Metrocar Funnel Analysis

Cecilia Martinez
10/29/2023

Summary

Using SQL, Python and Tableau for our comprehensive analysis of the Metrocar customer funnel, we examined the user journey from app download to customer review. Our findings unearthed a significant drop-off occurring notably at the "ride complete" stage of the funnel.

A deeper analysis of the data brought to light a crucial insight: customers who canceled their rides before completion experienced wait times (from ride request to driver acceptance) that were more than twice as long as those who did not cancel. Specifically, customers who did not cancel experienced an average wait time of 6 minutes, while those who canceled faced an extended, average wait time of 15 minutes.

Additionally, our study revealed that a dominant 61% of our customer base comprises iOS users, with the remainder divided between Android users and those utilizing the web platform. Moreover, our research pinpointed our target demographic: individuals in the 35-44 age group, offering valuable insights for targeted marketing strategies.

In our in-depth analysis of the reviews data, we discovered that the average rating stands at 3 out of 5 stars. A closer examination of the distribution revealed the following breakdown:

- 1 Star: 14,938 reviews
- 2 Stars: 4,942 reviews
- 3 Stars: 5,070 reviews
- 4 Stars: 12,610 reviews
- 5 Stars: 12,440 reviews

This detailed assessment provides a comprehensive view of the user sentiment, allowing us to identify specific areas of improvement and address user concerns effectively.

In light of our analysis, we propose the implementation of a dynamic pricing strategy during peak hours and high-demand areas. By introducing surge pricing during these bustling periods, we provide drivers with enhanced incentives to operate, thereby augmenting their availability, mitigating wait times, and elevating overall customer satisfaction.

Furthermore, our recommendations extend beyond surge pricing. To capitalize on the broader market landscape, we advocate allocating a significant portion of our upcoming year's marketing budget towards attracting Android users. Despite constituting a substantial majority in the market, Android users represent only 29% of our current user base. Focusing our marketing efforts in this direction could bridge this gap and expand our reach significantly.

Lastly, to delve deeper into the challenges faced at the "ride complete" stage, we propose the implementation of an in-app survey prompt. By integrating this prompt when a customer cancels a ride, we can glean invaluable insights into the reasons behind cancellations. This qualitative data would prove invaluable in refining our services and addressing customer concerns effectively.

These strategic recommendations, rooted in a thorough analysis of our user funnel, aim to enhance user experience and optimize service efficiency.

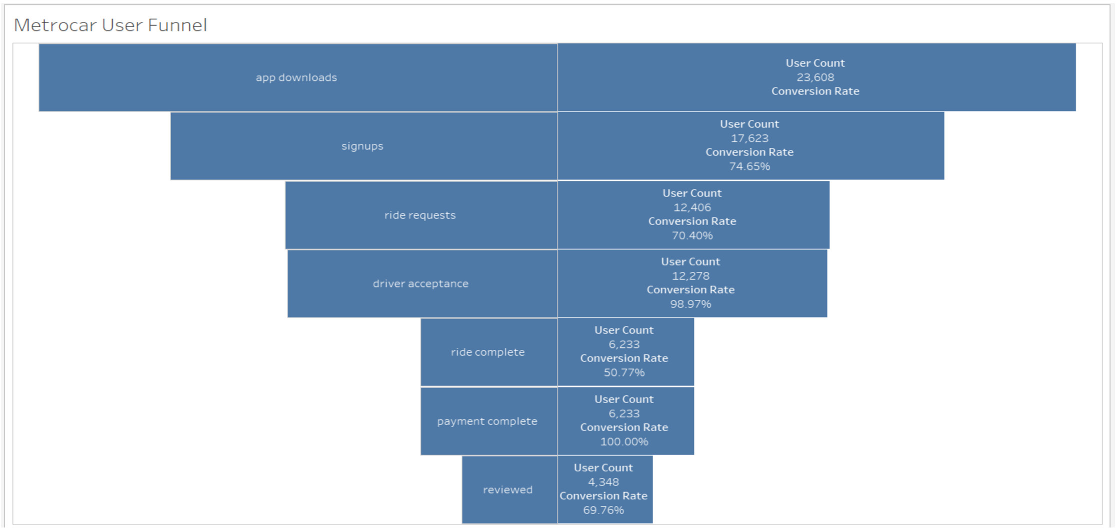
Context

In this analysis, we have examined the customer funnel of our ride-sharing service, Metrocar, to gain valuable insights into user behavior and enhance overall user experience. We used SQL to extract data and Tableau to visualize data. The funnel comprises of the following stages:

1. **App Download:** This initial step involves users downloading the Metrocar app.
2. **Signup:** Users register with the app by entering personal information, such as age and gender.
3. **Ride Request:** Users use app to search and select a ride. This action dispatches requests to nearby drivers for driver acceptance.
4. **Driver Acceptance:** Nearby drivers are notified of ride requests and driver accepts.
5. **Ride Complete:** The user experiences ride service from start to finish and reaches the destination.
6. **Payment Complete:** User completes payment, Metrocar receives payment for service.
7. **Reviewed:** User reviews the ride share experience.

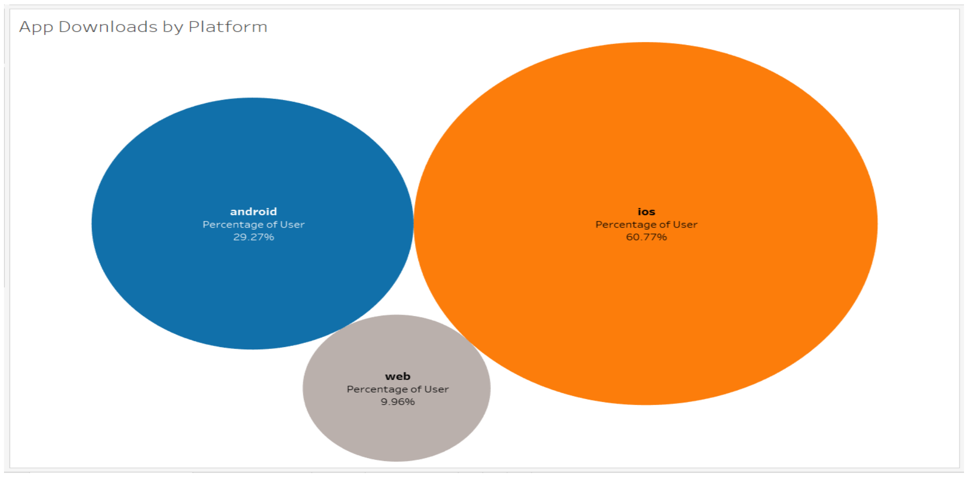
The analysis was conducted using a dataset collected over 1 year with a total of 17,623 users downloading the app. This dataset includes tables for app downloads, signups, ride requests, transactions, and reviews information.

Findings



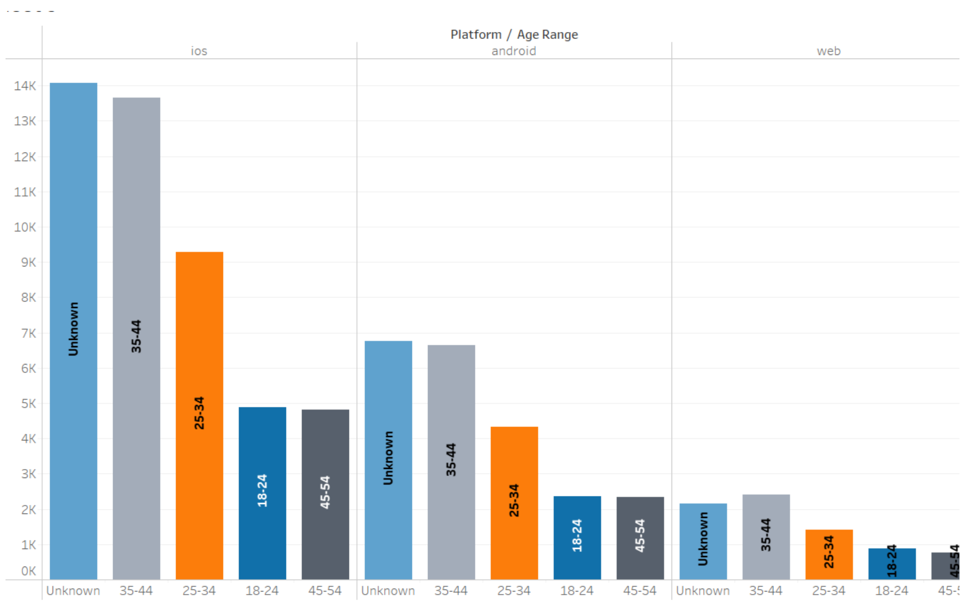
- Analyzing the user funnel, we can see that 23,608 users downloaded the Metrocar app.
- Out of the 23,608 users who downloaded the app, 74.65% of users ended up creating an account.
- 70.40% of users then requested a ride, while 98.97% of those requested rides were then accepted by a driver.
- The lowest conversion rate observed in the user funnel is in the ride complete stage, with only 50.77% of users completing their ride. This signifies that the user canceled the ride before it was completed. We do not have sufficient data at this time to determine exactly when or why the user canceled between the driver accepting the ride and the ride being completed. I suggest we collect data to determine the reasoning behind the user canceling their ride in the form of a survey or question if the user happens to cancel the ride.
- Of all users who requested and completed their rides, 100% completed payment.
- The next lowest conversion rate is in the review stage of our funnel, with 69.76% of users who completed a ride reviewing their ride.

Platform Details



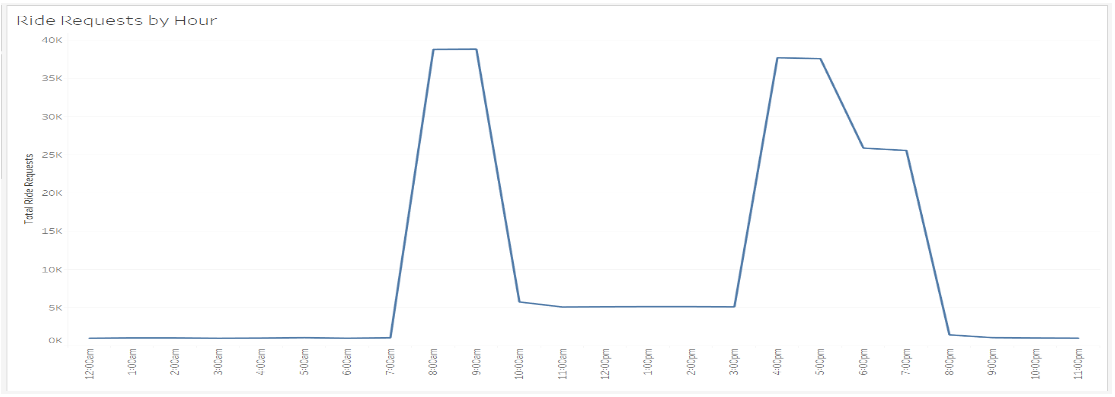
- Analyzing the data for downloads by platform, we see that 60.77% of our users are on the ios operating system, while 29.27% of users are on android and 9.96% of users are on web.
- Despite android users constituting the majority of market share, they only represent 29% of our user base. The difference in androids global market and its representation in our user base suggests a significant growth opportunity if we can tailor our marketing efforts toward android users.

Customer Base by Age



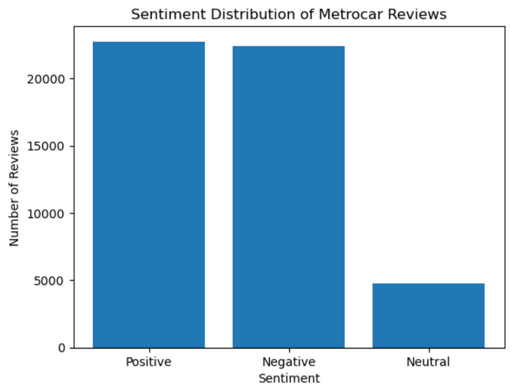
- A large number of the user's age is unknown, about 30%, which presents a limitation in the analysis.
- With the data we have, we can conclude that the 35-44 age group outperforms all other age groups in every step of the funnel.
- Users in the age group 35-44 are our likely target customers.

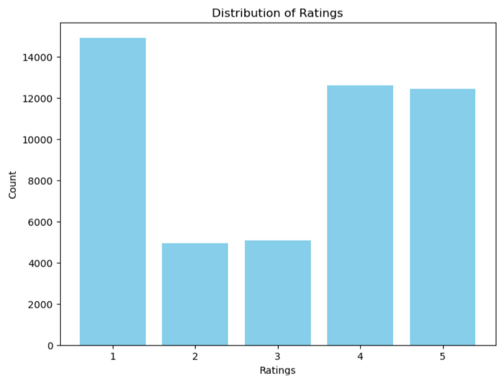
Ride Requests by Hour

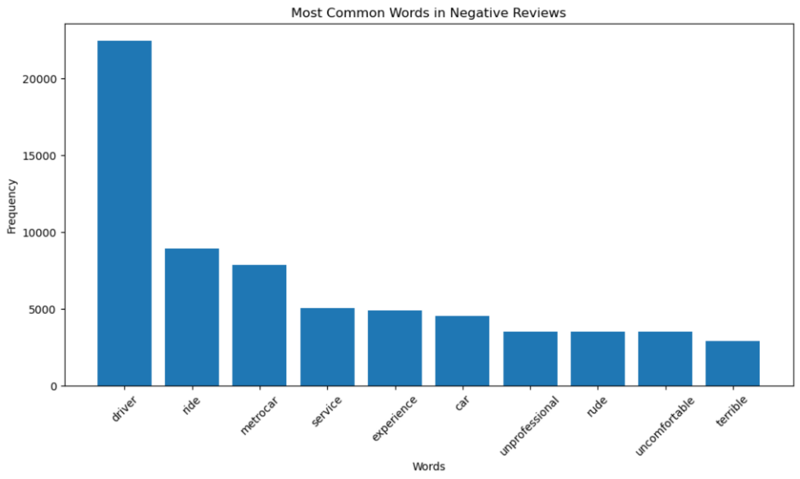


- Analyzing ride request patterns revealed 2 significant peaks in user demand, occurring consistently on a daily basis. The first surge takes place between 7 AM and 10 AM, coinciding with the morning rush hour.
- The second surge takes place later in the day, from 3 PM to 8 PM. This surge coincides with afternoon and early evening rush hours.
- We recommend the adoption of a price surge strategy during peak ride request hours. This strategic approach involves dynamically adjusting fares in response to high demand, incentivizing drivers to be available during these busy periods.

User Sentiment







- Applying Python for sentiment analysis on our reviews data, we observe a nearly equal distribution between positive and negative reviews. Specifically, we find 22,764 positive reviews and 22,439 negative reviews.
- Visualization of the most frequent words used in negative reviews can help us identify common issues or concerns most mentioned by users.

Recommendations

- Adopt a price surge strategy during peak rush hour times to incentivize more drivers to work during busiest times of the day, encouraging availability, reducing wait time and ensuring customer satisfaction.
- Investigate the low conversion rates of the ride complete stage of our funnel. Ask the user the reasoning for the cancellation through a survey or questionnaire when they cancel rides. We can do this by displaying a survey as a prompt within the app interface after a user cancels a ride.
- Focus our marketing budget for the next year on targeting android users. Android users account for the majority of global market share yet only make up 29% of our user base. Along with targeting android users, focus on users 35-44 as they make up the majority of our user base.

Appendix

SQL code to extract data for user funnel:

```
WITH downloads AS (
  select count(app_download_key) as num_of_downloads,
  platform,
  age_range,
  TO_CHAR(download_ts, 'YYYY-MM-DD') AS download_date
from app_downloads
left join signups on app_downloads.app_download_key = signups.session_id
group by platform, age_range, download_date
),

signup_cust AS (
  select count(user_id) as num_of_signups,
  platform,
  age_range,
  TO_CHAR(download_ts, 'YYYY-MM-DD') AS download_date
from signups
join app_downloads ad on signups.session_id = ad.app_download_key
group by platform, age_range, download_date
),

rides AS (
  select count(distinct ride_requests.user_id) as num_of_ride_req,
  platform,
  age_range,
  TO_CHAR(download_ts, 'YYYY-MM-DD') AS download_date
from ride_requests
join signups on ride_requests.user_id = signups.user_id
  join app_downloads on signups.session_id = app_downloads.app_download_key
group by platform, age_range, download_date
),

complete_ride AS (
  select count(distinct ride_requests.user_id) as num_of_comp_rides,
  platform,
  age_range,
  TO_CHAR(download_ts, 'YYYY-MM-DD') AS download_date
from ride_requests
join signups on ride_requests.user_id = signups.user_id
  join app_downloads on signups.session_id = app_downloads.app_download_key
where cancel_ts IS NULL
group by platform, age_range, download_date
),


driver AS (
  select count(distinct ride_requests.user_id) as num_of_driver_accep,
  platform,
  age_range,
  TO_CHAR(download_ts, 'YYYY-MM-DD') AS download_date
from ride_requests
  join signups on ride_requests.user_id = signups.user_id
  join app_downloads on signups.session_id = app_downloads.app_download_key
where accept_ts is not null
group by platform, age_range, download_date
),

payment AS (
  SELECT COUNT(DISTINCT ride_requests.user_id) as comp_payment,
  platform,
  age_range,
  TO_CHAR(download_ts, 'YYYY-MM-DD') AS download_date
```

```
FROM ride_requests
join signups on ride_requests.user_id = signups.user_id
  join app_downloads on signups.session_id = app_downloads.app_download_key
WHERE ride_id IN (
    SELECT ride_id
    FROM transactions
    WHERE charge_status = 'Approved')
group by platform, age_range, download_date),

reviewed AS (
  SELECT count(distinct reviews.user_id) AS review,
  platform,
  age_range,
  TO_CHAR(download_ts, 'YYYY-MM-DD') AS download_date
  from reviews
  left join signups on reviews.user_id = signups.user_id
  join app_downloads on signups.session_id = app_downloads.app_download_key
group by platform, age_range, download_date
)

SELECT
    funnel_step,
    funnel_name,
    platform,
    age_range,
    download_date,
    value as num_of_cust
FROM (
    SELECT
        1 AS funnel_step,
        'app downloads' AS funnel_name,
        num_of_downloads AS value,
                   platform,
                   age_range,
    download_date
    FROM downloads

    UNION

    SELECT
        2 AS funnel_step,
        'signups' AS funnel_name,
        num_of_signups AS value,
                   platform,
                   age_range,
    download_date
    FROM signup_cust

    UNION

    SELECT
        3 AS funnel_step,
        'ride requests' AS funnel_name,
        num_of_ride_req AS value,
                   platform,
                   age_range,
                   download_date
    FROM rides

    UNION

    SELECT
        4 AS funnel_step,
        'driver acceptance' AS funnel_name,
        num_of_driver_accep AS value,
                   platform,
                   age_range,
                   download_date
    FROM driver
    UNION

    SELECT
        5 AS funnel_step,
        'ride complete' AS funnel_name,
        num_of_comp_rides AS value,
                    platform,
                    age_range,
    download_date
    FROM complete_ride

    UNION

    SELECT
        6 AS funnel_step,
        'payment complete' AS funnel_name,
        comp_payment AS value,
                    platform,
                    age_range,
    download_date
    FROM payment

    UNION

    SELECT
        7 AS funnel_step,
        'reviewed' AS funnel_name,
        review AS value,
                    platform,
                    age_range,
    download_date
    FROM reviewed
) AS steps
ORDER BY funnel_step, platform, age_range, download_date;
```

SQL code for extracting time of day, wait time data:

```
SELECT
    EXTRACT(HOUR FROM request_ts) AS hour_of_day,
    SUM(CASE WHEN cancel_ts IS NOT NULL THEN 1 ELSE 0 END) AS cancel_count,
    SUM(CASE WHEN cancel_ts IS NULL THEN 1 ELSE 0 END) AS no_cancel_count,
    AVG(CASE WHEN cancel_ts IS NULL THEN EXTRACT(EPOCH FROM (accept_ts - request_ts)) / 60 ELSE NULL END) AS
avg_wait_time_no_cancel_minutes,
    AVG(CASE WHEN cancel_ts IS NOT NULL THEN EXTRACT(EPOCH FROM (accept_ts - request_ts)) / 60 ELSE NULL END) AS
avg_wait_time_cancel_minutes
FROM
    ride_requests
WHERE
    accept_ts IS NOT NULL AND request_ts IS NOT NULL
GROUP BY
    EXTRACT(HOUR FROM request_ts)
ORDER BY
    EXTRACT(HOUR FROM request_ts);
```