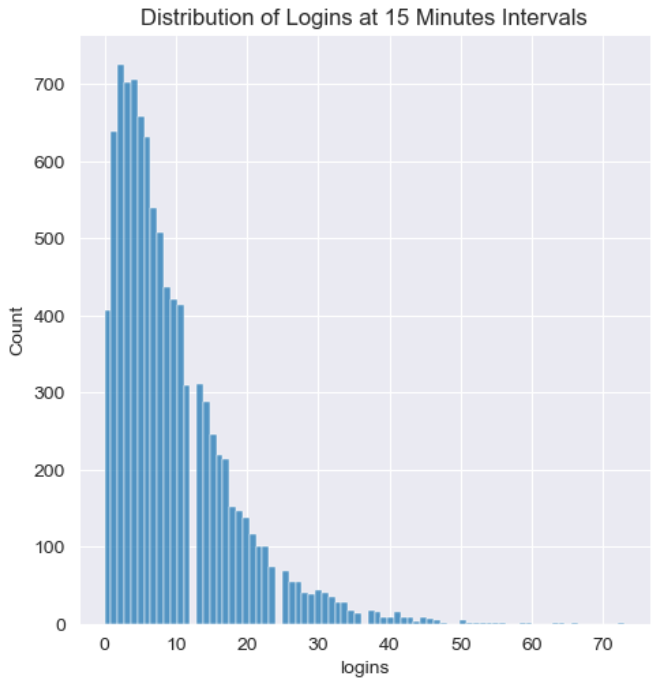
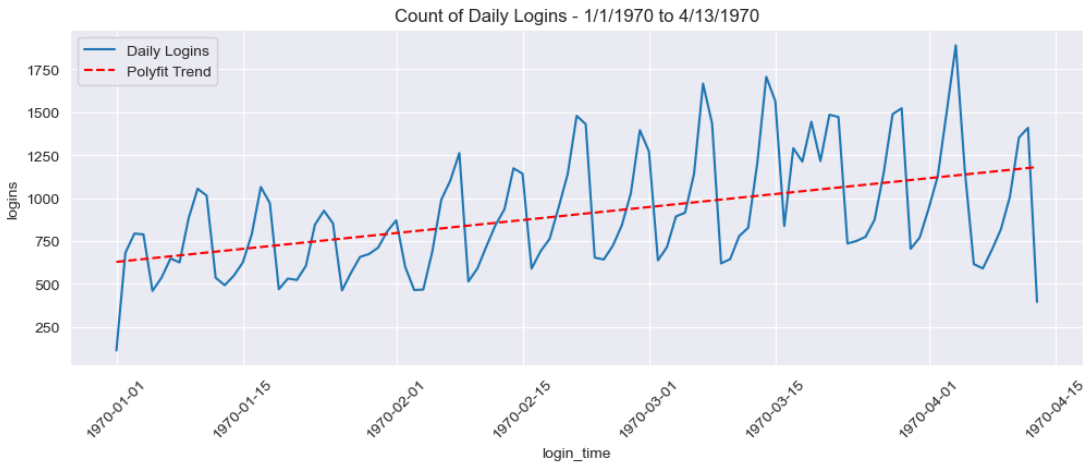
Data Analysis Interview Challenge

Part 1 – Exploratory Data Analysis

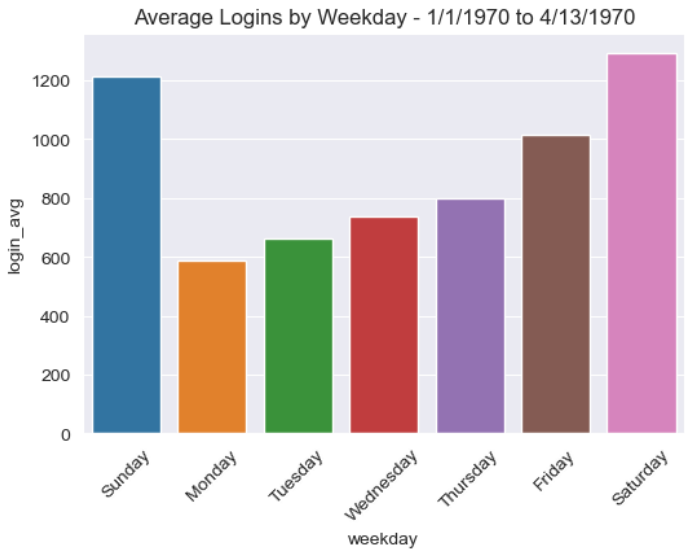
After loading the data, aggregating into 15-minute intervals, and doing some cleaning, I visualized with a histogram:



From this initial look at our 15-minute aggregate data it was clear we'd have some outliers, but I started by looking at a longer trend. The small intervals were a bit busy, so I created this daily view:

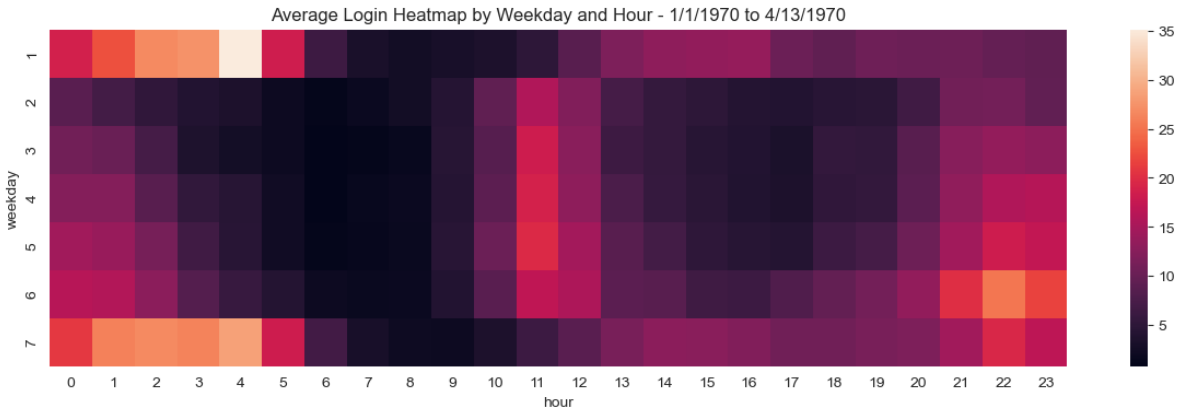


Login traffic appears to be on an upward trend, but there are a lot of recurring peaks. The reason for this is a change in traffic based on weekday. Here are the total daily averages for each weekday:



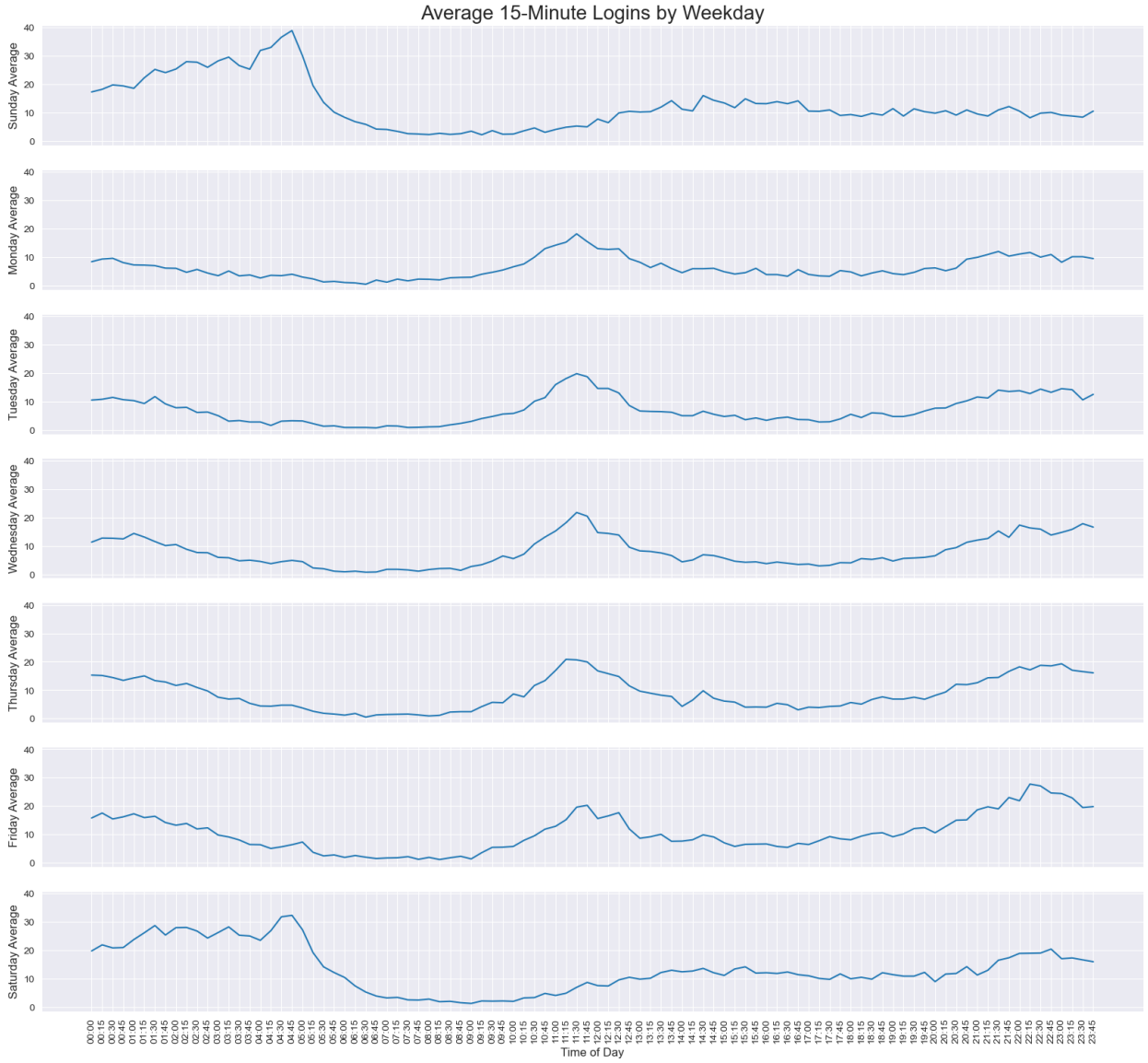
Logins are heavy on the weekends and grow through the week – peaking on Saturday. This is usual for the team managing the website, but it doesn't give enough details on the daily cycles themselves. To understand this piece, I took two different looks at similar data.

1. Heatmap by weekday and hour of day



This is a quick view, but we can see the lighter areas being more traffic on the weekends (as we already knew, and mostly in the early hours of the morning. While the weekdays are busiest around noon and in the evening. But this view is limited because the 'heat' is based on the global averages.

2. Independent daily cycle by weekday and 15-min intervals



This is a large graph, but it better shows the daily detail independent of the others. This really highlights the variation in cycle on weekends vs. weekdays. With peaks on the weekends likely due to riders needing safe transportation home from bars and clubs. This is important to know for our team that manages the servers but also our operations and growth teams to make sure we have enough of a driver pool available at the right time to supports these needs.

Part 2 – Experiment & Metrics Design

Key Measure of Success:

I would choose to measure driver partner bridge crossings. Within the realm of what is measurable as a city manager, I think this makes the most sense because our goal is more dual-city availability. But availability isn't a data point that is easy to access or aggregate. We could also have the balancing measure of total bridge crossings, and track driver partner crossing as a % of total crossings to not mistake overall traffic increase with a successful experiment.

Practical Experiment:

My idea for this experiment would be to include 3 different groups of driver partners: one control with no reimbursement, one with an after-the-fact process for manually adding how much you paid for tolls, and one a sort of "FastTrack" that allows for automatic free crossing without having to worry about paying and being reimbursed (assuming we could work with the bridge authority to set this up). Then compare the 90 days pre- and post-intervention with an Analysis of Variance (ANOVA) test on the results to see if the variation is statistically significant.

To interpret these results, we would check the results against the null hypothesis that there is no difference between the means of the individual group changes, rejecting if we get a p-value below the standard 0.05 (with one test). If we can reject that null hypothesis, we could then take the groups and compare individually against another using something like a Tukey Multiple Comparison of Means to see what is driving the difference of means. And recommend the intervention that is most statistically likely to incentivize more driver partners to operate in both cities.

Part 3 – Predictive Modeling

Cleaning & Exploring:

As a first step I checked the categorical features for consistency in the labeling – everything looked good. For numerical features I graphed the histograms and noticed what looked to be some outliers.

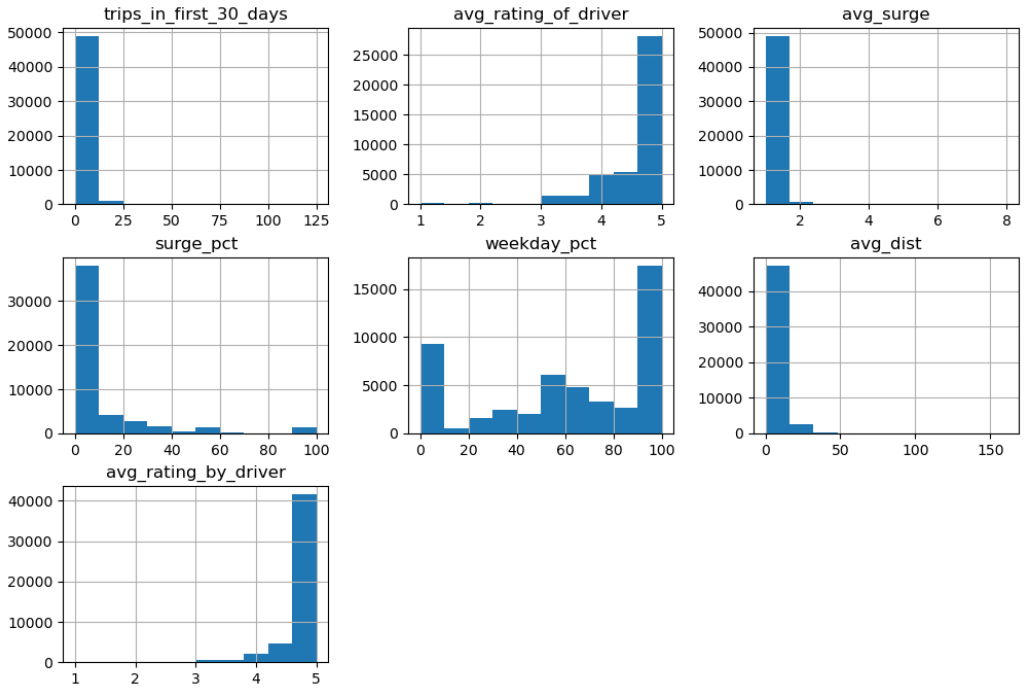


Figure 1: Histograms of our Numerical Features

I looked at each one and dropped records with:

- First 30-Day Trips > 12 – Even if the data is accurate, I felt this wasn't representative of our most common users. 12 was about 3 standard deviations from the mean.
- Average surge >= 3 – same logic as the first drop
- Average distance – this threshold allowed for more variation than the first two, but I didn't want to exclude riders that may make trips to an airport that could be further outside the city.
- Duplicated records (had 8 pairs of complete duplicates)
- Records with surge percent > 0, but average surge ==1
  - This math didn't make sense to me, so I dropped the records to avoid data inconsistency

These changes dropped 2,141 records leaving 96% of the original data.

Feature Engineering:

To prepare for better modeling I made these changes:

- Encoded the 'ultimate_black_user' field
- Added an account age based on the assumption of the data being pulled on 7/1/14
  - This assumption is based on the max 'last_trip_date' being 7/1
- Breakout sign-up and last-trip dates to weekdays, and drop the original dates
  - I didn't feel the datetimes would add value to the model
- Added a target variable for retention using the logic of having a trip within the last 30 days.

I thought about adding a field for days since last trip but felt that would be too correlated with our target variable.

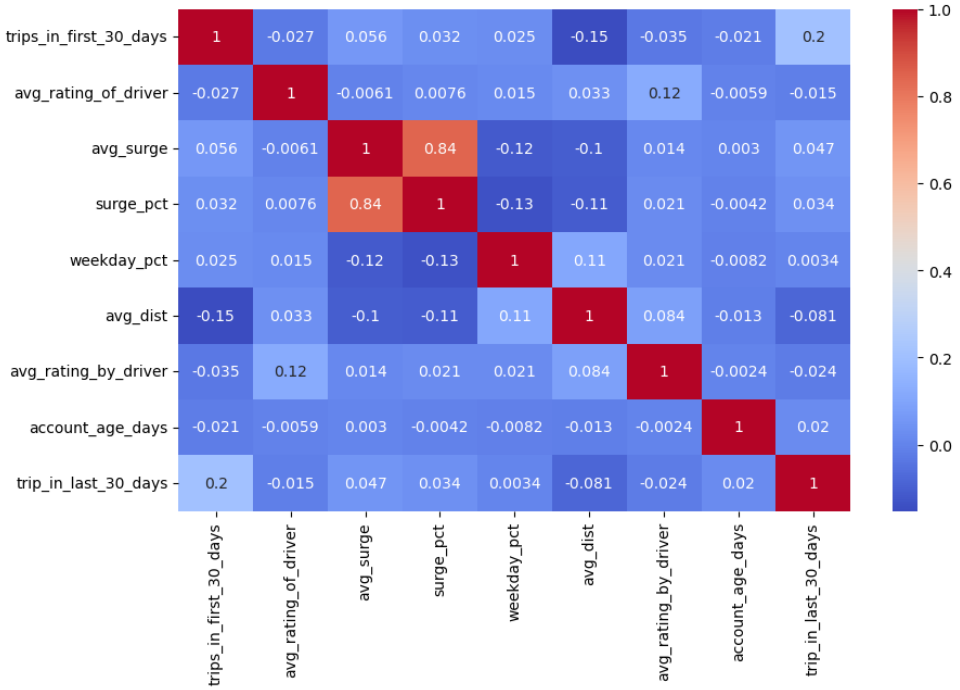


Figure 2: Correlation Matrix of Numeric Fields

There is some inter-correlation between our surge fields, but I'm curious how each of those will impact our customer churn, so I left both in the model.

Pre-Processing

After splitting training and test sets I setup these steps in a pipeline, separating categorical and numerical variables.

Categorical:

- Impute missing values using the 'most_frequent' strategy
- Encode categorical variables using OneHotEncoder

Numerical:

- Impute missing values using the 'median' strategy
- Scale using StandardScaler()

I considered applying different scalers based on the distribution of the numerical features but decided against it due to the simplicity and consistency of just using the StandardScaler.

Modeling

Using the pipeline for pre-processing, I tested two types of models: Logistic Regression and Random Forest Classifier. I fit various hyperparameters with GridSearchCV, prioritizing the precision score.

In the end, I settled on using the Random Forest Classifier due to its higher recall on the 'churn' class – meaning we are better at predicting the riders that will churn. And it's higher 'retain' precision – meaning we are less likely to predict riders to churn when they will not. To me this lowers the risk of over-spending on unnecessary initiatives.

```
RandomForestClassifier
RandomForestClassifier(max_depth=4, min_samples_split=0.4, n_estimators=200,
                       random_state=42)
```

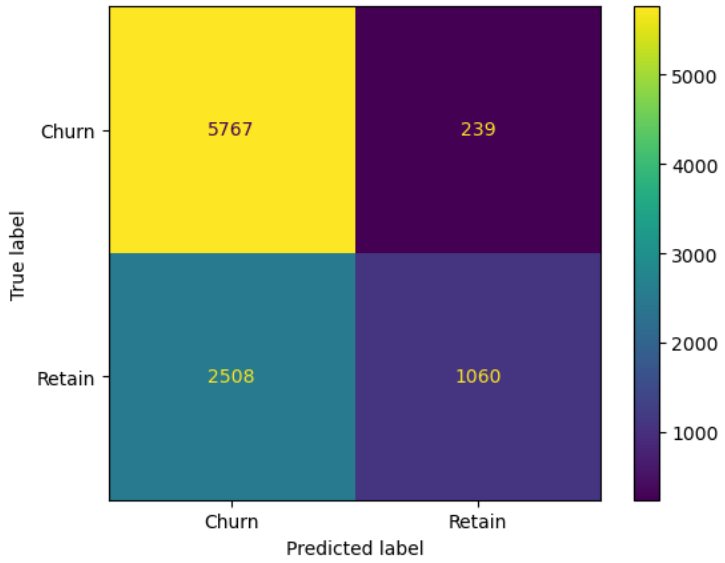


Figure 3&4: Model and Confusion Matrix for Model on Test Data

Classification Report

	Precision	Recall	F1-Score	Support
Churn	0.70	0.96	0.81	6,006
Retain	0.82	0.30	0.44	3,568
Accuracy			0.71	9,574
Macro Avg	0.76	0.63	0.62	9,574
Weighted Avg	0.74	0.71	0.67	9,574

Figure 5: Classification Report for Random Forest Model on Test Data

My main concern is that the trees be pruned enough to generalize well to new data. In comparing train and test sets, results were consistent, which made me feel better about the potential issue of overfitting.

```
Train Performance - Random Forest
---------------------------------
Model rf Predictions: AUC 0.62 | Accuracy 0.71 | Recall 0.29 | Precision 0.8 | F1 0.42

Test Performance - Random Forest
--------------------------------
Model rf Predictions: AUC 0.63 | Accuracy 0.71 | Recall 0.3 | Precision 0.82 | F1 0.44
```

Figure 6: Train vs. Test Performance Summary

In this model, these features were found to be the most important in understanding retention and churn:

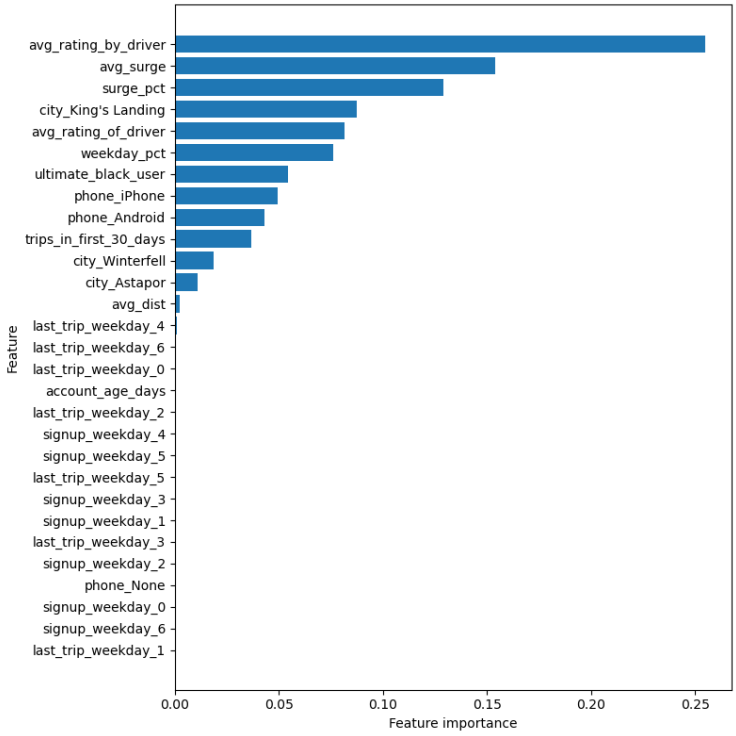


Figure 7: Feature Important of Random Forest Model

Leveraging the Data

With this model in place, Ultimate can use it to flag which riders are most likely to churn. It will be important to consider these most impactful fields – average rating by driver, surge, etc. to plan interventions in on the variables that make the biggest difference. In doing so I think they can reduce churn in a targeted, cost-effective way.