Ecommerce Logistics Analysis

DRAFT

Ryan Borchardt
February 21, 2020

Olist

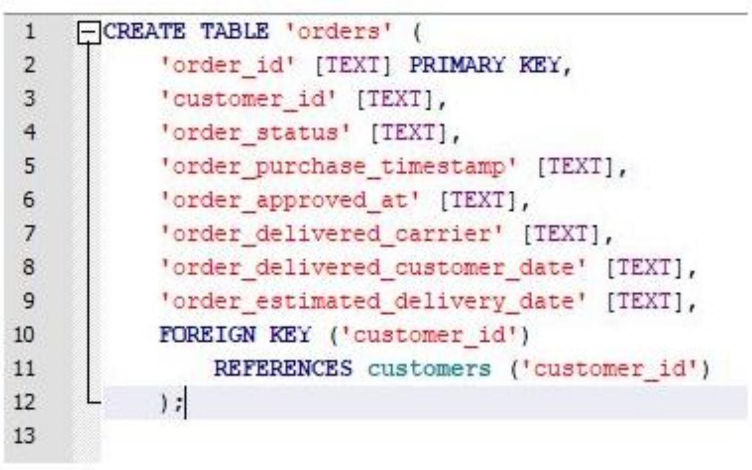
- Brazilian ecommerce company that connects sellers / small businesses to the main Brazilian marketplaces where customers can purchase their products
- Partners with sellers to shape and grow their digital presence by managing their analytics, marketing services, listings, orders, pricing, customer support, payments and shipping logistics
- https://olist.com/

Dataset

- Real, commercial data (messy)
- ~100,000 orders from Sept. 2016 – Oct. 2018
- Order, product, customer, seller, geographic data
- 10 csv files
- Kaggle (not a competition)

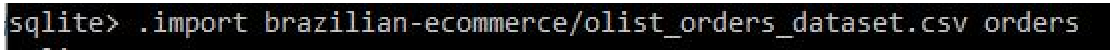
Creating the database

- Creating tables:



```
CREATE TABLE 'orders' (
    'order_id' [TEXT] PRIMARY KEY,
    'customer_id' [TEXT],
    'order_status' [TEXT],
    'order_purchase_timestamp' [TEXT],
    'order_approved_at' [TEXT],
    'order_delivered_carrier' [TEXT],
    'order_delivered_customer_date' [TEXT],
    'order_estimated_delivery_date' [TEXT],
    FOREIGN KEY ('customer_id')
        REFERENCES customers ('customer_id')
    );
```

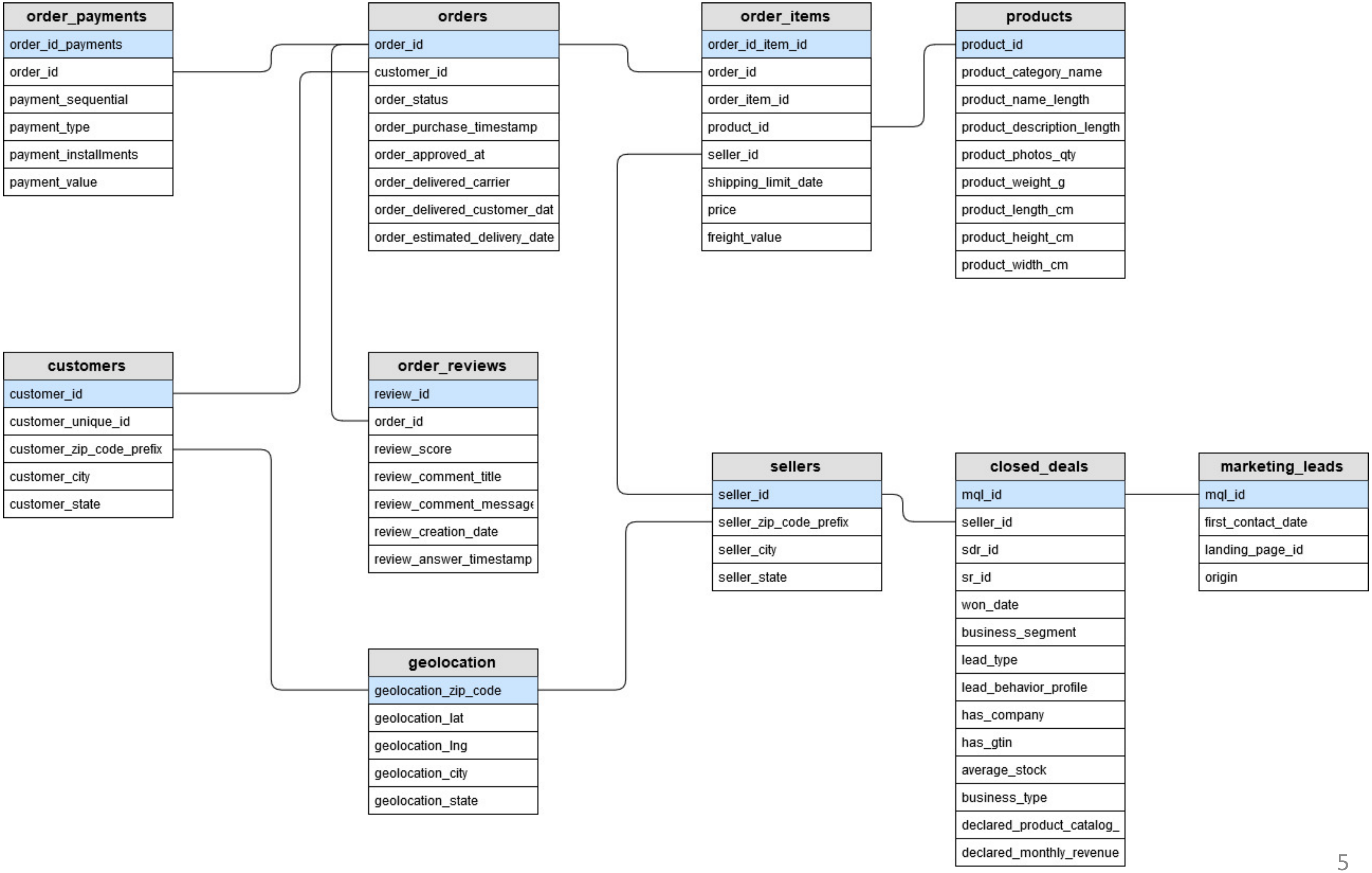
- Inserting data into tables:



```
sqlite> .import brazilian-ecommerce/olist_orders_dataset.csv orders
```

- Identifying primary and foreign keys
- Data normalization
  - Reducing data redundancy
- SQLite3 .db file

Schema



order_payments
order_id_payments
order_id
payment_sequential
payment_type
payment_installments
payment_value

orders
order_id
customer_id
order_status
order_purchase_timestamp
order_approved_at
order_delivered_carrier
order_delivered_customer_dat
order_estimated_delivery_date

order_items
order_id_item_id
order_id
order_item_id
product_id
seller_id
shipping_limit_date
price
freight_value

products
product_id
product_category_name
product_name_length
product_description_length
product_photos_qty
product_weight_g
product_length_cm
product_height_cm
product_width_cm

customers
customer_id
customer_unique_id
customer_zip_code_prefix
customer_city
customer_state

order_reviews
review_id
order_id
review_score
review_comment_title
review_comment_message
review_creation_date
review_answer_timestamp

sellers
seller_id
seller_zip_code_prefix
seller_city
seller_state

closed_deals
mql_id
seller_id
sdr_id
sr_id
won_date
business_segment
lead_type
lead_behavior_profile
has_company
has_gtin
average_stock
business_type
declared_product_catalog_
declared_monthly_revenue

marketing_leads
mql_id
first_contact_date
landing_page_id
origin

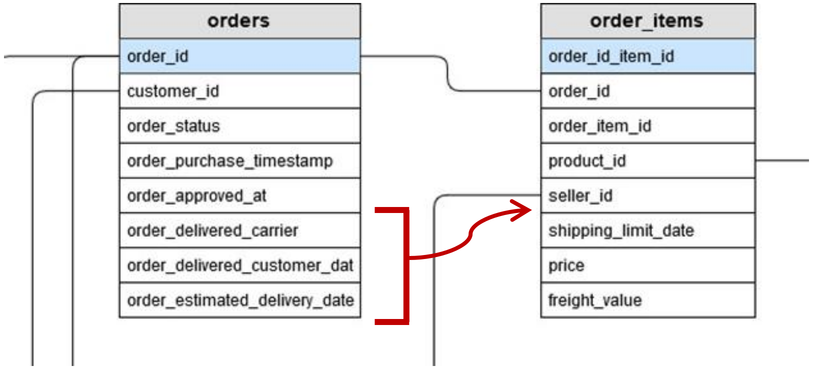
geolocation
geolocation_zip_code
geolocation_lat
geolocation_lng
geolocation_city
geolocation_state

Structural issue in Dataset

- Database structure: overly normalized -> missing important information as a result of the way that Olist structured their csv files:

Issue 1

- Fields in orders csv file ('order_delivered_carrier', 'order_delivered_customer_date', 'order_estimated_delivery_date') should be individual to each item, especially when the items in the order are from different sellers.
- Instead of having this information for each order_item within the order, based on the way this data has been structured by Olist, this information only exists for each order.

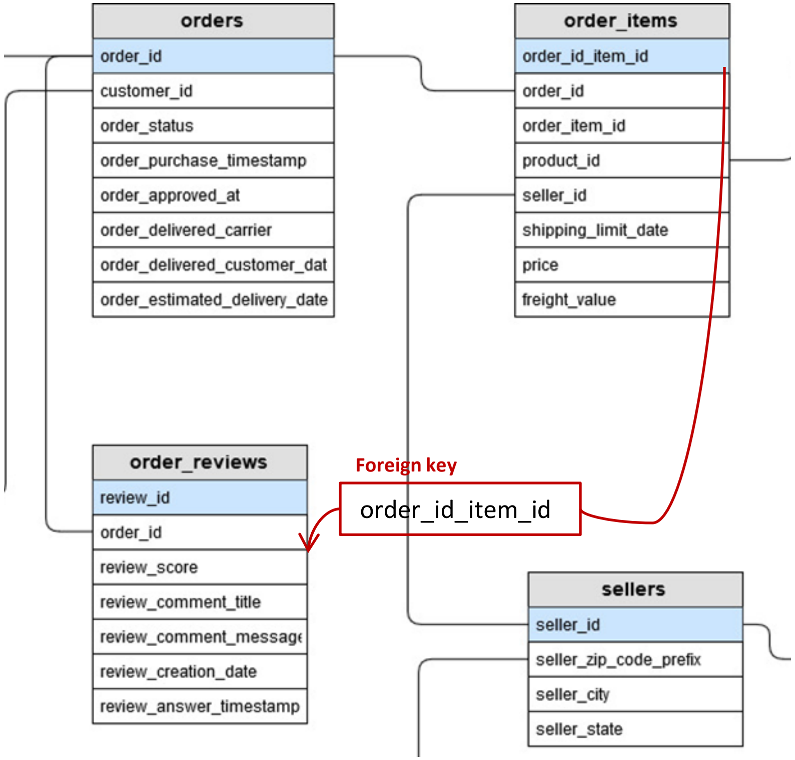


orders
order_id
customer_id
order_status
order_purchase_timestamp
order_approved_at
order_delivered_carrier
order_delivered_customer_dat
order_estimated_delivery_date

order_items
order_id_item_id
order_id
order_item_id
product_id
seller_id
shipping_limit_date
price
freight_value

Issue 2

- No field that can be used as connection/ relationship between the ‘order_items’ csv and the ‘order_reviews’ csv.
- For orders with multiple items, a customer may leave multiple reviews under that order (each review corresponding to a different item).
- With the way the datasets are currently structured, unable to tell which review corresponds to which item within an order.



orders
order_id
customer_id
order_status
order_purchase_timestamp
order_approved_at
order_delivered_carrier
order_delivered_customer_dat
order_estimated_delivery_date

order_items
order_id_item_id
order_id
order_item_id
product_id
seller_id
shipping_limit_date
price
freight_value

order_reviews
review_id
order_id
review_score
review_comment_title
review_comment_message
review_creation_date
review_answer_timestamp

sellers
seller_id
seller_zip_code_prefix
seller_city
seller_state

Foreign key

order_id_item_id

Solution

- Only looking at orders that have one unique product (orders can have multiple items as long as all of the items are the same product and from the same seller).
- Result is that the reviews and shipping information can be reliably matched with each item in an order.
- 95,430 (96%) of the orders meet these requirements

Data Sampling/Sourcing Issues

- Simple random sampling of orders that have reviews* ie excludes orders that don’t have reviews
- Forced to only analyze orders that only have one unique item due to structural issues mentioned earlier.
- This dataset is almost definitely NOT representative of the actual entire Olist dataset b/c will only be looking at orders that:
  - Have reviews submitted for them
  - Have only have one unique item.

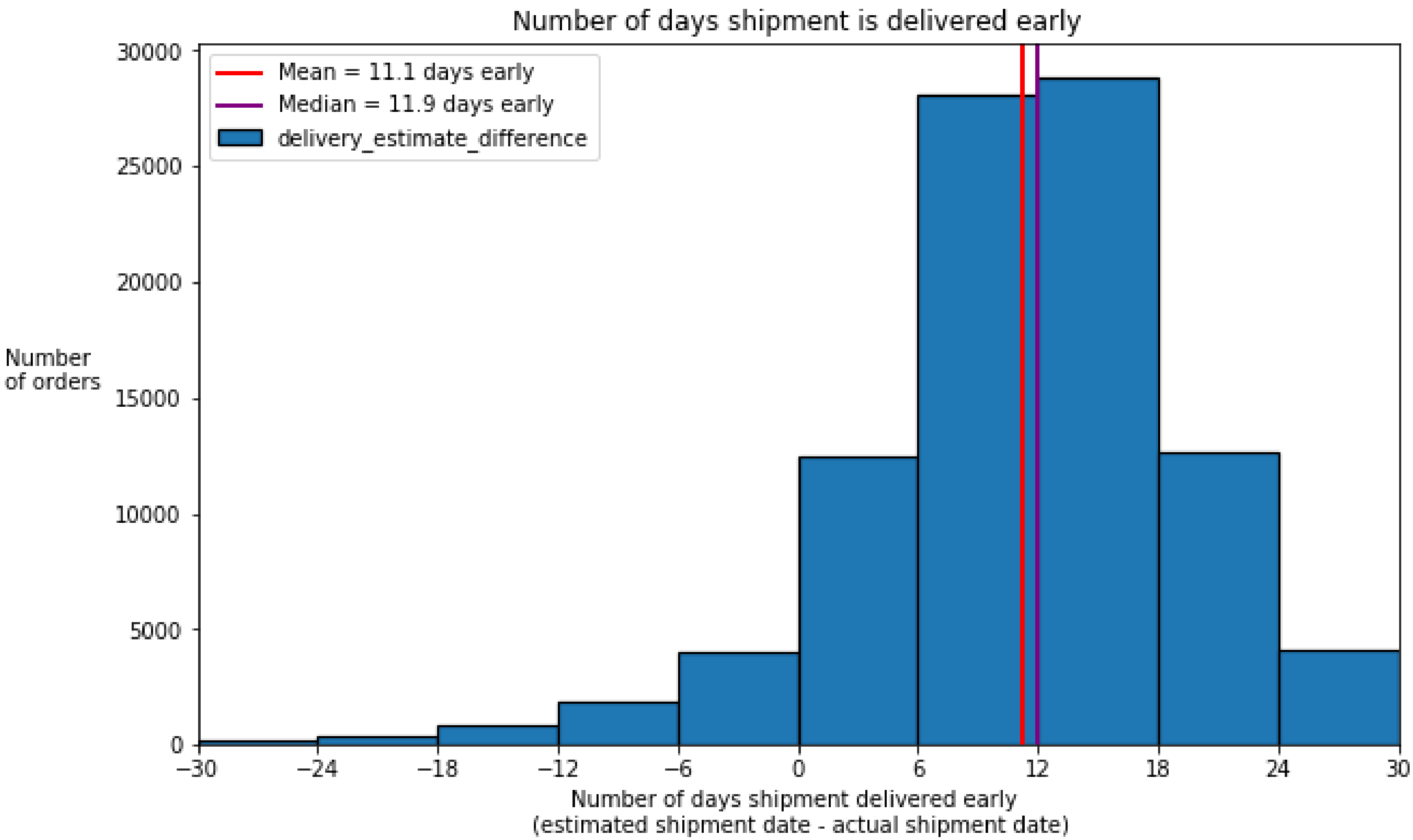
Missing Values

- Need to address missing values at every stage in analysis
- Removal vs imputation judgement calls
- See Appendix for details

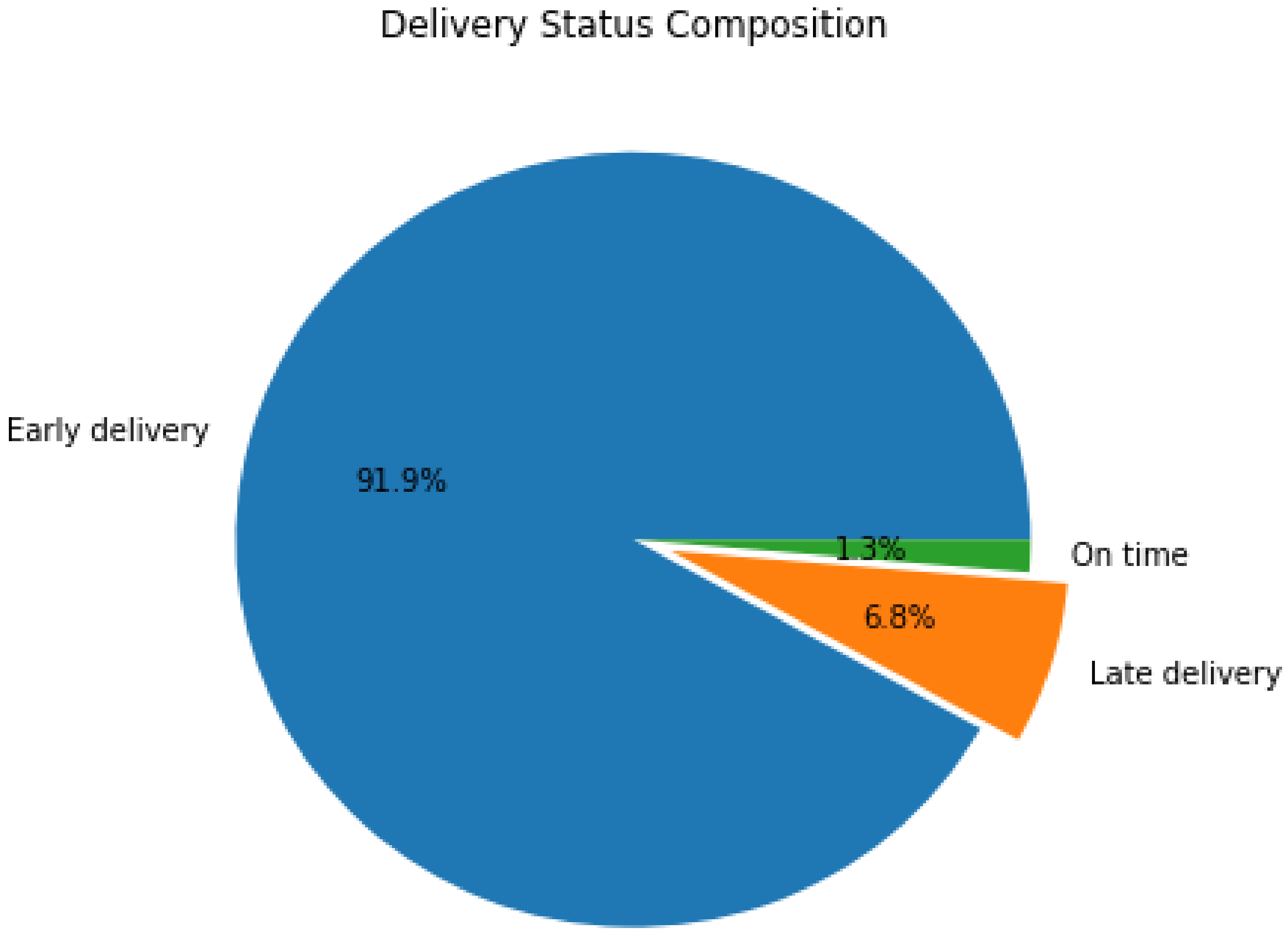
Logistics Analysis Overview

1. How accurate is Olist’s shipment delivery estimation system?
2. Does the delivery status (whether it is delivered late, on time or early) have an effect on the review scores of the products?
3. Late Delivery Investigation: Where can fault be attributed when the shipment is delivered later than estimated? Seller? Logistic partner? Olist’s delivery estimation system?
4. Geographic Analysis: Customer locations, seller locations, shipping distance estimation, shipping paths
5. Delivery Time prediction (regression)
   - Product weight/size, distance, estimated delivery time, price of product (5 features)
6. Shipment PCA
7. Dashboard(s)

1. How accurate is Olist’s shipment delivery estimation system (for delivered shipments)?

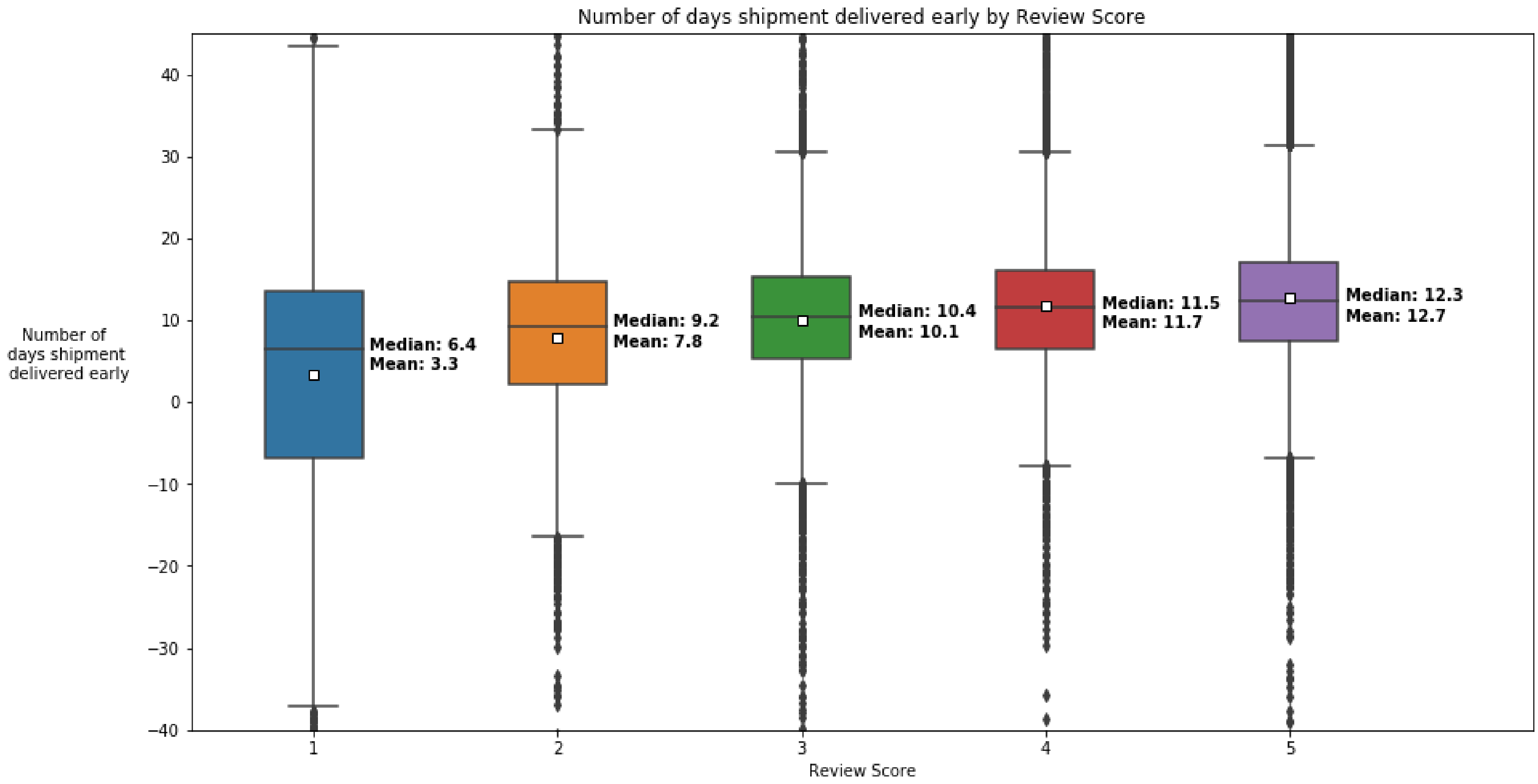


- Zoomed in, excluding outliers (2715 shipments (~2.7% of the shipments) were delivered >30 days early or >30 days late. Shipments ranged from 189 days late to 146 days early.



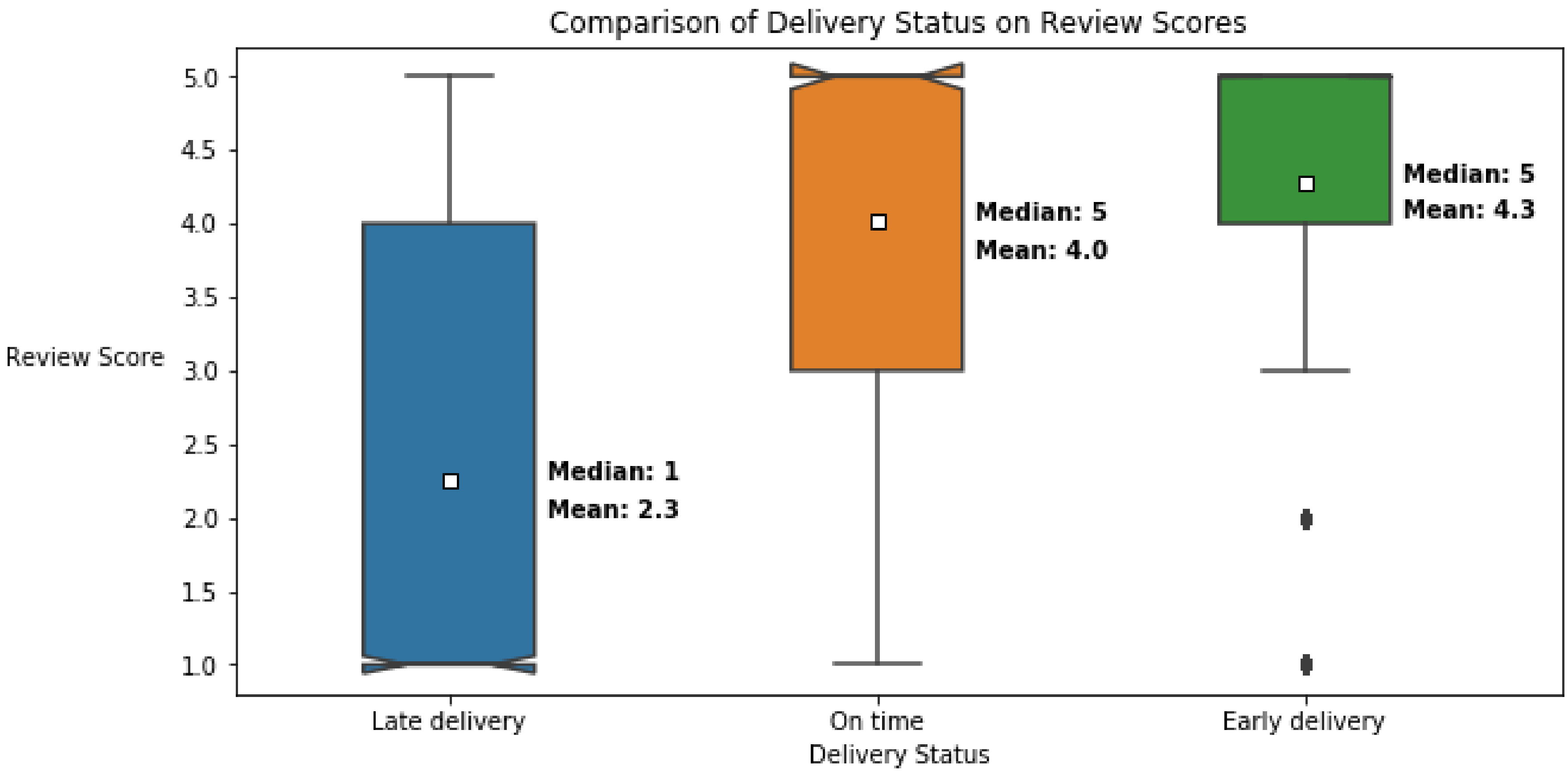
- Conservative and not accurate
- Current estimation system seems to involve a lot of guess-work – using conservative estimates to reduce frequency of late shipments (6.8%)

2. Does the delivery status (whether it is delivered late, on time or early) have an effect on the review scores of the products?



- Visually: the number of days the shipment is delivered early may have a relationship with the review score of the product that is shipped: The mean and median number of days the shipment is delivered early increases as the review score increases.

- A more in depth investigation is required to see if there is a statistically significant relationship between the number of days the shipment is delivered early and the review score of the product that is shipped.



- Visually: the delivery status of the shipment (whether it is delivered late, on time or early) may have a relationship with the review score of the product that is shipped: The mean and median review scores are dramatically increased for shipments that were delivered on time or early.
- A more in depth investigation is required to see if there is a statistically significant relationship between the delivery status of the shipment and the review score of the product that is shipped.
  - one-way ANOVA test : parametric assumptions not met
  - Kruskal-Wallis test to see if there is a statistically significantly difference between the median review scores for deliveries that are delivered late, on-time and early.

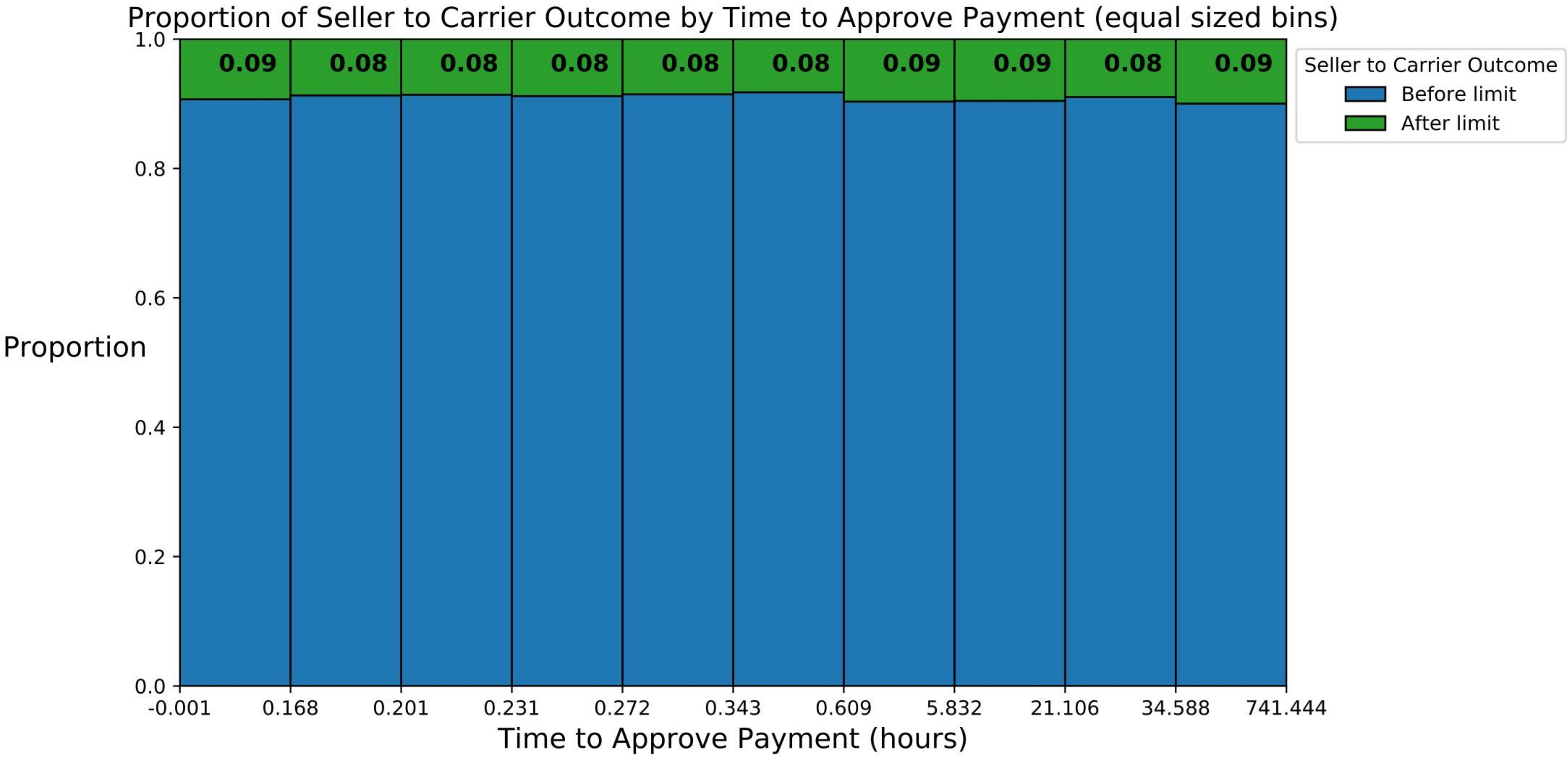
3. Late Delivery Investigation: Who is at fault when the shipment is delivered later than estimated?

- Payment approval system?
- Seller?
- Carrier?
- Olist’s shipment delivery estimation system?

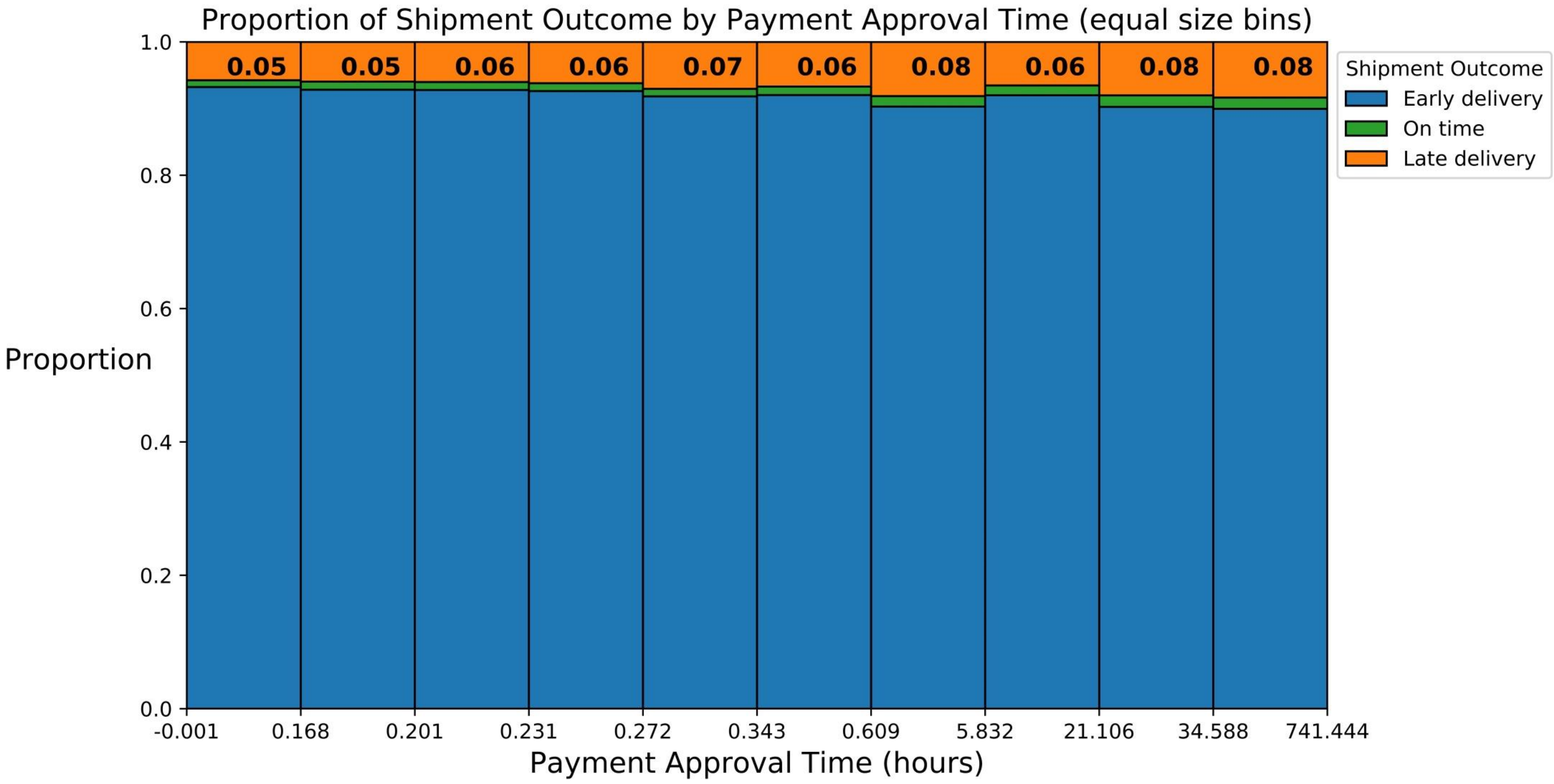
Late Delivery Investigation: Payment Approval System

- 33,032 shipments (35.4%) where seller brought order to carrier partner before the order payment was approved.
- 24,147 shipments (25.9%) where shipment was delivered to customer before the order payment was approved

Late Delivery Investigation: Payment Approval System



Late Delivery Investigation: Payment Approval System



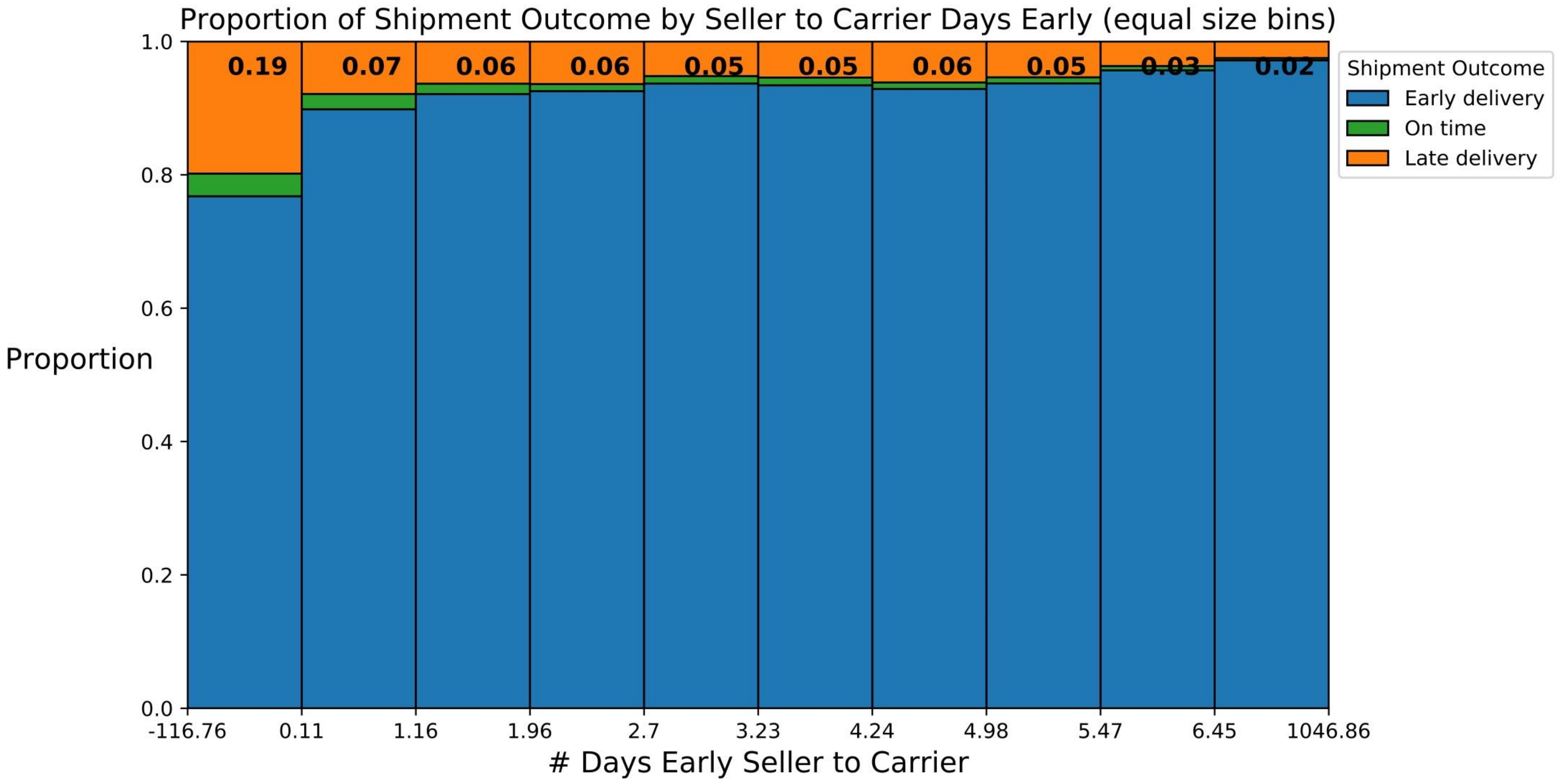
Late Delivery Investigation: Payment Approval System

Conclusion:

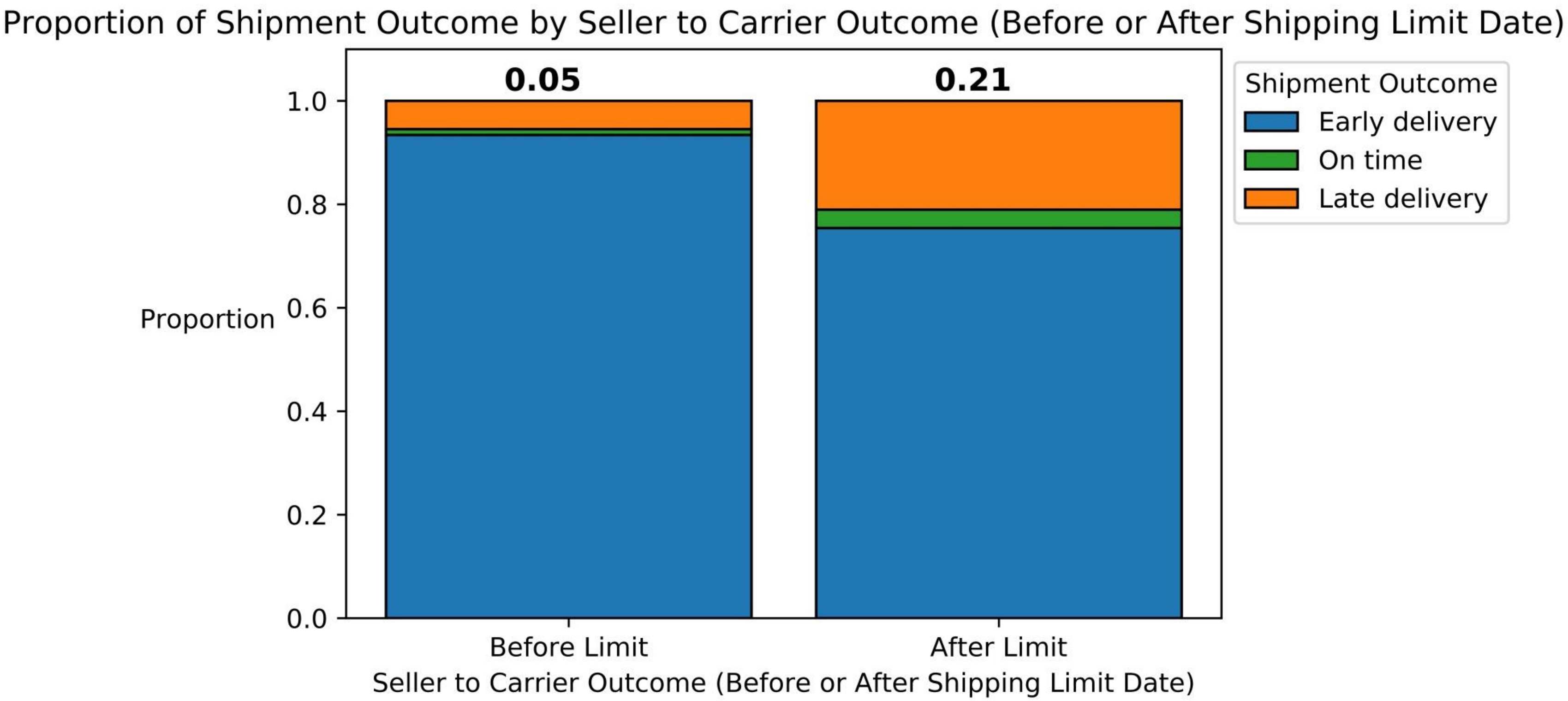
- The length of time to approve the payment doesn’t seem to have an impact on the shipment outcome.
- The payment approval system doesn’t to be at fault for shipments that are delivered late.

- See appendix for additional analyses

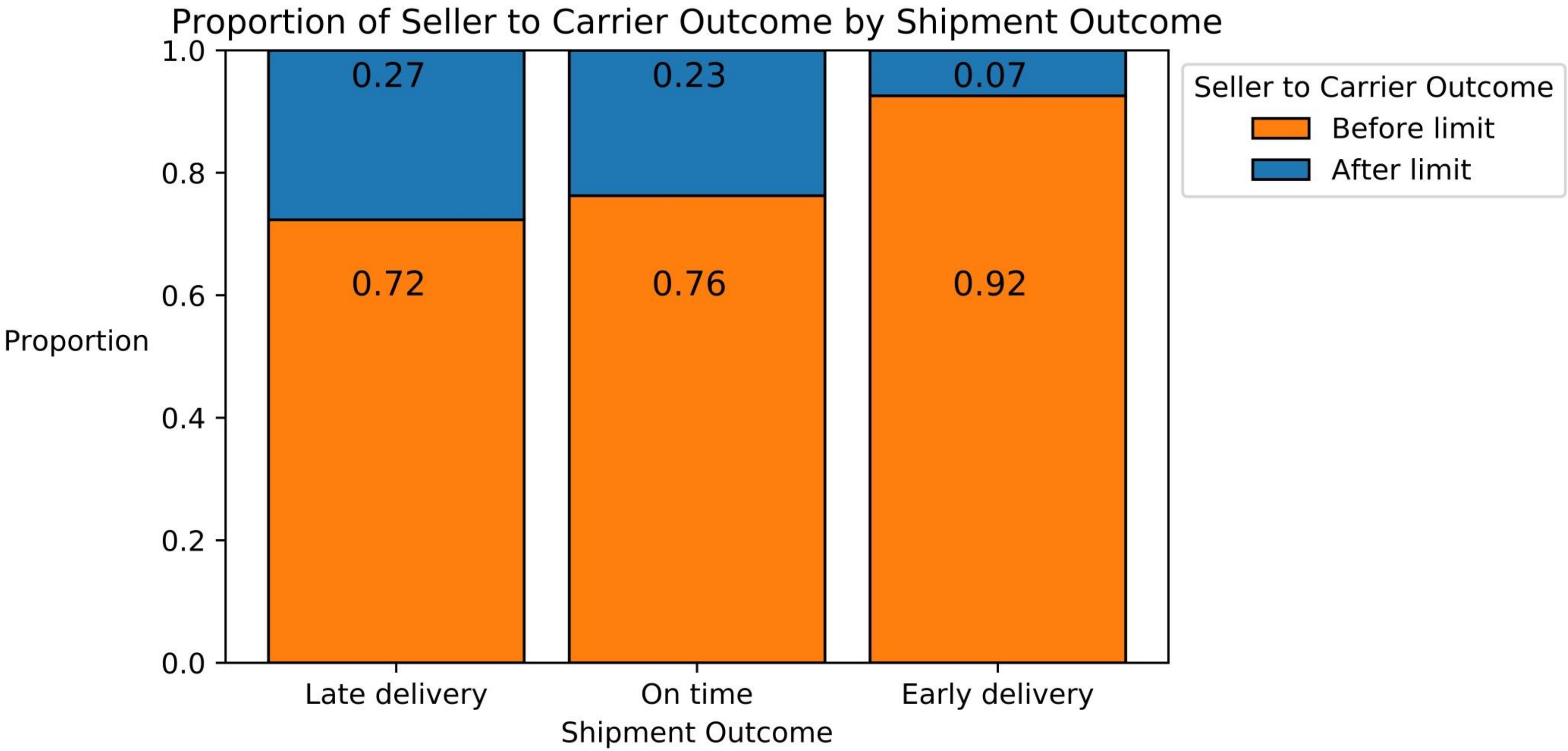
Late Delivery Investigation: Seller



Late Delivery Investigation: Seller



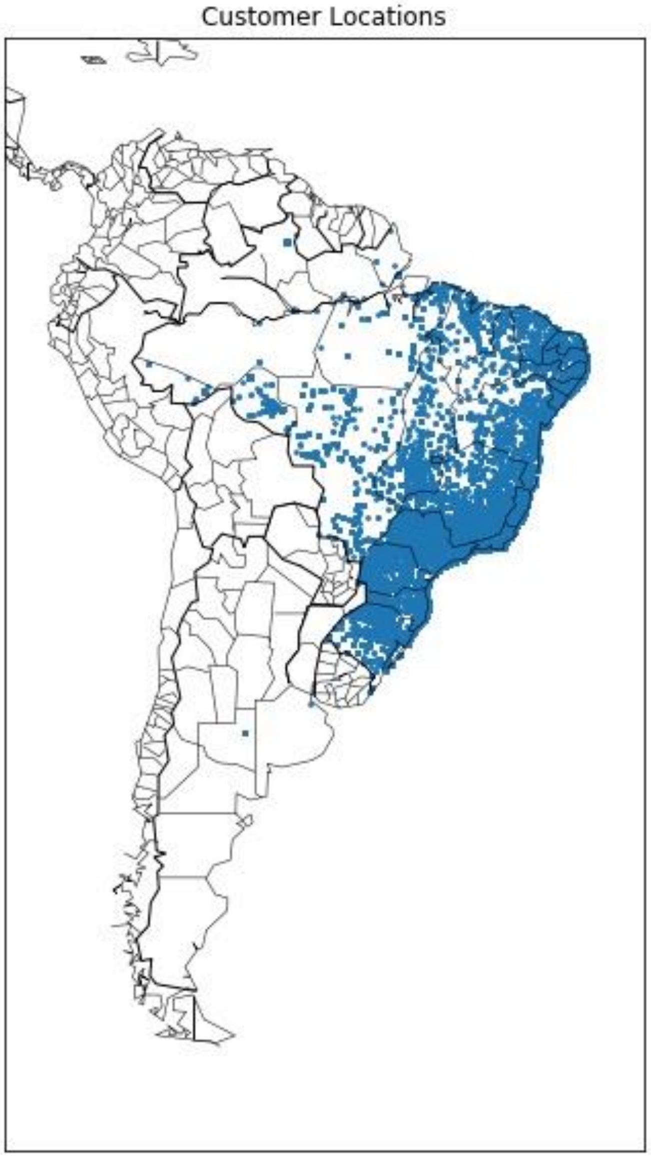
Late Delivery Investigation: Seller

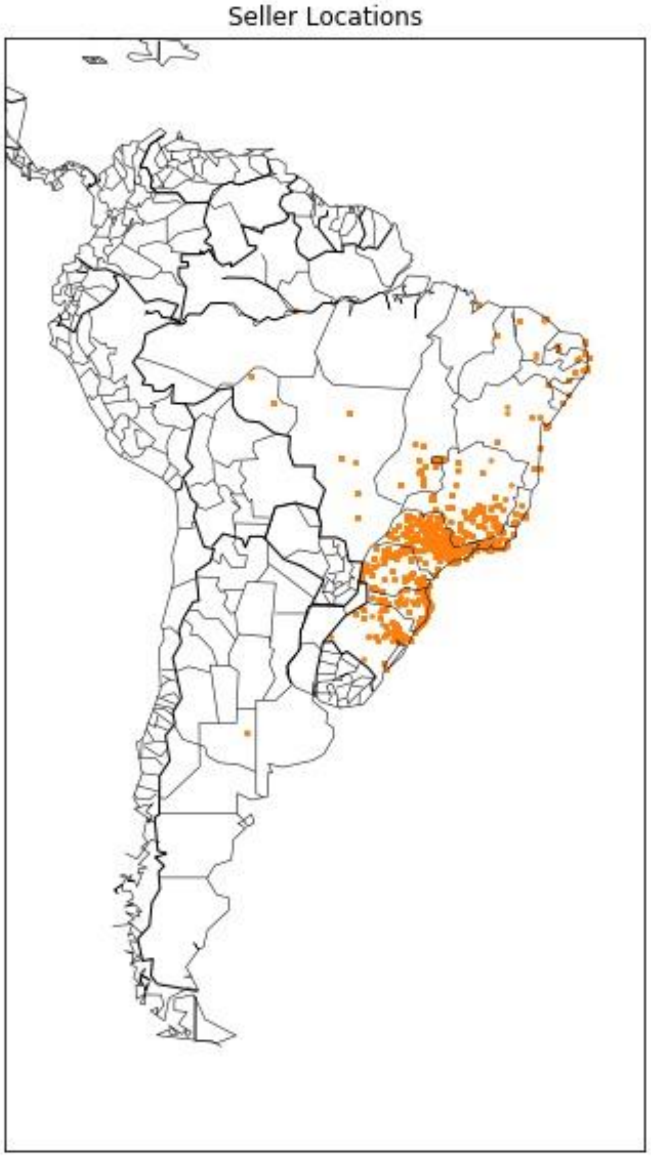


4. Geographic Analysis:

- Customer and Seller Locations
- Distance Traveled
- Shipment Paths
- Urban vs Rural Classification

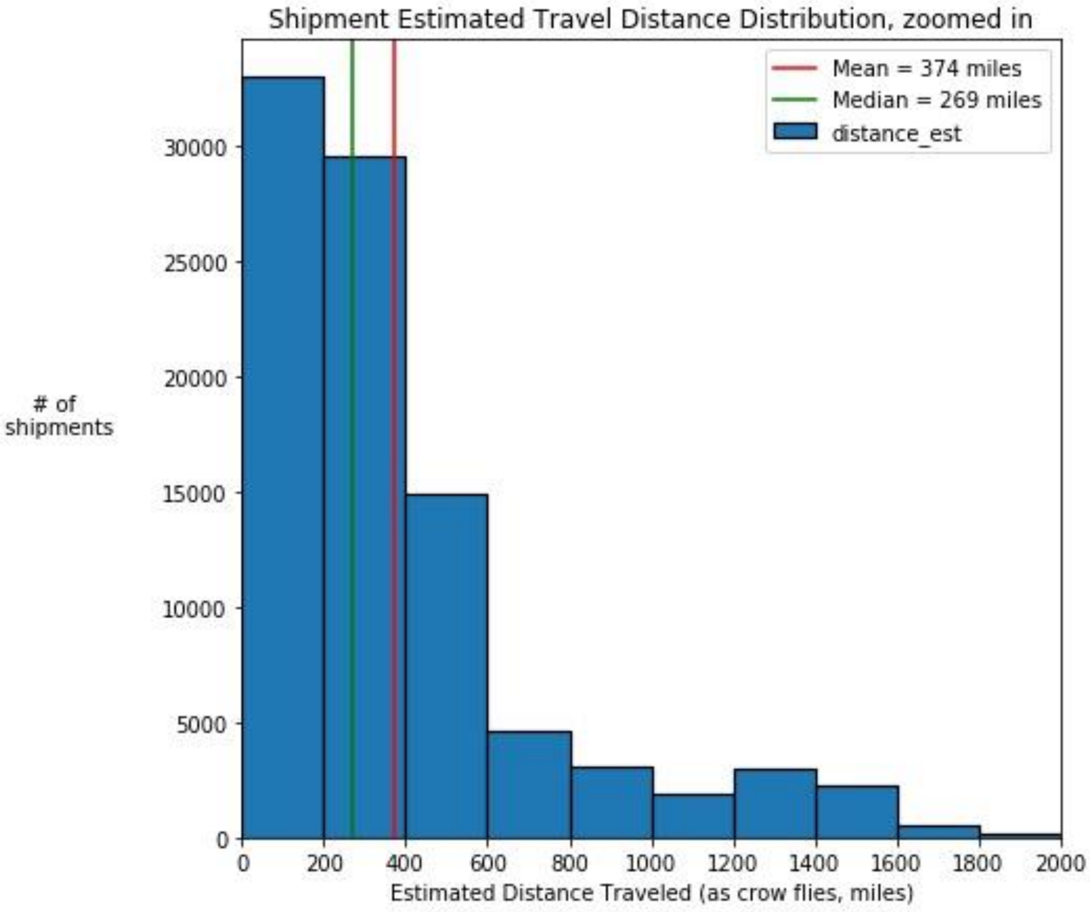
Locations


Customer Locations


Seller Locations

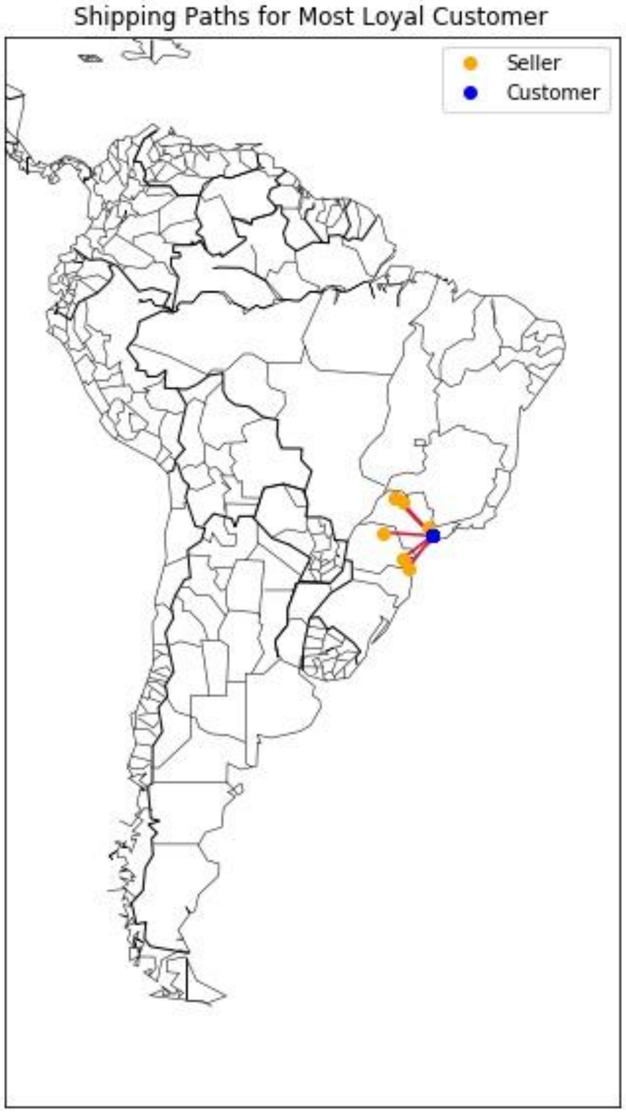
Shipment Distance Estimation

- As crow flies
- Haversine estimation
- Min: 0 miles
- Max: 2112 miles
- Number of shipments greater than 2000 miles (as the crow flies, miles): 35



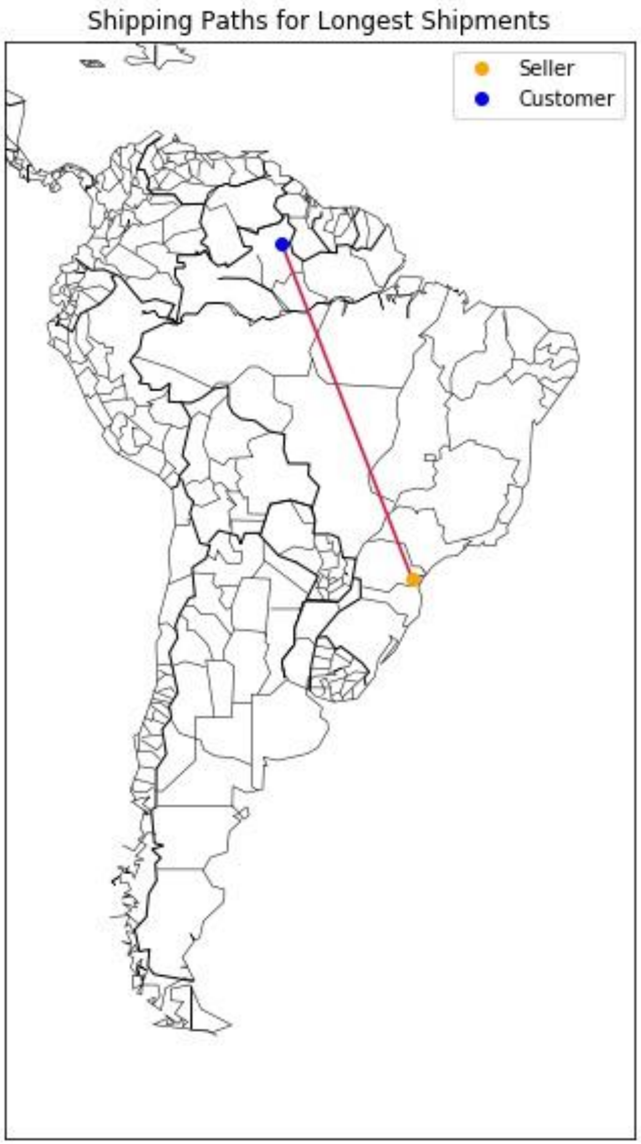
Shipment Paths: Most Loyal Customer

- Highest # of orders (16)
- Located in Sao Paulo, Sao Paulo.
- Shipments came from sellers in mostly in Sao Paulo, but also Parana and Santa Catarina.
- Estimated shipment travel distance from 3 miles to 310 miles.
- Mean: 105 miles
- Median: 4 miles



Shipment Path: Longest Shipment

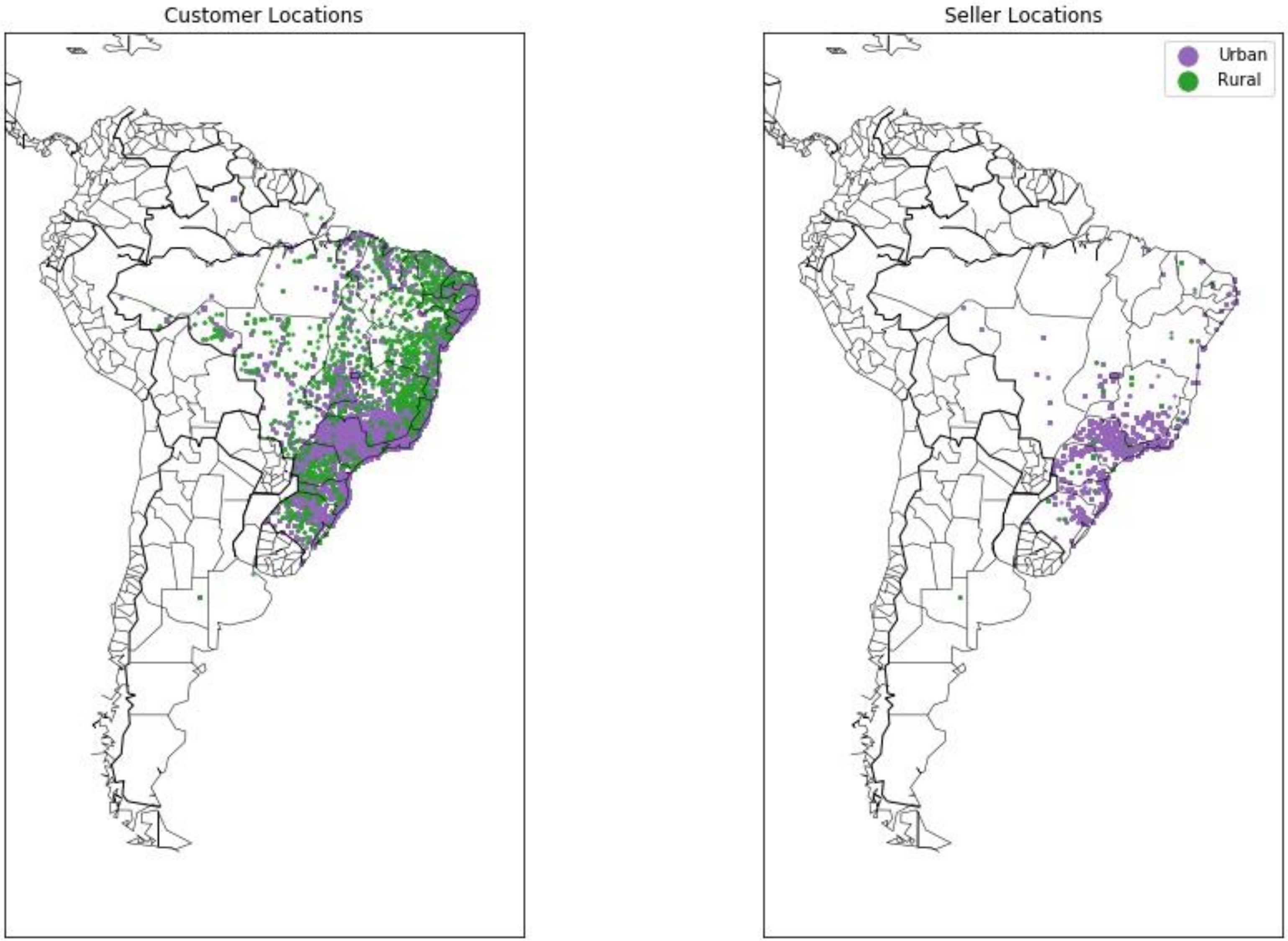
- The longest shipment was 2112 miles from Fazenda Rio Grande, PR to Boa Vista, RR.



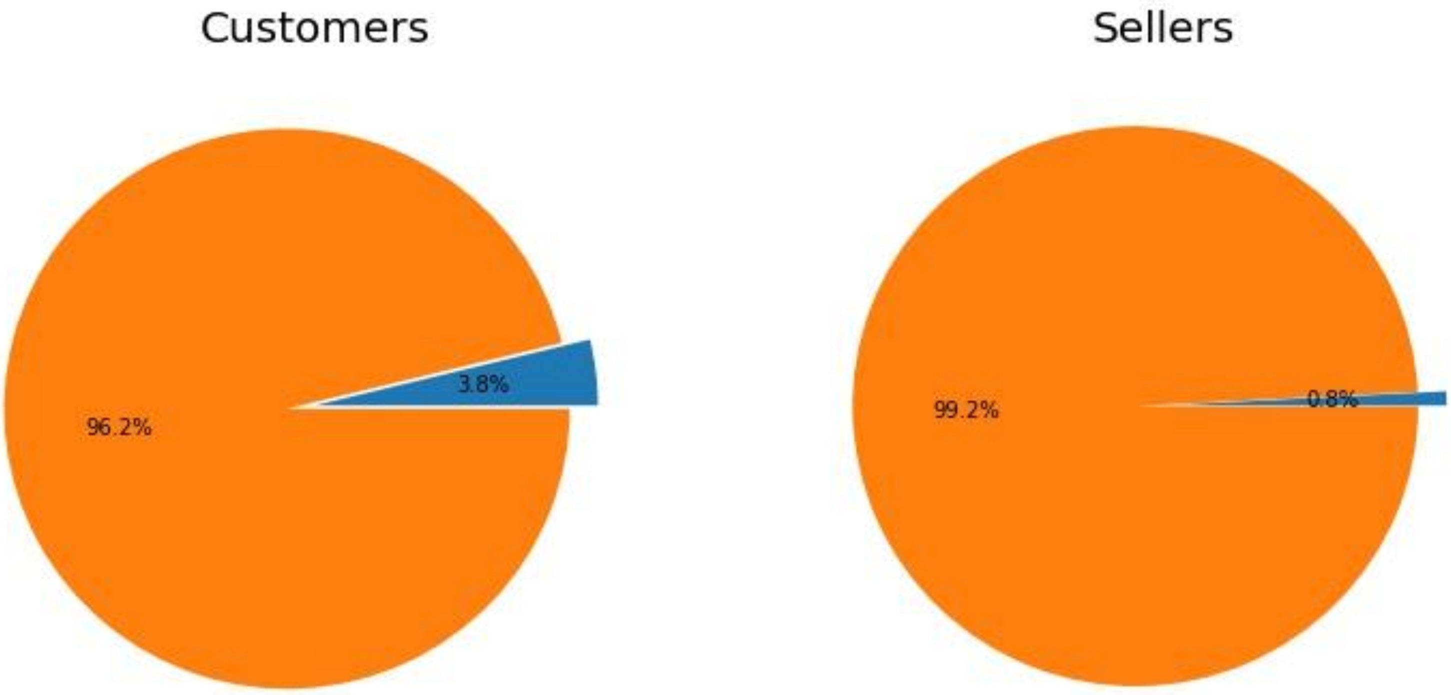
Urban vs Rural

- Dataset compiled from IBGE, contains data on all 5,573 Brazilian municipalities. Found on Kaggle: Source
- Each municipality is classified as urban or rural (1,456 are urban)
- Created function that for each unique geolocation zip code:
  - Identifies the closest urban municipality
  - The haversine distance to that urban municipality
  - Whether or not that location is in an urban area (within 20 miles of the nearest urban municipality)

Urban vs Rural

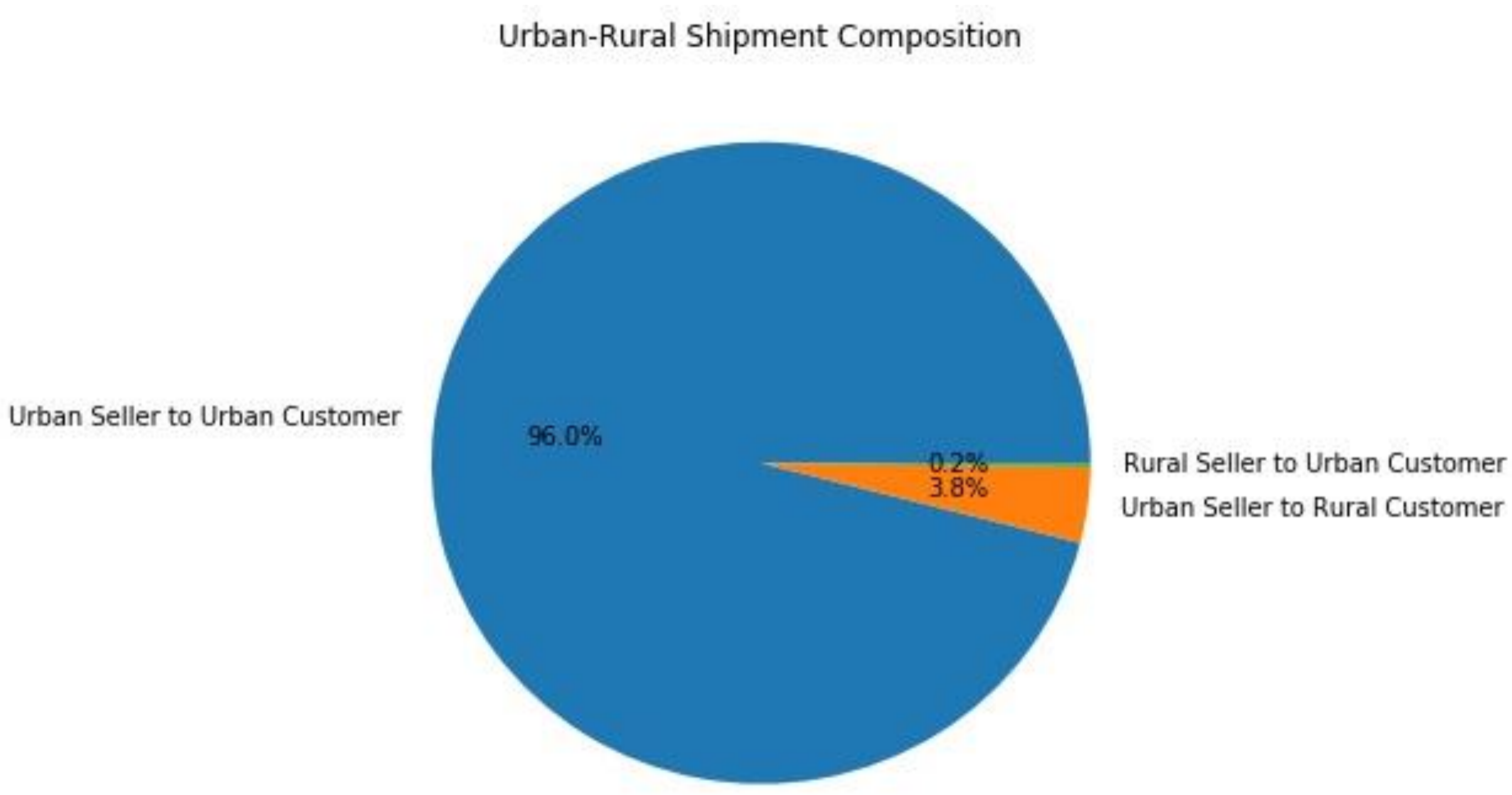


Urban-Rural Composition



- Overall Brazilian urban composition: 86.6% of population are in urban areas Source

Urban-Rural Shipment Composition



5. Delivery Time Prediction

Business Goal

1. Increase customer satisfaction: increase review scores by improving the delivery estimation system to reduce the number of late deliveries (shipments that are delivered after the estimated delivery date given to customer at the time of purchase)
2. Increase the number of purchases through Olist: Give less conservative delivery times (ie improve delivery estimate system to increase the number of on-time deliveries and decrease the number of early deliveries) so that customer will purchase through Olist –backed seller rather than through competitor (No data/statistics to back this. The business intuition is that potential customers may purchase through a competitor if they think that the delivery time will take longer through the Olist-backed seller).

Summary: Improve the delivery date estimation system such that:

1. Decrease the number of late deliveries
2. Decrease the number of early deliveries
3. Increase the number of on-time deliveries

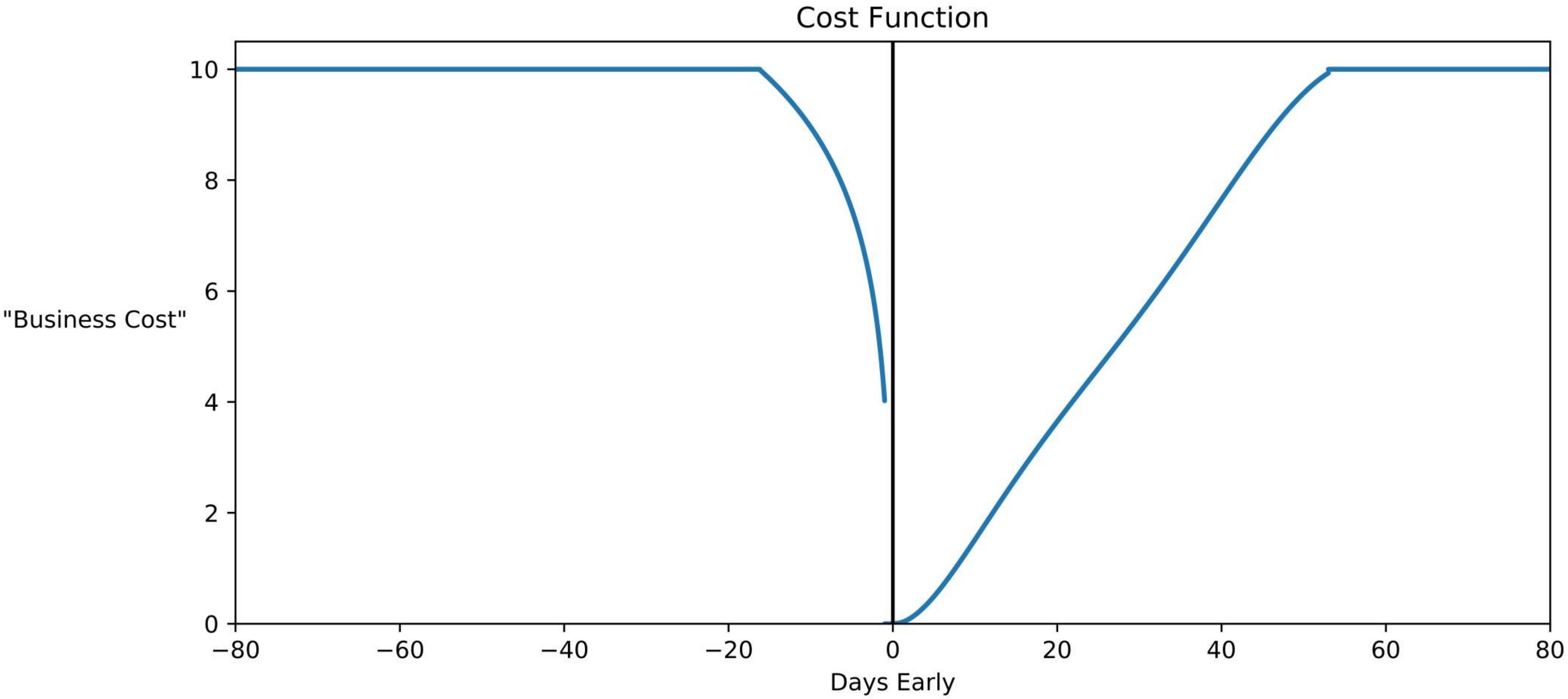
Uncertainty

- Don’t have enough data to quantify the business trade-off between how much “worse” a late delivery is compared to an early delivery.
    - A late delivery results in:
        - Customer may be more likely to leave a negative review and potential customers may be less likely to buy product due to these negative reviews
    - An early delivery means that:
        - The estimated delivery time given to customer at the time of purchase was longer than it needed to be. Potential customers may be more likely to purchase from competitor b/c they may be estimating a shorter delivery time.
- Assumption: Both late deliveries and early deliveries have a negative impact on revenue.
    - Is one outcome worse than the other? By how much?
    - Delivery that is 10 days late vs 2 days late vs 10 days early vs 2 days early? Etc
    - Need additional data and mathematical/statistical analyses to better answer these questions quantifiably

Choosing Scoring Metric

- Created custom cost function based on intuition/guess-work
- Serves as placeholder
- Strongly recommend further analysis (including additional data collection) to better determine custom cost function

Cost Function



Dummy Baseline Model(s)

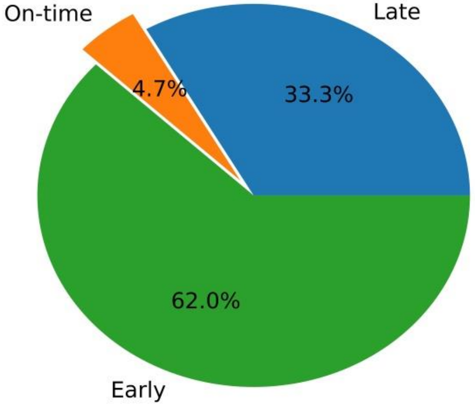
- Baseline prediction based on evaluation metric.

For regression:

- MSE, RMSE: Best prediction is mean value
- MAE: Best prediction is median value
  - See appendix for why

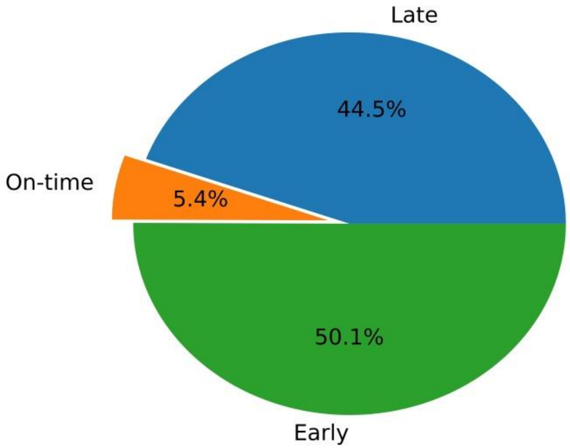
Dummy Baseline Model(s) Performance

Mean Baseline



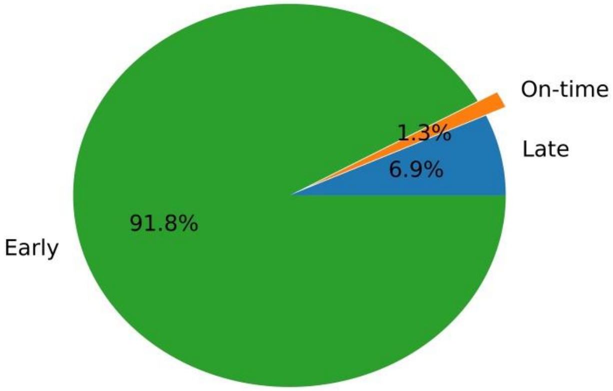
Business Cost: 92203

Median Basline



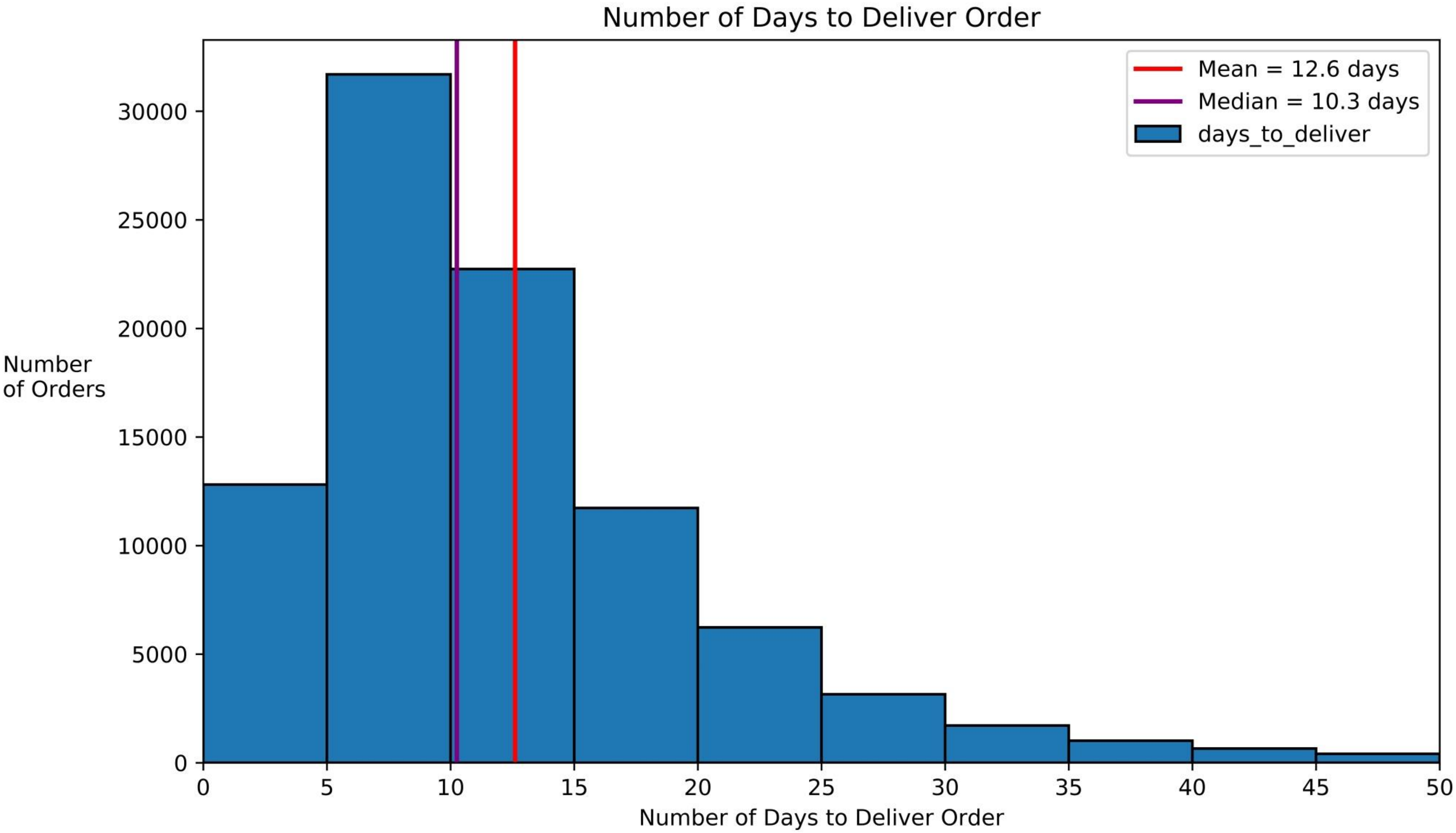
Business Cost: 113885

Olist_prediction

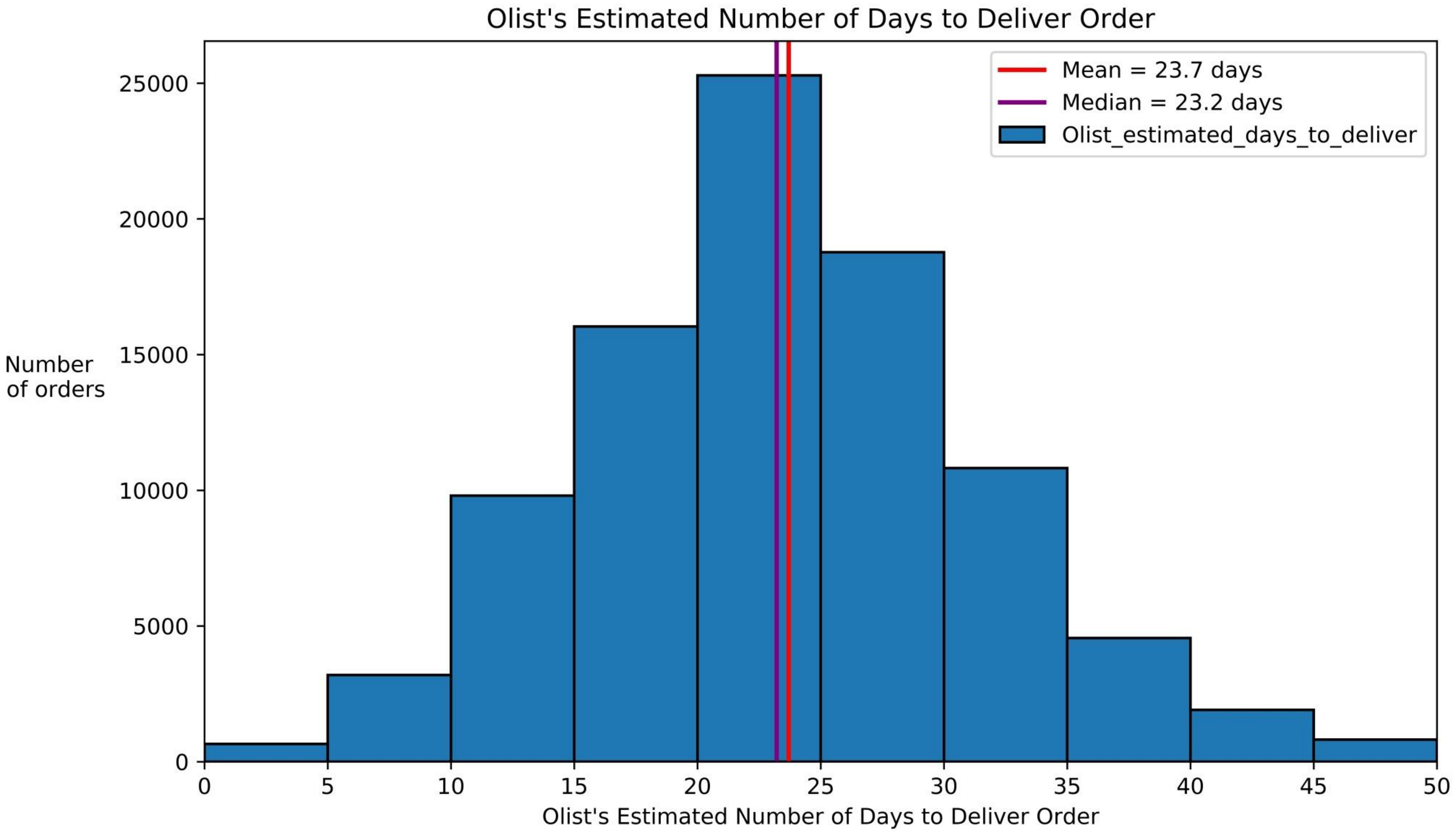


Business Cost: 78313

Target Variable



Olist’s Prediction System



Predictive Features

- Haversine Distance
- Total Payment
- Product Volume
- Product Length
- Seller to Carrier shipping limit date
- Rural vs Urban classification
- Same metropolitan area feature
- Freight Fee
- Number of photos
- Description length

Algorithm Selection

Linear regression models

- General (no penalty weight)
- Regularized: Lasso, Ridge, ElasticNet

Tree-based models

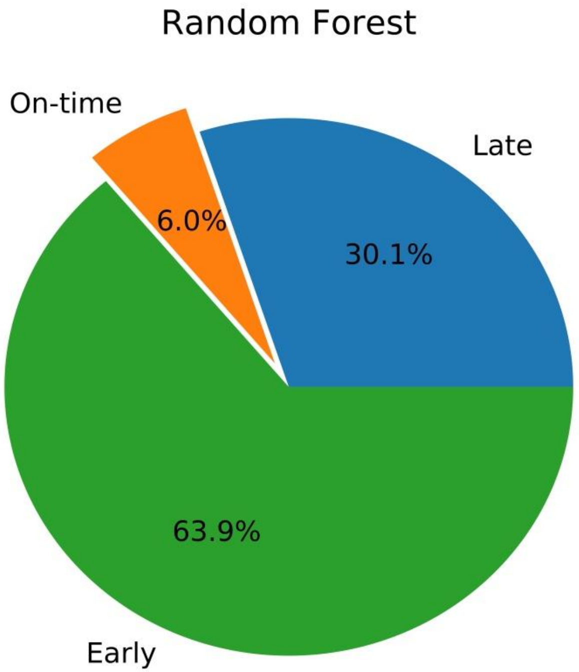
- Single decision tree
- Random Forest (bagging)
- Boosting Models

Nearest-Neighbors

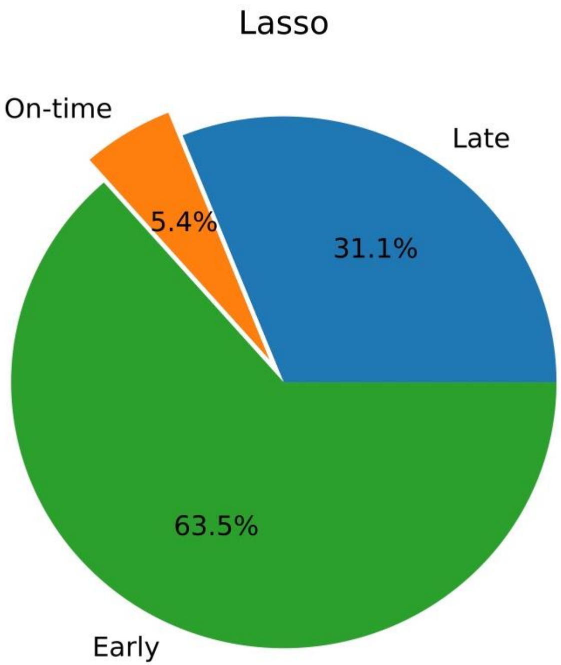
Support Vector Machine Models

See Appendix for deeper look at the algorithms*

Results


Random Forest

Business Cost: 78567

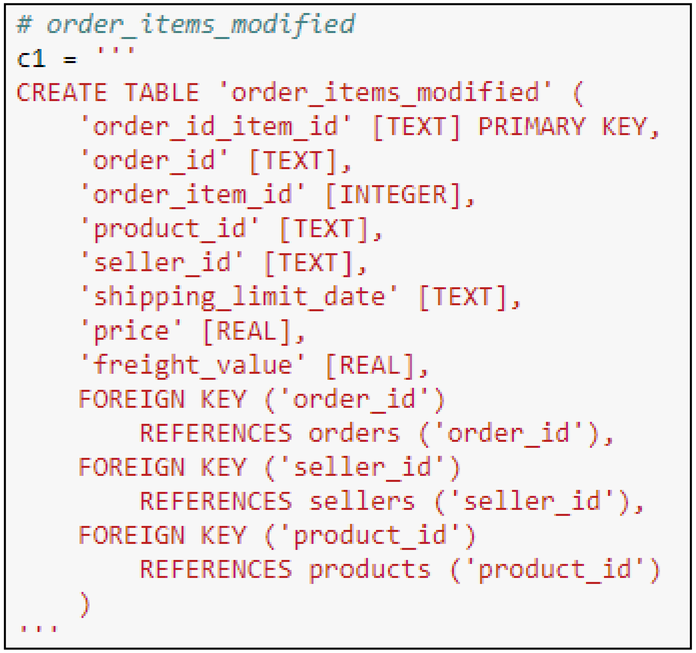

Lasso

Business Cost: 82260

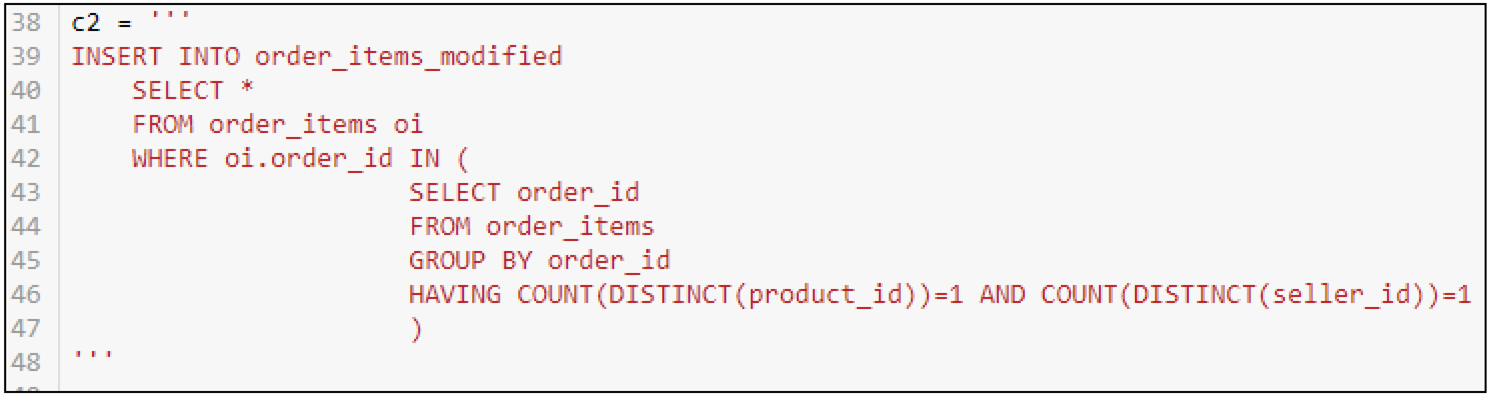
- Discussion of results to be continued*

Appendix

A0: Data Structure Solution



```
# order_items_modified
c1 = '''
CREATE TABLE 'order_items_modified' (
    'order_id_item_id' [TEXT] PRIMARY KEY,
    'order_id' [TEXT],
    'order_item_id' [INTEGER],
    'product_id' [TEXT],
    'seller_id' [TEXT],
    'shipping_limit_date' [TEXT],
    'price' [REAL],
    'freight_value' [REAL],
    FOREIGN KEY ('order_id')
        REFERENCES orders ('order_id'),
    FOREIGN KEY ('seller_id')
        REFERENCES sellers ('seller_id'),
    FOREIGN KEY ('product_id')
        REFERENCES products ('product_id')
    )
'''
```



```
c2 = '''
INSERT INTO order_items_modified
    SELECT *
    FROM order_items oi
    WHERE oi.order_id IN (
                      SELECT order_id
                      FROM order_items
                      GROUP BY order_id
                      HAVING COUNT(DISTINCT(product_id))=1 AND COUNT(DISTINCT(seller_id))=1
                      )
'''
```

A1: Missing Values for: how accurate is Olist’s shipment delivery estimation system?

- Exclusively used orders_modified table.

Field	# blank values
‘order_delivered_customer_date’	2152
‘order_delivered_carrier’	992
‘order_approved_at’	14

A1: Missing Values for: how accurate is Olist’s shipment delivery estimation system?

- For now: only shipments that were delivered and dropped the 23 shipments that were missing shipment dates.
- 93,258 shipments/orders

‘order_status’ value	Count
delivered	93,281
shipped	1,085
canceled	457
invoiced	304
processing	296
unavailable	5
approved	2

A2: Missing Values for: Does the number of days early the shipment is delivered have an effect on the review score of the product in the shipment?

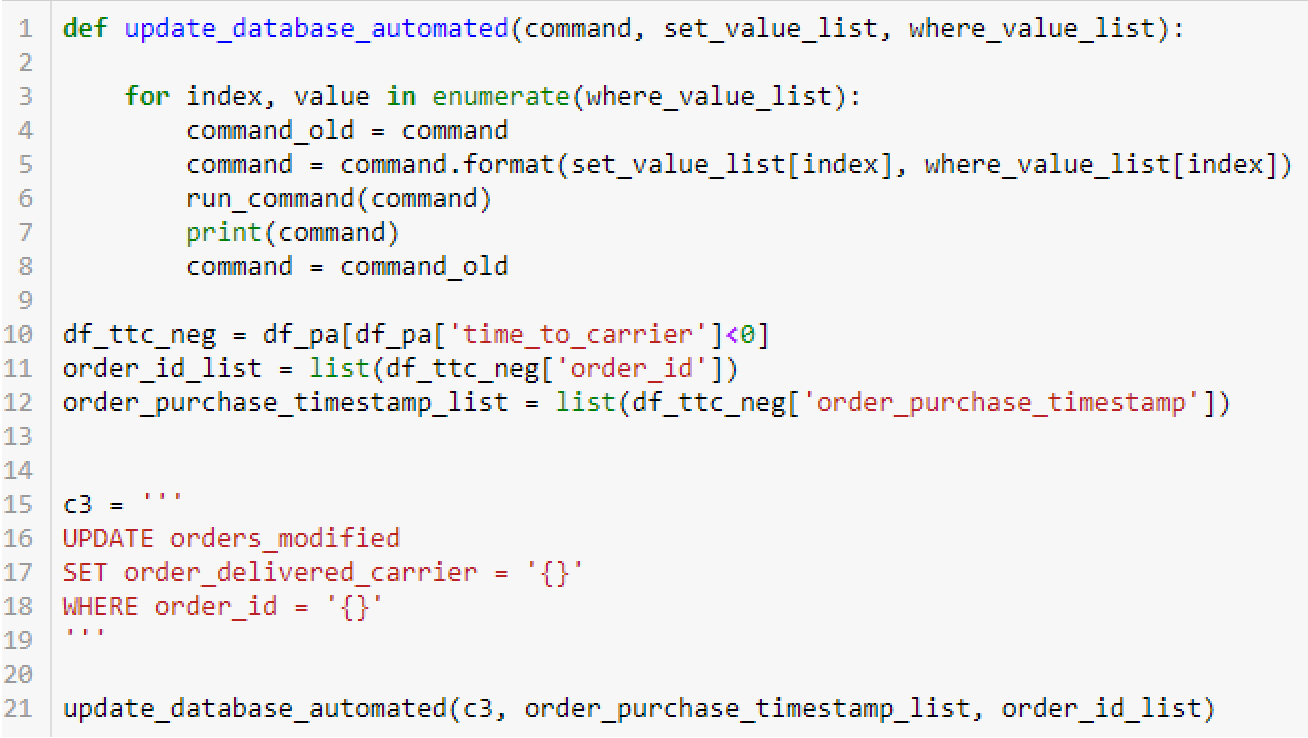
- Using ‘orders_modifed’ and ‘order_reviews_modified’ tables. Only looked at shipments that were delivered
- Dropped the 23 shipments that were missing shipment dates.
- Dropped the 441 shipments that had no corresponding review score.
- 92,817 shipments/orders.

A3: Incorrect Values for: Late Delivery Investigation

- 159 records: ‘order_delivered_carrier’ date that is earlier than the ‘order_purchase_timestamp’ date.
- Results in the ‘time_to_carrier’ field having negative value.
- Assuming that for these shipments, the item was already in the warehouse/storage area of the logistic partner
- Corrected these records so that the ‘order_delivered_carrier’ date is the same as the ‘order_purchase_timestamp’ so that the ‘time_to_carrier’ value is 0 – meaning that it took zero time for the item to get from the seller and to their logistic carrier partner (because the logistic carrier partner already this item in their warehouse).

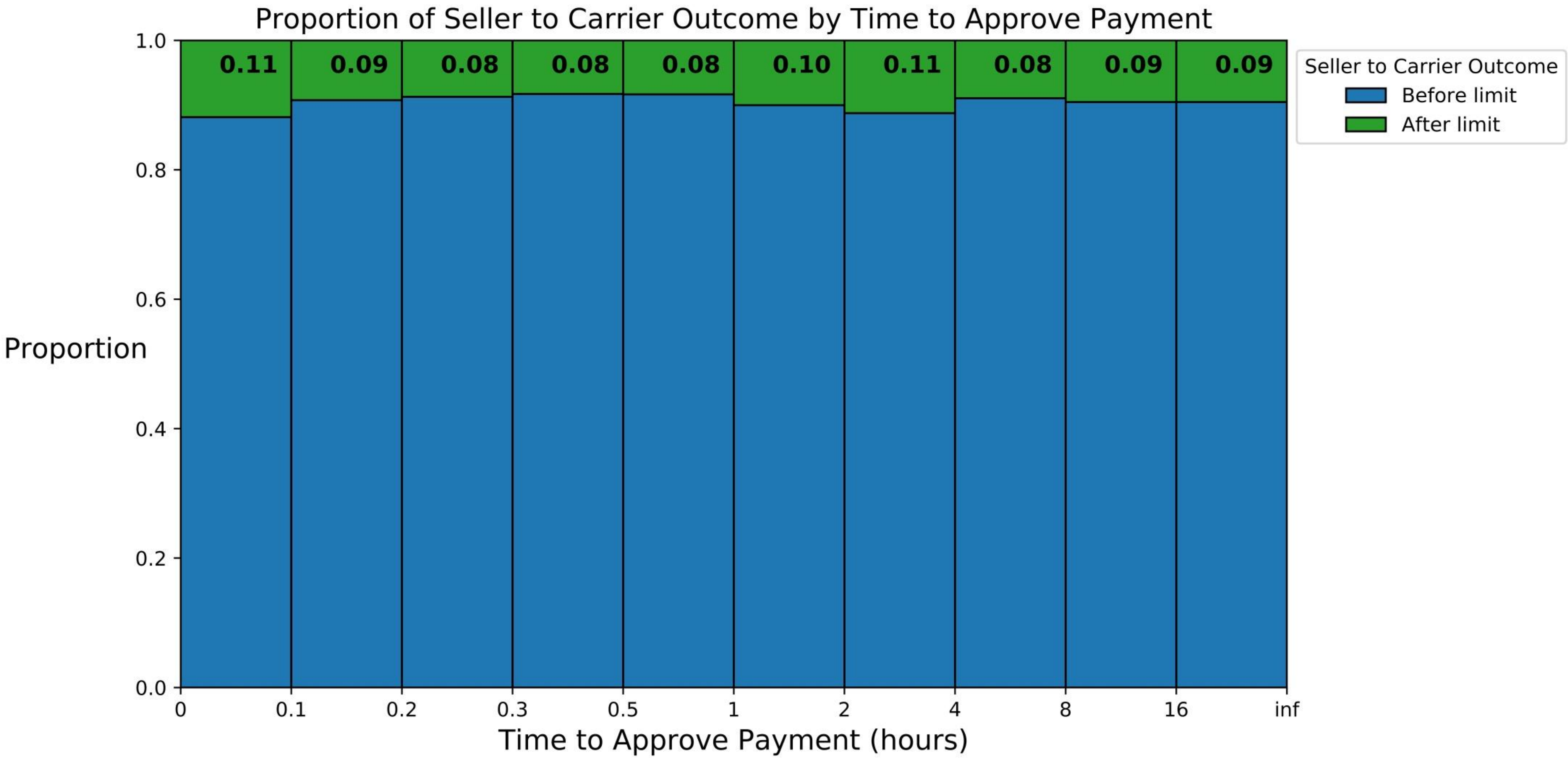
A3: Incorrect Values for: Late Delivery Investigation

- Created Python function to automate the task of updating each record:

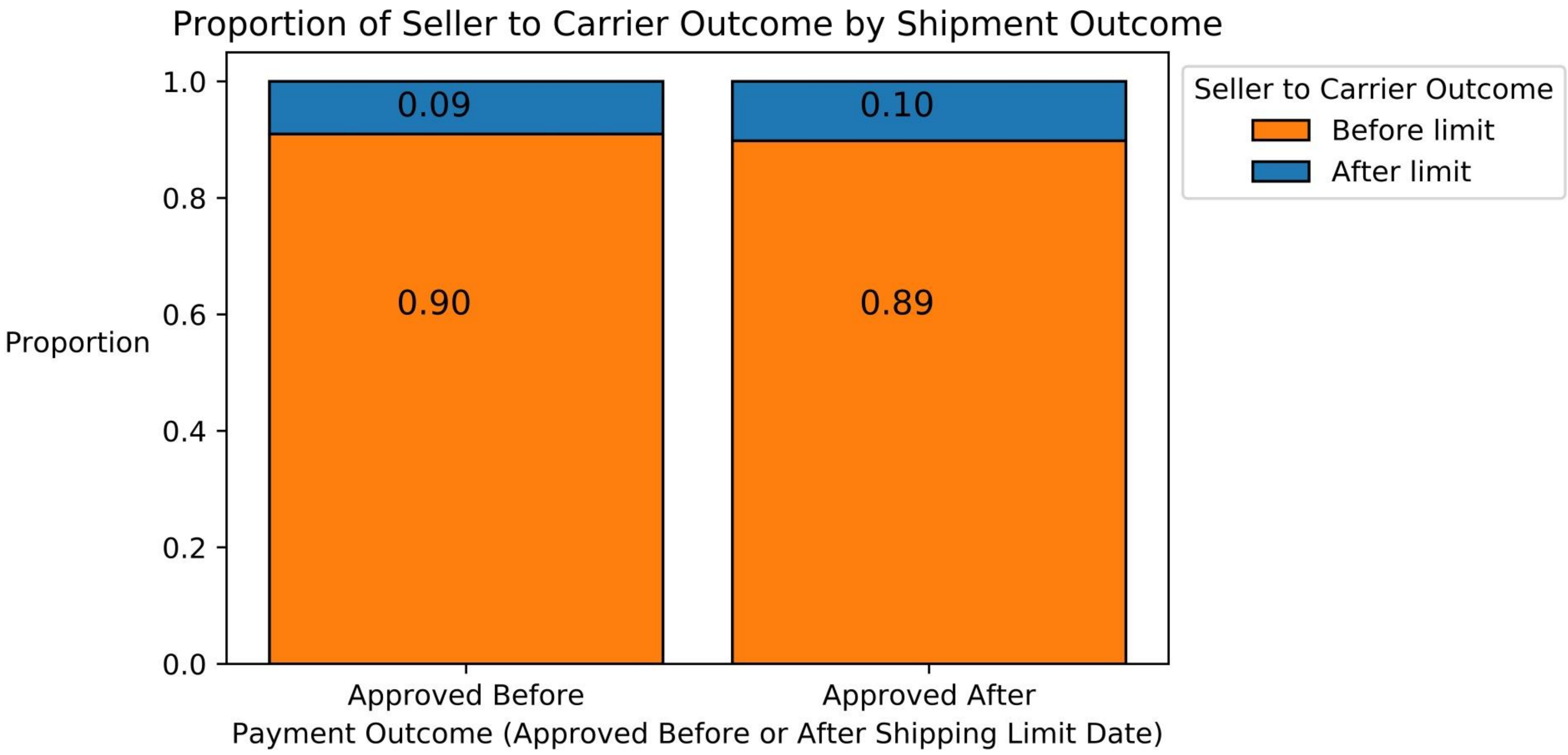


```
def update_database_automated(command, set_value_list, where_value_list):

    for index, value in enumerate(where_value_list):
        command_old = command
        command = command.format(set_value_list[index], where_value_list[index])
        run_command(command)
        print(command)
        command = command_old

df_ttc_neg = df_pa[df_pa['time_to_carrier']<0]
order_id_list = list(df_ttc_neg['order_id'])
order_purchase_timestamp_list = list(df_ttc_neg['order_purchase_timestamp'])


c3 = '''
UPDATE orders_modified
SET order_delivered_carrier = '{}'
WHERE order_id = '{}'
'''

update_database_automated(c3, order_purchase_timestamp_list, order_id_list)
```

A3: Payment Approval System

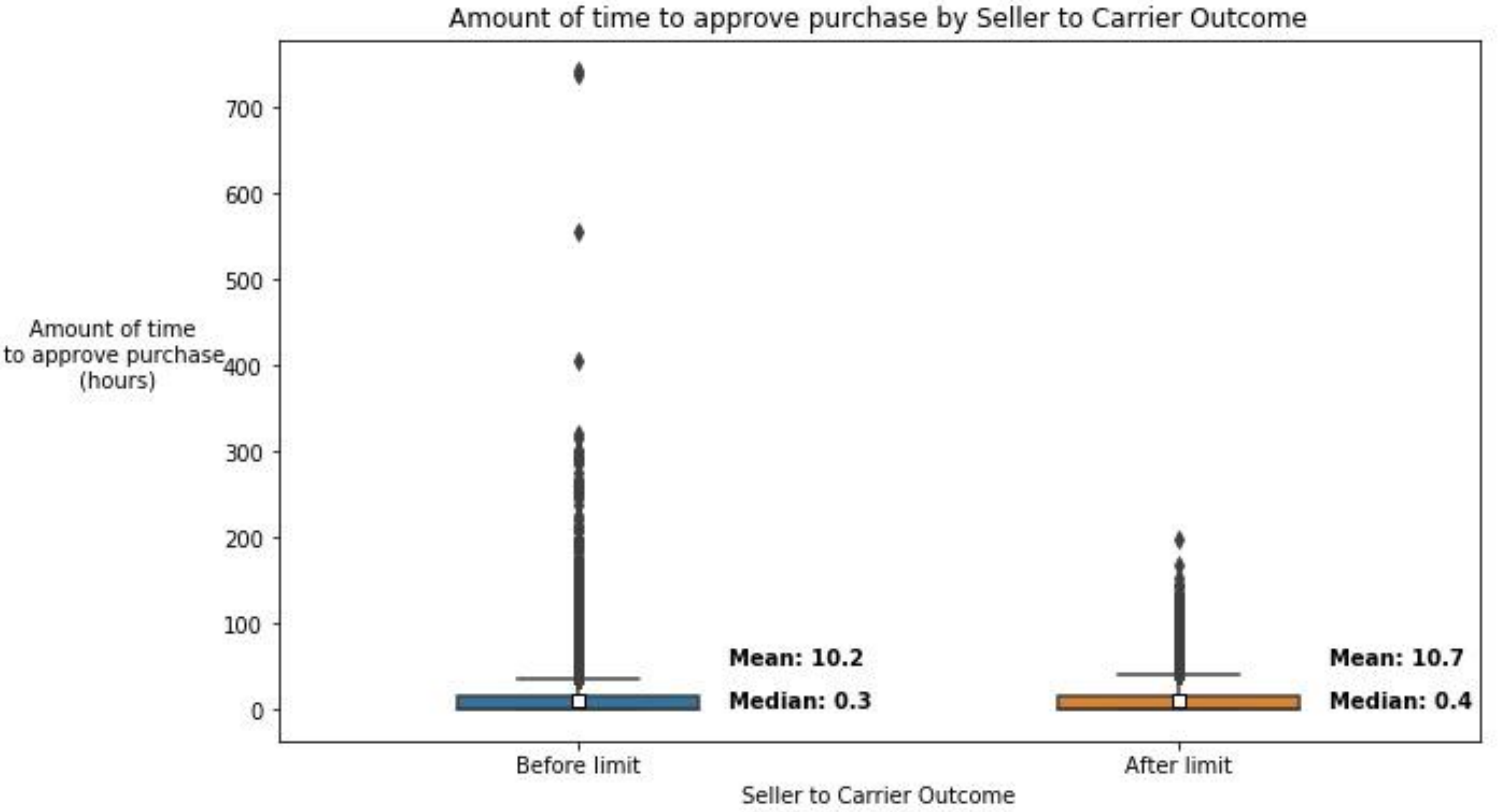


A3: Payment Approval System

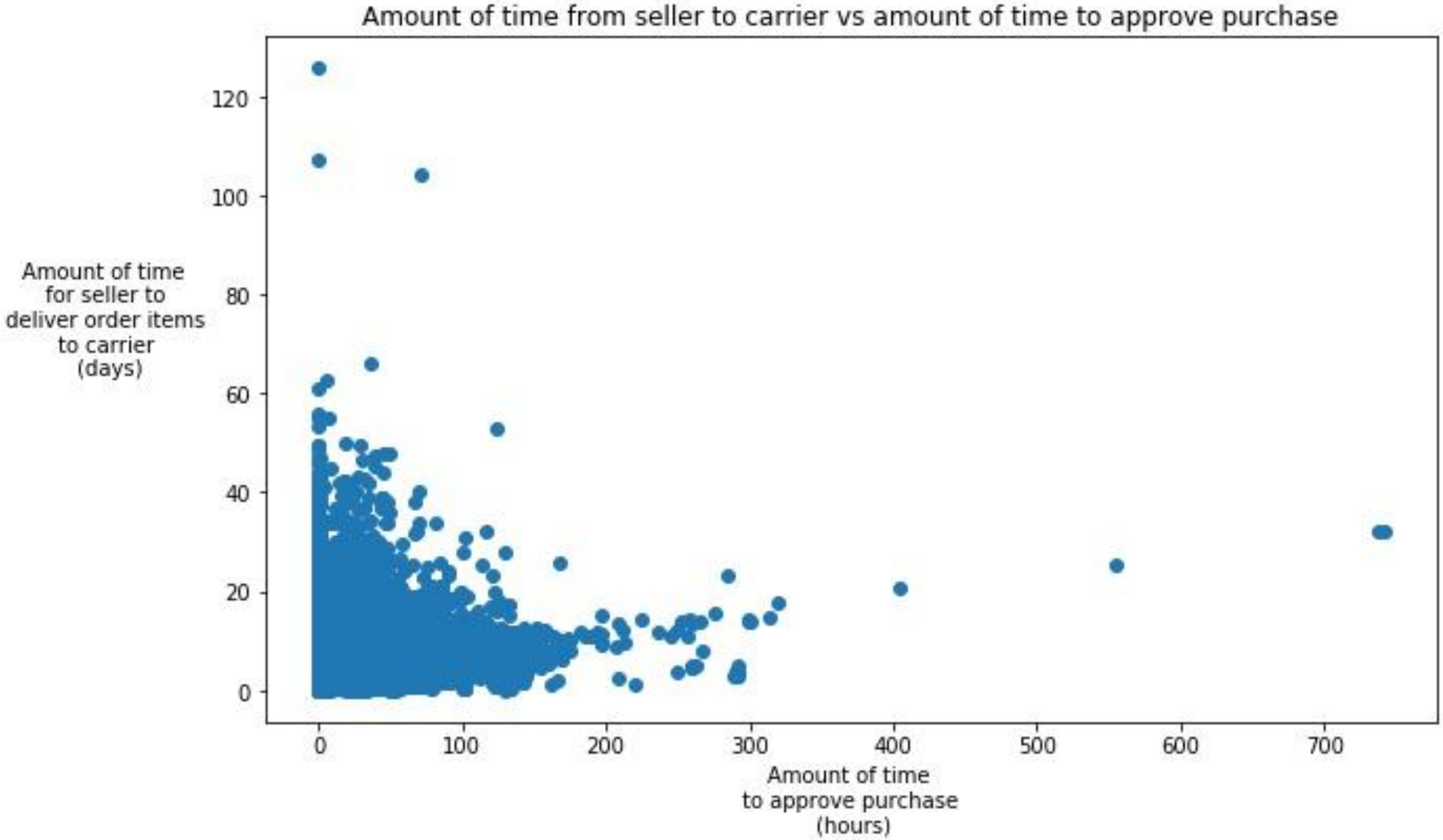


- NOT a reliable visual (‘Approved After’ bar is for 108 shipments vs 93,173 shipments in the ‘Approved Before’ bar.

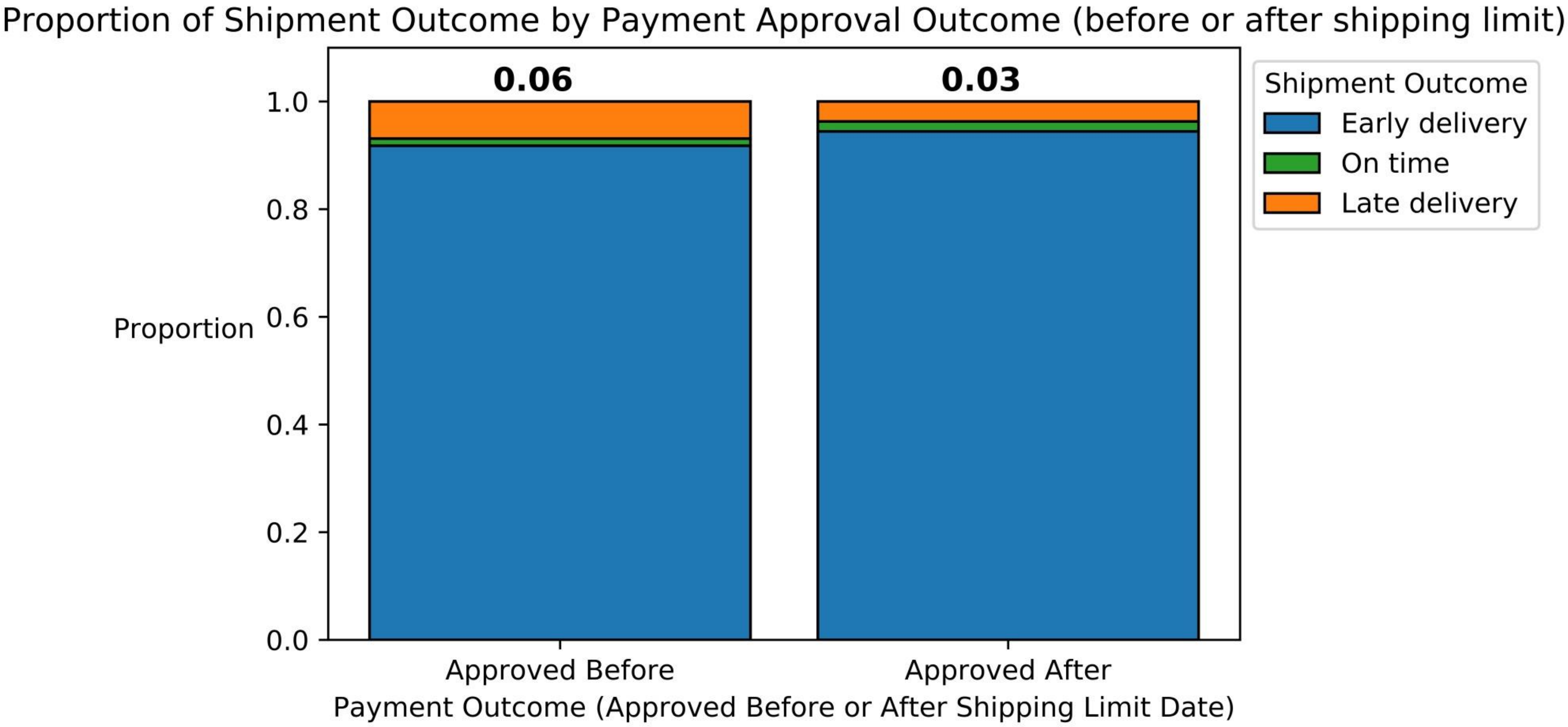
A3: Payment Approval System



A3: Payment Approval System

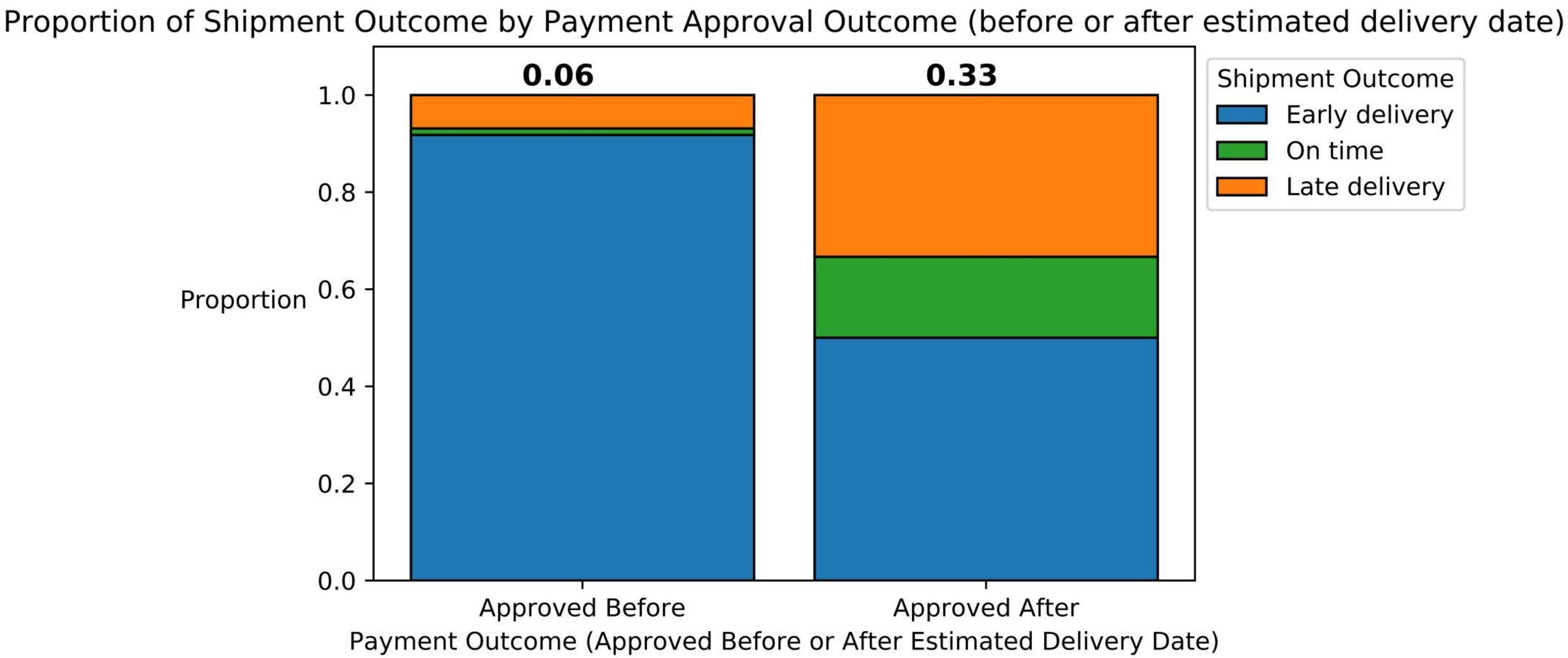


A3: Payment Approval System



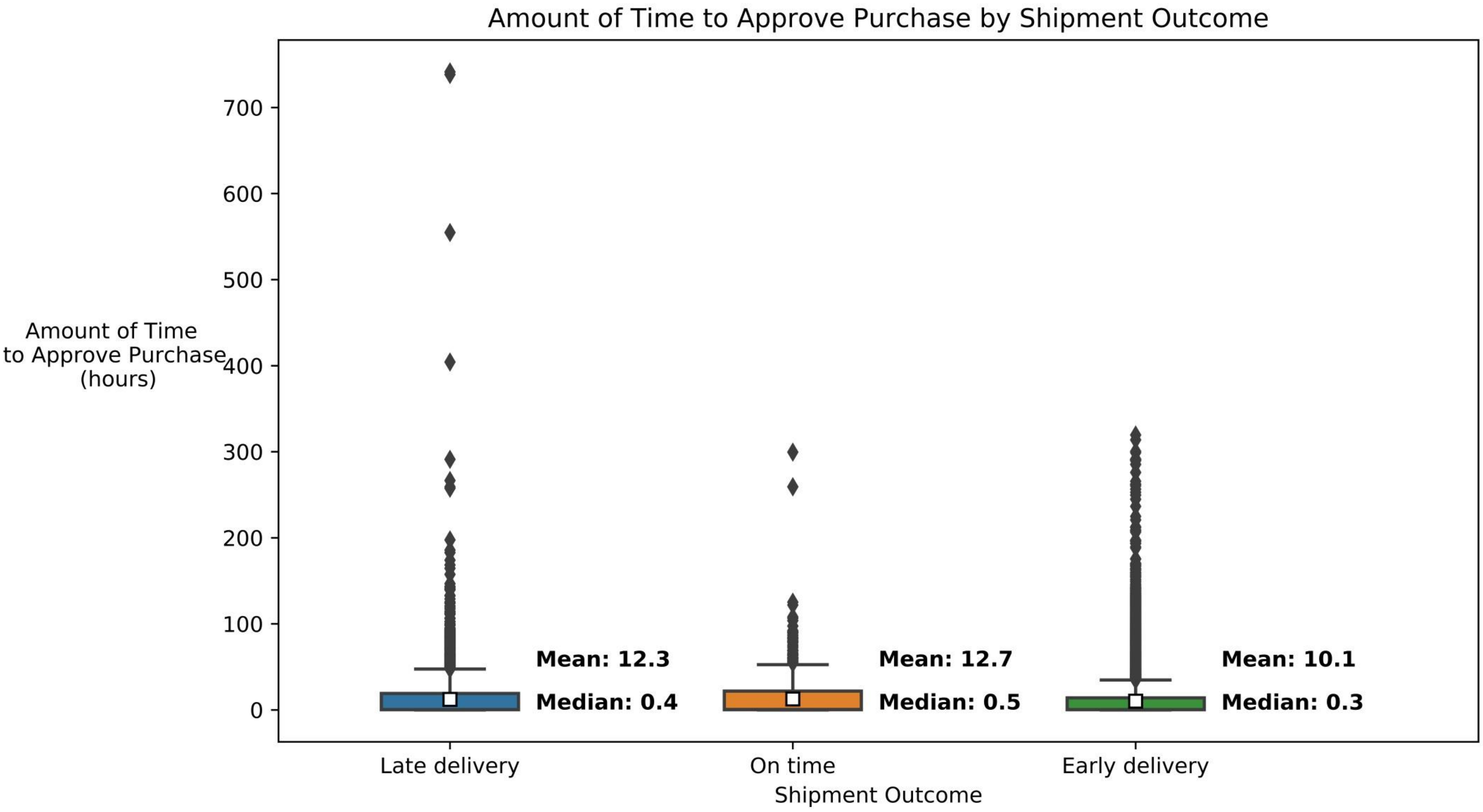
- NOT a reliable visual (‘Approved After’ bar is for 108 shipments vs 93,173 shipments in the ‘Approved Before’ bar.

A3: Payment Approval System

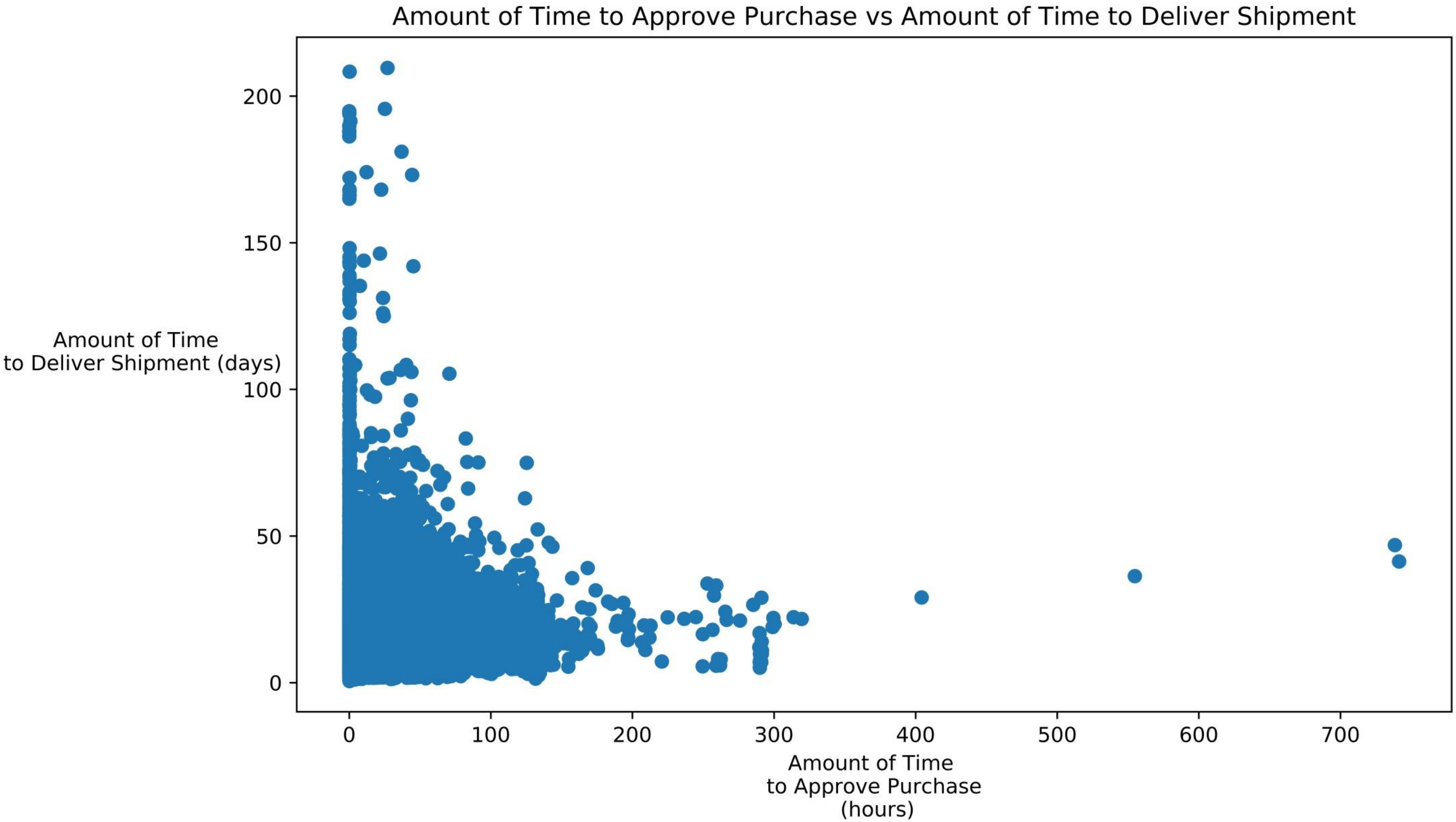


- NOT a reliable visual (‘Approved After’ bar is for 5 shipments vs 93,267 shipments in the ‘Approved Before’ bar. ‘Approved After’ bar extremely susceptible to being skewed by outliers.

A3: Payment Approval System



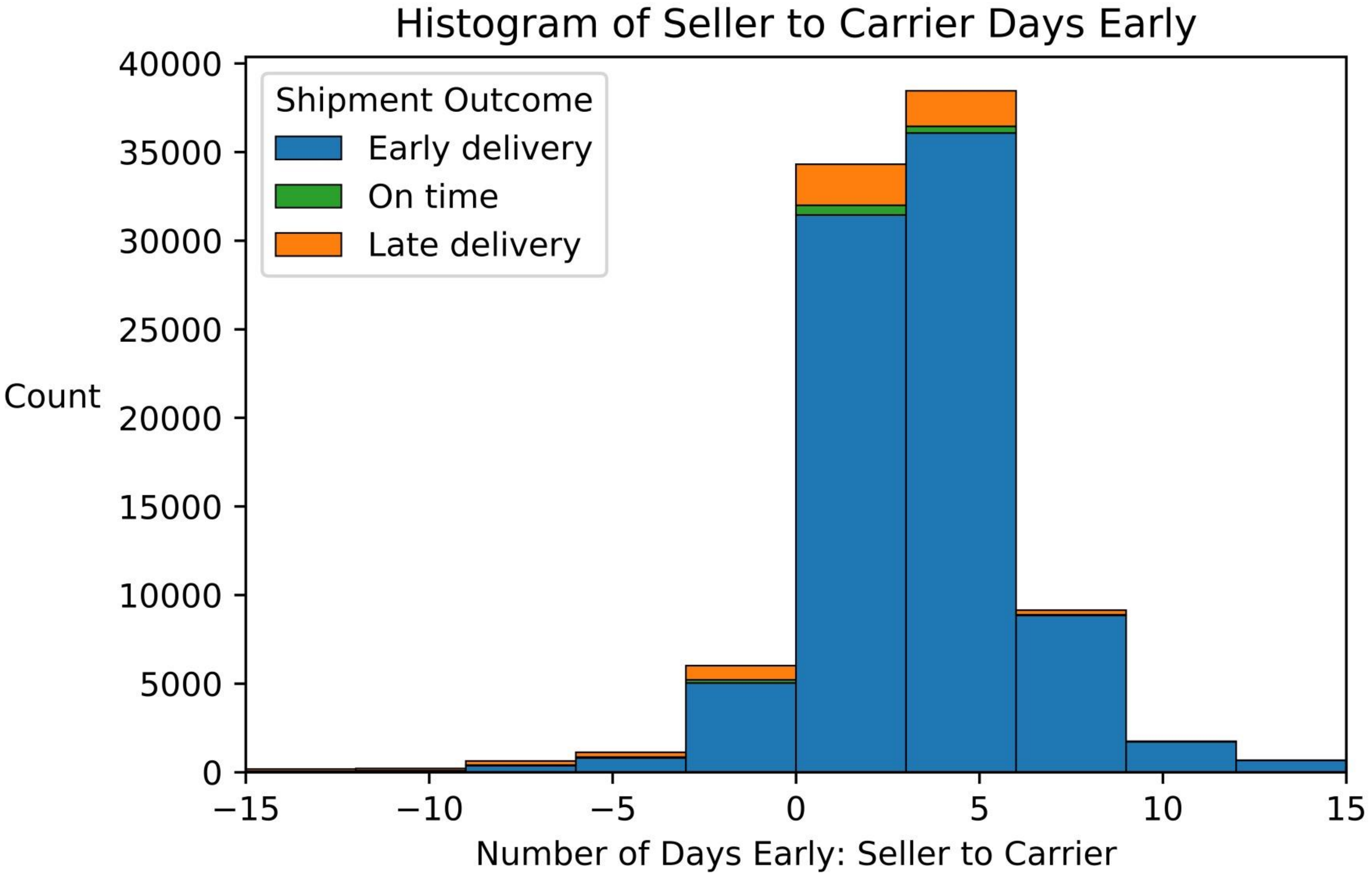
A3: Payment Approval System



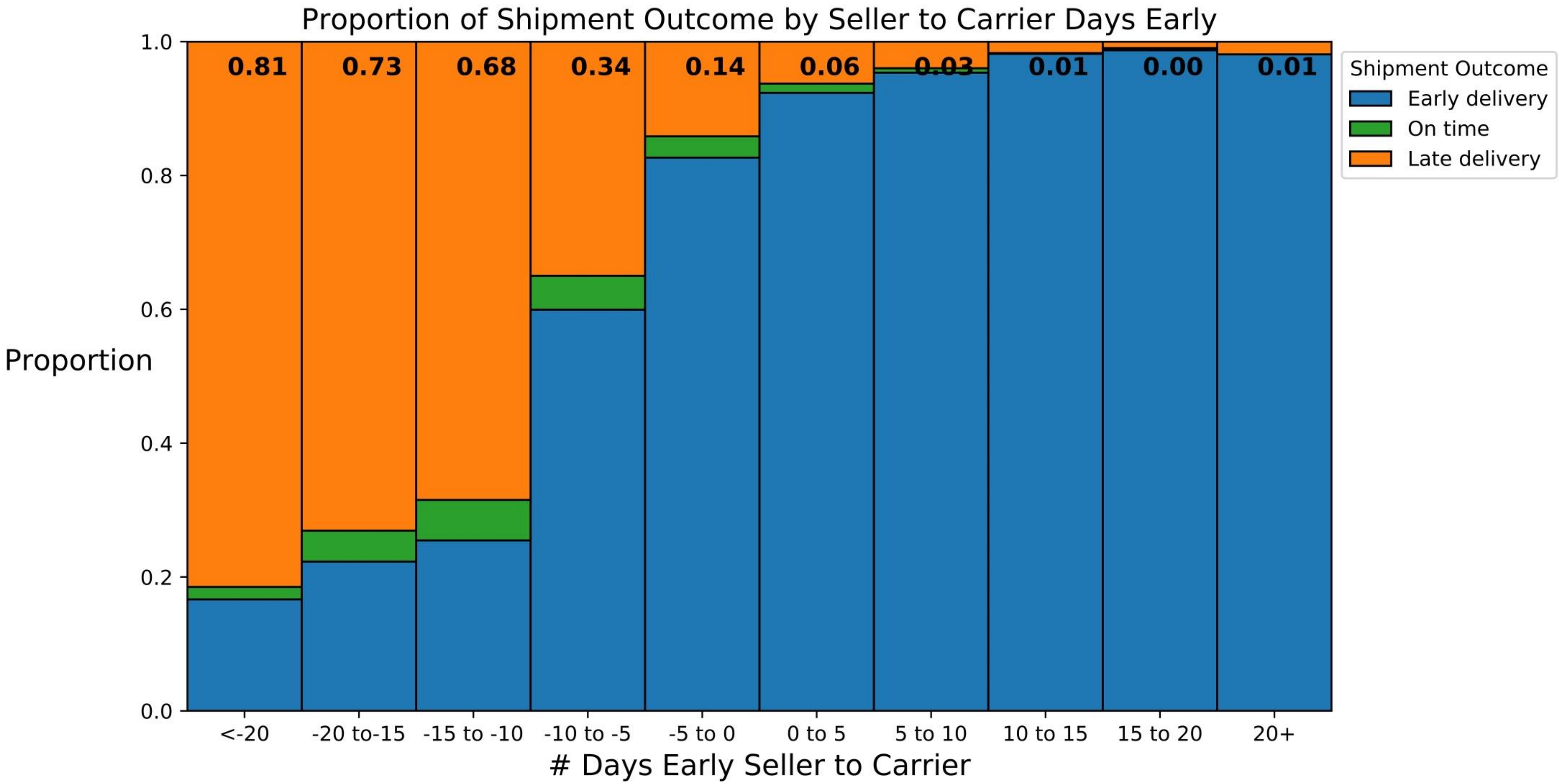
A3: Payment Approval System

- Only 11 orders where payment was approved after the shipping limit date and the seller delivered order to carrier after shipping limit date.
- 4 shipments where 315+ hours to approve the payment (all were late).*

A3: Seller


Histogram of Seller to Carrier Days Early
Shipment Outcome
Early delivery
On time
Late delivery
Count
40000
35000
30000
25000
20000
15000
10000
5000
0
−15
−10
−5
0
5
10
15
Number of Days Early: Seller to Carrier

A3: Seller



Proportion of Shipment Outcome by Seller to Carrier Days Early
Proportion
1.0
0.8
0.6
0.4
0.2
0.0
0.81
0.73
0.68
0.34
0.14
0.06
0.03
0.01
0.00
0.01
<-20
-20 to-15
-15 to -10
-10 to -5
-5 to 0
0 to 5
5 to 10
10 to 15
15 to 20
20+
# Days Early Seller to Carrier
Shipment Outcome
Early delivery
On time
Late delivery

A4: Missing Values for: Geographic Analysis?

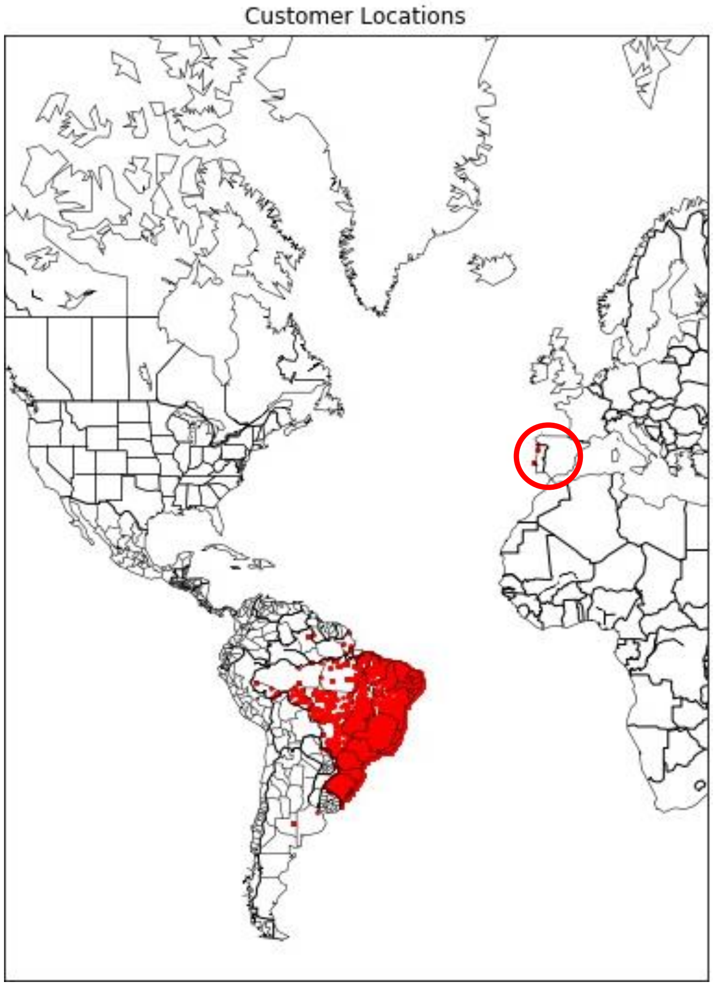
- Joining:
  - customers table
  - geolocation table
  - orders_modified table
  - order_items_modified table
  - sellers table
  - geolocation table (again)
- Such that row for each order/shipment (95,430 orders)

A4: Missing Values for: Geographic Analysis?

- Analyzing shipments that were delivered (93,281)
- Drop 265 missing customer latitudes/longitudes and 212 missing seller latitudes/longitudes. Ignoring other missing values.
- 92,818 shipments/orders

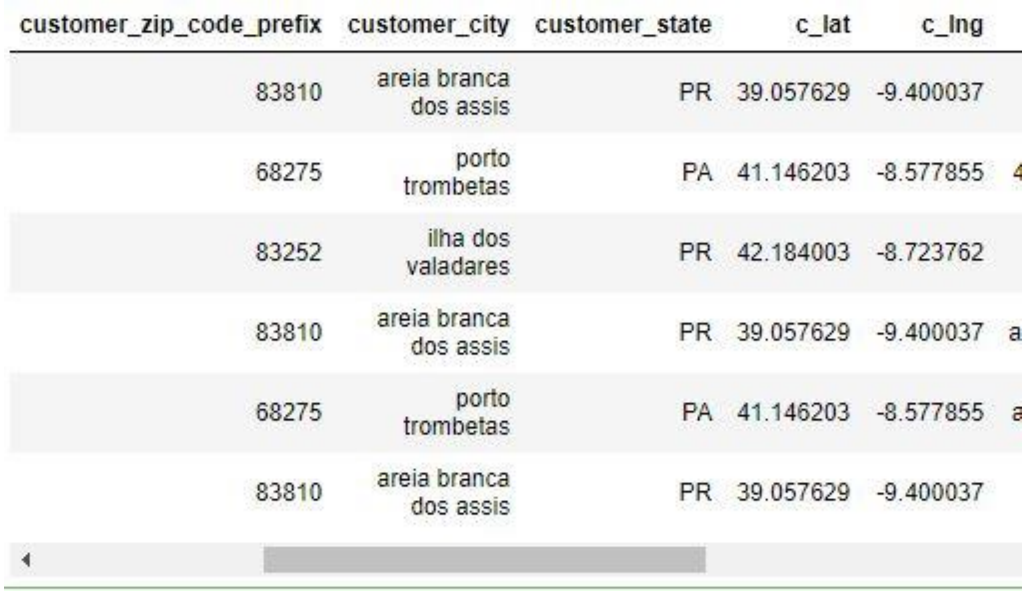
A4: Incorrect Values for: Geographic Analysis

- 5 shipments listed as being sent to longitudes/latitudes in Portugal

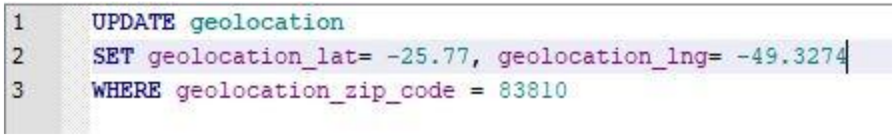


A4: Incorrect Values for: Geographic Analysis

- Olist incorrectly converted these zip codes
- Updated database w/ correct values:



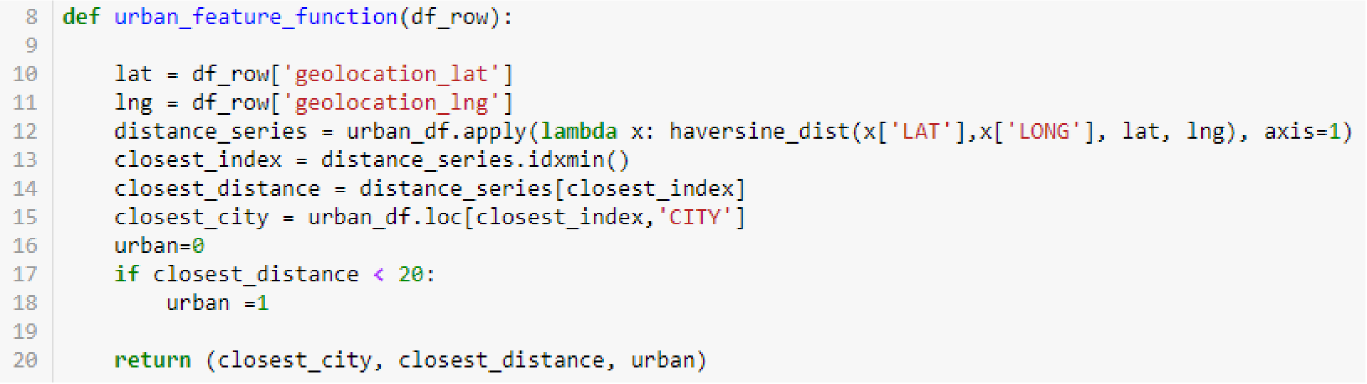
customer_zip_code_prefix	customer_city	customer_state	c_lat	c_lng	
83810	areia branca dos assis	PR	39.057629	-9.400037	
68275	porto trombetas	PA	41.146203	-8.577855	4
83252	ilha dos valadares	PR	42.184003	-8.723762	
83810	areia branca dos assis	PR	39.057629	-9.400037	a
68275	porto trombetas	PA	41.146203	-8.577855	a
83810	areia branca dos assis	PR	39.057629	-9.400037	



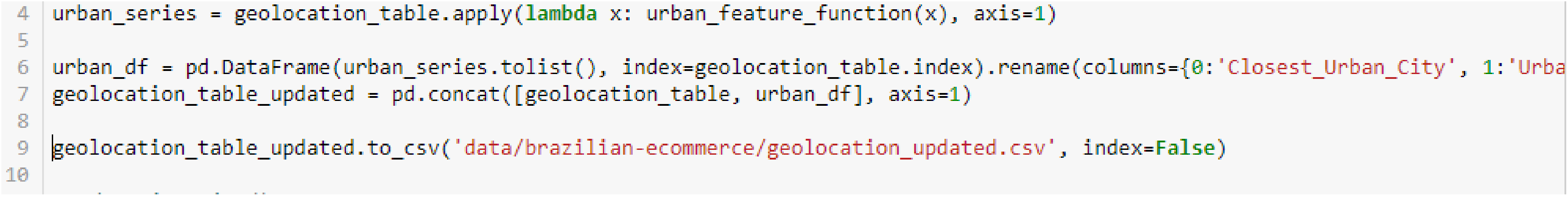
```
UPDATE geolocation
SET geolocation_lat= -25.77, geolocation_lng= -49.3274
WHERE geolocation_zip_code = 83810
```

A4: Urban features

- Add new urban features



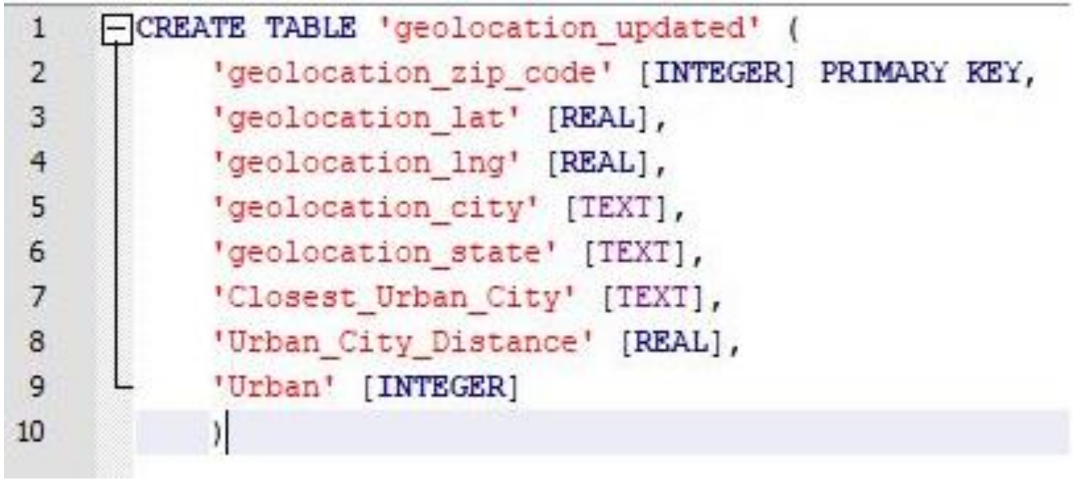
```python
def urban_feature_function(df_row):

    lat = df_row['geolocation_lat']
    lng = df_row['geolocation_lng']
    distance_series = urban_df.apply(lambda x: haversine_dist(x['LAT'],x['LONG'], lat, lng), axis=1)
    closest_index = distance_series.idxmin()
    closest_distance = distance_series[closest_index]
    closest_city = urban_df.loc[closest_index,'CITY']
    urban=0
    if closest_distance < 20:
        urban =1

    return (closest_city, closest_distance, urban)
```



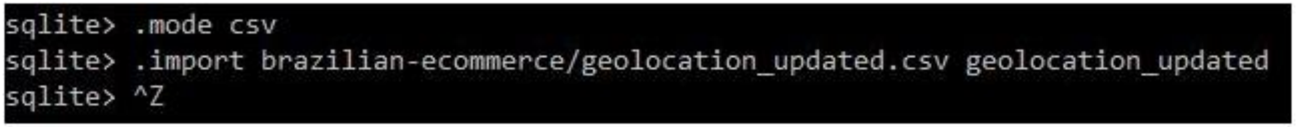
```python
urban_series = geolocation_table.apply(lambda x: urban_feature_function(x), axis=1)

urban_df = pd.DataFrame(urban_series.tolist(), index=geolocation_table.index).rename(columns={0:'Closest_Urban_City', 1:'Urba
geolocation_table_updated = pd.concat([geolocation_table, urban_df], axis=1)

geolocation_table_updated.to_csv('data/brazilian-ecommerce/geolocation_updated.csv', index=False)
```

A4: Urban features

- Create modified geolocation table*



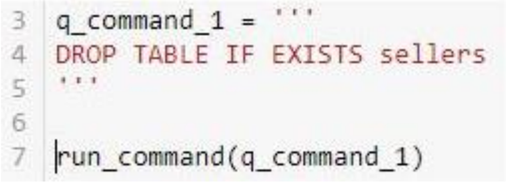
```
CREATE TABLE 'geolocation_updated' (
    'geolocation_zip_code' [INTEGER] PRIMARY KEY,
    'geolocation_lat' [REAL],
    'geolocation_lng' [REAL],
    'geolocation_city' [TEXT],
    'geolocation_state' [TEXT],
    'Closest_Urban_City' [TEXT],
    'Urban_City_Distance' [REAL],
    'Urban' [INTEGER]
    )
```



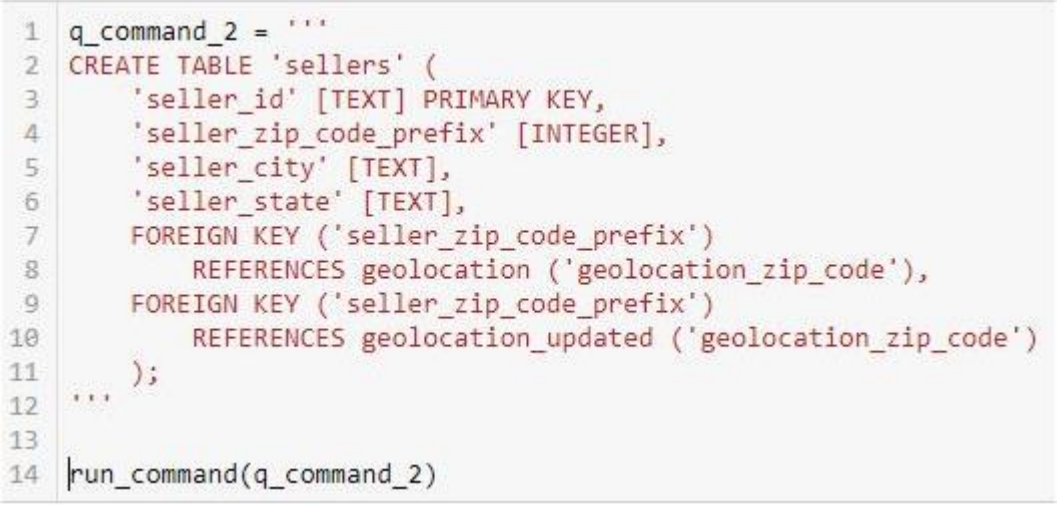
```
sqlite> .mode csv
sqlite> .import brazilian-ecommerce/geolocation_updated.csv geolocation_updated
sqlite> ^Z
```

A4: Urban features

- Add new foreign keys to customer and seller tables



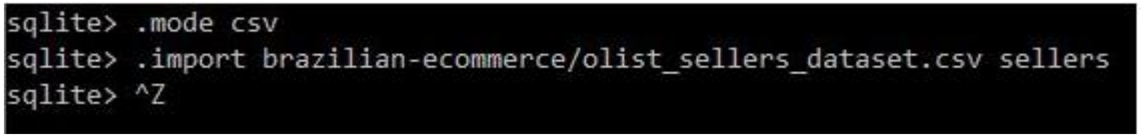
```
q_command_1 = '''
DROP TABLE IF EXISTS sellers
'''

run_command(q_command_1)
```



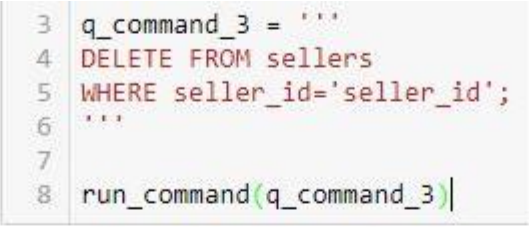
```
q_command_2 = '''
CREATE TABLE 'sellers' (
    'seller_id' [TEXT] PRIMARY KEY,
    'seller_zip_code_prefix' [INTEGER],
    'seller_city' [TEXT],
    'seller_state' [TEXT],
    FOREIGN KEY ('seller_zip_code_prefix')
        REFERENCES geolocation ('geolocation_zip_code'),
    FOREIGN KEY ('seller_zip_code_prefix')
        REFERENCES geolocation_updated ('geolocation_zip_code')
    );
'''

run_command(q_command_2)
```

A4: Urban features

- Add new foreign keys to customer and seller tables (continued)



```
sqlite> .mode csv
sqlite> .import brazilian-ecommerce/olist_sellers_dataset.csv sellers
sqlite> ^Z
```



```
q_command_3 = '''
DELETE FROM sellers
WHERE seller_id='seller_id';
'''

run_command(q_command_3)
```

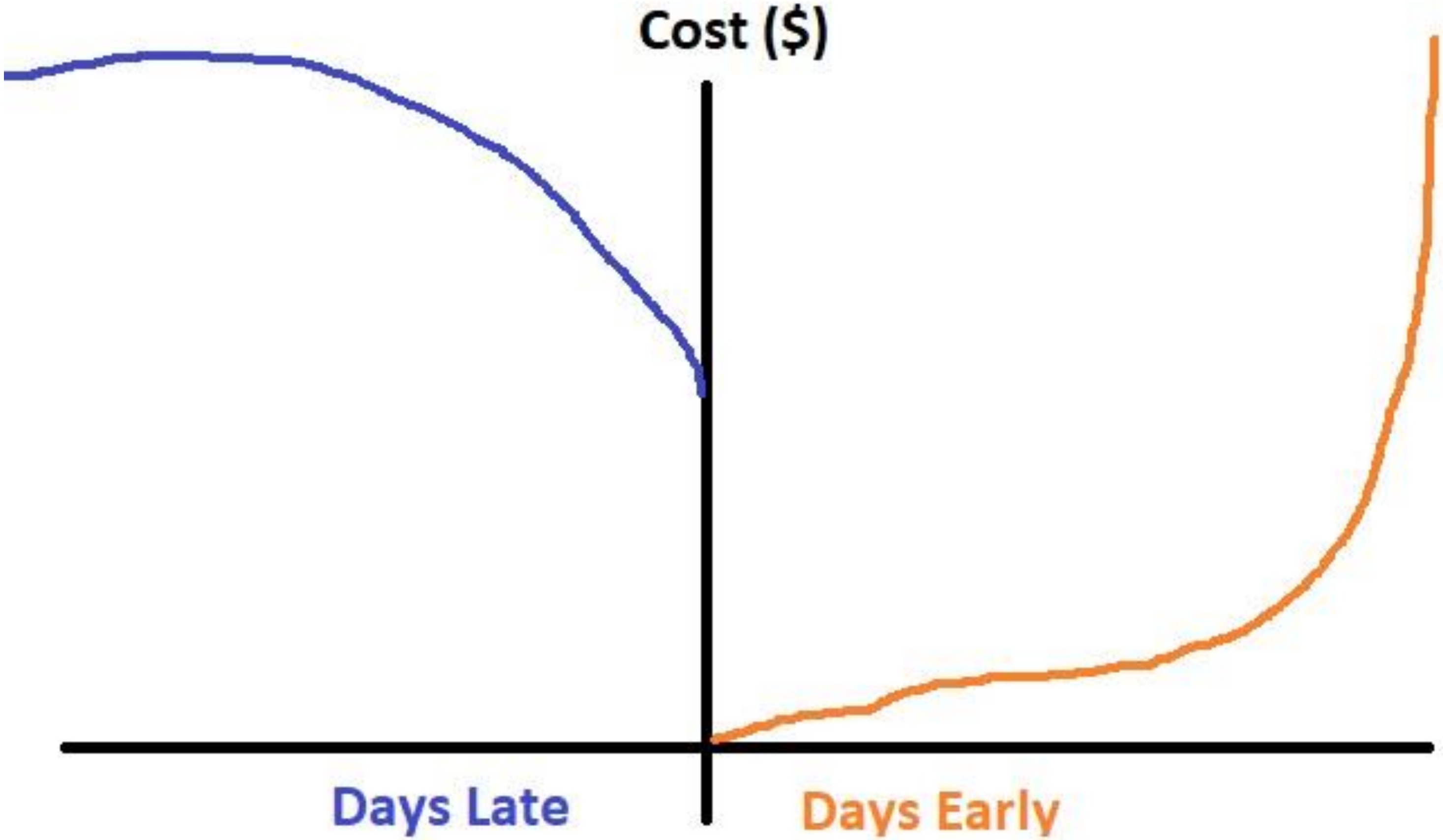
A5: Choosing the Evaluation Metric

- MSE, RMSE:
  - Bigger differences weighted more, more sensitive to outliers
  - Gradient is easy to interpret (show why)
  - Baseline prediction: Choose mean
- MAE:
  - Individual differences weighted equally, less sensitive to outliers
  - Gradient is complicated (show why)
  - Baseline prediction: Choose median
- R^2:
  - Bigger differences weighted more, more sensitive to outliers
  - Takes baseline model into account
  - Baseline prediction: Choose mean

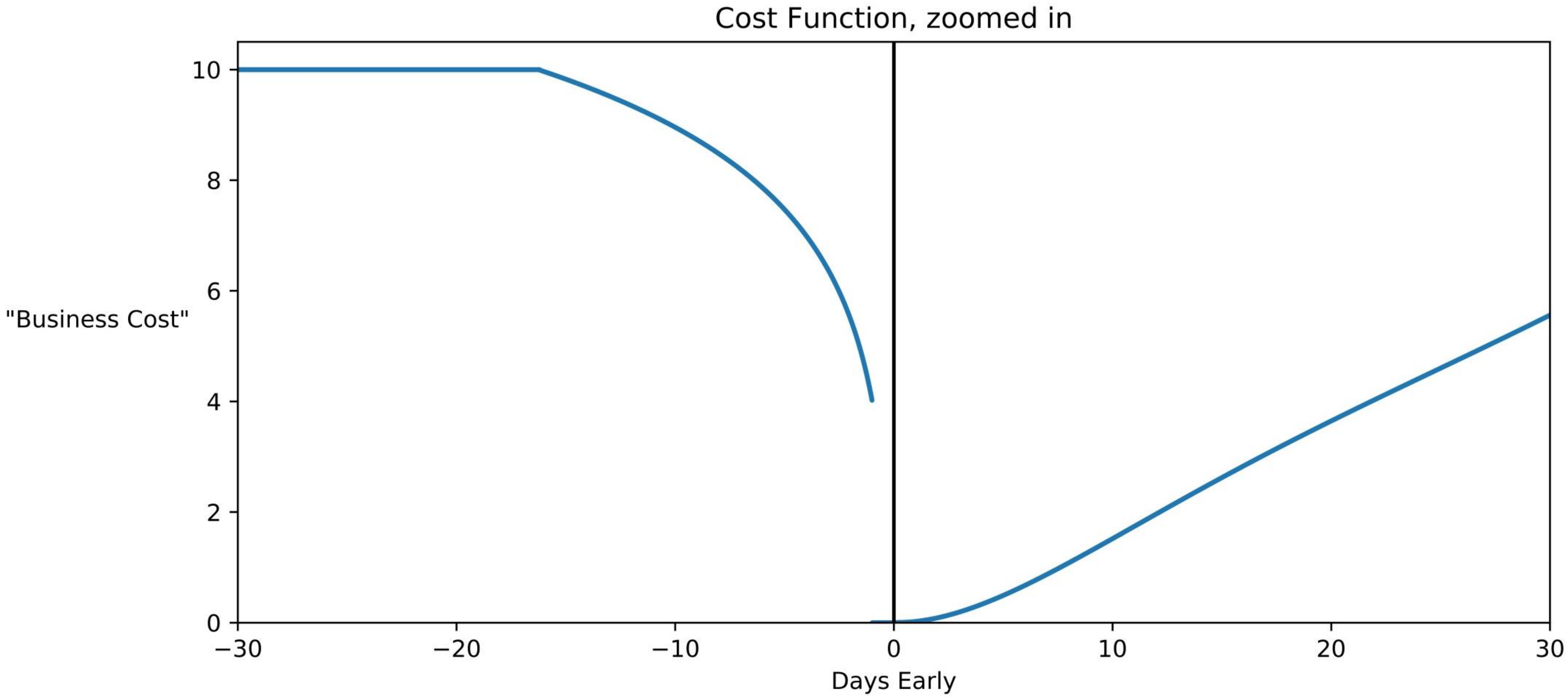
A5: Dummy Baseline Model

- Baseline prediction based on evaluation metric.
- RMSE -> mean of target variable
  - MSE, RMSE: Best prediction is mean value (include formulas to show why)
  - MAE: Best prediction is median value (include formulas to show why)

A5: Cost Function



A5: Cost Function, zoomed in



A5: Algorithm Choice

- No free lunch (trade offs to every algorithm)
- Tried many different algorithms while understanding the tradeoffs for each
- Have personally built many of the algorithms from scratch for better understanding

A5: Algorithm Considerations

- Some algorithms give “better” results depending on the goals and the data available
- Interpretability
- Computation Expense
- Tendencies to overfit/underfit