Ryan McCaffrey
7/27/17
Uber Data Challenge

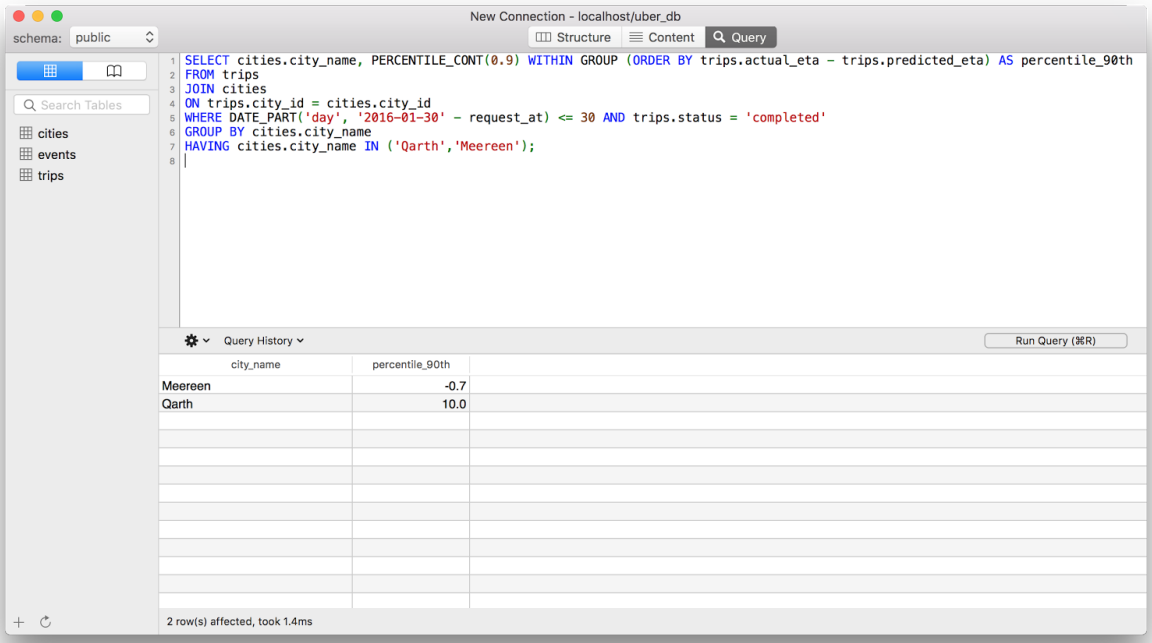
Part 1

To verify that the PostgreSQL logic and syntax of my queries was correct, for these problems I created an Uber database with tables “trips”, “cities” and “events” using Postgres.app and PSequel. I filled the tables with simulated data and tested the queries to verify that they work (though the solutions may not be the most elegant, they should work).

Question 1

For each of the cities 'Qarth' and 'Meereen', calculate 90th percentile difference between Actual and Predicted ETA for all completed trips within the last 30 days.

```
SELECT cities.city_name,
PERCENTILE_CONT(0.9) WITHIN GROUP (ORDER BY trips.actual_eta -
trips.predicted_eta) AS percentile_90th
FROM trips
JOIN cities
ON trips.city_id = cities.city_id
WHERE DATE_PART('day', NOW() - request_at) <= 30 AND trips.status =
'completed'
GROUP BY cities.city_name
HAVING cities.city_name IN ('Qarth','Meereen');
```



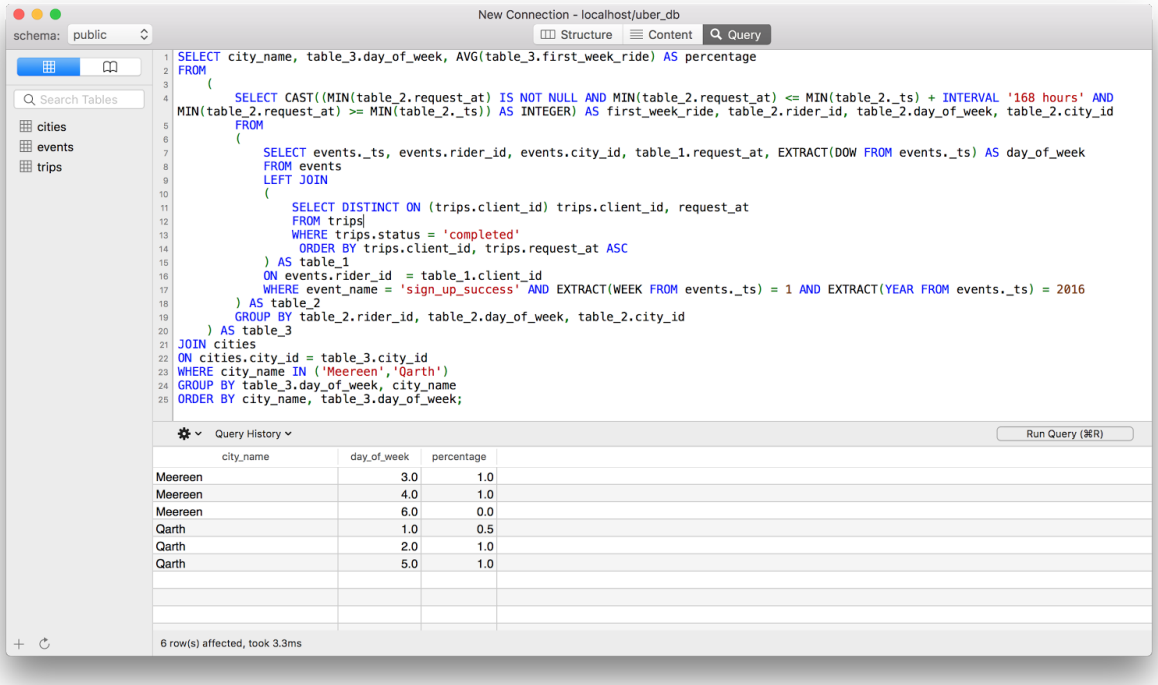
Screenshot of SQL query and database output (using sample data) for Question 1.

Question 2

A signup is defined as an event labeled 'sign_up_success' within the events table. For each city ('Qarth' and 'Meereen') and each day of the week, determine the percentage of signups in the first week of 2016 that resulted in completed a trip within 168 hours of the sign up date.

```
SELECT city_name, table_3.day_of_week, AVG(table_3.first_week_ride) AS
percentage
FROM
    (
        SELECT CAST((MIN(table_2.request_at) IS NOT NULL AND
MIN(table_2.request_at) <= MIN(table_2._ts) + INTERVAL '168 hours' AND
MIN(table_2.request_at) >= MIN(table_2._ts)) AS INTEGER) AS
first_week_ride, table_2.rider_id, table_2.day_of_week,
table_2.city_id
        FROM
        (
            SELECT events._ts, events.rider_id, events.city_id,
table_1.request_at, EXTRACT(DOW FROM events._ts) AS day_of_week
            FROM events
            LEFT JOIN
```

```
            (
                SELECT DISTINCT ON (trips.client_id) trips.client_id,
request_at
                FROM trips
                WHERE trips.status = 'completed'
                 ORDER BY trips.client_id, trips.request_at ASC
            ) AS table_1
            ON events.rider_id  = table_1.client_id
            WHERE event_name = 'sign_up_success' AND EXTRACT(WEEK FROM
events._ts) = 1 AND EXTRACT(YEAR FROM events._ts) = 2016
        ) AS table_2
        GROUP BY table_2.rider_id, table_2.day_of_week,
table_2.city_id
    ) AS table_3
JOIN cities
ON cities.city_id = table_3.city_id
WHERE city_name IN ('Meereen','Qarth')
GROUP BY table_3.day_of_week, city_name
ORDER BY city_name, table_3.day_of_week;
```



Screenshot of SQL query and database output (using sample data) for Question 2.

Part 2

Please see Jupyter notebook for full explanations of analysis and modeling.

Question 1

Perform any cleaning, exploratory analysis, and/or visualizations to use the provided data for this analysis (a few sentences/plots describing your approach will suffice). What fraction of the driver signups took a first trip? (2 points)

For the feature engineering I created time-dependent features (e.g., time between signup and background check, time between signup and vehicle registration, etc.) and segmented the dates into more useful datetime objects such as day of month and day of week in order to capture any effects related to promotional campaigns that would target weekends or the end of month. Some of the categorical features I dummified and some I label-encoded, which mostly depended on the number of categorical variables per feature. Additional features were created to flag the presence of NaN values. Also, I created two vehicle-related features: (1) a feature that flagged "old" cars as defined by those that are older than the average age of all cars currently in operation in the US, and (2) a feature that flagged "luxury" cars as defined by those that have both letters and numbers in the car name (supporting link). Both of the car-related "old" and "luxury" features seemed to have minimal importance in our best performing model. Finally, prior to modeling the design matrix was scaled and normalized to have unit variance.

Only 11.2% of driver applicants who signed up with Uber actually took a first trip as a driver.

Question 2

Build a predictive model to help Uber determine whether or not a driver signup will start driving. Discuss why you chose your approach, what alternatives you considered, and any concerns you have. How valid is your model? Include any key indicators of model performance. (2 points)

A gradient boosting model seemed to work the best based on all major binary classification evaluation metrics (i.e., accuracy, ROC AUC, F1 score, precision, and recall). I tried 7 other models including logistic regression, decision tree, random forest, extra trees, adaboost, KNN and SVM models. Model selection was determined using 5-fold cross-validation. Additionally, a classification report, confusion matrix and feature importance were calculated using a train-test-split cross-validation approach. Based on how I structured the target variable (1 = signup but trip not completed, 0 = signup and trip completed) and my understanding of the

business problem, I believe recall to be the best evaluation metric to use in this case as it will ensure we maximize our ability to predict the people who sign up but never complete their first trip as a driver. From a business perspective, I believe we should target those individuals and incentivize them to complete their first trip as a driver. The recall score for the gradient boosting model was 0.963 ± 0.002.

Question 3

Briefly discuss how Uber might leverage the insights gained from the model to generate more first trips (again, a few ideas/sentences will suffice). (1 point)

Uber can leverage the model to better predict those who will sign up but not actually complete their first trip as a driver. In the context of this problem, recall tells us the following: of all the applicants who actually signed up and did not complete a trip as a driver, what percentage of these applicants did we correctly predict? I believe the goal of this project is to correctly predict - as best as possible - the applicants who sign up but never complete a trip as a driver, which is equivalent to maximizing recall in this case. It is important for us to correctly identify these people so that Uber can take appropriate action with them by applying ideas like promotional rates, driver incentives, app alerts/reminders, and email campaigns.

In addition, if we look at the plot below it clearly shows that applicants who fail to register their vehicles 30 days after first signing up with Uber almost never complete their first trip as a driver. Uber should use this knowledge to send reminders or incentives to the applicants after two weeks of no updates by the applicant. This finding is significant since *signup_vehicle_delta* (i.e., the time between signup and vehicle registration) had the highest feature importance in our gradient boosting model. The second ranked feature in terms of feature importance to the model was *vehicle_year*. As a way of recruiting new drivers Uber should think about targeting people driving newer cars by perhaps concentrating advertisements at new car dealerships or striking partnerships with auto loan providers and/or online new car websites.


Days Between Driver Signup and Vehicle Registration with Uber
Drivers
Not Drivers
Count
2500
2000
1500
1000
500
0
0
10
20
30
40
50
60
Days