Jumpman23
[New Market Analysis]

Stephen Stark

October 2020

Presented to:

Postmates

ANALYSIS OVERVIEW

1 EXECUTIVE SUMMARY

The New York market is **STRONG with SIGNIFICANT GROWTH POTENTIAL**:

- Completed 4,700 transactions in the month of October 2014
- 45-minute average delivery time
- 2,900+ unique customers served; 16% order twice, 6% order three or more times
- Most popular delivery time is between 6PM-8PM

GROWTH RECOMMENDATION #1: Targeted campaigns to specific zip codes.

- [11226, 10025, 11211]: have the lowest market penetration rate, currently reaching 60 of the 290K potential customers
- [10282,10007,10069]: have the highest average household median income $200K+ and a low market penetration rate, currently reaching 110 of the 17K potential customers

GROWTH RECOMMENDATION #2: Focus on Jumpmen and Customer engagement. Survey the ones who use the service the most to to gather feedback on their hurdles and challenges. Also engage with the customers who only used once and haven't returned.

GROWTH RECOMMENDATION #3: Decrease wasted time at restaurants. Explore a restaurant partnership model to minimize the time spent waiting at the pickup location.

2 INTRODUCTION

Methodology:

1. Examine data integrity concerns
2. Exploratory data analysis
3. Feature engineering
4. Determine growth strategy

Assumptions:

- Place category, item name, item quantity, category name were all replaced with “not disclosed”
- Estimated time to order using an average value for the dataset as my analysis did not show meaningful deviation by place category
- Dropped records that did not have Jumpman pickup arrival and departure times
- Multiple items from the same order were broken out into separate records. I decided to only keep one record per delivery id
- Excluded records with suspicious values. One record suggested a bicyclist traveled 100+ mph

3 NEW MARKET ANALYSIS

KNOW THE BUSINESS

CUSTOMERS

+

JUMPMEN

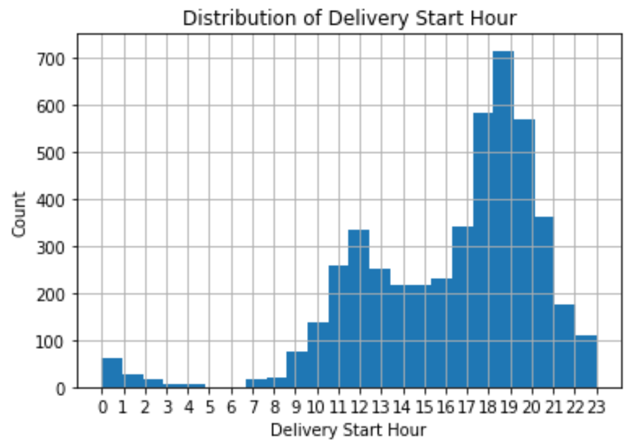
+

BUSINESSES

3.1 NEW MARKET ANALYSIS [CUSTOMERS]

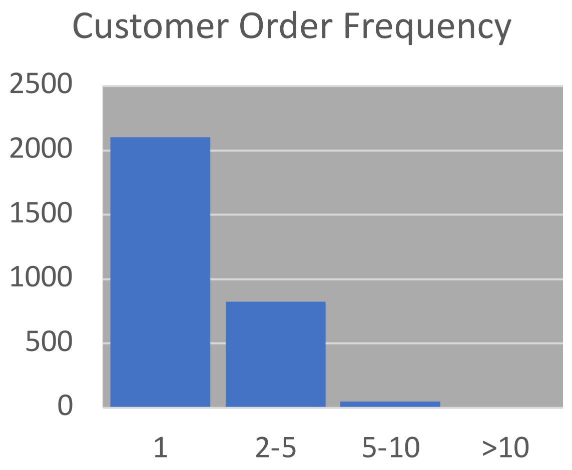
CUSTOMER PROFILE

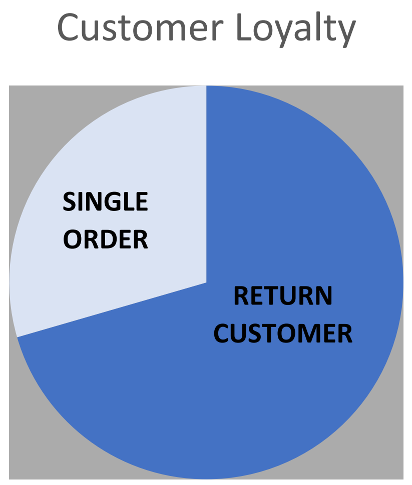
- Dinner is most popular (6PM – 8PM) followed by lunch
- Five most frequented drop off zip codes are [10001, 10002, 10003, 10011, 10012]
- Average median household income of $80K, with average population of 46K per zip code
- Most popular categories: Italian, American, burgers, Japanese, and dessert



CUSTOMER LOYALTY

- Most loyal customer, customer id #369272, ordered 23 times!
- Average order frequency per customer is 1.6x
- 30% of customers ordered more than once



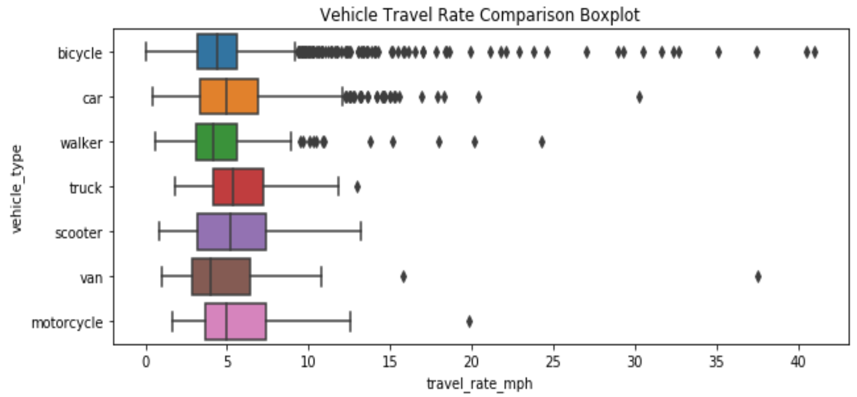


3.2 NEW MARKET ANALYSIS [JUMPMEN]

5,400+ total miles traveled*

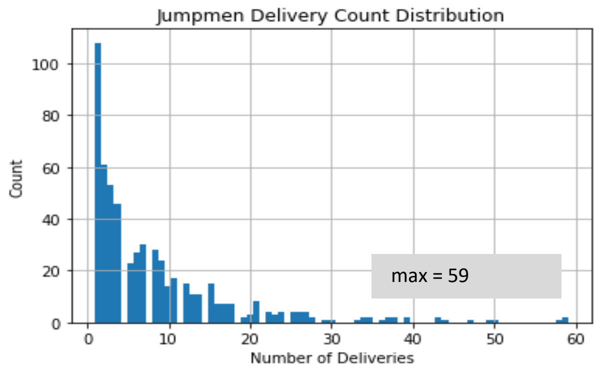
72% of deliveries via bicycle, 21% by car

*based on Haversine distance



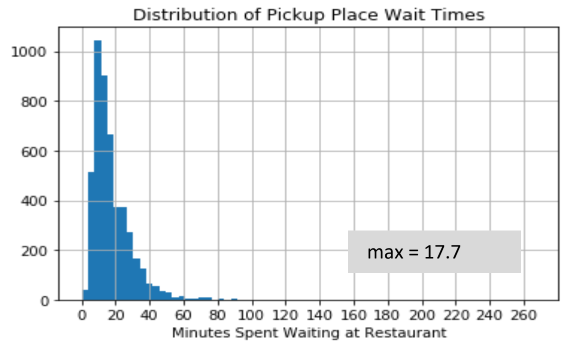
83,000 minutes spent waiting at businesses, or ~1,400 hours

Average wait time of 17 minutes per business



565 Jumpmen on the platform

The average Jumpman completed 8.3 deliveries, or one every four days



3.2 NEW MARKET ANALYSIS [JUMPMEN]

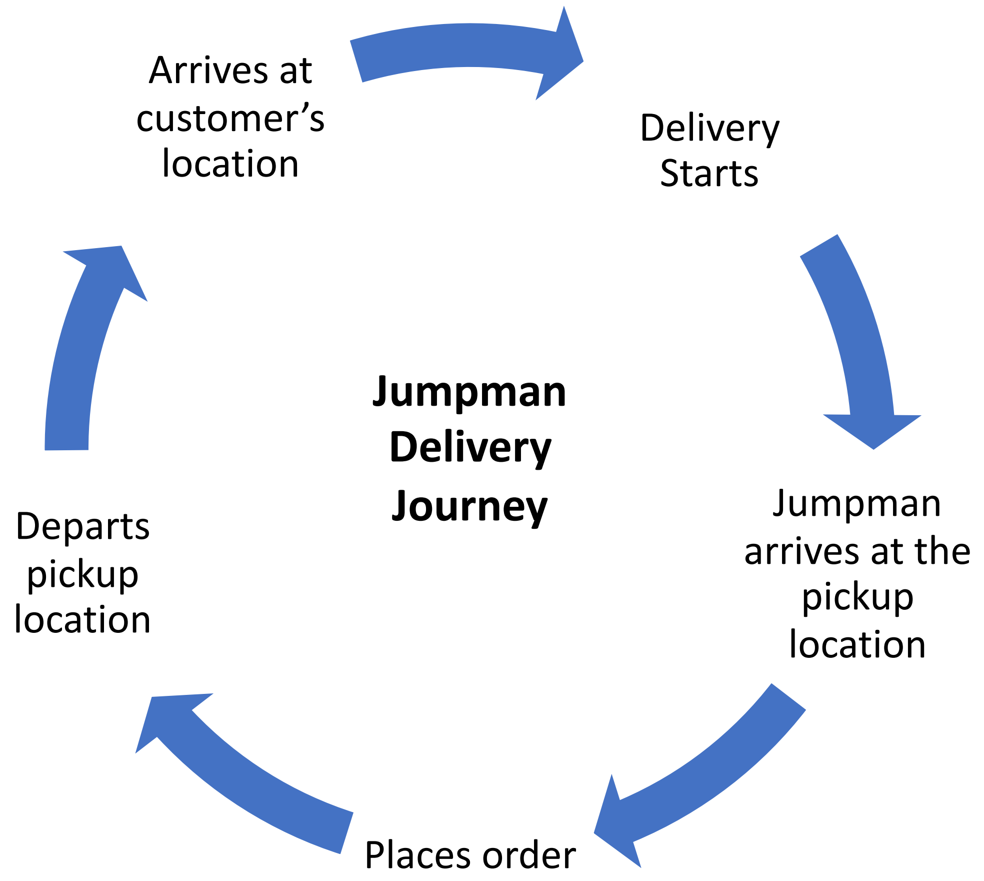
The Jumpman Journey

Jumpman23 is an on-demand delivery platform connecting "**Jumpmen**" and customers.

Jumpmen are sent to merchants to pickup any items requested by the customer.

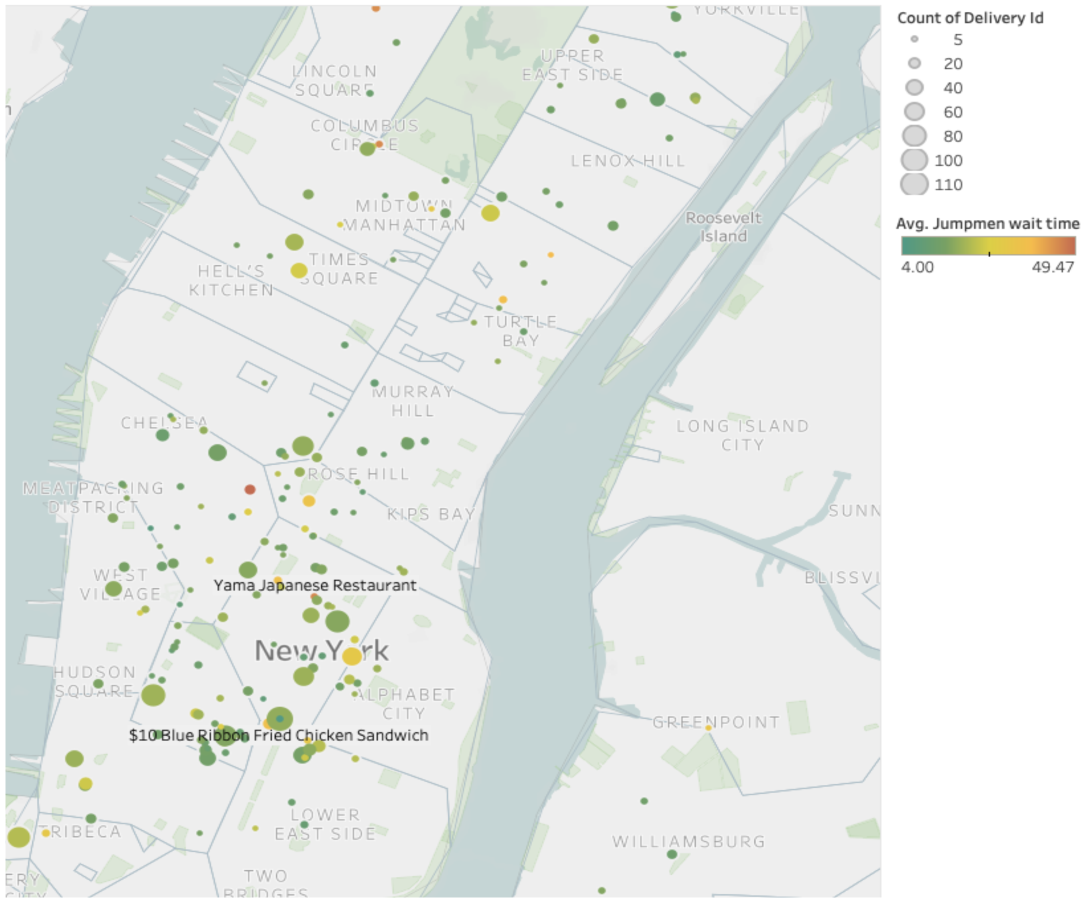
Whenever possible, Jumpman23 will order the requested items ahead to save the Jumpmen time.

Each time a Jumpman23 delivery is completed, a record is saved to the Jumpman23.

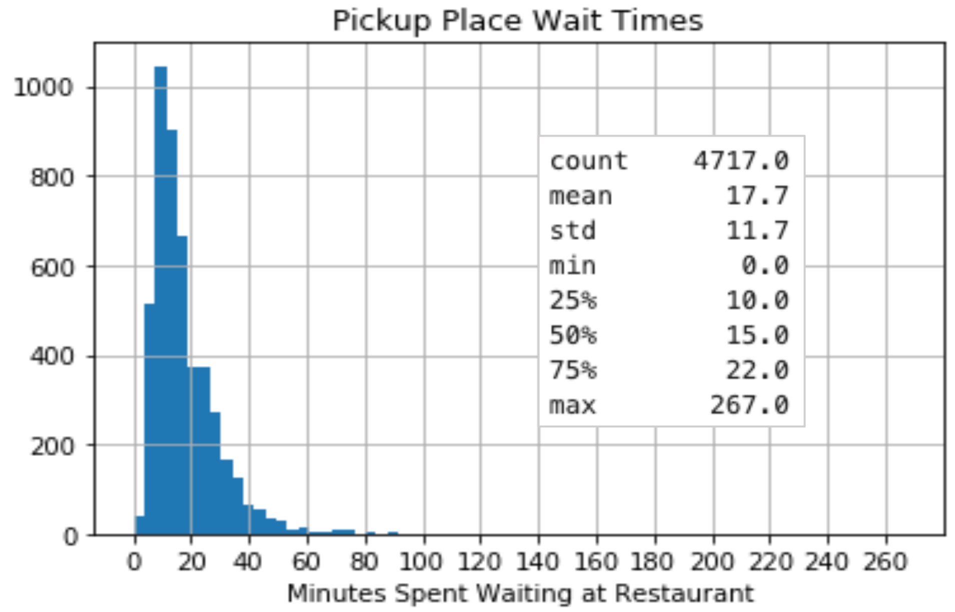


3.3 NEW MARKET ANALYSIS [BUSINESSES]

Some of the most popular restaurants can be found in the East Village, SoHo, and Lower East Side neighborhoods



Average Jumpmen wait time is almost 18 minutes

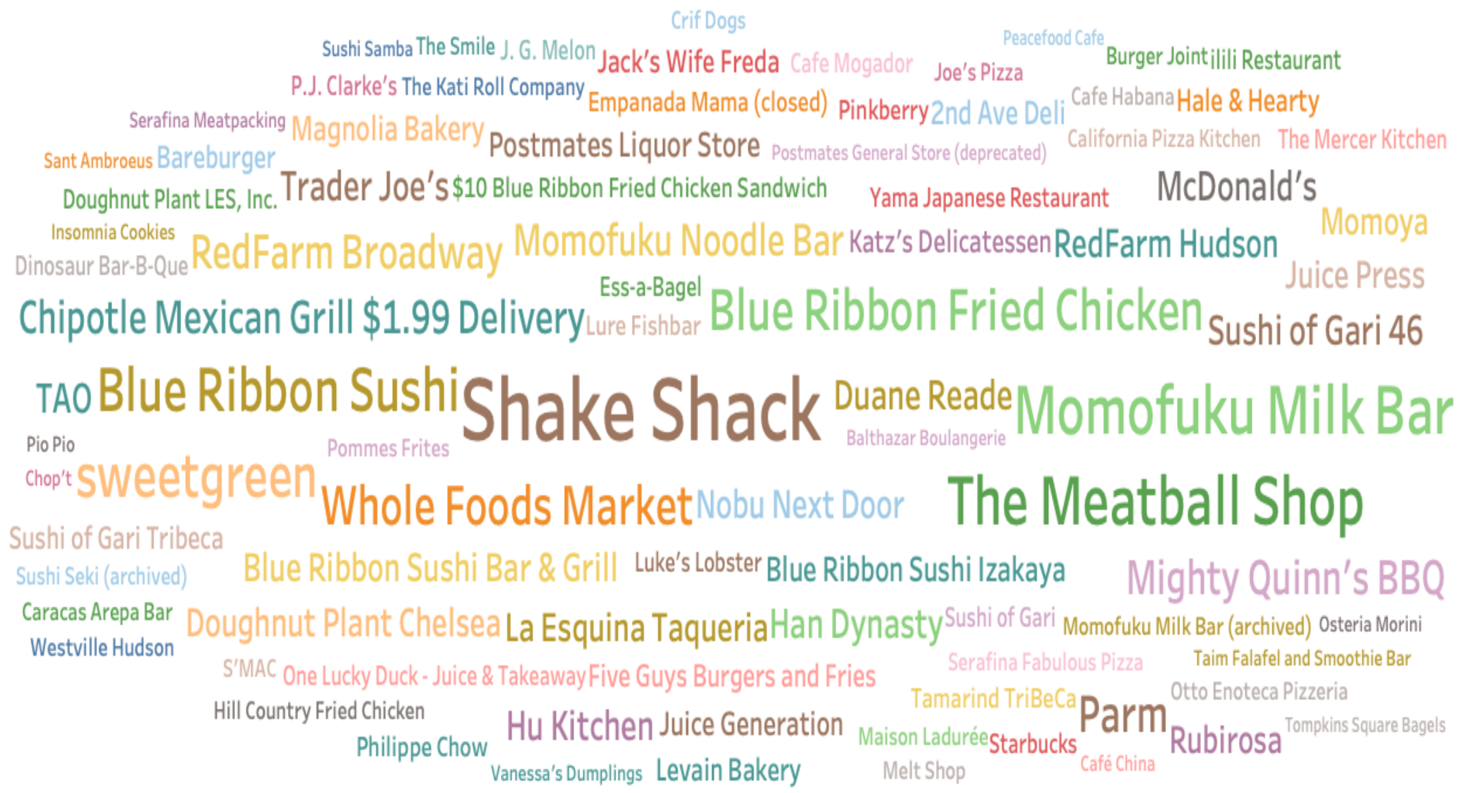


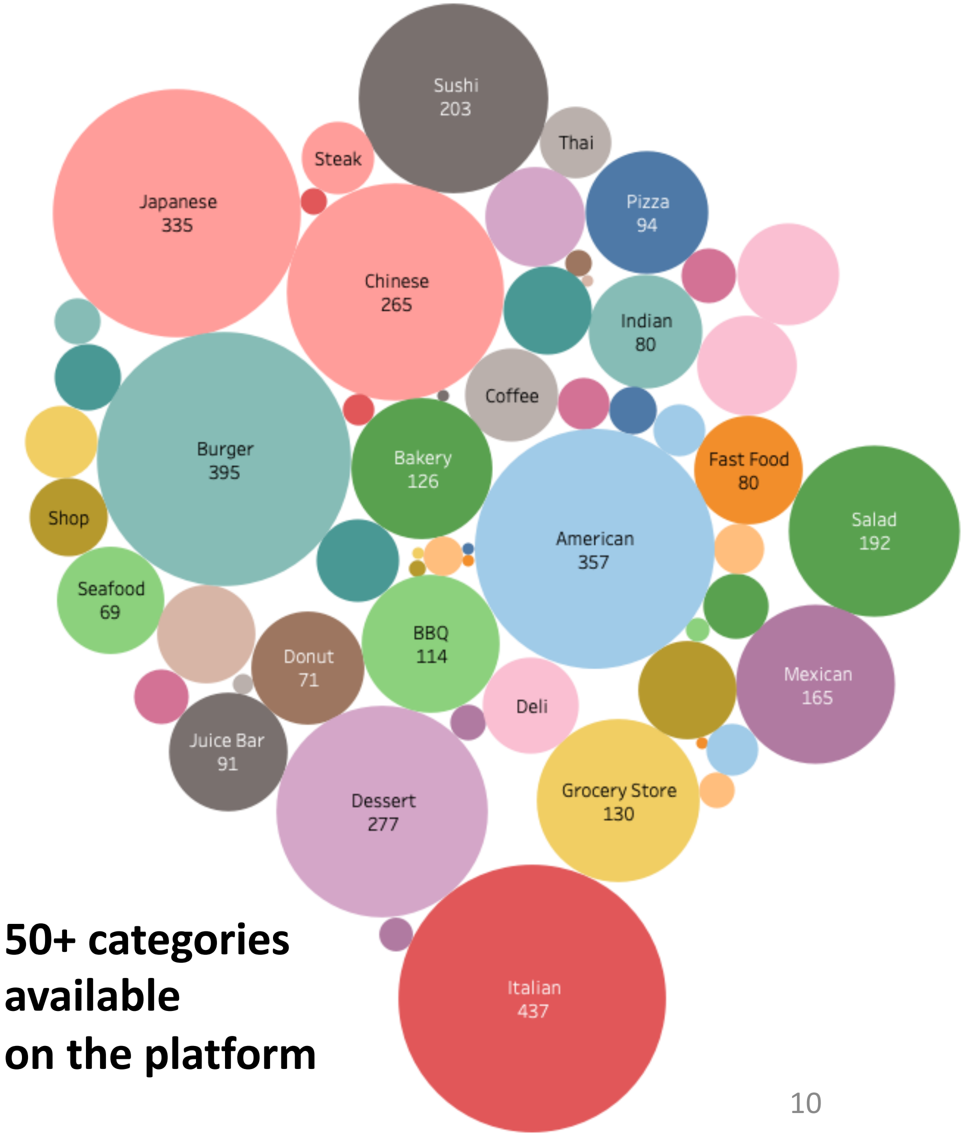
```
count    4717.0
mean       17.7
std        11.7
min         0.0
25%        10.0
50%        15.0
75%        22.0
max       267.0
```

The right skew indicates inefficiency. I recommend a partnership model to decrease time spend at restaurants

3.3 NEW MARKET ANALYSIS [BUSINESSES]

Most Frequented Pickup Spots

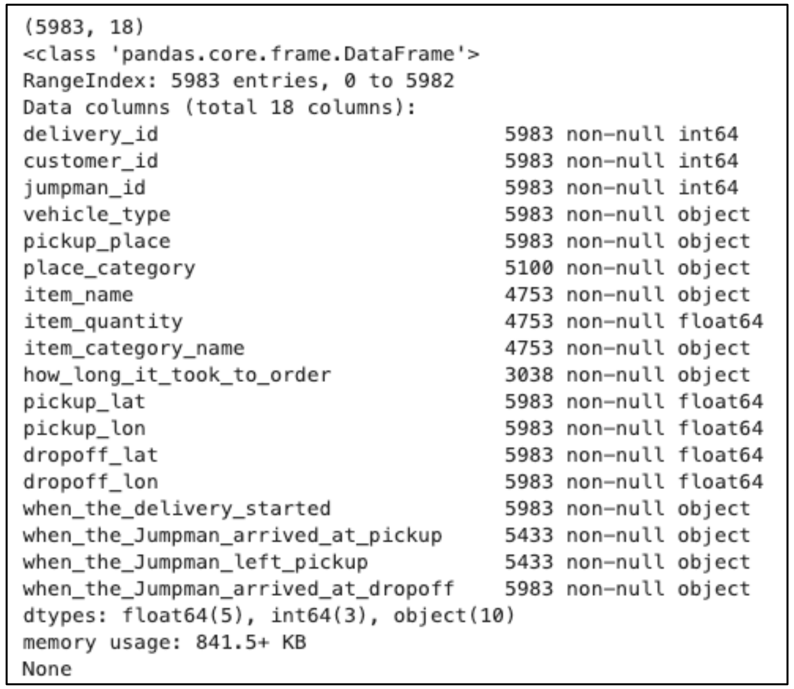




50+ categories available on the platform

4 DATA

Job_ID → a unique identifier of a delivery
Customer_id → a unique identifier for the Jumpman23 customer
Jumpman_id → a unique identifier for the Jumpman who completed the delivery
vehicle_type → The method of transport the Jumpman used to complete the delivery
pickup_place → The name of the Pickup location
place_category → A categorization of the Pickup location
Item_name → the name of the item requested
Item_quantity → how many of that item was requested
Item_category_name → categorization provided by the merchant, think "appetizers", "soups" etc
How_long_it_took_to_order → how long it took to place the order [interval]
pickup_lat → the coordinates of the pickup location
pickup_lon → the coordinates of the pickup location
dropoff_lat → the coordinates of the dropoff location
dropoff_lon → the coordinates of the dropoff location
when_the_delivery_started→ localized timestamp representing when the delivery began
when_the_Jumpman_arrived_at_pickup → localized timestamp representing when the Jumpman arrived at the pickup location
when_the_Jumpman_left_pickup → localized timestamp representing when the Jumpman left the pickup location



```
(5983, 18)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5983 entries, 0 to 5982
Data columns (total 18 columns):
delivery_id                          5983 non-null int64
customer_id                          5983 non-null int64
jumpman_id                           5983 non-null int64
vehicle_type                         5983 non-null object
pickup_place                         5983 non-null object
place_category                       5100 non-null object
item_name                            4753 non-null object
item_quantity                        4753 non-null float64
item_category_name                   4753 non-null object
how_long_it_took_to_order            3038 non-null object
pickup_lat                           5983 non-null float64
pickup_lon                           5983 non-null float64
dropoff_lat                          5983 non-null float64
dropoff_lon                          5983 non-null float64
when_the_delivery_started            5983 non-null object
when_the_Jumpman_arrived_at_pickup   5433 non-null object
when_the_Jumpman_left_pickup         5433 non-null object
when_the_Jumpman_arrived_at_dropoff  5983 non-null object
dtypes: float64(5), int64(3), object(10)
memory usage: 841.5+ KB
None
```

Initial dataset: 5,983 records, 18 features

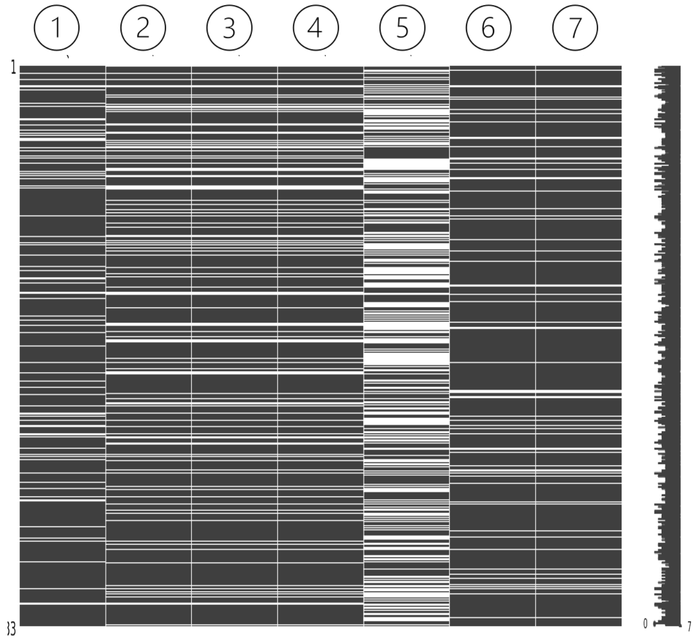
4.1 DATA [INTEGRITY CONCERNS]

No major data quality concerns that affect the analysis. Minor issues include missing data and suspicious observations.

Data quality examples:

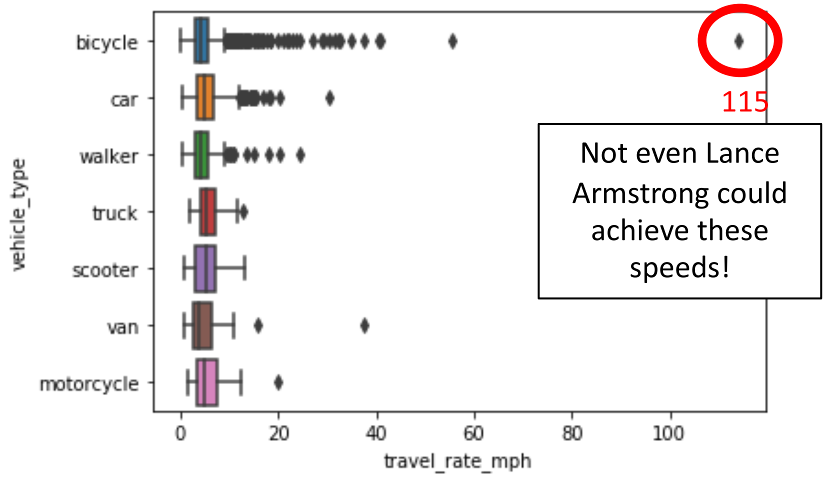
Heatmap showing seven columns of missing data, by record descending vertically. White space indicates missing data. **Significant.**

Notice data is usually missing for multiple columns for the same record. Systemic data collection issue!

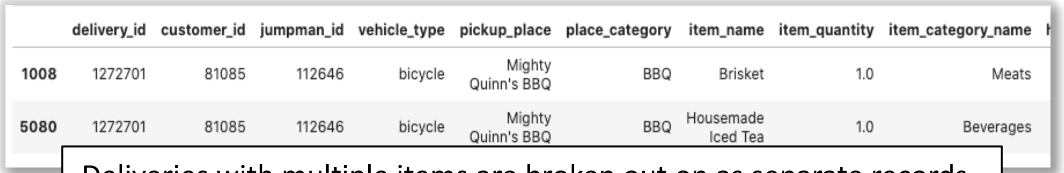


Attribute	Missing Value Count
delivery_id	0
customer_id	0
jumpman_id	0
vehicle_type	0
pickup_place	0
place_category	883
item_name	1,230
item_quantity	1,230
item_category_name	1,230
how_long_it_took_to_order	2,945
pickup_lat	0
pickup_lon	0
dropoff_lat	0
dropoff_lon	0
when_the_delivery_started	0
when_the_Jumpman_arrived_at_pickup	550
when_the_Jumpman_left_pickup	550
when_the_Jumpman_arrived_at_dropoff	0

SUSPICIOUS examples:



Not even Lance Armstrong could achieve these speeds!



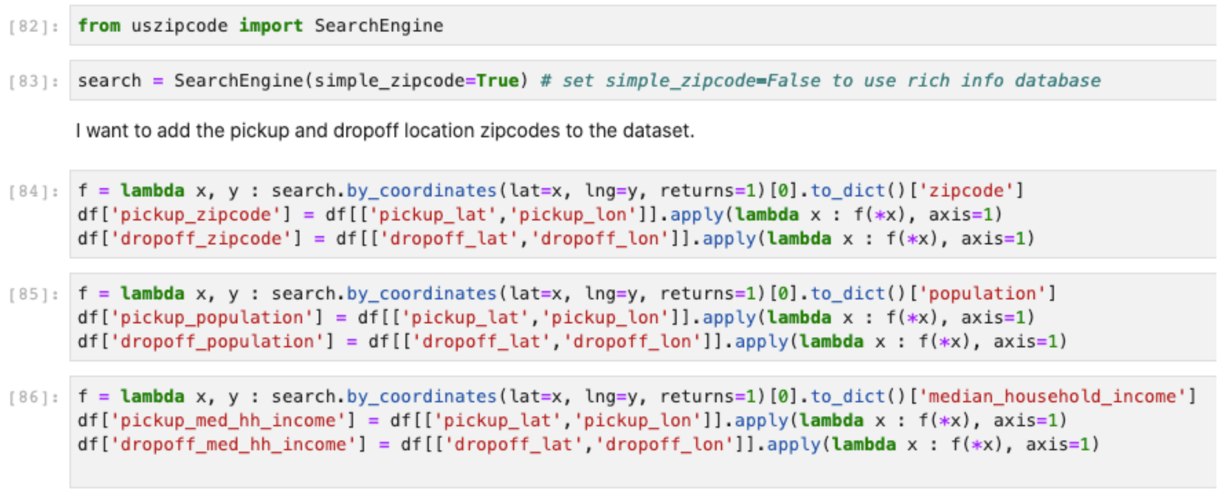
	delivery_id	customer_id	jumpman_id	vehicle_type	pickup_place	place_category	item_name	item_quantity	item_category_name
1008	1272701	81085	112646	bicycle	Mighty Quinn's BBQ	BBQ	Brisket	1.0	Meats
5080	1272701	81085	112646	bicycle	Mighty Quinn's BBQ	BBQ	Housemade Iced Tea	1.0	Beverages

Deliveries with multiple items are broken out on as separate records

4.2 DATA [FEATURE ENGINEERING]

Sample code to show the process behind feature engineering of attributes not in the original dataset.

DEMOGRAPHIC: Zip code, population, & median household income



```
[82]: from uszipcode import SearchEngine

[83]: search = SearchEngine(simple_zipcode=True) # set simple_zipcode=False to use rich info database
```

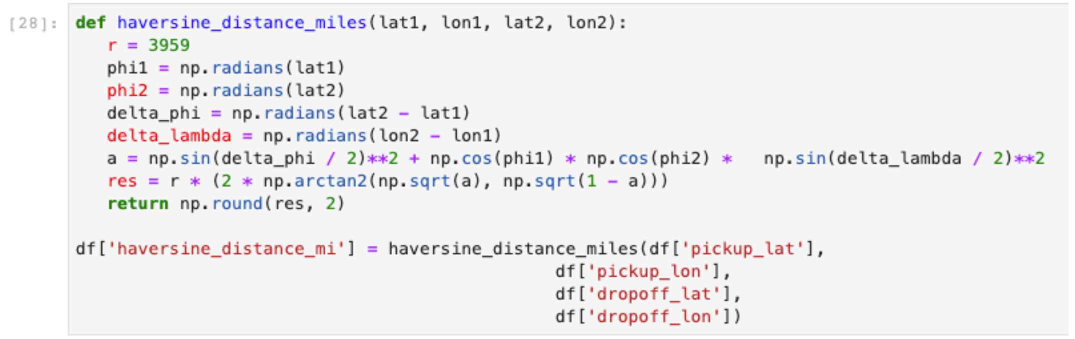
I want to add the pickup and dropoff location zipcodes to the dataset.

```
[84]: f = lambda x, y : search.by_coordinates(lat=x, lng=y, returns=1)[0].to_dict()['zipcode']
      df['pickup_zipcode'] = df[['pickup_lat','pickup_lon']].apply(lambda x : f(*x), axis=1)
      df['dropoff_zipcode'] = df[['dropoff_lat','dropoff_lon']].apply(lambda x : f(*x), axis=1)

[85]: f = lambda x, y : search.by_coordinates(lat=x, lng=y, returns=1)[0].to_dict()['population']
      df['pickup_population'] = df[['pickup_lat','pickup_lon']].apply(lambda x : f(*x), axis=1)
      df['dropoff_population'] = df[['dropoff_lat','dropoff_lon']].apply(lambda x : f(*x), axis=1)

[86]: f = lambda x, y : search.by_coordinates(lat=x, lng=y, returns=1)[0].to_dict()['median_household_income']
      df['pickup_med_hh_income'] = df[['pickup_lat','pickup_lon']].apply(lambda x : f(*x), axis=1)
      df['dropoff_med_hh_income'] = df[['dropoff_lat','dropoff_lon']].apply(lambda x : f(*x), axis=1)
```

“uszipcode” package provides access US census data in dictionary format

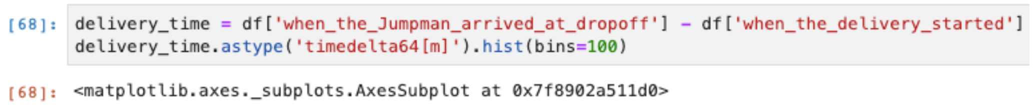
A sample function returns zip code based on latitude and longitude coordinates

GEOGRAPHIC: Haversine “as the crow flies” distance calculation

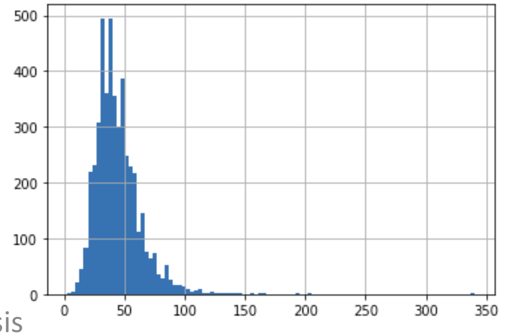


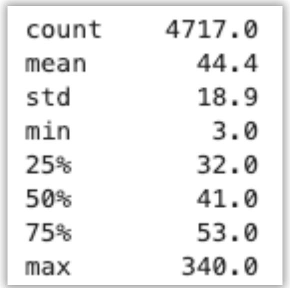
```
[28]: def haversine_distance_miles(lat1, lon1, lat2, lon2):
          r = 3959
          phi1 = np.radians(lat1)
          phi2 = np.radians(lat2)
          delta_phi = np.radians(lat2 - lat1)
          delta_lambda = np.radians(lon2 - lon1)
          a = np.sin(delta_phi / 2)**2 + np.cos(phi1) * np.cos(phi2) *   np.sin(delta_lambda / 2)**2
          res = r * (2 * np.arctan2(np.sqrt(a), np.sqrt(1 - a)))
          return np.round(res, 2)

      df['haversine_distance_mi'] = haversine_distance_miles(df['pickup_lat'],
                                                             df['pickup_lon'],
                                                             df['dropoff_lat'],
                                                             df['dropoff_lon'])
```

DESCRIPTIVE STATISTICS: Total Time to Delivery



```
[68]: delivery_time = df['when_the_Jumpman_arrived_at_dropoff'] - df['when_the_delivery_started']
      delivery_time.astype('timedelta64[m]').hist(bins=100)

[68]: <matplotlib.axes._subplots.AxesSubplot at 0x7f8902a511d0>
```





```
count    4717.0
mean       44.4
std        18.9
min         3.0
25%        32.0
50%        41.0
75%        53.0
max       340.0
```

5 GROWTH STRATEGY

GROWTH RECOMMENDATIONS

TARGETED CAMPAIGNS

+

ENGAGEMENT

+

WAIT TIMES

5.1 GROWTH STRATEGY [TARGETED CAMPAIGNS]

GROWTH RECOMMENDATION #1: Targeted campaigns to specific zip codes.

- [11226, 10025, 11211]: have the lowest market penetration rate, currently reaching 60 of the 290K potential customers
- [10282,10007,10069]: have the highest average household median income $200K+ and a low market penetration rate, currently reaching 110 of the 17K potential customers

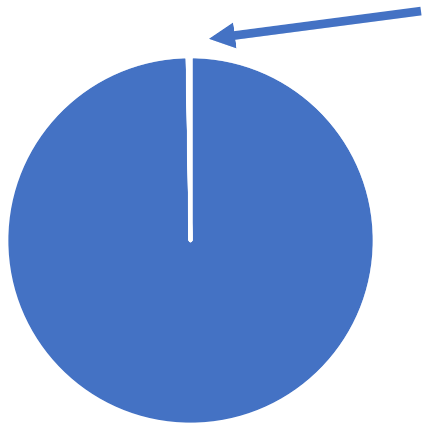
UNTAPPED MARKET POTENTIAL

290,000

Population of NY's three most populous zip codes

60

Number of current customers in those same three zip codes



0.26%
Current reach of the **10 highest earning* zip codes**

*by estimated average household income

5.1 GROWTH STRATEGY [TARGETED CAMPAIGNS]

10 MOST POPULOUS ZIP CODES

Zip Code	Estimated Population	# Unique Customers	Market Penetration (X 10)
11226	101,572	3	0.030%
10025	94,600	26	0.275%
11211	90,117	31	0.344%
11206	81,677	2	0.024%
10002	81,410	135	1.658%
11221	78,895	2	0.025%
10029	76,003	39	0.513%
11215	63,488	13	0.205%
10009	61,347	121	1.972%
10023	60,998	73	1.197%

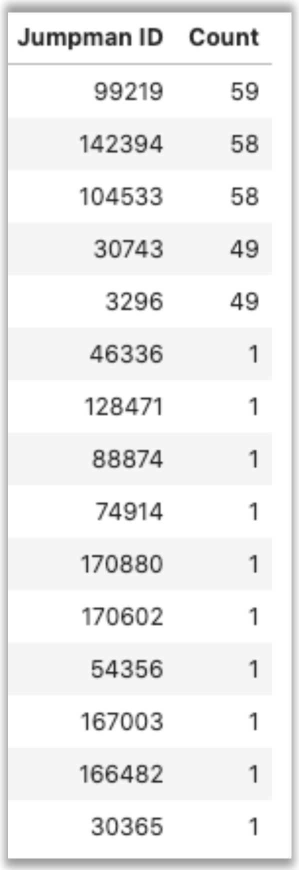
10 HIGEST EARNING ZIP CODES

Zip Code	Median Household Income	Estimated Population	# Unique Customers	Market Penetration (X 10)
10282	230,952	4,783	27	5.645%
10007	216,037	6,988	38	5.438%
10069	170,630	5,199	45	8.656%
10162	168,667	1,685	64	37.982%
10280	129,574	7,853	18	2.292%
11109	125,871	3,523	2	0.568%
10005	124,670	7,135	18	2.523%
10006	119,274	3,011	31	10.296%
10065	115,519	32,270	99	3.068%
10024	109,956	59,283	1	0.017%

Market Penetration (X10) =
(# Unique Customers / Estimated Population) *10

5.2 GROWTH STRATEGY [ENGAGEMENT]

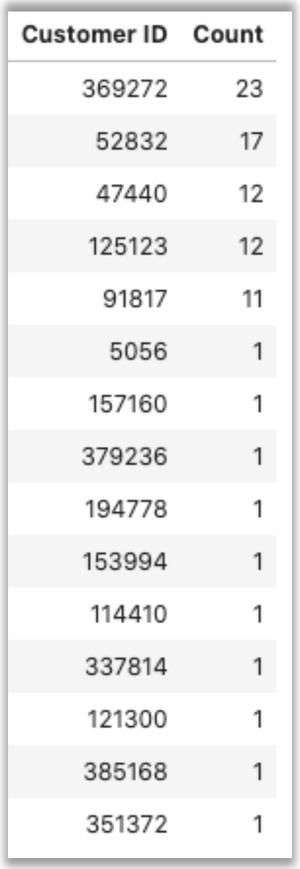
GROWTH RECOMMENDATION #2: Focus on Jumpmen and Customer engagement. Survey the people who use the platform the most to gather feedback on their hurdles and challenges. Also engage with the single use customers who have not yet returned.



Jumpman ID	Count
99219	59
142394	58
104533	58
30743	49
3296	49
46336	1
128471	1
88874	1
74914	1
170880	1
170602	1
54356	1
167003	1
166482	1
30365	1

Five most active Jumpmen/Customers on the platform

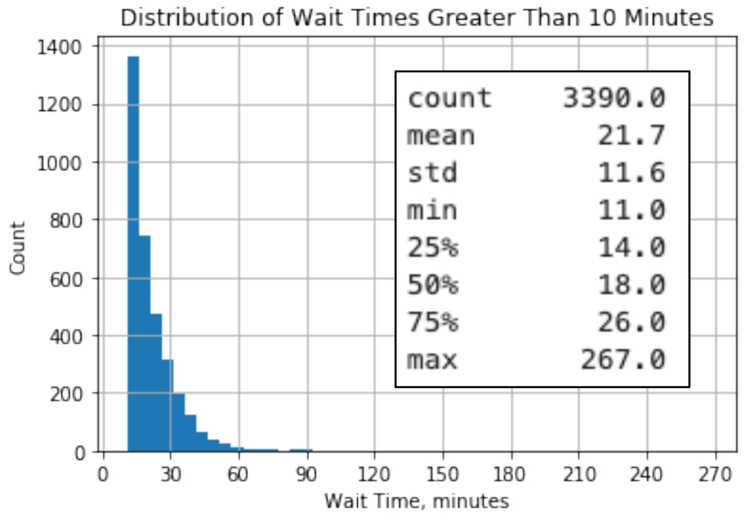
Ten minimally active Jumpmen/Customers (random sample)



Customer ID	Count
369272	23
52832	17
47440	12
125123	12
91817	11
5056	1
157160	1
379236	1
194778	1
153994	1
114410	1
337814	1
121300	1
385168	1
351372	1

5.3 GROWTH STRATEGY [WAIT TIMES]

GROWTH RECOMMENDATION #3: Decrease wasted time at restaurants by minimizing **WAIT TIMES** for the Jumpmen. Explore a restaurant partnership model to minimize the time spent waiting at the pickup location.



```
count    3390.0
mean       21.7
std        11.6
min        11.0
25%        14.0
50%        18.0
75%        26.0
max       267.0
```

- Set an ambitious goal to have Jumpmen spending **no longer than 15 minutes** at each restaurant. Achieve this goal by implementing a business partnership model.
- In October 2014, reducing all wait times to a max of 15 minutes would have saved Jumpmen a total of 14,000 minutes, or an average of 27 minutes per Jumpman
- Giving time back to Jumpmen allows them to put more time into the platform

6 NEXT STEPS [BEYOND THIS ANALYSIS]

1. Implement the outlined growth strategies
2. Define specific KPIs to better understand performance.
   a) The CEO wants to grow the business 20%. In terms of what? Number of transactions? Number of customer's reached? Number of restaurants on the platform? Completed transaction value?
3. Invest in quality assurance measures to investigate data inconsistencies
4. Broaden the scope of the analysis. More data!
   a) Time series modeling to account for seasonality. What insights from other markets could we leverage to better forecast the New York Market?

7 APPENDIX

I hope you enjoyed my analysis. Please reach out with any further questions.

The full analysis can be found on my GitHub profile.

Contact Information:

Stephen Stark

Email | LinkedIn | GitHub