## Foundations of Data Science - Capstone Project Milestone Report

Jonathon Thompson

01/02/2017

**Introduction to the problem**

Getaround's [current search function](https://www.getaround.com/search) is utilitarian in nature and requires users to have preexisting knowledge about the types of cars they want to rent. Results are presented to potential renters based on proximity, availability and cost, with simple vehicle filters limited to transmission type, parking, body type and make.

This project’s goal is to demonstrate a new approach to searching and presenting results to renters based on what they intend to do with the car during their trip. For example, instead of blindly sifting through listings, a renter could search for cars that are good for “running errands”, “economical” or “fun”.

“Getaround is an online car sharing or peer-to-peer car sharing service that allows drivers to rent cars from private car owners, and owners to rent out their cars for payment. Owners set their rental prices and earn a 60% commission from their rental revenue. The company currently operates rentals in the San Francisco Bay Area, San Diego, Austin, Portland, Chicago, and Washington, D.C.” <https://en.wikipedia.org/wiki/Getaround>

This project is intended to be a road map for product growth, designed to show Getaround how additional search features could enhance both the renters' and owners’ experience. While today’s customers may only come to Getaround to get from point A to point B, a more dynamic way of presenting listings could expand the variety of vehicles that repeat renters utilize and attract a broader customer base; think Discover Weekly for car sharing.

This concept could start with using 3rd-party data like Edmunds, but could shift towards Getaround becoming a content curator - using owners’ listings and descriptions to allow owners to have an impact into their listing’s rankings. The renter’s review process could be expanded to capture more feedback like rankings in specific categories and questions asking what the vehicle was used for.

Imagine a search that one day incorporates a renter’s own history, ratings from other renters, listing descriptions and features from owners and is able to present you with the perfect car for your trip.

**A deeper dive into the data set:**

**What important fields and information does the data set have?**

The data fields that I have preliminarily identified to use in this project are as follows:

[Getaround](https://index.getaround.com/v1.0/search?product=web&uid=100005837281185&user_lat=37.7717185&user_lng=-122.44389289999998&viewport=37.514591%2C-122.644614%2C38.028441%2C-122.243172&properties=car_id,car_name,car_photo,carkit_enabled,distance,latitude,longitude,make,model,price_daily,price_hourly,price_weekly,total_price,timezone,year,dedicated_parking&sort=best&page_sort=magic&page_size=1000" \t "_blank)

Details about the vehicles available for rent in a given location.

year – Vehicle Year

make – Vehicle Make

model - Vehicle Model

car\_name – Getaround unique “username” for each rental

car\_id - Getaround unique identifier for each rental

distance – Distance in miles the vehicle is from the searcher’s location

[Edmunds](http://developer.edmunds.com/api-documentation/overview/" \t "_blank)

SPEC: VEHICLE STYLE

This data gets us the styles.id and basic vehicle info based on passing make, model and year into the API.

styles.id – Edmunds primary identifier for a specific vehicle model

styles.trim – Use this to filter by “Base” models

styles.make.niceName – Vehicle Make in lowercase

styles.model.niceName – Vehicle Model in lowercase

styles.year.year – Vehicle Year

styles.submodel.niceName – Type of car like SUV or Sedan in lowercase

SPEC: VEHICLE EQUIPMENT

This is the heart of the details about the vehicles. The data is stored across two separate APIs, one of which provides basic attributes and another that provides a detailed feature breakdown. For the latter we pull back features that are only considered "STANDARD".

Basic

styles.id – Edmunds primary identifier for a specific vehicle model

styles.drivewheels – Front, rear or all-wheel drive

styles.doors – Number of doors the vehicle has

categories.market – Vehicle market like Luxury or Performance

categories.vehiclesize - Vehicle size like Compact or Midsize

categories.vehiclestyle – Vehicle style with more specific classifications than styles.submodel.niceName

engine.cylinder – Number of cylinders the engine has

engine.horsepower – Vehicle horsepower

engine.torque – Vehicle torque

Details

styles.id – Edmunds primary identifier for a specific vehicle model

equipment.name – Category of vehicle equipment

equipment.type – Subcategory of vehicle equipment

attributes.name – Equipment name

attributes.value – Equipment value; can be numerical data points like MPG or descriptive like “leather” or “cloth”.

CONTENT: EDMUNDS CAR RATINGS

Rankings in various categories by Edmunds reviewers as a score of 1-10 or a grade of A-F. This is only provided for model years 2013 or newer and covers less than half of our Getaround universe.

make.niceName – Vehicle Make in lowercase

model.niceName – Vehicle Model in lowercase

year.year – Vehicle Year

ratings.title – Category of review like Performance or Comfort

ratings.grade – Overall grade for category

ratings.score – Overall score for category

title – Sub-category of review like Acceleration or Braking

grade – Grade for sub-category

score – Score for sub-category

**What are its limitations i.e. what are some questions that you cannot answer with this data set?**

One of the limitations of this data set is that we do not know the exact vehicle make and model. There are many variations in the trim levels that manufacturers produce, but we are limited in the information provides us about the listing – Make, Model and Year. Because of this limitation, I will use the information provide by Edmunds for the “Base” model and “Standard” equipment where available. The impact of this method is that we could be misrepresenting some of the results without knowing vehicle specifics.

If this project were being run from within Getaround, we would have access to the vehicles’ VIN (vehicle identification number) and could extract specific vehicle model data. With the VIN, we would use [Edmunds SPEC:SQUISH VIN API](http://developer.edmunds.com/api-documentation/vehicle/spec_squishvin/v2/) which outputs a specific vehicle style\_id using the first 11 digits of the VIN without the 9th digit which is a check digit.

Another limitation with the data is that we have no information about a user’s previous rental preferences. Because of this, we will rely on the user to tell us what they want to do with the car – “run errands” or “go to Ikea”. If we were able to see the user’s history, we could make recommendations based on their previous rentals.

Lastly, I had hoped to leverage the reviews and review categories provided by Edmunds, but unfortunately this data is only available for vehicle model years of 2013 or newer. This is providing coverage for less than half of our Getaround population, so I will need to reevaluate how this can be leveraged.

**What kind of cleaning and wrangling did you need to do?**

I’ve leveraged data accessible via APIs from two primary sources - Getaround for the base listing of vehicles and Edmunds for car features and ratings data. I’ve built code that starts with a list of cars from Getaround then uses that data to make specific API calls to get further vehicle details from Edmunds.

Getaround

For Getaround, accessing the data is straightforward and I’ve imported a JSON using a fixed url string and converted to a data frame.

I then iterated through the data and made the following changes so that it is compatible with the Edmunds API and exported the results to .csv.

* Update make and model name to lowercase
* Replace spaces “ “ with dashes “-“
* Make model specific updates
  + For example “mazda3” needs updated to “3”. There are only 5 of these types of modifications needed across the data set.

Edmunds

All of the Edmunds results were in a JSON format, some of which were heavily nested. I primarily used jsonlite to massage this data into data frames.

SPEC: VEHICLE STYLE

Based on the Getaround results, I created a unique list of vehicles Make/Model/Year and made an API call for each. This returns a list of all styles for the combination along with most importantly the styles.id.

SPEC: VEHICLE EQUIPMENT

I filtered the results of the prior call down to one record per Make/Model/Year and used styles.trim to filter by “Base” models only. This left me with a unique styles.id to use to get basic vehicle equipment.

To pull the details of the vehicle equipment, I passed a “standard” filter through the API call that ensured the equipment returned was present on every vehicle we are proxying off of a specific styles.id.

CONTENT: EDMUNDS CAR RATINGS

The ratings API call required vehicle Make/Model/Year/Sub-model from the Vehicle Style API call to return results.

**Any preliminary exploration you’ve performed and your initial findings.**

So far I’ve come up with five use cases that I’d like to build functions for to help a renter locate the best vehicle for their desired usage. I’ve begun to map out the specific data points I will need to rank vehicles based on these cases. The data points will have weights assigned to them depending on the use case. These may evolve as the project moves along.

* “Run Errands” – Focuses on small vehicles and fuel efficiency
* “Trip to Ikea” – Focuses on interior cargo space and vehicle type (truck/SUV)
* “Go for a fun drive” – Focuses on acceleration time, rear-wheel drive, convertible/coupe
* “Fuel Efficient” – Focuses on EPA MPG ratings, electric cars
* “Ski Trip” – Focuses on four-wheel drive, SUV/wagons

**Based on these findings, what approach are you going to take? How has your approach changed from what you initially proposed, if applicable?**

Original Approach:

* Import the data into R via the APIs and save a static data set as .csv for each of the calls.
* Cleanse the data
* Join Getaround listings with Edmunds SPEC: VEHICLE STYLE style\_id on year, make and model.
* Get full list of features from SPEC: VEHICLE EQUIPMENT for the universe of Getaround listings.
* Create buckets of features that map to usage definitions like “running errands”, “economical” or “fun”.
* Get full list of review categories from CONTENT: EDMUNDS CAR RATINGS for the universe of Getaround listings.
* Look at relationships between features, categories and ratings.
* Create algorithm or look for pre-existing recommender systems and test results.

Current Approach:

I’m following the same general approach but there are a couple things I’ll need to adapt for going forward.

I am going to create one data table to store all of the Edmunds vehicle equipment data once I know all of my data points I will utilize in my functions. It is currently spread across a few tables, with much of the data not important to this project. Ideally I will end up with one table for all of the Edmunds vehicle features and another for the ratings.

I will also need to think about how I can leverage the Edmunds ratings data as it covers less than half of the Getaround universe. Perhaps it can be used as an overall rankings factor to skew results up or down slightly depending on good or bad reviews?