Olist Ecommerce Analysis

Brendan Ang Wei Jie (U2021332F)
Justyn Phoa Zairen (U2022593D)
Xia Tianyi (U2020801C)

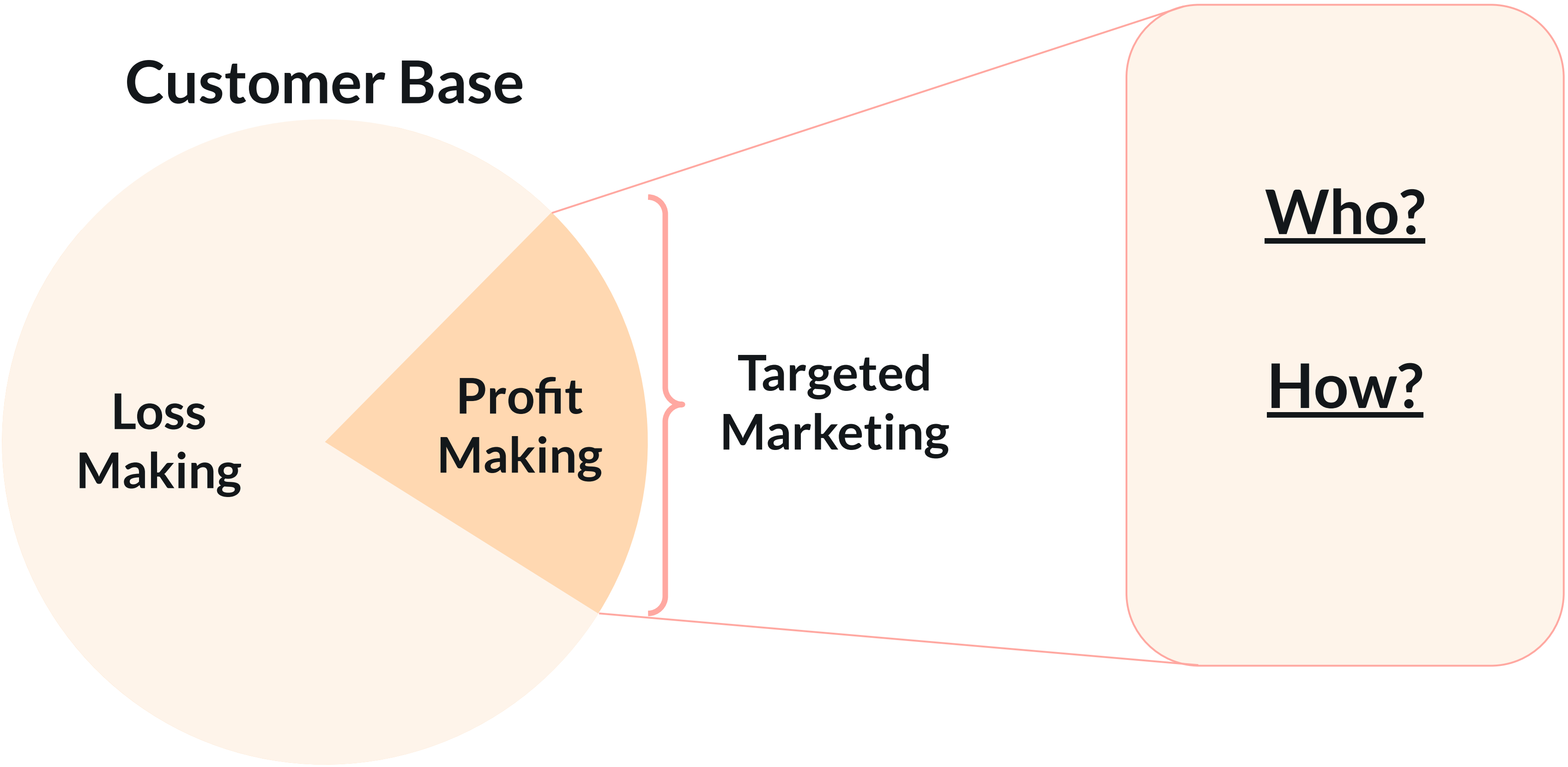
Brazilian E-commerce Industry

Highly competitive: there is no 1 major firm having the majority stake or market share

Prevalence of substitutes to Olist

Customer retention is very challenging and costly for E-commerce firms

Customer Retention



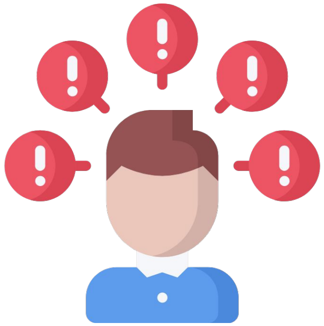
1. Who to retain?

Metric: Customer Lifetime Value (CLV)

CLV: Total amount of money a customer is expected to spend in your business during their lifetime

Estimated profit of a particular customer relationship

1. Who to retain?



Predicting CLV allows Olist to pinpoint groups and individuals who consistently generate revenue and are hence profitable to them.

2. How to retain?

Determine which variables within the vendor-customer experience affect customer retention

Data Preparation

Checking for Discrepancies

```
print(len(orderitems.order_id.unique()))
print(len(orders.order_id.unique()))
print(len(orderpayments.order_id.unique()))
print(len(orderreviews.order_id.unique()))
```

```
98666
99441
99440
99441
```

Missing data

```
#Finding list of missing order_id
missinglist = []
testset = set(orderitems.order_id)
for x in orders.order_id:
    if x in testset:
        continue
    else:
        missinglist.append(x)


#Extracting missing data from orders dataset
missingdf = pd.DataFrame
missingdf = orders[orders['order_id'].isin(missinglist)]
missingdf.info()
missingdf
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 775 entries, 266 to 99415
Data columns (total 8 columns):
 #   Column                         Non-Null Count  Dtype
---  ------                         --------------  -----
 0   order_id                       775 non-null    object
 1   customer_id                    775 non-null    object
 2   order_status                   775 non-null    object
 3   order_purchase_timestamp       775 non-null    object
 4   order_approved_at              629 non-null    object
 5   order_delivered_carrier_date   1 non-null      object
 6   order_delivered_customer_date  0 non-null      object
 7   order_estimated_delivery_date  775 non-null    object
dtypes: object(8)
memory usage: 54.5+ KB
```

Data Preparation

	customer_id	order_status	order_purchase_timestamp	order_approved_at	order_delivered_carrier_date
	64a254d30eed42cd0e6c36dddb88adf0	unavailable	2017-11-16 15:09:28	2017-11-16 15:26:57	NaN
	9582c5bbecc65eb568e2c1d839b5cba1	unavailable	2018-01-31 11:31:37	2018-01-31 14:23:50	NaN
	7607cd563696c27ede287e515812d528	unavailable	2017-08-14 17:38:02	2017-08-17 00:15:18	NaN
	470b93b3f1cde85550fc74cd3a476c78	unavailable	2018-01-08 19:39:03	2018-01-09 07:26:08	NaN
	3532ba38a3fd242259a514ac2b6ae6b6	canceled	2018-08-28 15:26:39	NaN	NaN
	...	...	...	...	...
	df20748206e4b865b2f14a5eabbfcf34	unavailable	2018-01-16 14:27:59	2018-01-17 03:37:34	NaN
	0b0d6095c5555fe083844281f6b093bb	canceled	2018-08-31 16:13:44	NaN	NaN
	2f0524a7b1b3845a1a57fcf3910c4333	canceled	2018-09-06 18:45:47	NaN	NaN
	726f0894b5becdf952ea537d5266e543	unavailable	2017-08-23 16:28:04	2017-08-28 15:44:47	NaN
	32c9df889d41b0ee8309a5efb6855dcb	unavailable	2017-10-10 10:50:03	2017-10-14 18:35:57	NaN

Cancelled / Unavailable / Unreceived Orders

Feature Engineering

Delivery time

```
finaldata[['order_purchase_timestamp','order_delivered_customer_date',
           'delivery_time']]
```

	order_purchase_timestamp	order_delivered_customer_date	delivery_time
0	2017-09-13 08:59:02	2017-09-20 23:43:48	7.0
1	2017-06-28 11:52:20	2017-07-13 20:39:29	15.0
2	2018-05-18 10:25:53	2018-06-04 18:34:26	17.0
3	2017-08-01 18:38:42	2017-08-09 21:26:33	8.0
4	2017-08-10 21:48:40	2017-08-24 20:04:21	14.0
...	...	...	...
144910	2017-03-15 17:16:36	2017-04-05 18:48:06	21.0
144911	2018-07-13 20:04:05	2018-07-23 19:44:45	10.0
144912	2018-08-18 10:00:59	2018-08-21 12:18:57	3.0
144913	2017-06-01 16:53:03	2017-06-08 13:04:40	7.0
144914	2017-12-18 16:33:07	2018-01-08 18:23:10	21.0

Review response time

```
finaldata[['review_creation_date','review_answer_timestamp',
           'review_response_days']]
```

	review_creation_date	review_answer_timestamp	review_response_days
0	2017-09-21 00:00:00	2017-09-22 10:57:03	1.0
1	2017-07-14 00:00:00	2017-07-17 12:50:07	3.0
2	2018-06-05 00:00:00	2018-06-06 21:41:12	1.0
3	2017-08-10 00:00:00	2017-08-13 03:35:17	3.0
4	2017-08-25 00:00:00	2017-08-28 00:51:18	3.0
...	...	...	...
144910	2017-04-06 00:00:00	2017-04-10 13:32:15	4.0
144911	2018-07-24 00:00:00	2018-07-25 00:25:51	1.0
144912	2018-08-22 00:00:00	2018-08-25 14:22:54	3.0
144913	2017-06-09 00:00:00	2017-06-12 11:05:17	3.0
144914	2018-01-09 00:00:00	2018-01-11 23:56:38	2.0

Feature Engineering

As you can see, those customers with retained = True have customer_unique_id mapped to at least 2 different customer_id

```
explore = finaldata[finaldata['retained'] == True]
explore1 = explore[['customer_id','customer_unique_id','retained','order_purchase_timestamp','price']]
explore1.sort_values(by=['customer_unique_id'])
```

	customer_id	customer_unique_id	retained	order_purchase_timestamp	price
42563	1b4a75b3478138e99902678254b260f4	004288347e5e88a27ded2bb23747066c	True	2017-07-27 14:13:03	229.99
61726	f6efe5d5c7b85e12355f9d5c3db46da2	004288347e5e88a27ded2bb23747066c	True	2018-01-14 07:36:54	87.90
52451	cbb68c721ba9ddb30d8a490cc1897fa1	00a39521eb40f7012db50455bf083460	True	2018-06-03 10:12:57	11.55
66659	876356df457f952458a764348e1858bc	00a39521eb40f7012db50455bf083460	True	2018-05-23 20:14:21	69.90
155333	102fc0966044243157bb81e4ee0a251e	00cc12a6d8b578b8ebd21ea4e2ae8b27	True	2017-03-21 19:25:23	69.90

Different

Same

CLV Modelling

CLV Modelling

Lifetimes Library

```
sdata = data[["customer_unique_id", "order_purchase_timestamp", "price"]]
sdata.sort_values(by =["customer_unique_id"], inplace = True, ignore_index = True)
```

```
summary = lifetimes.utils.summary_data_from_transaction_data(sdata,'customer_unique_id','order_purchase_timestamp','price')
summary = summary.reset_index()
summary
```

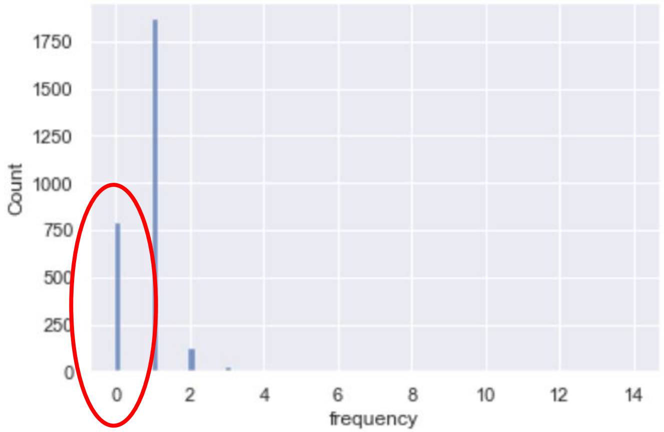
	customer_unique_id	frequency	recency	T	monetary_value
0	004288347e5e88a27ded2bb23747066c	1.0	171.0	397.0	87.900
1	00a39521eb40f7012db50455bf083460	1.0	11.0	97.0	11.550
2	00cc12a6d8b578b8ebd21ea4e2ae8b27	0.0	0.0	525.0	0.000
3	011575986092c30523ecb71ff10cb473	1.0	60.0	192.0	63.900
4	011b4adcd54683b480c4d841250a987f	1.0	177.0	371.0	227.880
...	...	...	...	...	...
2796	ff03923ad1eb9e32304deb7f9b2a45c9	1.0	33.0	127.0	220.640
2797	ff44401d0d8f5b9c54a47374eb48c1b8	0.0	0.0	466.0	0.000
2798	ff8892f7c26aa0446da53d01b18df463	1.0	186.0	461.0	99.900
2799	ff922bdd6bafcdf99cb90d7f39cea5b3	2.0	204.0	552.0	34.945
2800	ffe254cc039740e17dd15a5305035928	0.0	0.0	513.0	0.000

2801 rows × 5 columns

Anomalies

```
sb.histplot(data = summary, x="frequency")
```

<AxesSubplot:xlabel='frequency', ylabel='Count'>




```
sdata.head(10)
```

	customer_unique_id	order_purchase_timestamp	price
0	004288347e5e88a27ded2bb23747066c	2017-07-27 14:13:03	229.99
1	004288347e5e88a27ded2bb23747066c	2018-01-14 07:36:54	87.90
2	00a39521eb40f7012db50455bf083460	2018-06-03 10:12:57	11.55
3	00a39521eb40f7012db50455bf083460	2018-05-23 20:14:21	69.90
4	00cc12a6d8b578b8ebd21ea4e2ae8b27	2017-03-21 19:25:23	69.90
5	00cc12a6d8b578b8ebd21ea4e2ae8b27	2017-03-21 19:25:23	69.90
6	00cc12a6d8b578b8ebd21ea4e2ae8b27	2017-03-21 19:25:22	29.90
7	00cc12a6d8b578b8ebd21ea4e2ae8b27	2017-03-21 19:25:22	29.90
8	00cc12a6d8b578b8ebd21ea4e2ae8b27	2017-03-21 19:25:23	69.90
9	00cc12a6d8b578b8ebd21ea4e2ae8b27	2017-03-21 19:25:23	69.90

Beta-Geometric/Beta-Binomial Model

```python
# Fitting the BG/NBD model
bgf = lifetimes.BetaGeoFitter(penalizer_coef=0.0)
bgf.fit(summary['frequency'], summary['recency'], summary['T'])
```

```
<lifetimes.BetaGeoFitter: fitted with 2801 subjects, a: 2.13, alpha: 157.32, b: 0.29, r: 1.16>
```

```python
# Compute the customer alive probability
summary['probability_alive'] = bgf.conditional_probability_alive(summary['frequency'], summary['recency'], summary['T'])
summary.head(10)
```

	customer_unique_id	frequency	recency	T	monetary_value	probability_alive
0	004288347e5e88a27ded2bb23747066c	1.0	171.0	397.0	87.90	0.042640
1	00a39521eb40f7012db50455bf083460	1.0	11.0	97.0	11.55	0.053582
2	00cc12a6d8b578b8ebd21ea4e2ae8b27	0.0	0.0	525.0	0.00	1.000000
3	011575986092c30523ecb71ff10cb473	1.0	60.0	192.0	63.90	0.047191
4	011b4adcd54683b480c4d841250a987f	1.0	177.0	371.0	227.88	0.048872
5	012452d40dafae4df401bced74cdb490	1.0	330.0	436.0	1320.00	0.082800
6	012a218df8995d3ec3bb221828360c86	1.0	42.0	113.0	1369.90	0.066731
7	013ef03e0f3f408dd9bf555e4edcdc0a	1.0	29.0	68.0	59.90	0.083916
8	013f4353d26bb05dc6652f1269458d8d	1.0	4.0	277.0	256.00	0.015988
9	015557c9912277312b9073947804a7ba	1.0	39.0	523.0	59.90	0.009329

Predicted No. of Transactions

```python
#Predict future transaction for the next 30 days based on historical data
t = 30
summary['pred_num_txn'] = round(bgf.conditional_expected_number_of_purchases_up_to_time
                          (t, summary['frequency'], summary['recency'], summary['T']),2)
summary.sort_values(by='pred_num_txn', ascending=False).head(10).reset_index()
```

	index	customer_unique_id	frequency	recency	T	monetary_value	probability_alive	pred_num_txn
0	1551	8d50f5eadf50201ccdcedfb9e2ac8455	14.0	428.0	436.0	48.195	0.835836	0.61
1	622	394ac4de8f3acb14253c177f0e15bc58	4.0	236.0	249.0	176.900	0.567154	0.20
2	2426	dc813062e0fc23409cd255f7f53c7074	5.0	418.0	423.0	163.552	0.656989	0.20
3	1254	72e05cde2a8e5d8f08ca3ef3bcfadc63	0.0	0.0	11.0	0.000	1.000000	0.18
4	2047	b948343ff2e4e183e27e22ca63968d2b	0.0	0.0	12.0	0.000	1.000000	0.18
5	1623	931a4a1a3e2cf8b4b4d33922f1469dbe	0.0	0.0	11.0	0.000	1.000000	0.18
6	2357	d649357bd5b1b116bf9662f41259db37	0.0	0.0	6.0	0.000	1.000000	0.18
7	50	059e7585d8fcd2430a33375bdbcbb990	0.0	0.0	9.0	0.000	1.000000	0.18
8	1734	9b475216704f39cb18ce448648b995e1	0.0	0.0	18.0	0.000	1.000000	0.17
9	1284	7580e539c3d74ce5ff3946877db01dd2	0.0	0.0	14.0	0.000	1.000000	0.17

Gamma-Gamma Model

```
ggf = lifetimes.GammaGammaFitter(penalizer_coef=0.001)
ggf.fit(return_customers_summary['frequency'],
        return_customers_summary['monetary_value'])
```

```
<lifetimes.GammaGammaFitter: fitted with 2015 subjects, p: 8.00, q: 0.99, v: 7.56>
```

```
# Calculating the conditional expected average profit for each customer per transaction
summary = summary[summary['monetary_value'] >0]
summary['exp_avg_sales'] = ggf.conditional_expected_average_profit(summary['frequency'],
                                        summary['monetary_value'])
summary.head()
```

	customer_unique_id	frequency	recency	T	monetary_value	probability_alive	pred_num_txn	exp_avg_sales
0	004288347e5e88a27ded2bb23747066c	1.0	171.0	397.0	87.90	0.042640	0.00	95.618031
1	00a39521eb40f7012db50455bf083460	1.0	11.0	97.0	11.55	0.053582	0.01	19.144075
3	011575986092c30523ecb71ff10cb473	1.0	60.0	192.0	63.90	0.047191	0.01	71.579066
4	011b4adcd54683b480c4d841250a987f	1.0	177.0	371.0	227.88	0.048872	0.01	235.825290
5	012452d40dafae4df401bced74cdb490	1.0	330.0	436.0	1320.00	0.082800	0.01	1329.718362

```
# Checking the expected average value and the actual average value in the data to make sure the values are good
print(f"Expected Average Sales: {summary['exp_avg_sales'].mean()}")
print(f"Actual Average Sales: {summary['monetary_value'].mean()}")
```

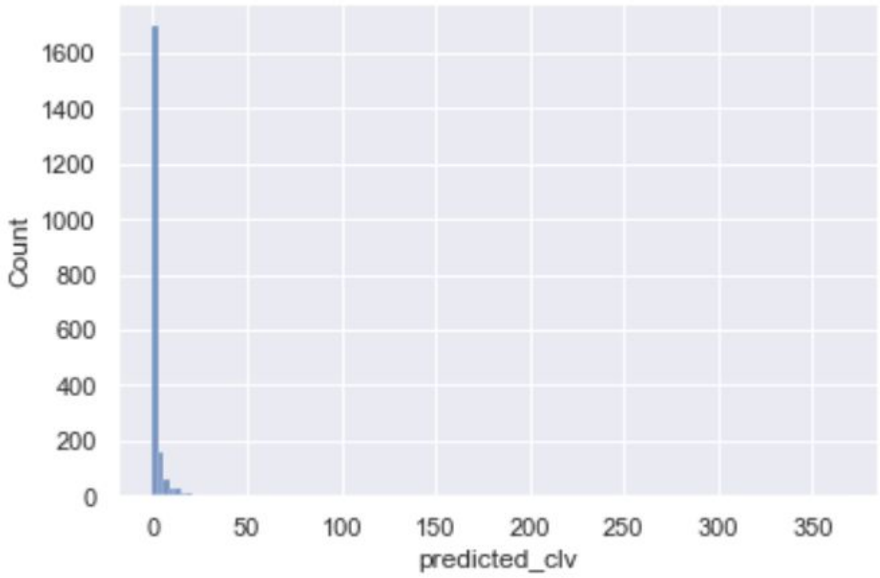
```
Expected Average Sales: 296.2688305063302
Actual Average Sales: 288.55097526881957
```

Predicted CLV

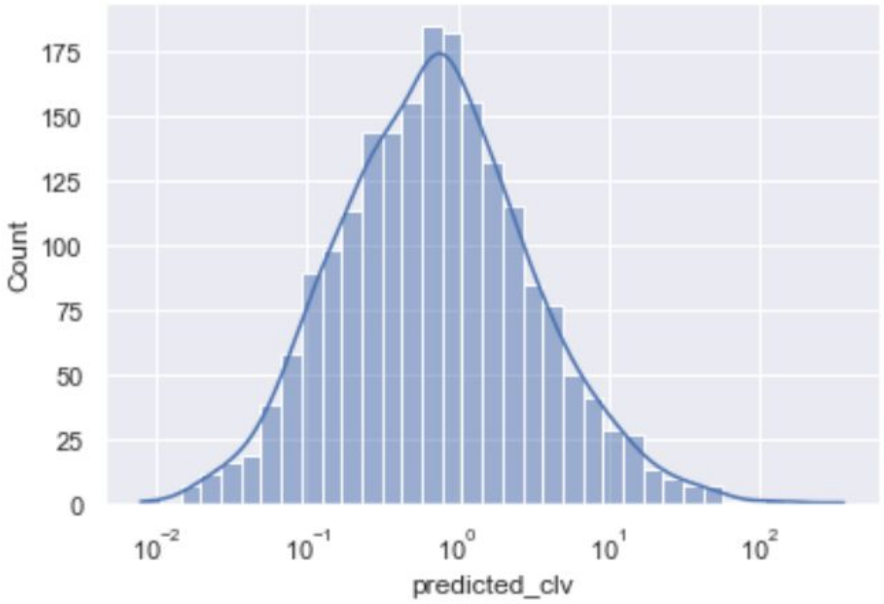
```
summary['predicted_clv'] =      ggf.customer_lifetime_value(bgf,
                                                           summary['frequency'],
                                                           summary['recency'],
                                                           summary['T'],
                                                           summary['monetary_value'],
                                                           time=1,
                                                           freq='D',
                                                           discount_rate=0.01)
summary.head()
```

	customer_unique_id	frequency	recency	T	monetary_value	probability_alive	pred_num_txn	exp_avg_sales	predicted_clv
0	004288347e5e88a27ded2bb23747066c	1.0	171.0	397.0	87.90	0.042640	0.00	95.618031	0.448267
1	00a39521eb40f7012db50455bf083460	1.0	11.0	97.0	11.55	0.053582	0.01	19.144075	0.231999
3	011575986092c30523ecb71ff10cb473	1.0	60.0	192.0	63.90	0.047191	0.01	71.579066	0.572376
4	011b4adcd54683b480c4d841250a987f	1.0	177.0	371.0	227.88	0.048872	0.01	235.825290	1.326227
5	012452d40dafae4df401bced74cdb490	1.0	330.0	436.0	1320.00	0.082800	0.01	1329.718362	11.347272

Predicted CLV



Right skewed

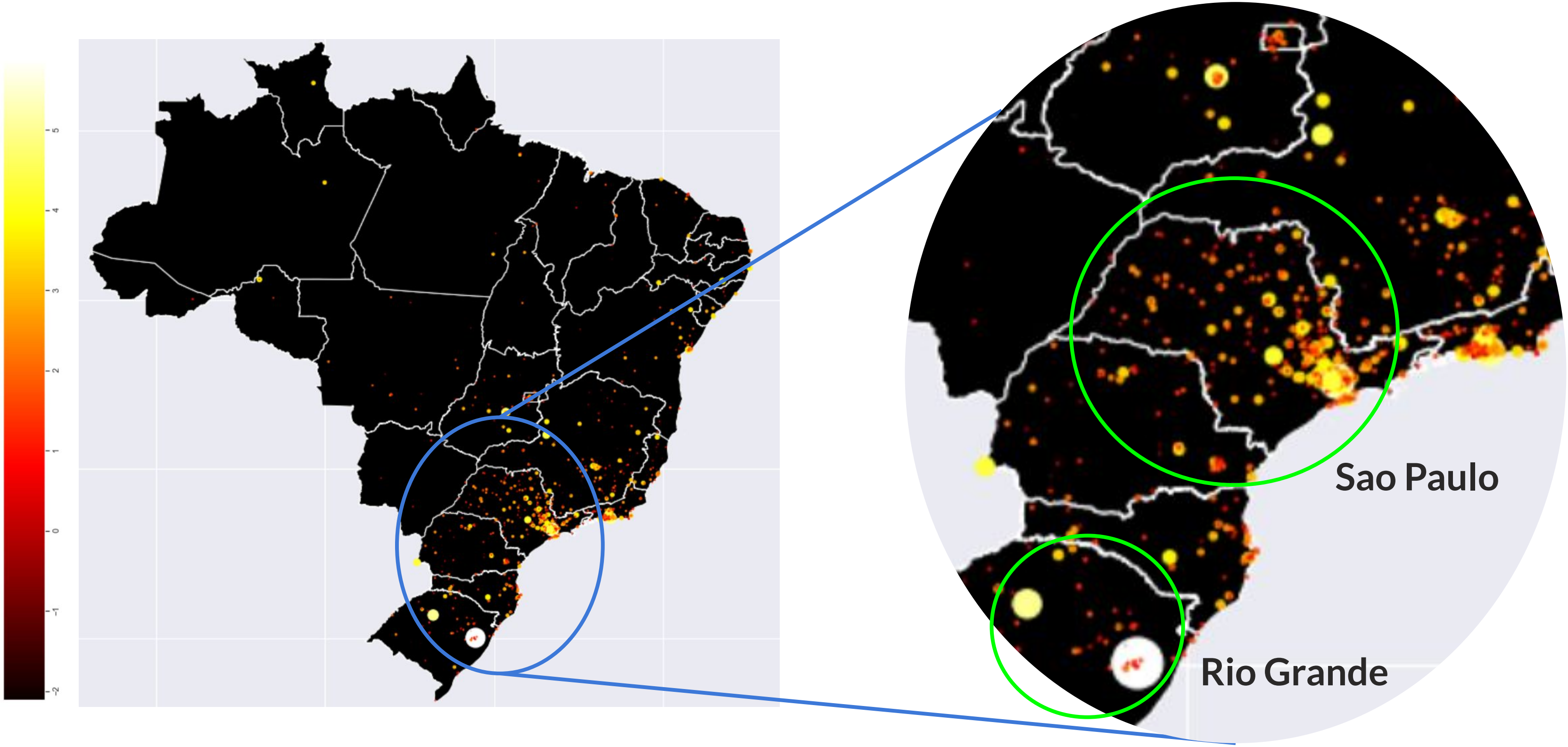


Log graph

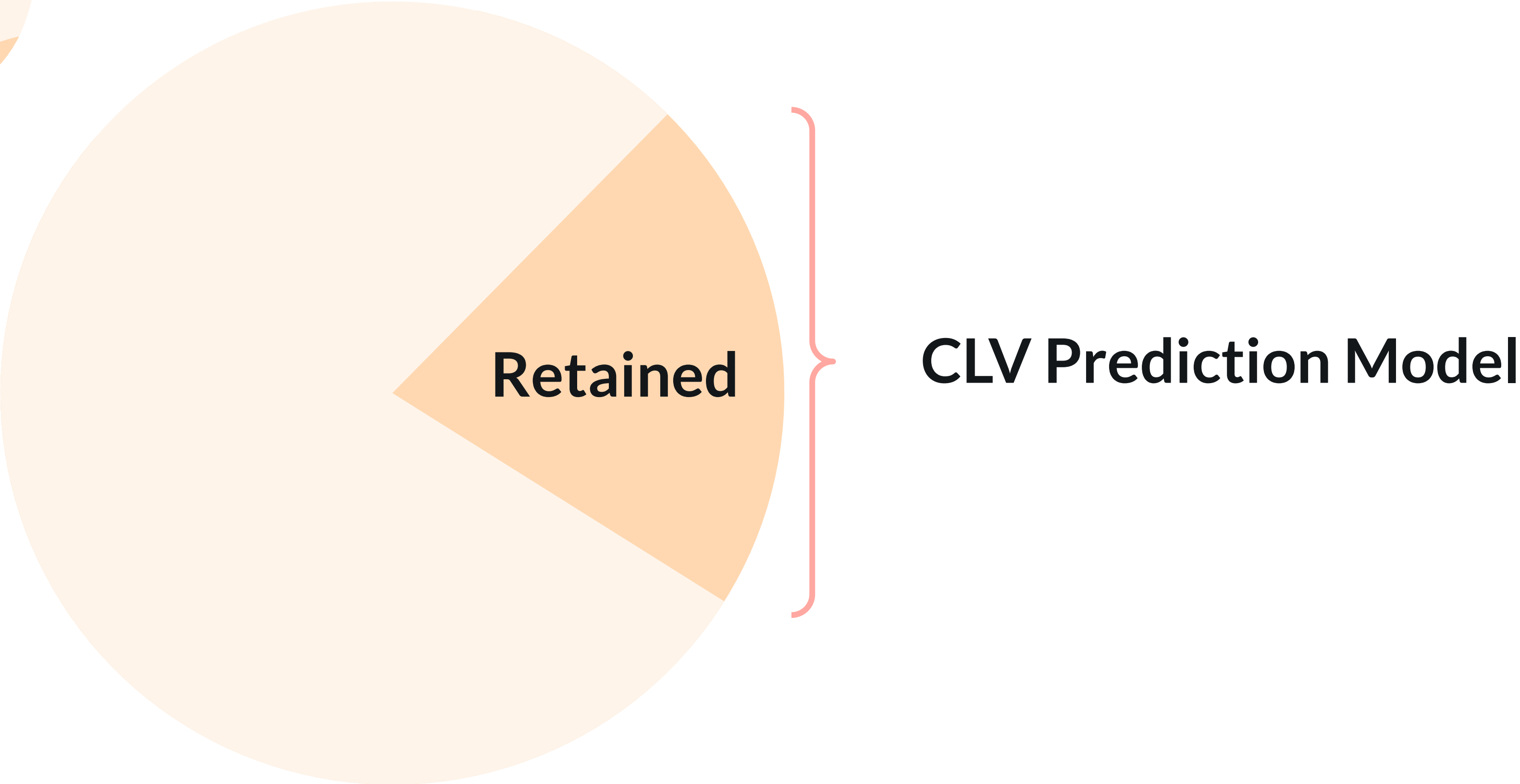
Geospatial Analysis

geolocation_lat	geolocation_lng	geolocation_city	geolocation_state	customer_unique_id	customer_city	customer_state	predicted_clv
-22.758886	-43.435672	nova iguacu	RJ	004288347e5e88a27ded2bb23747066c	nova iguacu	RJ	0.448267
-19.983999	-44.032573	belo horizonte	MG	011575986092c30523ecb71ff10cb473	belo horizonte	MG	0.572376
-12.548775	-38.712588	santo amaro	BA	011b4adcd54683b480c4d841250a987f	santo amaro	BA	1.326227
-23.590818	-46.551266	sao paulo	SP	012452d40dafae4df401bced74cdb490	sao paulo	SP	11.347272
-23.414784	-46.974449	pirapora do bom jesus	SP	012a218df8995d3ec3bb221828360c86	pirapora do bom jesus	SP	19.711841
...	...	...	...	...	...	...	...
-23.583436	-46.686463	sao paulo	SP	fe81bb32c243a86b2f86fbf053fe6140	sao paulo	SP	46.676690
-22.921733	-43.258689	rio de janeiro	RJ	fed519569d16e690df6f89cb99d4e682	rio de janeiro	RJ	0.628549
-23.122708	-48.921441	avare	SP	ff03923ad1eb9e32304deb7f9b2a45c9	avare	SP	2.563879
-17.197929	-40.220196	ibirajá	BA	ff8892f7c26aa0446da53d01b18df463	ibiraja	BA	0.397728
-22.486086	-48.547450	barra bonita	SP	ff922bdd6bafcdf99cb90d7f39cea5b3	barra bonita	SP	0.331310

Geospatial Analysis



Limitation



Retention Model

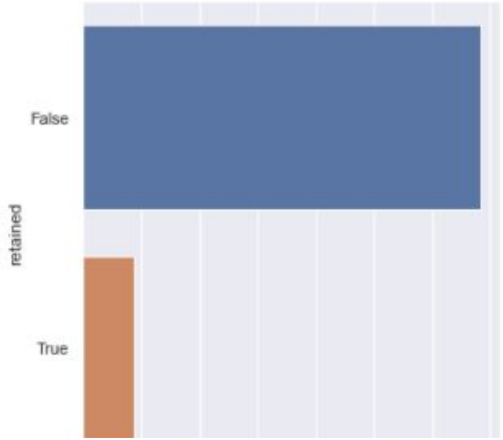
Exploratory Analysis

- Large imbalances between the 2 categories of TRUE and FALSE.
- Large proportion of customers are only 1-timers

Limitations

Classifiers will be biased towards the major classes and, hence, display poor classification rates on minor classes. It is also possible that the classifier predicts everything as a major class and ignores the minor class

```
In [2]: sb.catplot(y = 'retained', data = finaldata, kind = "count")
Out[2]: <seaborn.axisgrid.FacetGrid at 0x24cf1f123a0>
```



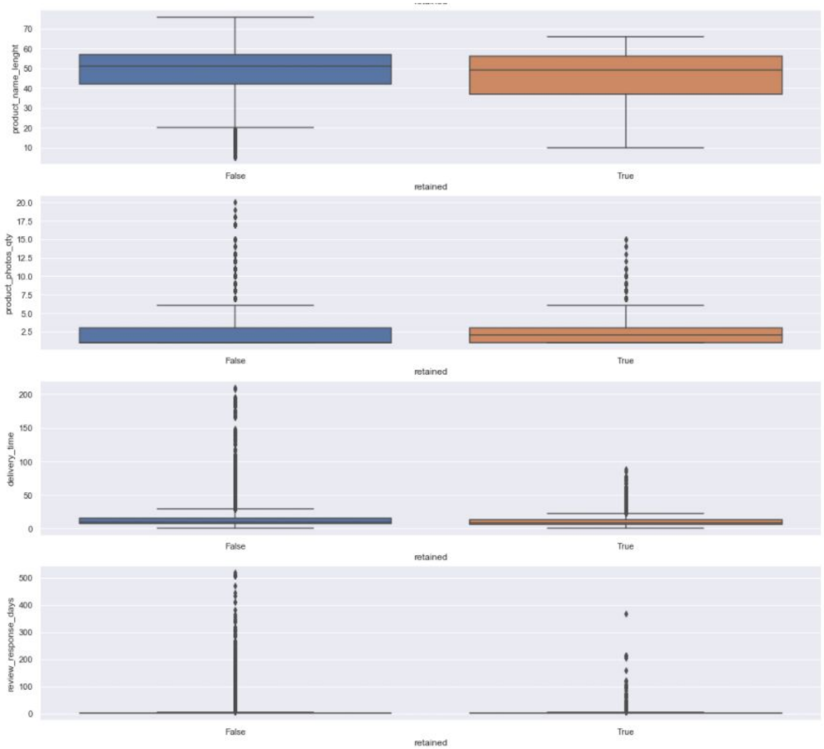

```
countY, countX = finaldata.retained.value_counts()
print("Ratio of classes is Y : N = ", countY, ":", countX)
print("Large imbalance between the classes")
```

```
Ratio of classes is Y : N =  170874 : 21887
Large imbalance between the classes
```

Exploratory Analysis

Univariate plot of Response (retained) and Predictors

- Largely no significant distribution caused by any of the variables
- Vendor variables may not have a significant impact in predicting retention.



Decision Tree Classifier

1. Extract predictor variables: review response time, delivery time, payment value etc.
2. Extract response variable: retain
3. Split data into train and test set
4. Fit decision tree with max depth = 3

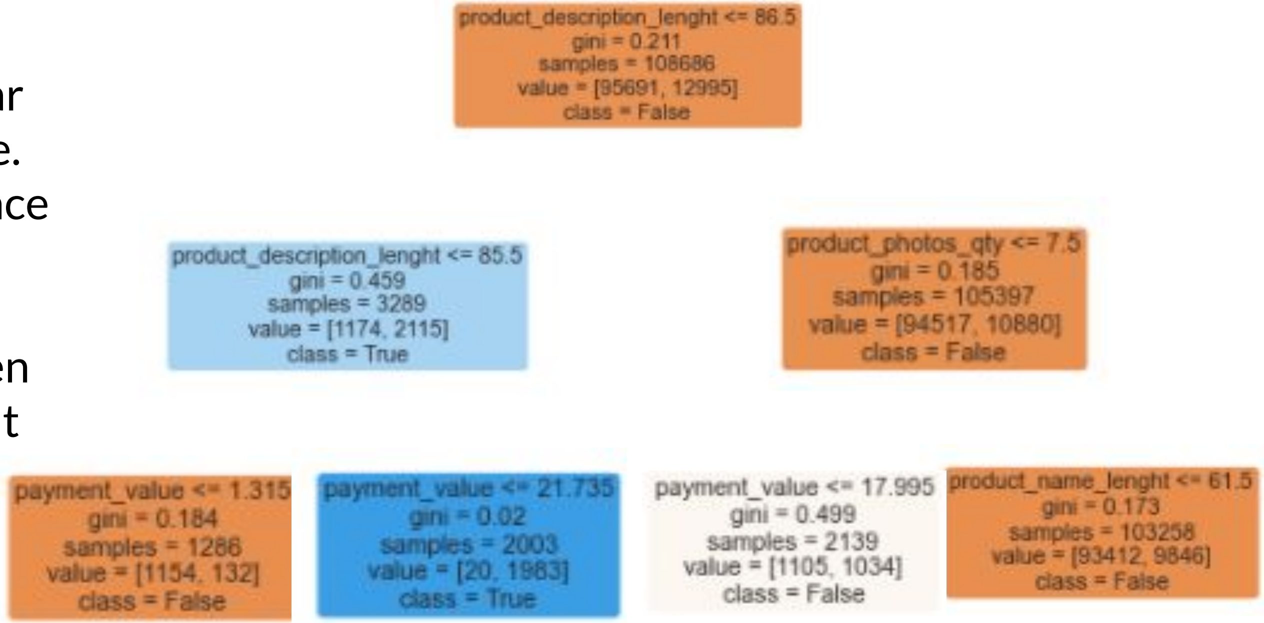
```
# Extract Response and Predictors
y = pd.DataFrame(finaldata['retained'])
X = pd.DataFrame(finaldata[['review_response_days','payment_value','product_description_lenght','review_score','product_name_leng

# Split the Dataset into Train and Test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25)

# Decision Tree using Train Data
dectree = DecisionTreeClassifier(max_depth = 3)  # create the decision tree object
dectree.fit(X_train, y_train)                    # train the decision tree model


f = plt.figure(figsize=(24,24))
plot_tree(dectree, filled=True, rounded=True,
          feature_names=X_train.columns,
          class_names=["False","True"])
```

Decision Tree Classifier

Gini coefficient analysis

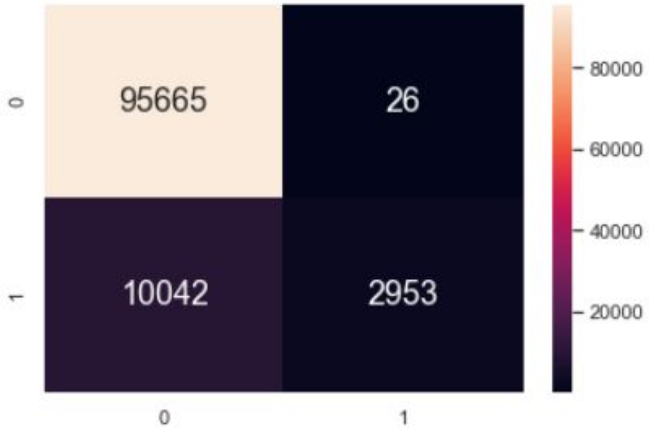
Payment value and description length appear multiple times in the tree. Suggests some significance towards retention.

They may be other hidden variables more important which we are unable to access.



Classification Accuracy

```
Train Data
Accuracy  :      0.9073661741162615

TPR Train :      0.22724124663332051
TNR Train :      0.9997282921068857

FPR Train :      0.00027170789311429497
FNR Train :      0.7727587533666795

<matplotlib.axes._subplots.AxesSubplot at 0x24ce4e6c160>
```




```
Test Data
Accuracy  :      0.9027850616909106

TPR Test :       0.22601232394366197
TNR Test :       0.9998421966230078

FPR Test :       0.00015780337699226762
FNR Test :       0.773987676056338

<matplotlib.axes._subplots.AxesSubplot at 0x24cebc1b2b0>
```

Conclusion

Conclusion

CLV Modelling
+
Geospatial Analysis

Segment regions of revenue generation

Pinpoint areas with growth potential

Conclusion

Retention Model

Customer Retention

Customer-Vendor Experience

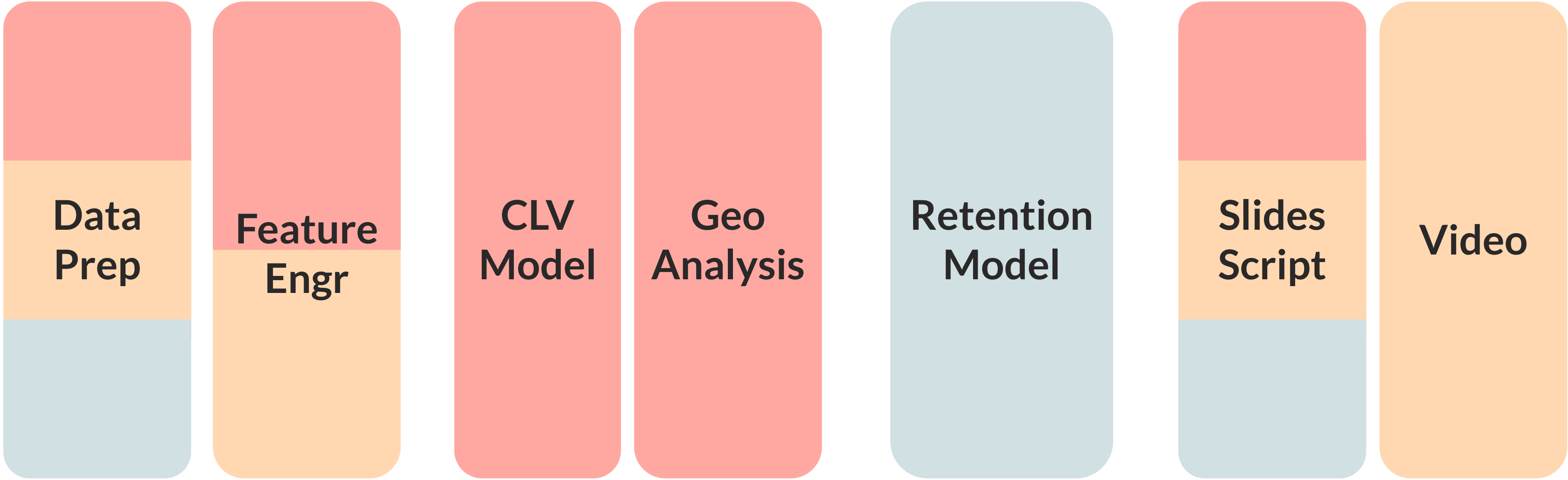
?

Platform Experience

- Customer Effort Score
- Customer Satisfaction

Workload Segregation

- Justyn
- Tianyi
- Brendan



Resources

https://www.kaggle.com/olistbr/brazilian-ecommerce/code
https://geopandas.org/gallery/plotting_with_geoplot.html
https://datashader.org/index.html
https://geopandas.org/docs/user_guide/mapping.html
http://darribas.org/gds15/content/labs/lab_03.html
https://www.qualtrics.com/au/experience-management/customer/customer-lifetime-value/
https://www.analyticsvidhya.com/blog/2020/10/a-definitive-guide-for-predicting-customer-lifetime-value-clv/
https://thepathforward.io/customer-lifetime-value-how-to-model-it-how-to-measure-it/
https://lifetimes.readthedocs.io/en/latest/
https://towardsdatascience.com/whats-a-customer-worth-8daf183f8a4f
https://dataorigami.net/blogs/napkin-folding/18868411-lifetimes-measuring-customer-lifetime-value-in-python
https://towardsdatascience.com/data-driven-growth-with-python-part-3-customer-lifetime-value-prediction-6017802f2e0f
https://services.google.com/fh/files/misc/exploratory_data_analysis_for_feature_selection_in_machine_learning.pdf
https://www.shopify.com.sg/encyclopedia/customer-lifetime-value-clv#:~:text=The%20lifetime%20value%20of%20a,your%20products%2C%20during%20their%20lifetime.
https://matplotlib.org/stable/contents.html#