Data Science Project
October, 27 2016

Question

Can you predict if a sitter will have churned 180 days after they were approved at the point when you approve their profile?

Defining Churn

At 180 Days after your profile was first approved, you meet the following conditions:

- Your profile is inactive (Explicit)

OR

- You have not had an **Event** in the past 100 days* (Implicit)

*Less than 10% of sitters return who haven't had an event in the past 100 days

Events:

- Booked a Stay
- Sent a Message
- Uploaded a Photo
- Scheduled a Meet and Greet
- Activated a Service
- Set Affirmative Weekend Availability

Creating Churn Prediction

```
SELECT DISTINCT p.id

FROM people_person p

LEFT JOIN (SELECT provider_id,
        MIN(approved_on) AS first_approved,
        MIN(searchable_date) AS first_searchable_date,
        MAX(active) AS currently_active,
        MAX(searchable) AS currently_searchable,
        MAX(deactivated_at) AS deactivated_at
        FROM services_service serv
        GROUP BY 1) serv ON serv.provider_id = p.id

LEFT JOIN (SELECT provider_id,
        MAX(stay_added) as last_booking
        FROM [b_stays]
        GROUP BY 1) booking ON booking.provider_id = p.id

LEFT JOIN (SELECT ss.provider_id,
        MAX(cd.added) as last_activated
        FROM common_deactivatableauditlog cd
        JOIN services_service ss on ss.id = cd.object_id AND cd.content_type_id = 155 AND
cd.activated = 1
        GROUP BY 1) activated ON activated.provider_id = p.id

LEFT JOIN (SELECT ss.provider_id,
        MAX(ss.searchable_date) as last_searchable
        FROM services_service ss
        WHERE searchable_date is not null
        GROUP BY 1) searchable ON searchable.provider_id = p.id

LEFT JOIN (SELECT person_id,
        MAX(added) as last_weekend_availability
        FROM marketing_weekendavailabilitysurvey
        WHERE is_available = 1
        GROUP BY 1) weekend_availability ON weekend_availability.person_id = p.id

LEFT JOIN (SELECT uploader_id,
        MAX(added) AS last_photo_added
        FROM images_image ii
        WHERE content_type_id in (17, 36)
        GROUP BY 1) photos ON photos.uploader_id = p.id

LEFT JOIN (SELECT provider_id,
        MAX(cmg.added) AS last_meet_and_greet
        FROM conversations_meetandgreet cmg
        JOIN conversations_conversation cc on cc.id = cmg.conversation_id
        GROUP BY 1) meet_greet ON meet_greet.provider_id = p.id

LEFT JOIN (SELECT cc.provider_id,
        MAX(cm.sent) as last_message_sent
        FROM conversations_message cm
        JOIN conversations_conversation cc ON cc.id = cm.conversation_id AND cm.sender_id = cc.provider_id
        GROUP BY 1) message ON message.provider_id = p.id

WHERE
p.id > 121934
AND
COALESCE(serv.first_approved,serv.first_searchable_date) < CURDATE() - INTERVAL 180 DAY
AND
(serv.deactivated_at < COALESCE(serv.first_approved,serv.first_searchable_date) + INTERVAL 180 DAY
 OR
(booking.last_booking BETWEEN COALESCE(serv.first_approved,serv.first_searchable_date) + INTERVAL 80 DAY AND
COALESCE(serv.first_approved,serv.first_searchable_date) + INTERVAL 180 DAY
 OR
 activated.last_activated BETWEEN COALESCE(serv.first_approved,serv.first_searchable_date) + INTERVAL 80 DAY AND
COALESCE(serv.first_approved,serv.first_searchable_date) + INTERVAL 180 DAY
 OR
 searchable.last_searchable BETWEEN COALESCE(serv.first_approved,serv.first_searchable_date) + INTERVAL 80 DAY AND
COALESCE(serv.first_approved,serv.first_searchable_date) + INTERVAL 180 DAY
 OR
 weekend_availability.last_weekend_availability BETWEEN COALESCE(serv.first_approved,serv.first_searchable_date) + INTERVAL 80
DAY AND COALESCE(serv.first_approved,serv.first_searchable_date) + INTERVAL 180 DAY
 OR
 photos.last_photo_added BETWEEN COALESCE(serv.first_approved,serv.first_searchable_date) + INTERVAL 80 DAY AND
COALESCE(serv.first_approved,serv.first_searchable_date) + INTERVAL 180 DAY
 OR
 meet_greet.last_meet_and_greet BETWEEN COALESCE(serv.first_approved,serv.first_searchable_date) + INTERVAL 80 DAY AND
COALESCE(serv.first_approved,serv.first_searchable_date) + INTERVAL 180 DAY
 OR
 message.last_message_sent BETWEEN COALESCE(serv.first_approved,serv.first_searchable_date) + INTERVAL 80 DAY AND
COALESCE(serv.first_approved,serv.first_searchable_date) + INTERVAL 180 DAY)
)
```

Data Set Sizing

All sitters approved > 180 days ago that have data sufficiency due to lack of data availability from very early sitters:

- 93,213

Of that group, number that were churned 180 days after approval:

- 47,657 (51.12%)

Feature Set

We can only look at variables known to us at the point at which the sitter's profile is sent for approval.

Initial Set:

Number of Photos	Location
Services Listed	Time to Complete Profile
Day of Week Availability	Dog owner
Protection Package	Children in Home
Building Type	Number of Testimonials Requested
Dogs Allowed on Furniture	Years of Dog Sitting Experience

Creating the Feature Set

```
SELECT DISTINCT p.id,
CASE WHEN photos.images_count IS NOT NULL THEN photos.images_count ELSE 0 END AS photos,
ppl.zip AS zip_code,
CASE WHEN servhome.id IS NOT NULL THEN 1 ELSE 0 END AS listed_boarding,
CASE WHEN servtravel.id IS NOT NULL THEN 1 ELSE 0 END AS listed_traveling,
CASE WHEN servwalk.id IS NOT NULL THEN 1 ELSE 0 END AS listed_walking,
CASE WHEN servdropin.id IS NOT NULL THEN 1 ELSE 0 END AS listed_dropin,
CASE WHEN servdaycare.id IS NOT NULL THEN 1 ELSE 0 END AS listed_daycare,
datediff(COALESCE(serv_info.first_approved,serv_info.first_searchable_date),serv_info.first_service) AS
service_added_to_complete,
datediff(COALESCE(serv_info.first_approved,serv_info.first_searchable_date),p.added) AS account_added_to_complete,
aap.monday,
aap.tuesday,
aap.wednesday,
aap.thursday,
aap.friday,
aap.saturday,
aap.sunday,
CASE WHEN pet.id IS NOT NULL THEN 1 ELSE 0 END AS has_dog,
CASE WHEN iip.id IS NOT NULL THEN 1 ELSE 0 END AS protection_package,
CASE WHEN spp.years_of_experience IS NOT NULL THEN spp.years_of_experience ELSE 0 END AS years_of_experience,
COALESCE(shsp.dogs_allowed_on_bed,shsp.dogs_allowed_on_furniture) AS dogs_on_furniture,
CASE WHEN children_0_5 = 1 OR children_6_12 = 1 THEN 1 ELSE 0 END AS children_in_home,
CASE WHEN building_type = 'hs' THEN 1 ELSE 0 END AS building_home,
CASE WHEN building_type = 'apt' THEN 1 ELSE 0 END AS building_apartment,
CASE WHEN building_type = 'farm' THEN 1 ELSE 0 END AS building_farm,
CASE WHEN building_type IS NULL THEN 1 ELSE 0 END AS building_unknown,
CASE WHEN testimonials.testimonial_requests IS NOT NULL THEN testimonials.testimonial_requests ELSE 0 END AS
testimonial_requests

FROM people_person p
JOIN people_personlocation ppl ON ppl.person_id = p.id
LEFT JOIN services_service servhome ON (servhome.provider_id = p.id AND servhome.service_type_id = 1)
LEFT JOIN services_service servtravel ON (servtravel.provider_id = p.id AND servtravel.service_type_id = 2)
LEFT JOIN services_service servwalk ON (servwalk.provider_id = p.id AND servwalk.service_type_id = 3)
LEFT JOIN services_service servdropin ON (servdropin.provider_id = p.id AND servdropin.service_type_id = 4)
LEFT JOIN services_service servdaycare ON (servdaycare.provider_id = p.id AND servdaycare.service_type_id = 5)

LEFT JOIN (SELECT provider_id,
                  MIN(added) AS first_service,
        MIN(approved_on) AS first_approved,
        MIN(searchable_date) AS first_searchable_date,
        MAX(active) AS currently_active,
        datediff(MIN(serv.added), MIN(serv.met_requirements_on)) AS added_to_complete
        FROM services_service serv
        GROUP BY 1) serv_info ON serv_info.provider_id = p.id

LEFT JOIN (SELECT uploader_id,
        COUNT(ii.id) AS images_count
        FROM images_image ii
        JOIN (SELECT provider_id,
            MIN(approved_on) AS first_approved,
            MIN(searchable_date) AS first_searchable_date
            FROM services_service serv
            GROUP BY 1) serv ON serv.provider_id = ii.uploader_id
        WHERE ii.added <= COALESCE(serv.first_approved, serv.first_searchable_date)
        GROUP BY 1) photos ON photos.uploader_id = p.id

LEFT JOIN (SELECT aap.provider_id,
        MAX(aap.monday) AS monday,
        MAX(aap.tuesday) AS tuesday,
        MAX(aap.wednesday) AS wednesday,
        MAX(aap.thursday) AS thursday,
        MAX(aap.friday) AS friday,
        MAX(aap.saturday) AS saturday,
        MAX(aap.sunday) AS sunday
        FROM availability_availabilitypreferences aap
        GROUP BY 1) aap ON aap.provider_id = p.id

LEFT JOIN pets_pet pet ON pet.owner_id = p.id
LEFT JOIN insurance_insurancepurchase iip ON iip.person_id = p.id
LEFT JOIN services_providerprofile spp ON spp.provider_id = p.id
LEFT JOIN services_hostingservicepreferences shsp ON shsp.provider_id = p.id

LEFT JOIN (SELECT requester_id,
        COUNT(ptr.id) AS testimonial_requests
        FROM people_testimonialrequest ptr
        JOIN (SELECT provider_id,
            MIN(approved_on) AS first_approved,
            MIN(searchable_date) AS first_searchable_date
            FROM services_service serv
            GROUP BY 1) serv ON serv.provider_id = ptr.requester_id
        WHERE ptr.added <= COALESCE(serv.first_approved, serv.first_searchable_date)
        GROUP BY 1) testimonials ON testimonials.requester_id = p.id

WHERE p.id > 121934
AND COALESCE(serv_info.first_approved, serv_info.first_searchable_date) < CURDATE() - INTERVAL 180 DAY
GROUP BY 1
```

Evaluating the Feature Set

I ran into a few issues when evaluating the feature set:

- I did a lot of data cleaning while writing the SQL queries, but a few things managed to get through and caused some issues:
  - People input erroneous data for the number of years since they have been a dog sitter. One person put in 10,000 years and really screwed up the box plot.
  - Some people upload an insane amount of photos, or request an insane amount of testimonials. These data points caused evaluating via box plots to be difficult, but evaluating via groupby/mean was also difficult because there are so many possibilities for the number of photos or number of testimonials requested
- For location, I initially have used zip code, which I learned is too granular. Also, sometimes in our database a 9 digit zip code is used which gave an error re: multiple types of data in 1 column.

Evaluating the Feature Set

After initial analysis, only a couple features which were initially evaluated stand out as likely to be an indicator for churn.

Services Listed

```
In [16]: rover.groupby(['listed_traveling']).churned.mean()

Out[16]: listed_traveling
         0    0.448685
         1    0.544339
         Name: churned, dtype: float64
```

Saturday Availability

```
In [27]: rover.groupby(['saturday']).churned.mean()

Out[27]: saturday
         0    0.439882
         1    0.513986
         Name: churned, dtype: float64
```

Next Steps

Given the initial outcome and difficulties I encountered, here are my next steps:

- Add to & revise data set
  - Find new potential features, including month of approval
  - Remove erroneous & ridiculous data points

- Review with internal teams at Rover
  - Gather feedback on if there are any other features the recommend
  - Review churn definition

- Develop Model!