetuning it, and evaluating if it meets our business goals.


Data preprocessing
Feature Engineering (Create Tensor datasets, loading and batching)
Select and Train model (Sentiment Network with Pytorch)
Model Evaluation（Loss / ROC Curve）
Testing Accuracy Confusion matrix Create a new review for testing
Finding negative words Dictionary
Negative words analysis for business

For this part of the analysis, we only kept the records that had text reviews - meaning records where "review_comment_message" included a text description.
There were 41,753 review records which we converted to lowercase and removed punctuation. 29 records had only empty spaces and they were removed.

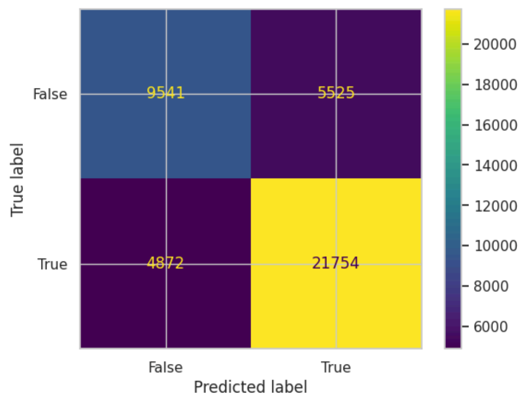
5. NLP

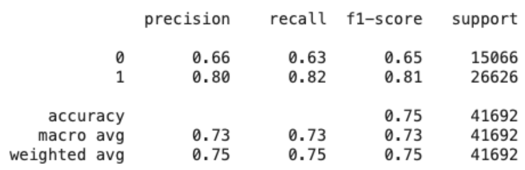
We tokenized the text, removed stop words, and lemmatized it, before rejoining the tokens. This means we separated each review into individual words, removed common and irrelevant words so that they don't interfere with our analysis, brought words to their base form based on the word's meaning, and then rejoined the review (Data Camp, 2023). We discovered that keeping the stop words led to better results, so we kept them.

5.1 Natural Language Toolkit (NLTK)

NLTK is a Python library for NLP, that simplifies tasks such as tokenization, stemming, lemmatization, parsing, and sentiment analysis (NLTK Project, 2023).

We performed sentiment analysis on the tokenized reviews. We created a new feature called "sentiment" which included the results of the sentiment analysis - marking 1 for positive or 0 for negative. We compared the result to the “review_score” and created the confusion matrix below. For “review_score”, we classified scores of 4-5 as positive - marked as 1, and scores of 1-3 as negative - marked as 0. As can be seen, overall model sentiment analysis accuracy was 75% (Data Camp, 2023).

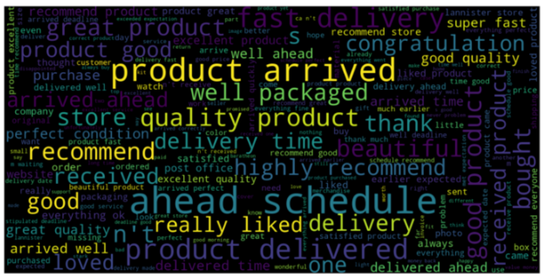




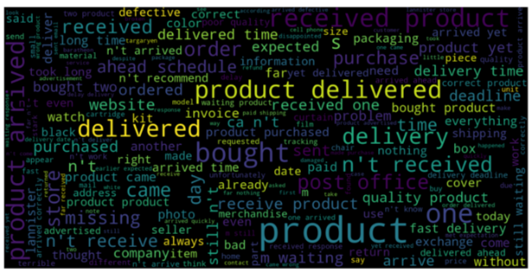
```
              precision    recall  f1-score   support

           0       0.66      0.63      0.65     15066
           1       0.80      0.82      0.81     26626

    accuracy                           0.75     41692
   macro avg       0.73      0.73      0.73     41692
weighted avg       0.75      0.75      0.75     41692
```

We also created a word map to visualize prominent words based on the sentiment analysis score (Geeks for Geeks, 2022).

Positive sentiment analysis score:



Negative sentiment analysis score:

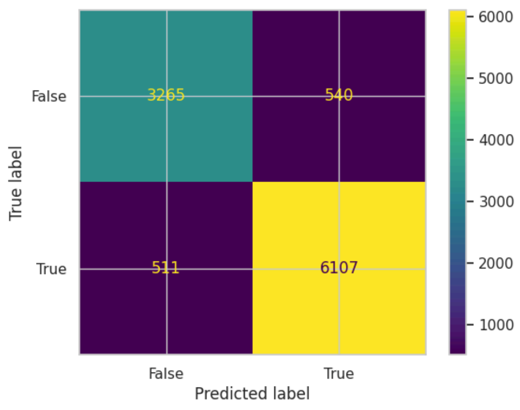


Results

From these pictures, it looks like product and delivery related words are more frequent in reviews, but further analysis is required.

5.2 TF-IDF

We then performed TF-IDF and Logistic Regression which is suitable for binary classification. The reviews were vectorized, shuffled (with controlled randomness), and then split into 75% training data and 25% test data (Geeks for Geeks, 2022; Saturn Cloud, 2023).



```
              precision    recall  f1-score   support

           0       0.86      0.86      0.86      3805
           1       0.92      0.92      0.92      6618

    accuracy                           0.90     10423
   macro avg       0.89      0.89      0.89     10423
weighted avg       0.90      0.90      0.90     10423
```

Results

Overall model sentiment analysis accuracy was 90%, an improvement from the 75% we previously got.

5.3 Recurrent Neural Network (RNN)

RNN is a family of neural networks that is ideal for text classification tasks due to its ability to capture sequential dependencies in data (Geeks for Geeks, 2024; Thomas, 2019).

5.3.1 Data Preprocessing

- Tokenization - removed all punctuations and separated each review into individual words, (0-41,752 items, 548,213 words).
- Encoded them, which resulted in 9,665 unique words.
- Padding -we added padding to ensure the encoding was the same size - Length= 50.
- Split the data to 80% - training, 10% - test, and 10% - validation.

  {Feature Shapes: Train set: (33295, 50); Validation set: (4162, 50); Test set: (4162, 50)}

5.3.2 Select and Train model (Sentiment Network with PyTorch)

Defined network with the following layers:

1. An embedding layer that converts our word tokens (integers) into embeddings of a specific size.
2. An LSTM layer that is defined by a hidden state size and number of layers.
3. A fully connected output layer that maps the LSTM layer outputs to a desired output size.

4. A sigmoid activation layer that turns all outputs into a value 0-1; returns only the last sigmoid output as the output of this network.

The Embedding Layer

We need to add an embedding layer because there are over 9,000 words in our vocabulary. It is inefficient to one-hot encode that many classes therefore, we can have an embedding layer and use it as a lookup table. Just make a new layer, use it for dimensionality reduction only, and let the network learn the weights (PyTorch, n.d.a).

The LSTM Layer(s)

We've created an LSTM layer(s) to use in our recurrent network, which takes in an input size, a hidden dim, several layers, a dropout probability (for dropout between multiple layers), and a batch first parameter (PyTorch, n.d.b).
Usually, a network will have better performance with more layers which enables it to learn complex relationships. We used 2.

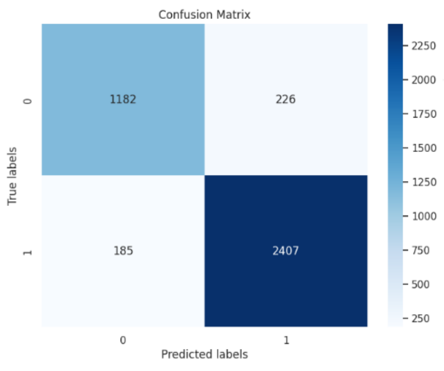
Set hyperparameters and define lr scheduler

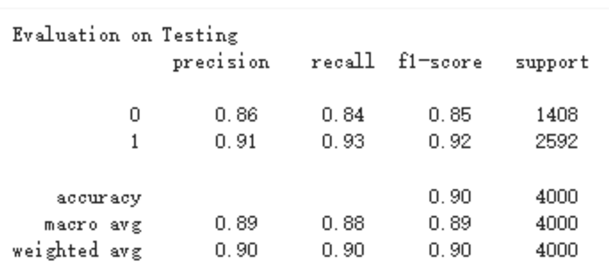
BCELoss, or Binary Cross Entropy Loss, is designed to work with a single Sigmoid output. It applies cross-entropy loss to a single value between 0 and 1.

- lr: learning rate for our optimizer.
- epochs: number of times to iterate through the training dataset.
- clip: the maximum gradient value to clip at (to prevent exploding gradients).

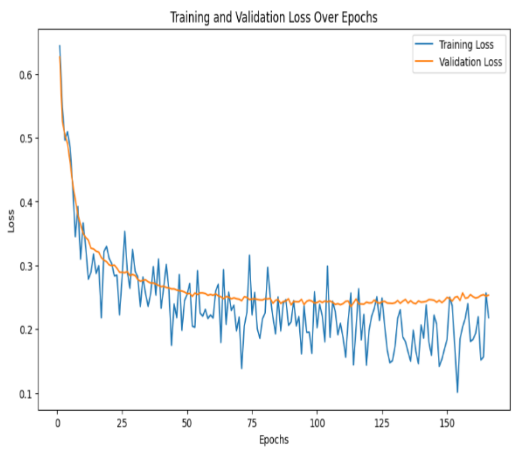
5.3.3 Performance

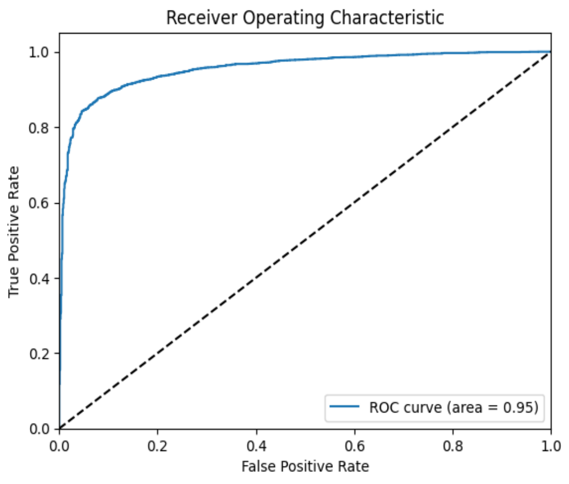
As can be seen below, the model is well-trained.





```
Evaluation on Testing
              precision    recall  f1-score   support

           0       0.86      0.84      0.85      1408
           1       0.91      0.93      0.92      2592

    accuracy                           0.90      4000
   macro avg       0.89      0.88      0.89      4000
weighted avg       0.90      0.90      0.90      4000
```





5.3.4 Testing (inference on user-generated data):

We also tested the model by entering two reviews that we created and reviewed the qualitatively accurate results. Below are the detailed results.

```
test_review_neg = 'So bad, left me with only bad effects, will not come again.'  →  (0.011665682308375835, 0.0)
test_review_pos = 'Very good personal service, I was very impressed. It will come again next time.'  →  (0.5842257738113403, 1.0)
```

5.3.5 Results

The model also revealed that some words were related to a low satisfaction score as below.

	words	prediction	sentiment	Frequency
86	not	0.185101	0.0	5873
63	terrible	0.219538	0.0	271
509	poor	0.255759	0.0	264
455	bad	0.268946	0.0	306
483	defective	0.366252	0.0	284
106	delay	0.369741	0.0	541
272	break	0.387206	0.0	339
354	wait	0.399479	0.0	1291
372	didnt	0.403751	0.0	2284
1349	late	0.445502	0.0	205
589	low	0.451298	0.0	139
1875	offices	0.454110	0.0	28
61	disappoint	0.462625	0.0	328
1120	lack	0.463513	0.0	236
1511	turn	0.474624	0.0	83

This helped us decide that we should further explore deliveries, as delay was an important word for dissatisfaction.

6 XGBoost

For Timeseries

The data had some issues that needed to be addressed.

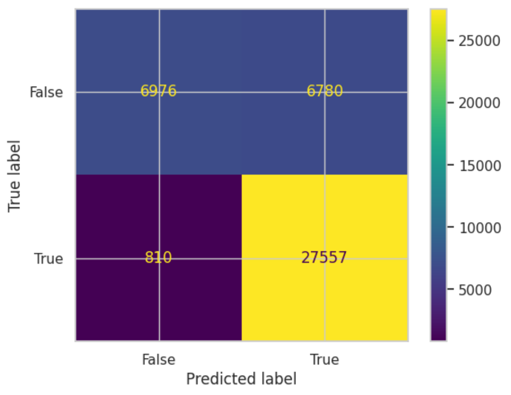
Remove Outliers

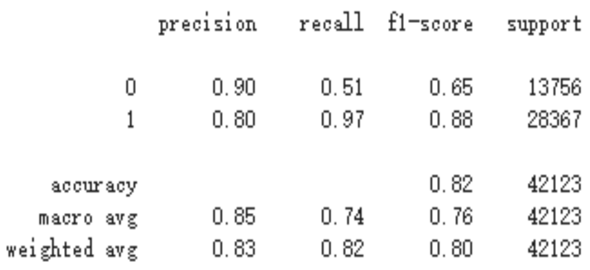
There were several mistakes with various timestamps. For example, for one record estimated delivery date was 3 years after the order date. For such a problem, we decided to delete the data received before delivery. The delay rate (transit time/estimated time) has been screened by the IQR to ensure the transit time is not too long or too short.

XGBoost is a gradient lift tree algorithm that can be applied to text data to learn patterns and correlations in the text and map them to predicted results. We created an XGBoost model for training and used the trained model to make predictions on the test set. Depending on the size of the predicted value, we marked the predicted value as 0 or 1. Then the confusion matrix was used to evaluate the classification performance of the model.

We choose to use XGBoost as it has benefits such as:

- Regularization - helps in reducing overfitting.
- Parallel Processing - XGBoost implements parallel processing and is much faster compared to GBM.
- Handling missing values - it has an in-built routine to handle missing values.
- Built-in cross-validation - allows the user to run cross-validation at each iteration of the boosting process.





```
              precision    recall  f1-score   support

           0       0.90      0.51      0.65     13756
           1       0.80      0.97      0.88     28367

    accuracy                           0.82     42123
   macro avg       0.85      0.74      0.76     42123
weighted avg       0.83      0.82      0.80     42123
```

XGBoost had lower accuracy than previous methods.

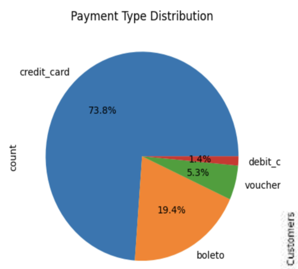
7 Business Analysis

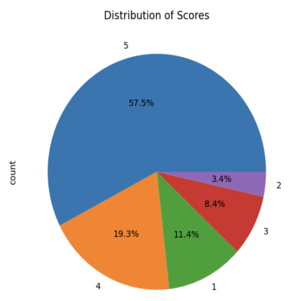
For this part of the analysis, date columns were updated to data type “datetime” and new features were created. In total, we used 113,782 records and 49 features for analyzing customers, sellers, orders, and product information. On top of that, geographical data was also used.

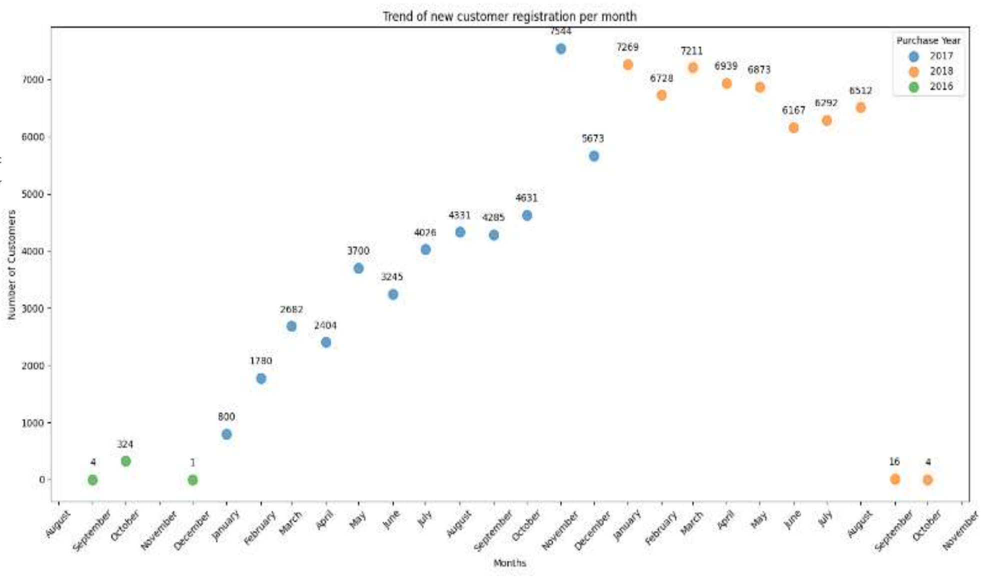
Kmeans was used as it is more suitable for unbalanced data than KNN.

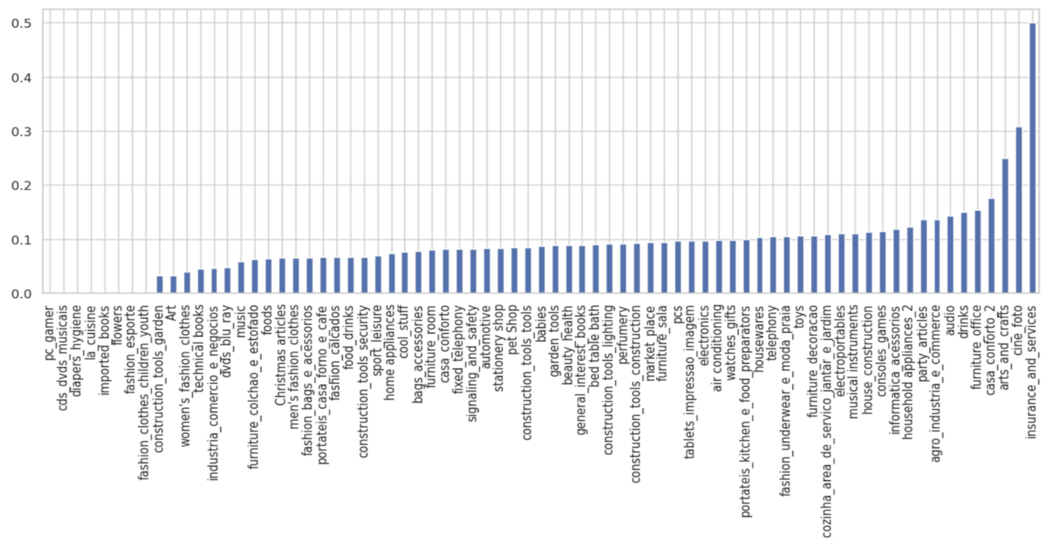
1. More than 70% of users pay with credit cards, and nearly 20% use boleto.
2. 57.5%, more than half, scored 5 points, and nearly 20% scored 4 points.
3. E-commerce in Brazil is growing. We can see some seasonality with peaks at specific months, February, March and November, December. But in general, customers are more prone to buy things online than before.
4. The following categories had the highest ranking: life items, health care, sports and leisure, and home products.
5. Products in the following categories: bed, tables, and bath, had the most negative reviews, but it may be due to the large quantity purchased.
6. Most of the high delay regions are inland or at the northeast edge of the country.

Negative word analysis for product *['delay']>0) | ['slow']>0)) | ['late']>0)]*

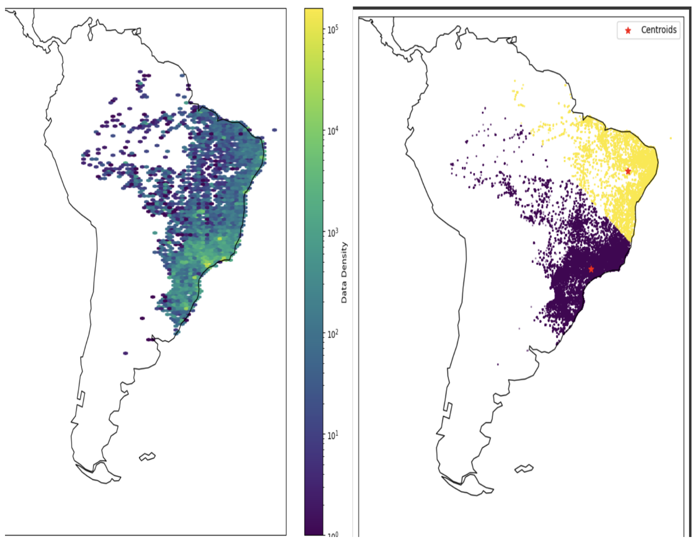




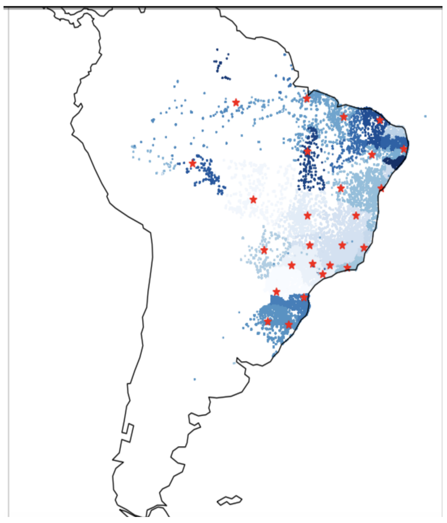




Normal orders location with heatmap



Map of delay rates region



8 Improvement Suggestions

We analyzed the data to learn what our customers care about, and we focused on the dissatisfaction that is associated with delays in shipment. From the information we learned, we have the following suggestion for Olist.

As we discovered, **certain regions in the country suffer more from delays,** we offer the below three improvement suggestions:

1. Short-term - increase delivery time on the website
2. Mid-term - find an additional courier company
3. Long-term - build a logistical center closer to that area

During high-selling **months there are more delays**:

1. Short-term - increase delivery time on the website
2. Mid-term - find an additional courier company and offer pre-sale, which could be a premium service

Oversized or heavy products:

1. Short-term - increase delivery time on the website
2. Mid-term - find a specialized courier company or build with courier partners to build this capacity

On top of that, Olist needs to work with sellers who are consistently delayed in approving orders and shipping them.

Finally, as Olist wishes to expand out of Brazil, Amazon services such as Translate, Lex, and Lambda, can help the company localize its services and offer better customer support.

9AWS Applied

An S3 bucket was used to store the data. AWS Simple Cloud Storage or S3 will enable various parties to easily access and work with the data while maintaining security.

10 Summary for ML Model

- It is important to choose a suitable direction and target at the beginning.
- Hyperparameter settings should be constantly tried, compared, and improved for tuning.
- Trying newer models to improve the accuracy of review predictions is still needed.

References

Crunchbase (n.d.). *Olist*. Crunchbase. https://www.crunchbase.com/organization/olist

Data Camp (2023). *NLTK Sentiment Analysis Tutorial for Beginners*. Data Camp. https://www.datacamp.com/tutorial/text-analytics-beginners-nltk

Geeks for Geeks (2022, November 21). *Amazon Product Reviews Sentiment Analysis in Python*. Geeks for geeks. https://www.geeksforgeeks.org/amazon-product-reviews-sentiment-analysis-in-python/

Geeks for Geeks (2024, January 2). *RNN for Text Classifications in NLP*. Geeks for Geeks. https://www.geeksforgeeks.org/rnn-for-text-classifications-in-nlp/

McCarthy, M. (2021, December 15). Olist Becomes Brazil's Newest Unicorn, Raises $186M: The Brazil-based startup announced Wednesday the close of a Series E, just a few months after its Series D round. Bloomberg Linea. https://www.bloomberglinea.com/english/olist-becomes-brazils-newest-unicorn-raises-186m/

NLTK Project (2023, January 2). *Natural Language Toolkit*. NLTK Project. https://www.nltk.org

Olist (n.d.). *Brazilian E-Commerce Public Dataset by Olist: 100,000 Orders with product, customer and reviews info*. Kaggle. https://www.kaggle.com/datasets/olistbr/brazilian-ecommerce/data?select=olist_order_reviews_dataset.csv

PyTorch (n.d. a). *TORCH.NN*. PyTorch. https://pytorch.org/docs/stable/nn.html#embedding

PyTorch (n.d. b). *TORCH.NN*. PyTorch. https://pytorch.org/docs/stable/nn.html#lstm

Saturn Cloud (2023, July 6). *What Is 'random_state' in sklearn.model_selection.train_test_split Example?*. Saturn Cloud. https://saturncloud.io/blog/what-is-randomstate-in-sklearnmodelselectiontraintestsplit-example/#:~:text=random_state%20is%20a%20parameter%20in,same%20splits%20of%20the%20data.

Thomas, C. (2019, June 9). *Recurrent Neural Networks and Natural Language Processing*. Towards Data Science. https://towardsdatascience.com/recurrent-neural-networks-and-natural-language-processing-73af640c2aa1