The objective of the project is to get an understanding of conversion rates for people who landed on a given sign-up page of an existing company through the different states of the signup process. These states are; session, lead, opportunity, complete.

A session occurs when a user lands on the page in question. The moment a user completes the first step in the signup page, which requires them to fill in them email and phone number, the user becomes a lead. They have shown some intent to complete the sign-up process.

The user is then directed to a page requiring them to fill in more information, upon completing said user information, the user converts to an opportunity. The user can then opt to order a device, complete the sign-up process and become a complete signup. An integral part of this venture's success is optimizing the conversion rate from session to complete, while other conversion rates are measures of marketing effectiveness. The higher the conversion from lead to opportunity for example can be used as a proxy to determine the quality of the leads acquired by the marketing team through various channels and campaigns. It is therefore important to understand the channels that appear to bring the highest quality leads (leads with a higher propensity to convert to the opportunity stage). Understanding this conversion rate across different filters will help build an understanding of: i.) how to assign a value to different marketing channels and campaigns-while this was the main initial question, as a result of limitations in my dataset and possibly in the clean-up process to accurately assign a marketing campaign and source to a single activity I opted to leave this part out, ii.) the best days and times to presumably acquire quality leads could help build an understanding of market behavior, iii.) monthly seasonality trends to understand which part of the month yields the best conversion results. We could for example learn that while the company acquires less leads on weekends for a higher price than other weekdays, these users actually have a higher propensity to convert to opportunities for whatever reason.

With this context in mind, I opted to build a Monte Carlo Markov Chain to extract the probabilities of users moving through different signup states and create simulations of hypothetical users passing through these states using the probabilities extracted from my dataset.

Monte Carlo Markov Chain

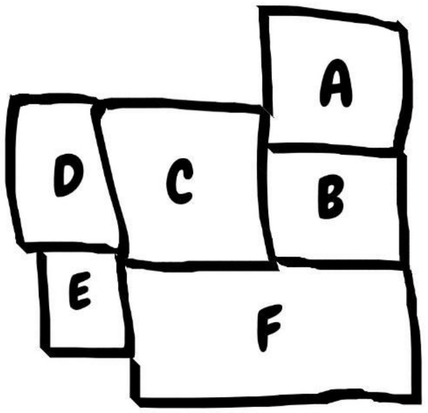
The Monte Carlo Markov Chain (MCMC) model can be defined as a method that can be used 'to approximate the posterior distribution of a parameter of interest by random sampling in a probabilistic space'. This model combines two properties[1]:

i.) Monte Carlo – this can be understood as a statistical technique that is used to model probabilistic systems by creating a predefined number of simulations to produce probable outcomes. This is especially useful for probability issues existing in complicated systems where definitive prediction is not feasible. In the context of this project, the monte carlo part of my project consists of creating multiple simulations of different users landing on the company's signup landing page. This is done in order to get a more accurate view of the conversion rates between each state in the signup process. The greater the number the closer I can get to a more accurate presentation of observed probabilities

ii.) Markov Chain – refers to a mathematical system where we look at the probability of something moving from one state to another. The process of moving to another state is

[1] A simple introduction to Markov Chain Monte-Carlo sampling, available at: https://link.springer.com/article/10.3758/s13423-016-1015-8

referred to as transitioning and typically uses the conditional probability concept from Bayesian statistics. We can for example think of this as modeling the probability of a person moving between different rooms in a house. Using the image below as an analogy.



Assuming the only entrance is in room A, we know that a person cannot move from room A to room F, if you are in a room you can only move to an adjacent room. Additionally, we know that a person can opt to stay in the room they are currently in. For each existing transition we have a transition probability that simply states what the probability is of that transition occurring based on the current room the person is (this is the conditional aspect of the probability). Using this we can build a transition matrix with conditional probabilities for each state change. In the context of this project I am looking at the conditional probability of a user moving to a different signup state based on their current state. However, in this context transitions can only be in one direction, a user cannot for example move from backwards from lead to session, nor can the user transition to the same state s/he is in. Where this happens, the user has not moved through the signup process.

Data Clean Up

My dataset contained rows for each activity a user carried out on the company's site. This included activities that while important where not particularly relevant to the core question I sought to answer. In order to focus purely on users who landed on a given landing page, I limited my dataset to users who had landed on pages that appeared to be associated with the main signup pages used by the company. I added labels to each user's state based on the landing page associated with the row. Additionally, I removed possible duplicates by looking at the anonymous id associated with the user, along with their referral url (the page they came from directly before landing on the page currently associated with their row) and the time in which this activity occurred. Filtering for similar anonymous ids would remove all other activity carried out by that user, while filtering for cases where the anonymous id, timestamp and referral url would ensure that I am only filtering out cases where a clear duplicate exists. For the purposes of this analysis I limited the date range of my data to 2 years, focusing on more recent data while trying to ensure that I would have enough data to make my model useful.

Associated SQL Script:

Upon completion of the extraction process, the data was then relabeled for consistency and ease of interpretation. In this part of the clean process assumptions were made that were for example the utm source contained the word adwords or the referrer domain contained DoubleClick, the associated source was Google Ads. Heading where given to each column and the data was saved as a csv for further analysis – this file will be referred to as the 'final attribution' file.

Python cleanup:

Separating the csv into multiple csv files representing each step

For the purposes of building a pipeline that would run through each step necessary to produce my final output (the simulations of users passing through different states), with the advice from my mentor, I opted to use the Luigi package. According to its documentation this is a package initially developed to 'address all the plumbing typically associated with long-running batch processes', giving the end-user the ability to stitch different takes together enabling me to run a pipeline of tasks while also enabling me to parallelize several processes in a given task. There is also the option to access a web interface showing options to view pending, running and completed tasks along with a visualization of the running workflow and any dependencies associated with each task. This is a good way to get a more intuitive understanding of how my pipeline runs, it can be accessed by running the luigid command in terminal to start up the daemon and opening the interface through this url: http://localhost:8082/.

To create this pipeline, I broke down every major task into a class and created separate python files for each class. This would serve the purpose of making any sort of debugging easier by simplifying the process of finding bugs. Each class created generally has three functions:

i.) A requires function – this function runs first and looks at the dependencies required to run the task(s) in the task. In the context of my pipeline, the 'requires' function was used to assign values to the parameters in the previous task. This uses the concept of inheritance making each preceding class a child of the following class inheriting the parameter values defined in the parent requires class (the concept of inheritance and how it applies to this project is explained in more detail in a later paragraph).

ii.) A run function – this contains the core logic of the task that needs to be run.

iii.) An output target where the output of the task is written.

The first task created in this pipeline involved taking the final attribution file, looking at the unique states as labelled in said file and creating multiple csv files for each unique state identified.

Separate _csv file:
https://github.com/EmmS21/Springboard-Capstones/blob/master/AttributionModel/Capstone/separate_csv.py

State to state transitions

The next two tasks involve:

i.) creating a column indicated the different utm sources a user clicked on before completing the signup process. The purpose of this was to create a visualization of different ad variations used by users before completing the signup process and was more for visualization purposes

ii.) returning a boolean label for each user depending on whether or not they moved from the previous state to the next state. 1 represents that the user moved from one state to another (e.g the user moved from session to lead) and 0 the opposite. Three files where then created for each state transition with the anonymous id and the user's associated Boolean value for each state change.

State to state transitions 1:
https://github.com/EmmS21/Springboard-Capstones/blob/master/AttributionModel/Capstone/state_to_state_transitions.py

State to state transitions 2:
https://github.com/EmmS21/Springboard-Capstones/blob/master/AttributionModel/Capstone/state_to_state_transitions2.py

Getting probability distributions using gaussian kde module

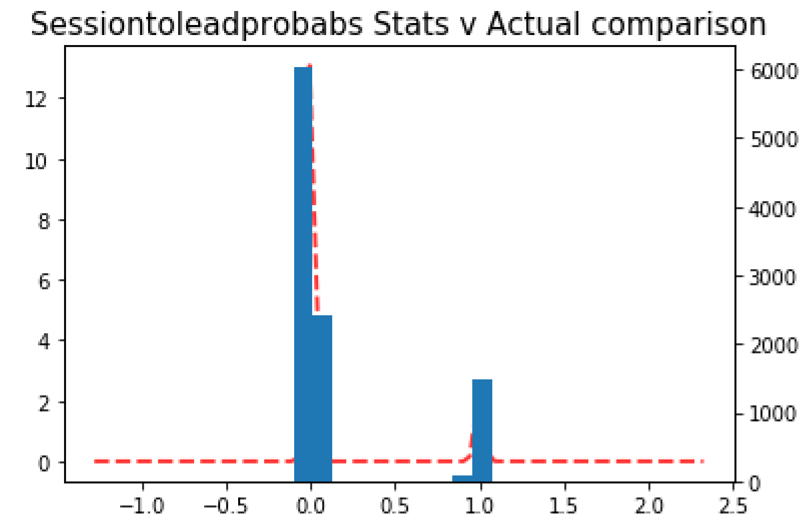
In order to return the probability distribution associated with each state transition I used the gaussian_kde module in the SciPy package to get an estimation of the probability distributions using kernel density estimation (which returns an estimation of the probability density function). The output of this task was 3 files for each state change along with the associated probability distribution. These files were saved as pickle files, which appears to be more efficient when dealing with big and complex data, additionally, atleast according to what I read, serializing my data as a pickle is faster to both read and write than saving it to a csv file.

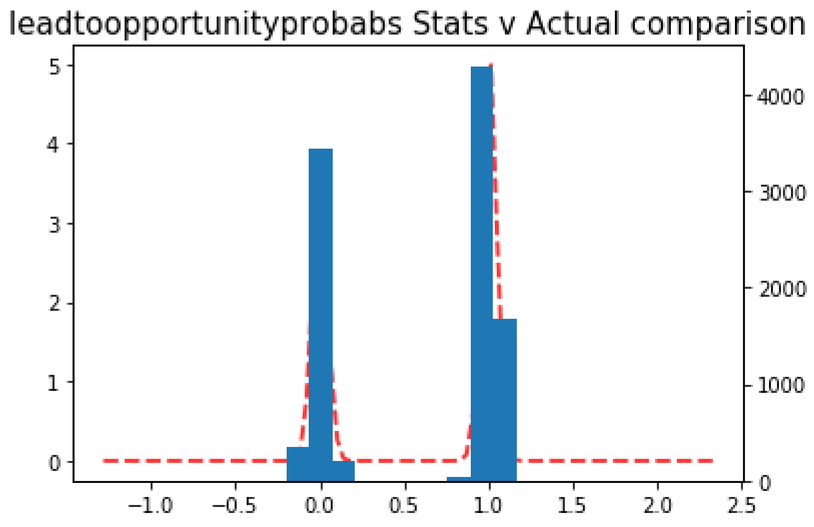
Gaussian_kdefit:
https://github.com/EmmS21/Springboard-Capstones/blob/master/AttributionModel/Capstone/gaussian_kdefit.py

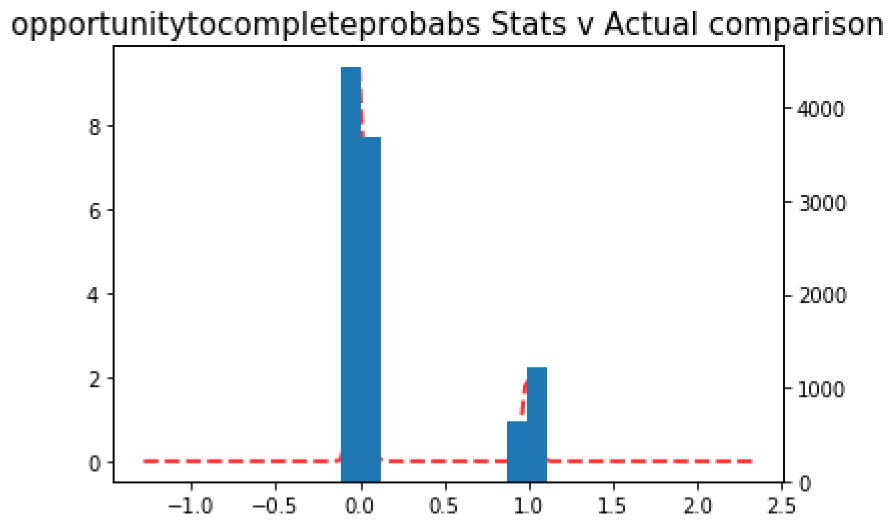
Extracting samples of probabilities and comparing samples with population

An important part of the pipeline involved extracting a given number of samples from the probability distributions created in the previous class. For the purposes of this project, I extracted a sample of 10000 probabilities, these samples would be used to run the simulations required in this model. In order to assess whether the samples represented an accurate snapshot of the population file. I created a file to visualize the 1's 0's associated with 10000 users sampled by this file and compared this distribution to that of my population file (represented by the 3 pickles for each state change).

As the output of this file required a visualization to compare these files, this 'task' did not constitute part of my pipeline and was instead carried out in a Jupyter file to allow for easier comparison.







As indicated in the above visualizations, distribution of transitions between states appear to be similar for both the sample and population data indicating that the samples are representative of the population data.

Getting Samples:
https://github.com/EmmS21/Springboard-Capstones/blob/master/AttributionModel/Capstone/get_samples.py

Visualizing output:
https://github.com/EmmS21/Springboard-Capstones/blob/master/AttributionModel/Notebooks/Samplesvactuals.ipynb

State to state machine

This class can be regarded as the work-horse of the entire pipeline as it builds the core logic of my MCMC model. The class creates the monte carlo simulations of hypothetical users landing on the company's signup landing page and moving across the different signup states. To determine whether or not a given simulated user moved from one state to another, I compared a probability extracted from my samples with a random float between 0 and 1, if the probability is found to be greater than the random float, the user moves to the next state, with the opposite applying in the instance that the probability is not greater than the random float. The logic being if enough simulations are created, this comparison will yield a similar distribution of state changes as the population data, we will get the probability of hypothetical users moving to different states that is almost identical to the observed probabilities. This happens because we are dealing with probability distributions, the probability closest to the 'real observed probability' will be selected more often than other probabilities (there is a higher probability of us choosing the numbers at closer to the center of the probability distribution curve).

The output of this class is multiple csv files for each observation (between 0 and the number of samples extracted), each with Boolean values to determine whether or not a given user converted from session to lead to opportunity to complete

State to state machine:
https://github.com/EmmS21/Springboard-Capstones/blob/master/AttributionModel/Capstone/state_to_state_machine.py

Wrapper

The wrapper file ties everything together. To understand how, it is imperative to understand the inheritance mechanism in object-oriented programming. To put it simply there is a parent and a child class each of these have certain properties and behaviours. Much like in the real world, we can think of this relationship is a hierarchy of sorts, the child can inherit either individual or multiple properties and behaviours directly from its parents. The hierarchy forms when a child class can also be considered as a parent of the child class that inherits from it, the more child classes are created the longer the hierarchy. In this instance the wrapper is the parent class which pre defines the size of the sample we are extracting (and subsequently the number of observation simulations we create), it assigns these values to the parameters in the state to state machine task. The class then use the glob module to find all the path names associated with the csv files created by the state to state machine and concatenates them into a single file.

Parent wrapper:
https://github.com/EmmS21/Springboard-Capstones/blob/master/AttributionModel/Capstone/parent_wrapper.py

Comparing simulations to population distribution

Using code similar to the code used to visualize and compare my samples to the population data, I used this to visualize the conversion rates using different filter; ranging from understanding the conversion rates between different utm sources, weekdays, hours of the day, and mobile/pc devices to understand what the most effective marketing source has been (in acquiring high value leads) and other parameters that appear to have a significant impact on conversion rates. It is important to reiterate that the limitations of this project are that: i.) the data is limited to users who at some point over the last two years landed on specific signup landing page with the assumption that all users who became customers must have, at some point, gone through the signup process (or possibly different variations of this signup process), ii.) assumptions were made about the signup states a user was in given the url associated with a given row, the definition of these states may not have been static over the last 2 years, iii.) assumptions were made with regards to how to classify a user’s associated utm source (e.g as Facebook or Google Ads), iv.) not all marketing campaigns are intended to drive signups, in certain cases the purpose of a campaign is to drive brand awareness which increases the potential customer’s understanding of the product and its benefits consequently reducing the cost of acquiring said potential customer when s/he is exposed to an ad that has signup completion as its objective, v.) I made the assumption that all completed signups would contain the phrase ‘complete’ in their url. However, this may not necessarily be the case. It may for example have been advantageous to evaluate whether a given anonymous id could be linked to a business won (this is defined as a business that has completed the signup process and internal vetting). For the sake of this project these assum

From observing the sequence of ads a user may have clicked before completing the signup process it is necessary to understand that since a user may have for example, clicked an ad on Facebook, Google Ads and then an email campaign, it is imperative to A/B test the conversion rates for each utm source and determine which particular source appeared to play a large role in bringing high quality leads. With advice from my mentor I opted to use the Mann Whitney U Test

Comparing simulations:
https://github.com/EmmS21/Springboard-Capstones/blob/master/AttributionModel/Capstone/parent_wrapper.py

Mann Whitney U Test

In order to compare the conversion rates from two different filters that have been used to create simulations (e.g day of the week someone started signing up), the Mann Whitney U test can be used to determine whether there is a statistical difference between the two conversion rates results. This test is especially useful in cases where the results come from the same population, which is the case in this instance. The assumptions for this test are that the data must come from random samples and the samples must be independent of each other. Implementation of this test is made easy using the mannwhitneyu module existing in the SciPy package.

Analysis

For the analysis carried out using this model I focused on two questions;

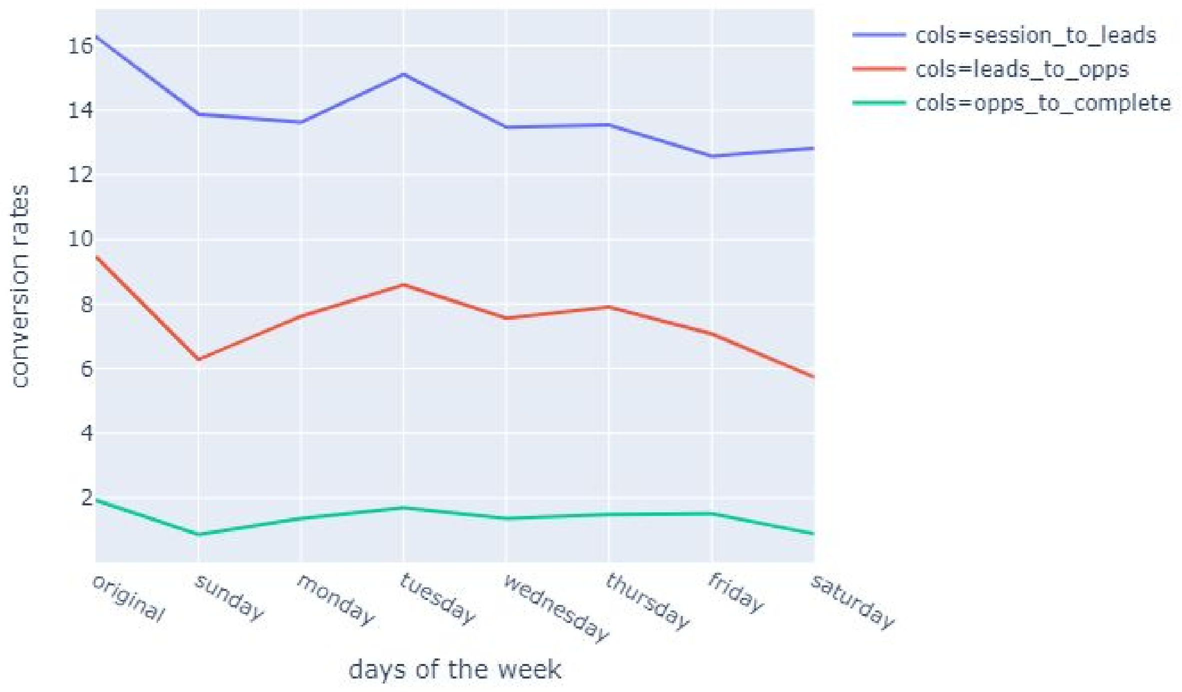
i.) On what day and time range is there a higher propensity for the company to get higher quality leads. Quality in this context is measured by leads converting to opportunities.
ii.) Understanding conversion rate across different devices

Using the following questions, I used filters related to time of day, day and device used to create simulations for conversion rates focused on each parameter. For this group of filters (e.g days of the week) I created a pandas dataframe appended with a row with the conversion rate for each state transition per filter.

Days of the week

The transition from session to leads and more so the transition from lead to opportunity is useful metric in measuring how effective marketing efforts are at acquiring users who have a strong intention to become customers. The more you are able to purchase eyeballs that convert to an opportunity. In this instance, eyeballs that take the time to fill in business and personal details indicating a stronger interest to become a customer, the more targeted your marketing efforts are at your ideal customer. As such this is a metric marketing teams would presumably want to optimize and understand how to optimize it more effectively.

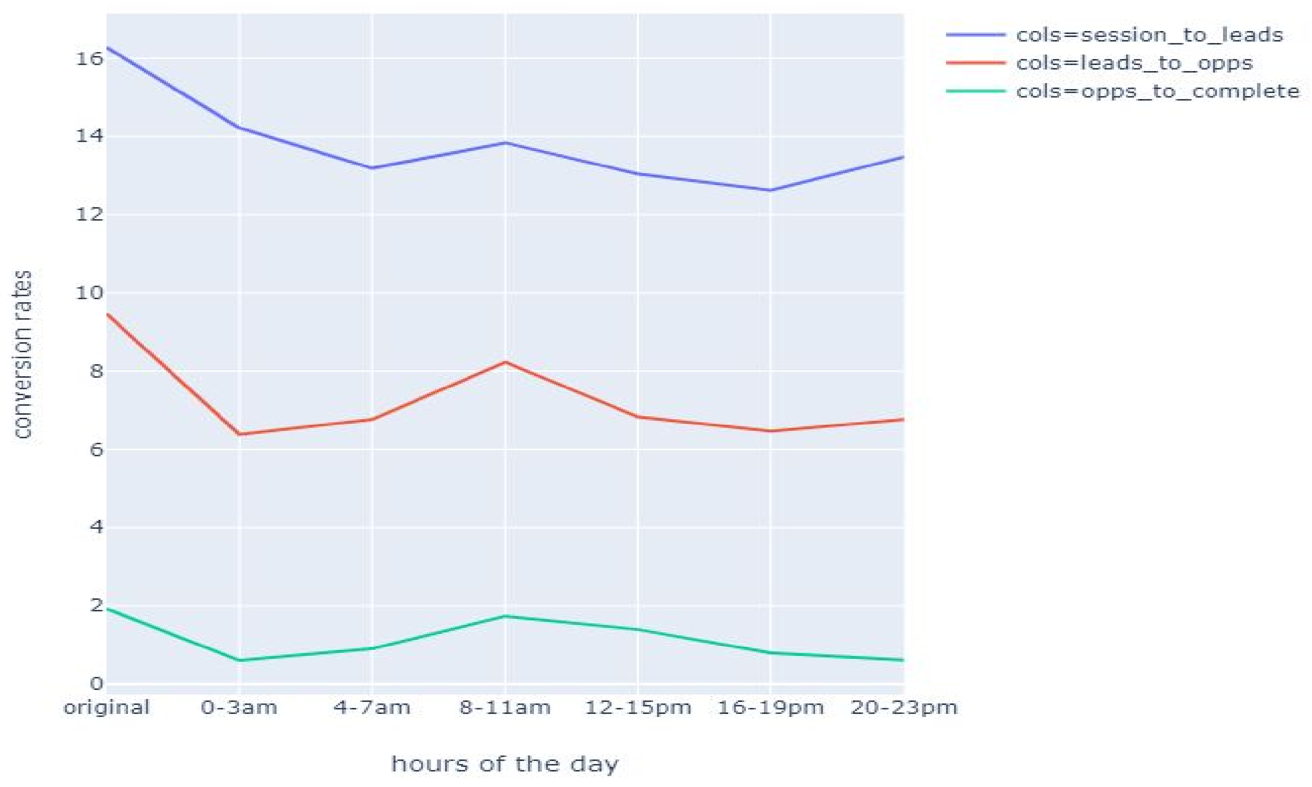
While it would have been a lot more valuable understanding which campaigns and marketing channels seem to generate higher conversion rates, since this is not feasible with the limitations of the current dataset, the day of the week should at least help us understand which days marketing efforts are yielding the best results (possibly as a result of potential users being more inclined to sign up on that particular day) and comparing and contrasting conversions on weekends and weekdays.



From the insights derived, the value of leads coming through on weekends appears to be questionable at best with the lowest two leads to opportunities conversion rates appearing on Saturday and Sunday while these rates appear to peak on Tuesday with the other weekdays delivering somewhat similar results. Similar results appear for conversions from opportunity to completion of the signup process.

Time of Day

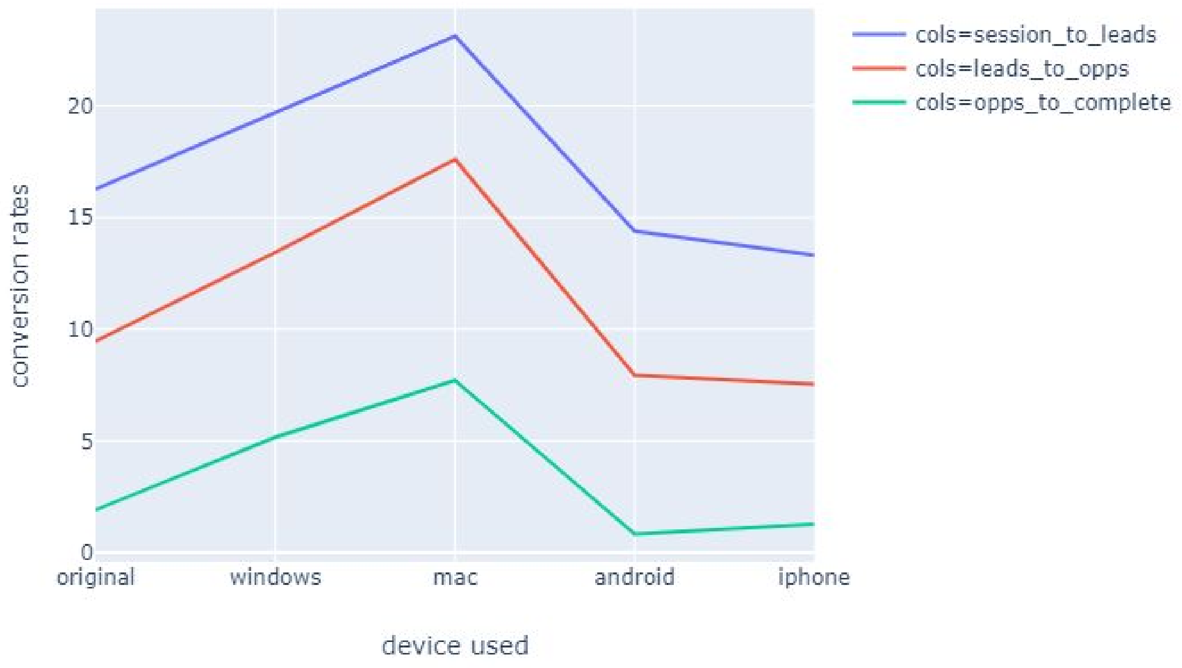
Another group of parameters evaluated was the time of day a given user started their signup process. From inception the assumption was that I would see higher conversion from lead to opportunity and opportunity to complete during the day, primarily because this company's core market are merchants - ie. people who run small businesses. These people would presumably be more active during working hours.



The results seemed to validate this assumption. Conversion rates through the different signup states appear to peak at around 8-11am, people appear to be less inclined to convert from opportunity to completing the signup process after 8pm.

Devices Used

For the devices split I labelled devices as windows,mac, linux, android, windows phone, blackberry, tizen, iphone, but specifically focused on the most used devices because with certain devices, there was no data for each state transition. As such, I specifically focused on computers running on a windows operating system, Android, Mac and iPhone devices . Much like the previous group splits the objective was to measure conversion rates differences between each device



The visualization created seems to indicate a clear difference between mobile and PC, higher conversion rates are observed for both computers running on windows and macs, put in numerical terms conversion rates for windows machines range from 19.67% (from session to lead) to 5.17 for the final transition from opportunity to completing the signup page, mac devices show a higher transition from session to leads (23.12%) and 7.71% conversion from opportunity to complete. While mobile devices show conversion rates of 14.39% and 13.31% for android and iphone users transitioning from session to leads and 0.83% and 1.26% from opportunity to completion.

Since the data I am looking at is over a 2 year range, it may indicate a signup process that either;

i.) may not have been easy to interact with for mobile users,

ii.) may have required users to switch devices, maybe because it required a lot of typing or people are more inclined to fill in the details required in the sign up process when they are at home and have more time. My assumption here is that people are more inclined to fill in short forms via mobile and longer forms that require more time and typing on their laptops or PCs.

The difference in conversion rates may also be explained by a mixture of the above two factors. A question then arises whether or not the difference in conversion rates between mac and windows devices is by pure chance or not. To test whether the distributions from these two devices are similar (this is the null hypothesis) using the Mann Whitney U Test. With this the question being asked was assuming these two filters have been sampled from identical distributions, what is the probability that the differences are purely by chance (as a result of random sampling). In this case the low p-value of

9.865876450376946e-10 allows me to reject the null hypothesis and assume that the differences are not due to chance.